МЕЖДУНАРОДНЫЙ СТАНДАРТ

ISO 8879

Первое издание

1986-10-15

ОБРАБОТКА ИНФОРМАЦИИ. ТЕКСТОВЫЕ И ОФИСНЫЕ СИСТЕМЫ. СТАНДАРТНЫЙ ОБОБЩЕННЫЙ ЯЗЫК РАЗМЕТКИ (SGML)

INFORMATION PROCESSING. TEXT AND OFFICE SYSTEMS. STANDARD GENERALIZED MARKUP LANGUAGE (SGML)

Регистрационный номер

ISO 8879-1986

ПРЕДИСЛОВИЕ

Международная организация по стандартизации (ИСО) представляет собой всемирную федерацию национальных органов по стандартизации (членов ИСО). Работа по подготовке международных стандартов обычно ведется через технические комитеты ИСО. Каждый член, заинтересованный в тематике, ради которой был организован технический комитет, имеет право быть представленным в этом комитете. Правительственные и неправительственные международные организации, наряду с ИСО, также принимают участие в этой работе.

Проекты международных стандартов, одобренные техническими комитетами, перед их принятием Советом ИСО в качестве международных стандартов рассылаются членам ИСО на утверждение. Они утверждаются в соответствии с процедурами ИСО, требующими утверждения не менее чем 75% членов ИСО, принимающих участие в голосовании.

Международный стандарт ИСО 8879 был подготовлен Техническим комитетом ИСО/ТК 97 “*Системы обработки информации*”.

Пользователи должны иметь ввиду, что все международные стандарты подлежат периодическому пересмотру, и что в любой ссылке, сделанной в настоящем документе на любой другой международный стандарт, подразумевается его последнее издание, если не указано иное.

УДК 681.3.06

Дескрипторы: обработка данных, документация, логическая структура, программирование (компьютеры), искусственные языки, языки программирования.

СОДЕРЖАНИЕ

0 Введение 1
0.1 Предпосылки 1
0.2 Цели 4
0.3 Организация 9
1 Область распространения 10
2 Область применения 11
3 Нормативные ссылки 12
4 Определения 13
5 Нотация 63
5.1 Синтаксические маркеры 63
5.2 Символы упорядочивания и выбора 64
6 Структура объекта 65
6.1 Документ SGML 65
6.2 Объекты SGML 66
6.2.1 S разделитель 67
6.2.2 Конец объекта 67
6.2.3 Подразумеваемое объявление SGML 67
6.3 Объект данных, не относящихся к SGML 67
7 Структура элемента 68
7.1 Пролог 68
7.2 Элемент документа 69
7.2.1 Пределы 69
7.3 Элемент 69
7.3.1 Минимизация пропущенным тегом 69
7.3.1.1 Пропуск начального тега 69
7.3.1.2 Пропуск конечного тега 70
7.3.2 Минимизация тегом данных 70

7.3.3 Количества 72
7.4 Начальный тег 72
7.4.1 Минимизация 72
7.4.1.1 Пустой начальный тег 72
7.4.1.2 Незакрытый начальный тег 73
7.4.1.3 Начальный тег, обеспечивающий NET 73
7.4.2 Количества 73
7.5 Конечный тег 74
7.5.1 Минимизация 74
7.5.1.1 Пустой конечный тег 74
7.5.1.2 Незакрытый конечный тег 74
7.5.1.3 Нулевой конечный тег 75
7.6 Содержание 75
7.6.1 Границы записи 76
7.7 Спецификация типа документа 78
7.7.1 Ссылки на общие объекты 78
7.7.2 Ссылки на объекты параметра 78
7.8 Спецификация родового идентификатора (GI) 79
7.8.1 Функция ранга 79
7.8.1.1 Полный родовой идентификатор 79
7.8.1.2 База ранга 79
7.9 Список спецификаций атрибутов 80
7.9.1 Минимизация 80
7.9.1.1 Спецификация пропущенного атрибута 80
7.9.1.2 Имя пропущенного атрибута 81
7.9.2 Количества 81
7.9.3 Спецификация значения атрибута 81
7.9.3.1 Минимизация 82
7.9.4 Значение атрибута 82

7.9.4.1 Синтаксические требования 82
7.9.4.2 Фиксированный атрибут 83
7.9.4.3 Имя общего объекта 83
7.9.4.4 Нотация 83
7.9.4.5 Количества 84
8 Команда обработки 84
8.1 Количества 85
9 Общие конструкции 85
9.1 Заменяемые символьные данные 85
9.2 Символьные данные 85
9.2.1 Символ SGML 86
9.2.2 Символ функции 86
9.3 Имя 86
9.3.1 Количества 86
9.4 Ссылки на объекты 87
9.4.1 Количества 87
9.4.2 Пределы 87
9.4.3 Запутывающие ссылки на объект 88
9.4.4 Поименованная ссылка на объект 89
9.4.5 Конец ссылки 89
9.4.6 Краткая ссылка 89
9.4.6.1 Строка эквивалентной ссылки 89
9.5 Символьная ссылка 91
9.6 Распознавание разграничителей 91
9.6.1 Режимы распознавания 92
9.6.2 Контекстные ограничения 93
9.6.3 Порядок распознавания 94
9.6.4 Разграничители, начинающиеся с одинаковых символов 94
9.6.5 Краткие ссылки с последовательностями пропусков 95

9.6.5.1 Количества 96
9.6.6 Символы имени 96
9.7 Подавление разметки 96
9.8 Доступный объем 97
10 Объявления разметки: общие сведения 99
10.1 Части объявлений 99
10.1.1 Разделитель параметров 99
10.1.2 Литерал параметра 100
10.1.2.1 Количества 101
10.1.3 Группа 101
10.1.3.1 Количества 102
10.1.4 Разделитель объявлений 102
10.1.5 Тип связанного элемента 102
10.1.6 Внешний идентификатор 102
10.1.6.1 Количества 103
10.1.6.2 Доступные объемы 103
10.1.7 Минимальный литерал 103
10.1.7.1 Количества 104
10.2 Формальный публичный идентификатор 104
10.2.1 Идентификатор владельца 104
10.2.1.1 Идентификатор владельца ИСО 105
10.2.1.2 Зарегистрированный идентификатор владельца 105
10.2.1.3 Незарегистрированный идентификатор владельца 106
10.2.2 Текстовый идентификатор 106
10.2.2.1 Класс публичного текста 107
10.2.2.2 Описание публичного текста 108
10.2.2.3 Язык публичного текста 108
10.2.2.4 Последовательность, обозначающая публичный текст 109
10.2.2.5 Версия отображения публичного текста 109

10.3 Объявление комментария 110
10.4 Объявление отмеченного раздела 110
10.4.1 Количества 111
10.4.2 Спецификация ключевого слова состояния 111
10.5 Объявление объекта 113
10.5.1 Имя объекта 113
10.5.1.1 Количества 113
10.5.1.2 Доступные объемы 113
10.5.2 Текст объекта 114
10.5.3 Текст данных 114
10.5.4 Текст, заключенный в скобки 115
10.5.4.1 Количества 116
10.5.5 Спецификация внешнего объекта 116
11 Объявления разметки: определение типа документа 117
11.1 Объявление типа документа 117
11.2 Объявление элемента 118
11.2.1 Тип элемента 118
11.2.1.1 Ранжированный элемент 119
11.2.1.2 Количества 119
11.2.2 Минимизация пропущенным тегом 119
11.2.3 Объявленное содержание 120
11.2.4 Модель содержания 120
11.2.4.1 Соединитель 122
11.2.4.2 Индикатор появления 122
11.2.4.3 Неоднозначная модель содержания 123
11.2.4.4 Группа тегов данных 123
11.2.4.5 Количества 124
11.2.5 Исключения 124
11.2.5.1 Включения 125

11.2.5.2 Изъятия 125
11.3 Объявление списка определений атрибутов 126
11.3.1 Количества 126
11.3.2 Имя атрибута 127
11.3.3 Объявленное значение 127
11.3.4 Значение по умолчанию 128
11.3.4.1 Количества 129
11.3.4.2 Доступные объемы 129
11.4 Объявление нотации 129
11.5 Объявление отображения краткой ссылки 130
11.6 Объявление использования краткой ссылки 130
11.6.1 Использование в объявлении типа документа 131
11.6.2 Использование в экземпляре документа 131
11.6.3 Текущее отображение 131
12 Объявления разметки: определение процессов связи 132
12.1 Определение типа связи 132
12.1.1 Спецификация простой связи 133
12.1.2 Спецификация неявной связи 133
12.1.3 Спецификация внешней связи 133
12.1.3.1 Предельные значения 134
12.1.4 Подмножество объявления типа связи 134
12.1.4.1 Объекты параметров 134
12.1.4.2 Атрибуты связи 134
12.1.4.3 Простая связь 135
12.2 Объявление набора связей 135
12.2.1 Спецификация элемента – источника 135
12.2.2 Спецификация элемента – результата 136
12.3 Объявление использования набора связей 136
12.3.1 Использование в объявлении типа связи 137

12.3.2 Использование в экземпляре документа 137
12.3.3 Текущий набор связей 137
13 Объявление SGML 138
13.1 Набор символов документа 139
13.1.1 Описание набора символов 140
13.1.1.1 Основной набор символов 140
13.1.1.2 Часть описываемого набора символов 141
13.1.2 Идентификация символов, не относящихся к SGML 142
13.2 Набор доступных объемов 142
13.3 Область конкретного синтаксиса 143
13.4 Конкретный синтаксис 145
13.4.1 Публичный конкретный синтаксис 146
13.4.2 Идентификация номера избегаемого символа 146
13.4.3 Набор символов синтаксиса – ссылок 147
13.4.4 Идентификация символа функции 147
13.4.5 Правила наименования 148
13.4.6 Набор разграничителей 150
13.4.6.1 Общие разграничители 150
13.4.6.2 Разграничители кратких ссылок 150
13.4.7 Использование зарезервированного имени 152
13.4.8 Набор количеств 152
13.5 Использование функции 153
13.5.1 Функции минимизации разметки 153
13.5.2 Функции типа связи 154
13.5.3 Другие функции 155
13.6 Специфичная для приложения информация 155
14 Конкретные синтаксисы ссылок и ядра 156
15 Соответствие 156
15.1 Документ, удовлетворяющий требованиям SGML 156

15.1.1 Основной документ SGML 157
15.1.2 Минимальный документ SGML 157
15.1.3 Иной документ, удовлетворяющий требованиям SGML 157
15.2 Приложение, удовлетворяющее требованиям SGML 157
15.2.1 Условные соглашения приложения 157
15.2.2 Соответствие документов 158
15.2.3 Соответствие документации 158
15.3 Система, удовлетворяющая требованиям SGML 158
15.3.1 Соответствие документации 159
15.3.2 Соответствие объявлению системы 159
15.3.3 Поддержка конкретного синтаксиса ссылок 160
15.3.4 Поддержка набора доступных объемов ссылок 162
15.3.5. Совместимость анализа 162
15.3.6. Условные соглашения приложения 162
15.4. Синтаксический анализатор SGML с проверкой ошибок 163
15.4.1 Распознавание ошибок 163
15.4.2 Идентификация сообщений SGML 164
15.4.3 Содержание сообщений SGML 164
15.5 Требования к документации 164
15.5.1 Стандартная идентификация 165
15.5.2 Идентификация конструкций SGML 165
15.5.3 Терминология 166
15.5.4 Иной конкретный синтаксис 166
15.6 Объявление системы 166
15.6.1 Поддерживаемые конкретные синтаксисы 167
15.6.1.1 Изменения конкретного синтаксиса 168
15.6.1.2 Трансляция набора символов 168
15.6.2 Услуги проверки данных 169
Приложение А 171

A.1. Процесс разметки 171
A.2 Описательная разметка 174
A.3 Строгая разметка 179
A.4 Заключение 184
A.5 Благодарности 186
A.6 Библиография 186
Приложение B 188
B.1 Документы, определения типа документа и процедуры 188
B.1.1 Документы 188
B.1.2 Определения типа документа 189
B.1.3 Процедуры 190
B.2 Разметка 191
B.3 Различение разметки и текста 194
B.3.1 Теги Описательной разметки 194
B.3.2 Другая разметка 195
B.3.3 Границы записи 196
B.3.3.1 Границы записи в данных 196
B.3.3.2 Границы записи в разметке 197
B.4 Структура документа 197
B.4.1 Определения типа документа 198
B.4.2 Объявления элемента 199
B.4.2.1 Модели содержания 199
B.4.2.2 Соединители и индикаторы появления 200
B.4.2.3 Ссылки на объекты в моделях 202
B.4.2.4 Группы имен 202
B.4.2.5 Символы данных 203
B.4.2.6 Пустое содержание 204
B.4.2.7 Данные, не относящиеся к SGML 204
B.4.2.8 Сводка разграничителей модели 205

B.5 Атрибуты 206
B.5.1 Определение атрибутов 206
B.5.1.1 Имена 207
B.5.1.2 Значения атрибута 208
B.5.2 Объявление атрибутов 209
B.5.2.1 Синтаксис определения атрибута 209
B.5.2.2 Комплексные значения атрибутов 211
B.5.2.3 Группы маркеров имени 212
B.5.2.4 Изменение значений по умолчанию 213
B.6 Объекты 213
B.6.1 Синтаксис ссылок на объект 214
B.6.2 Объявление объектов 215
B.6.2.1 Команды Обработки 216
B.6.2.2 Объекты со ссылками на объект 216
B.6.2.3 Внешние объекты 217
B.6.2.4 Публичные объекты 217
B.7 Символы 218
B.7.1 Классификация символов 219
B.7.2 Ссылки на символ 221
B.7.3 Использование символов - разделителей как данных 222
B.8 Отмеченные разделы 224
B.8.1 Игнорирование отмеченного раздела 224
B.8.2 Версии одиночного документа 226
B.8.3 Не анализируемые разделы 227
B.8.4 Временные разделы 228
B.8.5 Спецификации ключевых слов 228
B.8.6 Определение отмеченного раздела как объекта 229
B.9 Атрибуты уникального идентификатора 230
B.10 Атрибуты ссылок на содержание 231

B.11 Исключения в модели содержания 232
B.11.1 Включенные элементы 233
B.11.2 Выключенные элементы 233
B.12 Объявление типа документа 234
B.13 Содержание данных 236
B.13.1 Представления содержания данных 236
B.13.1.1 Символьные данные (PCDATA, CDATA и RCDATA) 236
B.13.1.2 Данные, не относящегося к SGML (NDATA) 238
B.13.2 Нотации содержания данных 239
B.13.2.1 Нотации для символьных данных 239
B.13.2.2 Нотации для данных, не относящихся к SGML 241
B.13.2.3 Определение нотаций содержания данных 242
B.14 Настройка 243
B.14.1 Объявление SGML 243
B.14.1.1 Факультативные функции 243
B.14.1.2 Иной конкретный синтаксис 244
B.14.2 Воздействие настройки 245
B.15 Соответствие 245
Приложение C 245
C 1 Функции минимизации разметки 248
C 1.1 SHORTTAG: Теги с опущенной разметкой 249
C.1.1.1 Незакрытые краткие теги 249
C.1.1.2 Пустые теги 249
C.1.1.3 Минимизация атрибута 250
C.1.2 OMITTAG: теги могут быть опущены 252
C.1.2.1 Концепция пропуска тега 252
C 1.2.2 Специфицирование минимизации 254
C.1.2.3 Пропуск конечного тега: внедрение начального тега 254
C 1.2.4 Пропуск конечного тега: конечный тег содержащего элемента 255

C.1.2.5 Пропуск начального тега: контекстно обязательный элемент ...256
C.1.2.6 Комбинация с минимизацией кратким тегом 256
C 1.2.7 Соображения по минимизации разметки 257
C 1.3 SHORTREF: разграничители краткой ссылки могут заменять полные ссылки на объект.....258
C 1.3.1 Набор на пишущей машинке: обобщенный режим WYSIWYG ..258
C 1.3.2 Пример набора на пишущей машинке: определение отображения краткой ссылки 259
C.1.3.3 Пример набора на пишущей машинке: активизация отображения краткой ссылки 261
C.1.3.4 Пример с таблицами 263
C.1.3.5 Специальные требования 264
C 1.4 DATATAG: данные могут также быть тегом 265
C.1.5.РАНГ: ранги могут опускаться в тегах 270
C.2 Функции СВЯЗИ: ПРОСТАЯ, НЕЯВНАЯ и ЯВНАЯ 271
C 2.1 Определения процесса связи.....272
C 3 Другие функции 273
C 3.1 CONCUR: экземпляры документа могут появляться.....274
одновременно.....274
C 3.2 SUBDOC: могут появляться вложенные объекты поддокументов.....275
C 3.3 FORMAL: публичные идентификаторы формальны.....276
Приложение D.....274
D.1 Наборы элементов 278
D.1.1 Обычные типы элементов.....278
D.1.2 Формальные типы элементов.....279
D.2 Нотации содержания данных 279
D.3 Иные конкретные синтаксисы.....280
D.3.1 Многокодовые конкретные синтаксисы 281

D.4 Наборы объектов 282
D.4.1 Общие соображения 284
D.4.1.1 Формат объявлений 284
D.4.1.2 Соответствующие отображающие наборы объектов 285
D.4.1.3 Имена объектов 286
D.4.1.4 Организация наборов объектов 287
D.4.2 Алфавитные символы 288
D.4.2.1 Латиница 288
D.4.2.2 Символы греческого алфавита 294
D.4.2.3 Символы кириллических алфавитов 297
D.4.3 Общее употребление 300
D.4.3.1 Цифровые и специальные графические символы 300
D.4.3.2 Символы диакритических знаков 302
D.4.3.3 Издательские символы 303
D.4.3.4 Символы рамок и заполнения 305
D.4.4 Техническое употребление 307
D.4.4.1 Общие объекты 308
D.4.4.2 Греческие символы 310
D.4.4.3 Альтернативные греческие символы 311
D.4.5 Дополнительные математические символы 313
D.4.5.1 Одиночные символы 313
D.4.5.2 Двоичные и большие операторы 314
D.4.5.3 Отношения 315
D.4.5.4 Отрицательные отношения 317
D.4.5.5 Стрелочные отношения 319
D.4.5.6 Открывающие и закрывающие разграничители 321
Приложение E 312
E.1 Определение типа документа 322
E.2 Метафайл машинной графики 331

E.3 Аппаратно - независимое расширение кода....332
E.3.1 Средства расширения кода....333
E.3.1.1 Предотвращение ложного распознавания разграничителей....334
E.3.1.2 Устранение аппаратных и кодовых зависимостей....338
Приложение F....326
F.1 Модель синтаксического анализа SGML....340
F.1.1 Физический ввод....341
F.1.1.1 Объекты....341
F.1.1.2 Границы записей....342
F.1.2 Режимы распознавания....342
F.1.3 Минимизация разметки....344
F.1.4 Трансляция....345
F.1.5 Аналогия командного языка....345
F.2 Инициализация....345
F.2.1 Начальное отображение процедуры....346
F.2.2 Спецификация процесса связи....346
F.2.3 Параллельные экземпляры документа....346
F.3 Динамическое отображение процедур....347
F.4 Обработка ошибок....347
Приложение G....335
G.1 Код классификации....349
G.1.1 Код функции....350
G.1.2 Код подтверждения правильности....352
G.1.3 Код синтаксиса....352
G.2 Соображения по сертификации....353
Приложение H....341
H.1 Нотация группы модели....355
H.2 Применение теории автоматов....356
H.3 Расхождение с теорией автоматов....357

Приложение I 345

I.1 Родовые идентификаторы фиксированной длины 359

I.2 Одиночный разграничитель 360

Рисунки

Рис. 1. Классы символов: абстрактный синтаксис 87
Рис. 2. Классы символов: конкретный синтаксис 90
Рис. 4. Набор разграничителей ссылок: краткие ссылки..... 98
Рис. 5. Набор доступных объемов ссылок 145
Рис. 6. Набор количеств ссылки 153
Рис. 7. Конкретный синтаксис ссылок 159
Рис. 8. Типичное объявление SGML для основного документа SGML ..161
Рис. 9. Разметка элемента.....195
Рис. 10. Начальный тег с двумя атрибутами 208
Рис. 11. Многокодовый базовый конкретный синтаксис.....283
Рис. 12. Атрибуты графического метафайла (1 из 2): кодирование и просмотр 334
Рис. 13. Атрибуты графического метафайла (2 из 2): размер и поворот336
Рис. 14. Символы функций для аппаратно - независимых многокодовых конкретных синтаксисов 337
Рис. 15. Классификация соответствия fsv 351

ОБРАБОТКА ИНФОРМАЦИИ. ТЕКСТОВЫЕ И ОФИСНЫЕ СИСТЕМЫ. СТАНДАРТНЫЙ ОБОБЩЕННЫЙ ЯЗЫК РАЗМЕТКИ (SGML)

0 Введение

Настоящий международный стандарт определяет язык для представления документа, именуемый "Стандартный Обобщенный Язык Разметки" (SGML). SGML может использоваться для публикаций в их самом широком определении, в пределах от одиночной публикации на обычном носителе до публикации мультимедийной базы данных. SGML может также использоваться в офисной обработке документа, когда требуются преимущества удобочитаемости для человека и обмена с издательскими системами.

0.1 Предпосылки

Абстрактно документ может рассматриваться как структура элементов различных типов. Автор организовывает книгу в главы, которые содержат, например, параграфы и рисунки, которые содержат названия рисунка. Редактор организовывает журнал в статьи, которые содержат параграфы, которые содержат слова, и так далее.

Процессоры обрабатывают эти элементы другими способами. Программа форматирования может печатать заголовки заметным шрифтом, оставлять место между параграфами и иным образом визуально передавать читателю структуру и другие атрибуты. Информационно-поисковая система возможно присвоит дополнительное значение словам в заголовке при создании своего словаря.

Хотя эта связь между атрибутами документа и его обработкой сейчас кажется очевидной, в ранних методах обработки текстов она была

более скрыта. Во времена до появления автоматизированного ввода редактор "разметил" бы рукопись специальными командами обработки, которые обеспечивали необходимый формат при работе наборщика. Любая связь между этими командами и структурой документа существовала только в голове редактора.

В ранних компьютеризированных системах этот подход был продолжен путем добавления «разметки», зависящей от обработки, к машинно-читаемому файлу документа. Разметка все еще состояла из определенных команд обработки, но теперь они были на языке программы форматирования, а не для человека-наборщика. Файл не мог легко использоваться для иной цели или в иной компьютерной системе без изменения всей разметки.

По мере того, как пользователи становились более искушенными, а текстовые процессоры более мощными, разрабатывались подходы для облегчения этой проблемы. Для идентификации тех мест в документе, где должна была произойти обработка, использовались "макровызовы" (или "вызовы формата"). Фактические команды обработки сохранялись вне документа, в "процедурах" (или "макроопределениях" или "сохраненных форматах"), где они могли изменяться более легким способом.

Хотя макровызовы могли размещаться в любом месте документа, пользователи стали замечать, что большая их часть размещается в начале или конце элементов документа. Поэтому было естественно выбирать такие имена для этих макрокоманд, которые были бы "родовыми идентификаторами" типов элементов, нежели именами, которые предлагали бы конкретный вид обработки (например, "заголовок" вместо "формат-17"), и таким образом возникла практика "родового кодирования" (или "обобщенного тегирования").

Родовое кодирование стало главным шагом на пути к созданию

автоматизированных систем обработки текста, отражающих естественные отношения между атрибутами документа и обработкой. Появление "обобщенных языков разметки" в начале 1970-ых развило эту тенденцию далее, обеспечив ядро формального языка для родового кодирования. В обобщенном языке разметки соблюдаются два основных принципа:

a) Описательная разметка преобладает и отличается от команд обработки .

Описательная разметка включает родовые идентификаторы и другие атрибуты элементов документа, которые мотивируют команды обработки. Команды обработки, которые могут быть на любом языке, обычно собираются вне документа в процедурах.

По мере сканирования исходного файла для разметки распознавания различных элементов система обработки выполняет процедуры, связанные с каждым элементом и атрибутом для того процесса. Для других процессов с теми же самыми элементами и атрибутами могут быть связаны другие процедуры без изменения разметки документа.

Когда команда обработки должна быть введена непосредственно в документе, она выделяется отлично от описательной разметки таким образом, чтобы ее можно было легко обнаружить и изменить для различных процессов.

b) Разметка формально определена для каждого типа документа.

Обобщенный язык разметки формализует разметку документа, включая "определения типов документа". Определения типов включают спецификацию (подобно формальной грамматике), определяющую, какие элементы и атрибуты могут появляться в документе и в каком порядке. С этой информацией возможно определить, является ли разметка для индивидуального документа

правильной (т. е., удовлетворяет определению типа), а также обеспечить отсутствующую разметку, поскольку она может быть однозначно выведена из другой разметки, которая присутствует.

ПРИМЕЧАНИЕ. Более подробное введение в концепцию родового кодирования и в Стандартный Обобщенный Язык Разметки можно найти в Приложении А.

0.2 Цели

Стандартный Обобщенный Язык Разметки стандартизирует приложение родового кодирования и концепций обобщенной разметки. Он обеспечивает логически последовательный и однозначный синтаксис для описания всего, что пользователь хочет идентифицировать в пределах документа. Язык включает:

— "абстрактный синтаксис" для описательной разметки элементов документа.

— "Конкретный синтаксис ссылки", который связывает абстрактный синтаксис со символами и количествами конкретных разделителей. Пользователи могут определять альтернативные конкретные синтаксиса для выполнения своих требований.

— Объявления разметки, которые позволяют пользователю определять специальный словарь родовых идентификаторов и атрибутов для различных типов документа.

— Условие для содержания произвольных данных. В обобщенной разметке, "данные" - это все, что не определено языком разметки. Они могут включать специализированные "нотации содержания данных" которые требуют интерпретации, отличной от общего текста: формулы, изображения, нелатинские

алфавиты, предварительно отформатированный текст или графика.

— ссылки на объекты: не зависящий от системы метод для ссылки на содержание, расположенное вне основного материала документа, например отдельно написанные главы, символы команд обработки, фотографии и т.д.

— Специальные разделители для команд обработки, чтобы отличить их от описательной разметки. Команды обработки могут вводиться при необходимости для ситуаций, которые не могут быть обработаны процедурами, но они могут быть легко найдены и изменены позже, когда документ передан в другую систему обработки.

Однако, чтобы обобщенный язык разметки мог быть приемлемым стандартом, требуется больше, чем только обеспечение требуемых функциональных возможностей. Язык должен иметь металингвистические свойства, чтобы выполнить ограничения, наложенные потребностью использовать его в множественных средах. Главные ограничения, и средства, которыми Стандартный Обобщенный Язык Разметки выполняет их, могут быть кратко сформулированы следующим образом:

а) Документы, "размеченные" с помощью языка, должны иметь возможность обрабатываться широким диапазоном систем обработки текстов.

Полная форма языка, со всеми факультативными функциями, предлагает общность и гибкость, которая может использоваться сложными системами; менее мощные системы не должны поддерживать функции. Для облегчения обмена между несходными системами "объявление SGML" описывает любые функции разметки или изменения конкретного синтаксиса, используемые в документе.

b) Должны поддерживаться миллионы существующих устройств ввода текста.

Документы SGML с конкретным синтаксисом бетона ссылок могут легко набираться и пониматься людьми без машинной помощи. В результате:

— Использование SGML не должно ожидать развития и принятия нового поколения аппаратных средств — только программного обеспечения для обработки документов на существующих машинах.

— Переход к такому новому поколению (когда оно появится) будет проще, поскольку пользователи уже будут знакомы с SGML.

c) Не должно существовать зависимости от набора символов, поскольку документы могут вводиться с множества устройств.

Язык не имеет зависимости от конкретного набора символов. Приемлем любой набор символов, который имеет разрядные комбинации для символов, цифр, пробела и разделителей.

d) Не должно быть зависимости от никакой обработки, системы или устройства.

Обобщенная разметка является преобладающим образом описательной и поэтому внутренне свободна от таких зависимостей. Случайная команда обработки специально выделяется, поэтому ее можно найти и конвертировать для обмена, либо когда другой процесс преобразовывает команду в неприменимую.

Ссылки на внешние части документа являются косвенными. Отображения на память реальной системны сделаны в виде "объявлений внешних объектов", которые появляются в начале документа, где они могут быть легко быть изменены для обмена.

Конкретный синтаксис может быть изменен в объявлениях SGML,

чтобы включить любые зарезервированные системные символы.

e) Не должно быть отклонений в отношении национальных языков.

Символы, используемые для имен, могут быть расширены любыми специальными национальными символами. Родовые идентификаторы, имена атрибутов и другие имена, используемые в описательной разметке, определяются пользователем в объявлениях элементов и объектов.

Имена объявлений и ключевые слова, используемые в объявлениях разметки, также могут изменяться.

Поддерживаются множественные символьные алфавиты, используемые в многоязыковых документах.

f) Язык должен допускать известные соглашения пишущей машинки и текстового процессора.

Возможности "краткой ссылки" и "тега данных" поддерживают соглашения ввода текста на пишущей машинке. Нормальный текст, содержащий параграфы и цитаты, может быть интерпретирован как SGML, хотя он набран без видимой разметки.

g) Язык не должен зависеть от специфического потока данных или физической организации файла.

Язык разметки имеет модель виртуальной памяти, в которой документы состоят из одного или большего числа объектов памяти, каждый из которых является последовательностью символов. Весь реальный доступ к файлу обрабатывается системой обработки, которая может решать, должна ли последовательность символов рассматриваться как непрерывная, или она должна отражать границы физической записи.

h) "Размеченный" текст должен сосуществовать с другими данными.

Система обработки может допускать появление в потоке данных с

другим материалом текста, соответствующего этому международному стандарту, до тех пор пока система может определить начало и конец этого текста.

Точно так же система может допустить логическое появление содержания данных, не определенного SGML, в пределах соответствующего документа. Появление таких данных обозначается объявлениями разметки для облегчения обмена.

i) Разметка должна быть пригодна для использования и людьми и программами.

Стандартный Обобщенный Язык Разметки предназначен для использования в качестве подходящего интерфейса для набора текста и обмена без препроцессоров. Это допускает широкое приспособление для учета предпочтений пользователя в соглашениях по вводу текста и требований множества клавиатур и дисплеев.

Однако следует признать, что многие разработчики захотят воспользоваться преимуществами возможностей языка по сбору информации для обеспечения интеллектуального редактирования или создания документов SGML из входной среды системы обработки текстов. SGML учитывает такие использования, обеспечивая следующие возможности:

— Содержание элемента может храниться отдельно от разметки.

— Управляющие символы могут использоваться в качестве разделителей.

— В документе допускаются смешанные режимы представления данных.

— Поддерживаются множественные параллельные логические и форматные структуры.

0.3 Организация

Этот международный стандарт организован следующим образом:

a) Физическая организация документа SGML как структуры объекта определена в 6.

b) Логическая организация документа SGML как структуры элемента, и его представление с помощью описательной разметки определено в 7.

c) Команды обработки описаны в 8.

d) Общие конструкции разметки, такие как символы, ссылки на объекты и команды обработки рассмотрены в 9.

e) Объявления разметки с общей применимостью (комментарий, объект и отмеченный раздел) определены в предложении 10.

f) Объявления разметки, которые используются преимущественно для создания определений типа документа (тип документа, элемент, нотация, отображение краткой ссылки и использование краткой ссылки), определены в 11.

g) Объявления разметки, которые используются преимущественно для создания определений процесса связи (тип связи, атрибут связи, множество связей и использование множества связей), определены в 12.

h) Объявление SGML, которое определяет набор символов документа, набор возможностей, конкретный синтаксис и функции, определено в 13.

i) Конкретный синтаксис ссылки определен в 14.

j) Соответствие документов, приложений и систем определено в 15.

Имеется также несколько приложений, содержащих дополнительную информацию; они не являются неотъемлемой частью основного документа настоящего международного стандарта.

ПРИМЕЧАНИЕ. Настоящий международный стандарт является формальной спецификацией машинного языка, которая может оказаться трудным чтением для тех, чей опыт сосредоточен более в области создания документов нежели компиляторов. В Приложениях A. B, и C в неформальном учебном стиле обсуждаются основные концепции, которые должны быть более доступны для большинства читателей. Однако, читатель должен знать, что эти приложения не охватывают ни все конструкции SGML, ни все подробности тех конструкций, которые рассмотрены, и тонкие различия часто игнорируются в целях представления ясного краткого обзора.

1 Область распространения

Настоящий международный стандарт:

a) Определяет абстрактный синтаксис, известный как Стандартный Обобщенный Язык Разметки (SGML). Язык выражает описание структуры документа и других атрибутов, также как и другой информации, которая делает разметку интерпретируемой.

b) Определяет конкретный синтаксис ссылок, который связывает абстрактный синтаксис с определенными символами и числовыми значениями, и критерии для определения различных конкретных синтаксисов.

c) Определяет документы, удовлетворяющие стандарту, в терминах использования ими компонентов языка.

d) Определяет системы, удовлетворяющие стандарту, в терминах их способности обработки документов, удовлетворяющих стандарту, и распознавания в них ошибок разметки.

e) Определяет, каким образом данные, не определенные этим международным стандартом (такие как изображения, графика или

форматированный текст) могут быть включены в документ, удовлетворяющие стандарту.

ПРИМЕЧАНИЕ. Данный международный стандарт не:

a) идентифицирует или определяет "стандартные" типы документов, архитектуры документов или текстовые структуры;

b) определяет выполнение, архитектуру или обработку ошибок разметки систем, удовлетворяющие стандарту;

c) определяет, каким образом должны создаваться документы, удовлетворяющие стандарту;

d) определяет поток данных, систему обработки сообщений, структуру файла или другое физическое представление, в котором производится хранение или обмен документов, удовлетворяющих стандарту, либо какой-либо набор символов или схему кодирования, в которые или из которых могли бы для таких целей транслироваться документы, удовлетворяющие стандарту;

e) определяет представление содержания данных или нотацию для изображений, графики, форматированного текста и т.д., которые включены в документ, удовлетворяющий стандарту;

2 Область применения

Стандартный Обобщенный Язык Разметки может использоваться для документов, которые обработаны любой системой обработки текстов. Он, в частности, применим к:

a) документам, которыми обмениваются системы с различными языками обработки текста;

b) документам, которые обработаны больше чем одним способом, даже когда процедуры используют один и тот же язык обработки текста;

Документы, которые существуют исключительно в окончательном форматированном виде, не относятся к области применения настоящего международного стандарта.

3 Нормативные ссылки

ИСО 639 *Коды для представления названий языков*[1)]

ИСО 646 *Обработка информации. Набор с 7-бтовых кодирующих символов для обмена информацией*

ИСО 9069 *Обработка информации. Средства поддержки SGML. Формат обмена документами SGML (SDIF)*[1)]

ИСО 9070 *Обработка информации. Средства поддержки SGML. Процедуры регистрации для общедоступного текста*[1)]

Приведенные ниже ссылки используются в связи с иллюстративными материалами:

ИСО 2022 *Обработка информации. Наборы с 7-битовых и 8-битовых кодирующих символов ИСО. Методы расширения кода*

ИСО 3166 *Коды для представления названий стран*

ИСО 4873 *Обработка информации. 8-битовый код ИСО для обмена информацией. Структура и правила реализации*

ИСО 6937 *Обработка информации. Кодированные наборы символов для обмена текстами*

ИСО 8632/2 *Системы обработки информации. Компьютерная графика. Метафайл для хранения и передачи описательной информации изображений. Часть 2: Кодирование символов*[1)]

ИСО 8632/4 *Системы обработки информации. Компьютерная графика. Метафайл для хранения и передачи описательной информации*

[1] В настоящее время существует в форме проекта.

изображений. Часть 4: Простое кодирование текста[1)]

4 Определения

ПРИМЕЧАНИЕ. В этом разделе используются типографские соглашения, описанные в 5.1.

В настоящем международном стандарте используются следующие определения:

4.1 Абстрактный синтаксис (SGML): правила, которые определяют, каким образом разметка добавляется к данным документа, безотносительно специальных символов, используемых для представления разметки.

4.2 Тип активного документа (объявление): объявление типа документа, в отношении которого анализируется объект SGML .

4.3 Активная связь (объявление): объявление типа связи, в отношении которого анализируется объект SGML.

4.4 Неоднозначная модель содержания: *модель содержания*, в которой элемент или строка символов, встречающаяся в экземпляре документа, может удовлетворять более чем одному *примитивному маркеру содержания* без анализа правого контекста.

ПРИМЕЧАНИЕ. Неоднозначные модели содержания запрещены в SGML.

4.5 Приложение: приложение обработки текста.

4.6 Условное соглашение приложения: Специфическое для приложения правило, управляющее текстом документа в областях, которые SGML оставляет на выбор пользователя.

ПРИМЕЧАНИЕ. Имеются два вида условных соглашений: условные соглашения содержания и условные соглашения разметки.

4.7 Специфичная для приложения информация: параметр *объявления SGML*, который определяет информацию, требуемую приложением и/или его архитектурой.

ПРИМЕЧАНИЕ. Например, информация могла бы идентифицировать архитектуру и/или приложение, или иным образом давать возможность системе определить, может ли она обрабатывать документ.

4.8 Тип связанного элемента: тип элемента, связанного с предметом объявления разметки его параметром *типа связанного элемента*.

4.9 Атрибут (элемента): характерное качество, иное нежели тип или содержание.

4.10 Определение атрибута: элемент списка определений атрибутов; он определяет имя атрибута, допустимые значения и значение по умолчанию.

4.11 Список определений атрибутов: набор одного или большего числа определений атрибута, определенный параметром *списка определений атрибутов* объявления списка определений атрибутов.

4.12 Объявление списка (определений) атрибутов: объявление разметки, которое сопоставляет список определений атрибутов с одним или большим числом типов элементов.

4.13 Список атрибутов: список спецификаций атрибутов.

4.14 Объявление списка атрибутов: объявление списка определений атрибутов.

4.15 Спецификация атрибута: элемент списка спецификаций атрибутов;

он определяет значение отдельного атрибута.

4.16 Список (спецификаций) атрибутов: разметка, которая является набором одной или большего числа спецификаций атрибутов.

ПРИМЕЧАНИЕ. Списки спецификаций атрибутов встречаются в начальных тегах и наборах связей.

4.17 Литерал значения атрибута: разграниченная строка символов, которая интерпретируется как *значение атрибута* путем замены ссылок и игнорирования или трансляции символов функций.

4.18 Доступный публичный текст: публичный текст, который является доступным широкой публике, хотя его владелец может требовать оплаты или выполнения других условий.

4.19 B-последовательность: непрерывная последовательность символов прописной буквы "B"; в строке, назначенной краткой ссылкой, она обозначает последовательность пропусков, чья минимальная длина - длина B-последовательности.

4.20 Элемент базового документа: элемент документа, который является базовым документом.

4.21 Тип базового документа: тип документа, указанный первым объявлением типа документа в прологе.

4.22 Базовый документ SGML: удовлетворяющий требованиям SGML документ, который использует конкретный синтаксис ссылок и набор доступных объемов и функции минимизации разметки SHORTTAG и OMITTAG .

ПРИМЕЧАНИЕ. Он также использует функцию SHORTREF в силу использования конкретного синтаксиса ссылок.

4.23 Бит: двоичная цифра; т. е. нуль или один.

4.24 Битовая комбинация: упорядоченный набор битов, интерпретируемый как двоичный номер.

4.25 Последовательность пропусков: непрерывная последовательность символов ***SPACE*** и/или ***SEPCHAR***.

4.26 Доступный объем: поименованный предел некоторого аспекта размера или сложности документа, выраженного как число пунктов, которые могут быть накоплены для вида объекта или для всех объектов.

ПРИМЕЧАНИЕ. Набор доступных объемов определяется абстрактным синтаксисом, но значения присваиваются индивидуальными документами и системами SGML.

4.27 Набор доступных объемов: набор присвоений числовых значений именам доступных объемов.

ПРИМЕЧАНИЕ. В объявлении SGML набор доступных объемов идентифицирует максимальные требования к доступным объемам документа (его фактические требования могут быть более низкими). Набор доступных объемов может также быть определен приложением для ограничения требований к доступным объемам документов, которые должны обработать реализации приложения, или системой для определения требований к доступным объемам, которые она может удовлетворить.

4.28 CDATA: символьные данные.

4.29 Объект CDATA: объект символьных данных.

4.30 Цепочка процессов (связи): процессы, выполняемые последовательно, которые формируют цепочку, в которой источник

первого процесса является образцом типа базового документа, а результат каждого процесса кроме последнего является источником для следующего. Любая часть цепочки может выполняться с помощью итераций.

ПРИМЕЧАНИЕ. Например, приложение комплексного форматирования страниц может включать три типа документа — логический, гранку и страницу - и два процесса связи - выравнивание и расчет набора. Процесс выравнивания создает образец гранки из образца логического документа, а процесс расчета набора в свою очередь создаст страницы из гранок. Два процесса могут выполняться с помощью итераций, поскольку решения, сделанные в ходе расчета набора могут потребовать повторного выравнивания гранок в иных размерах.

4.31 Символ: минимальная часть информации с индивидуальным значением, определяемая символьным набором.

ПРИМЕЧАНИЯ:

1. Имеются два вида символов: графический символ и управляющий символ.
2. Символ может возникать в контексте, в котором он имеет значение, определяемое разметкой или нотацией содержания данных, и которое имеет преимущество или дополняет его значение в символьном наборе.

4.32 Класс (символов): набор символов, которые имеют общее назначение в абстрактном синтаксисе, такие как символы, не относящиеся к SGML, или разделители.

ПРИМЕЧАНИЕ. Конкретные символы относятся к символьным классам четырьмя различными способами:

a) явно, абстрактным синтаксисом (***Special***, ***Digit***, ***LC Letter*** и ***UC Letter***);

b) явно, конкретным синтаксисом (***LCNMSTRT***, ***FUNCHAR***, ***SEPCHAR*** и т.д.);

c) неявно, в результате явных назначений, присвоенных ролям разграничителей или другим символьным классам (***DELMCHAR*** и ***DATACHAR***); или

d) явно, набором символов документа (***NONSGML***).

4.33 Символьные данные: нуль или большее число символов, которые возникают в контексте, в котором не опознана разметка, не являющиеся разграничителями, которые заканчивают *символьные данные*. Такие символы классифицируются как символы данных, поскольку они были таковыми объявлены.

4.34 Объект символьных данных: объект, чей текст рассматривается как *символьные данные* при ссылке и не зависит от конкретной системы, устройства или прикладного процесса.

4.35 Набор символьных объектов: набор публичных объектов, состоящий из общих объектов, которые являются графическими символами.

ПРИМЕЧАНИЯ:

1. Символьные объекты используются для символов, которые не имеют никакого кодированного представления в наборе символов документа, или не могут быть легко введены, или для достижения независимости от устройства для тех символов, чьи битовые комбинации не вызывают надлежащего отображения на всех

устройствах вывода.

2. Имеются два вида наборов символьных объектов: определительные и отображающие.

4.36 Номер символа: *номер*, который представляет десятичный целый эквивалент кодированного представления символа, полученный в результате представления последовательности битовых комбинаций как отдельных целых двоичных чисел.

4.37 Ссылка на символ: ссылка, которая заменяется одиночным символом.

ПРИМЕЧАНИЕ. Имеются два вида ссылок: поименованная ссылка на символ и цифровая ссылка на символ.

4.38 Символьный алфавит: набор символов, которые используются вместе. Значения определены для каждого символа, и могут также быть определены для управляющих последовательностей множественных символов.

ПРИМЕЧАНИЕ. Когда символ возникает в управляющей последовательности, значение последовательности имеет преимущество перед значением индивидуальных символов.

4.39 Набор символов: отображение символьного алфавита на кодовый набор таким образом, что каждому символу сопоставлено его кодированное представление.

4.40 (Символьная) строка: последовательность символов.

4.41 Класс: класс символов.

4.42 Расширение кода: использование отдельного кодированного

представления для больше чем одного символа без изменения набора символов документа.

ПРИМЕЧАНИЕ. Когда в документе имеется несколько национальных языков, может быть полезно расширение кода графического алфавита .

4.43 Кодовый набор: набор битовых комбинаций равного размера, упорядоченных по их числовым значениям, которые должны быть последовательны.

ПРИМЕЧАНИЕ. Например, кодовый набор, чьи битовые комбинации имеют 8 битов ("8-разрядный код") может состоять из 256 битовых комбинаций, в диапазоне от 00000000 до 11111111 (0-255 в десятичном исчислении), либо он может состоять из любого непрерывного подмножества этих битовых комбинаций.

4.44 Позиция кодового набора: числовое значение битовой комбинации в кодовом наборе.

4.45 Кодированное представление: представление символа как последовательность одного или большего числа битовых комбинаций равного размера.

4.46 Комментарий: часть объявления разметки, которая содержит объяснения или замечания, предназначенные для помощи людям, работающим с документом.

4.47 Объявление комментария: объявление разметки, которое содержит только комментарии.

4.48 Конкретный синтаксис (SGML): привязка абстрактного синтаксиса к специфическим символам - разграничителям, количествам,

именам объявлений разметки и т.д.

4.49 Параметр конкретного синтаксиса: параметр объявления SGML, который идентифицирует конкретный синтаксис, используемый в элементах документа и (обычно) прологах.

ПРИМЕЧАНИЕ. Параметр состоит из параметров, которые идентифицируют набор символов синтаксиса - ссылок, символы функций, избегаемые символы, правила наименования, использование разграничителей, использование зарезервированных имен и количественные характеристики.

4.50 Приложение, удовлетворяющее требованиям SGML: приложение SGML, которое требует, чтобы документы удовлетворяли требованиям SGML, и чья документация выполняет требования этого международного стандарта.

4.51 Документ, удовлетворяющий требованиям SGML: документ SGML, который выполняет все условия этого международного стандарта.

4.52 Содержащий элемент: элемент, в пределах которого имеется подэлемент.

4.53 Содержание: символы, которые имеются между начальным тегом и конечным тегом элемента в экземпляре документа. Они могут интерпретироваться как данные, надлежащие подэлементы, включенные подэлементы, другая разметка или их сочетание.

ПРИМЕЧАНИЕ. Если элемент имеет явную ссылку на содержание, или его объявленное содержание "EMPTY", содержание пусто. В таких случаях само приложение может генерировать данные и обрабатывать их так, как если бы это были данные содержания.

4.54 Условное соглашение содержания: условное соглашение приложения, управляющее содержанием данных, такое как ограничение на длину, допустимые символы или использование символов верхнего и нижнего регистра.

ПРИМЕЧАНИЕ. Условное соглашение содержания - по существу неофициальная нотация содержания данных, обычно ограничиваемая одиночным типом элемента.

4.55 Модель (содержания): параметр объявления элемента, который указывает *группу модели* и *исключения*, которые определяют допустимое содержание элемента.

4.56 Уровень вложенности модели содержания: максимальное число последовательных разграничителей ***grpo*** или ***dtgo***, которые возникают в *модели содержания* без соответствующих разграничителей ***grpc*** или ***dtgc***.

4.57 Ссылка на содержание (атрибут): возможный атрибут, на чье значение ссылается приложение для генерирования данных содержания.

ПРИМЕЧАНИЕ. Когда элемент имеет явную ссылку на содержание, содержание элемента в образце документа пусто.

4.58 Контекстная последовательность: последовательность одного или большего числа символов разметки, которые должны следовать за строкой разграничителей в пределах того же объекта для распознавания строки в качестве разграничителя.

4.59 Контекстно факультативный элемент: элемент

a) который может возникать только потому, что он является включением; или

b) чей *маркер содержания* в текущей применимой группе модели

является контекстно факультативным маркером.

4.60 Контекстно факультативный маркер: *маркер содержания*, который

a) является неотъемлемо факультативным маркером; или

b) имеет индикатор появления ***plus*** и был удовлетворен; или

c) находится в группе модели, которая сама является контекстно факультативным маркером, никакие маркеры которой не были удовлетворены.

4.61 Контекстно обязательный элемент: элемент, который не является контекстно факультативным элементом и

a) чей *родовой идентификатор* является *именем типа документа*; или

b) чей текущий применимый маркер модели является контекстно требуемым маркером.

ПРИМЕЧАНИЕ. Элемент может не быть ни контекстно требуемым, ни контекстно факультативным: например, элемент, чей текущий применимый маркер модели находится в группа or, которая не имеет неотъемлемых факультативных маркеров.

4.62 Контекстно обязательный маркер: *маркер содержания*, который

a) является единственным в своей группе модели; или

b) находится в группе ***seq***

i) которая

— сама является контекстно требуемым маркером;

или

— содержит маркер, который был удовлетворен;

и

ii) все предшествующие маркеры которой

— были удовлетворены; или

— контекстно факультативны.

4.63 Управляющий символ: символ, который управляет интерпретацией, представлением или другой обработкой символов, которые следуют за ним; например, символ табуляции.

4.64 Управляющая последовательность: последовательность символов, начинающаяся с управляющего символа, который управляет интерпретацией, представлением или другой обработкой символов, которые следуют за ним; например, escape-последовательность.

4.65 Конкретный синтаксис ядра: вариант конкретного синтаксиса ссылок, в котором нет разграничителей краткой ссылки.

4.66 Соответствующее содержание (маркера содержания): элемент(ы) и/или данные в экземпляре документа, которые соответствуют *маркеру содержания*.

4.67 Текущий атрибут: атрибут, чье текущее (т.е. последнее определенное) значение является его значением по умолчанию.

ПРИМЕЧАНИЕ. При первом появлении элемента с текущим атрибутом начальный тег не может быть пропущен .

4.68 Текущий элемент: открытый элемент, чей *начальный тег* появился последним (или был пропущен посредством минимизации разметки).

4.69 Текущий набор связей: набор связей, сопоставленный текущему элементу *объявлением использования набора связей* в содержании элемента или определением типа связи. Если текущий элемент не имеет сопоставленного ему набора связей, предыдущий текущий набор связей продолжает оставаться текущим набором связей .

4.70 Текущее отображение: отображение краткой ссылки, связанное с текущим элементом объявлением *использования краткой ссылки* в

элементе содержания или определении типа документа. Если текущий элемент не имеет сопоставленного отображения, предыдущее текущее отображение продолжает оставаться текущим отображением.

4.71 Текущий ранг: номер, который добавляется к базе ранга в теге, чтобы получить родовой идентификатор. Для *начального тега* это *суффикс ранга* последнего элемента с идентичной *базой ранга* или *база ранга* в той же *ранжированной группе*. Для *конечного тега* это *суффикс ранга* последнего открытого элемента с идентичной *базой ранга*.

4.72 Данные: символы документа, которые представляют собственно информационное содержание; символы, которые не распознаются как разметка.

4.73 Символ данных: *символ SGML*, который интерпретируется как данные в контексте, в котором он появляется, либо поскольку он был объявлен как данные, либо поскольку он не был распознан как разметка.

4.74 Содержание данных: часть *содержания* элемента, которое является скорее данными, нежели разметкой или подэлементом.

4.75 Нотация содержания данных: специфическая для приложения интерпретация содержания данных элемента, или информационного объекта, не относящегося к SGML, которая обычно расширяет или отличается от нормального значения набора символов документа.

ПРИМЕЧАНИЕ. Она определяется для содержания элемента атрибутом нотации, а для объекта данных, не относящегося к SGML, параметром имени нотации объявления объекта.

4.76 Тег данных: строка, которая соответствует шаблону тега данных открытого элемента. Он служит и как *конечный тег* открытого элемента, и как *символьные данные* в элементе, который его содержит.

4.77 Группа тегов данных: маркер группы модели, который связывает

шаблон тега данных с целевым элементом.

ПРИМЕЧАНИЕ. В образце целевого элемента содержание данных и содержание любых подэлементов просматривается в поиске строки, которая соответствует шаблону ("тег данных").

4.78 Шаблон тега данных: маркер группы тегов данных, определяющий строки, которые, появляясь в надлежащем контексте, составляли бы тег данных.

4.79 Объявление: объявление разметки.

4.80 Подмножество объявления: разграниченная часть объявления разметки, в которой могут появляться другие объявления .

ПРИМЕЧАНИЕ. Подмножества объявления возникают только в объявлениях типа документа, типа связи и отмеченного раздела.

4.81 Объявленный конкретный синтаксис: конкретный синтаксис, описанный параметром *конкретного синтаксиса объявления SGML*.

4.82 Специализированные символы данных: класс символов, состоящий из каждого *символа SGML*, который не имеет никакого возможного значения как разметка; элемент никогда не рассматривается как что-либо, кроме как *символ данных*.

4.83 Объект по умолчанию: объект, на который ссылаются общей ссылкой на объект с необъявленным именем.

4.84 Значение по умолчанию: часть определения атрибута, которая определяет значение атрибута, используемое в случае отсутствия для него *спецификации атрибута* .

4.85 Определительный набор (символьных) объектов: набор символьных объектов, чье назначение состоит в том, чтобы определить

имена объектов для графических символов, но не отобразить их фактически. Его *публичный идентификатор* не включает *версию отображения публичного текста*.

ПРИМЕЧАНИЕ. В ходе обработки система заменяет определительный набор объектов соответствующим набором объектов шрифтов отображения для соответствующего устройства вывода.

4.86 Символы - разграничители: класс символов, который состоит из каждого *символа SGML*, не являющегося *символом имени* или *символом функции*, и который появляется в строке, назначенной на роль разграничителя конкретным синтаксисом.

4.87 Контекстный разграничитель: строка символов, которая состоит из строки разграничителей, непосредственно за которой в том же объекте следует контекстная последовательность.

4.88 Роль разграничителя: роль, определенная абстрактным синтаксисом, и заполняемая строкой символов, назначенной конкретным синтаксисом, которая включает идентификацию частей разметки и/или отличие разметки от данных.

4.89 Набор разграничителей: набор назначений строк разграничителей ролям разграничителей абстрактного синтаксиса.

4.90 Параметр набора разграничителей: параметр объявления SGML, который идентифицирует набор разграничителей, используемый в объявленном конкретном синтаксисе.

4.91 Разграничитель (строка): строка символов, назначенная на роль разграничителя конкретным синтаксисом.

4.92 Описательная разметка: Разметка, которая описывает структуру и другие атрибуты документа не системно специфическим способом, независимо от какой-либо обработки, которой он может подвергнуться.

В частности, она использует теги для выражения структуры элемента.

4.93 Аппаратно-зависимая версия (публичного текста): публичный текст, чей *формальный публичный идентификатор* отличается от идентификатора другого публичного текста только добавлением *версии отображения публичного текста*, которая идентифицирует поддерживаемые устройства отображения или используемую схему программирования.

4.94 Цифры: класс символов, состоящий из 10 арабских цифр от "0" до "9".

4.95 Отображающий набор (символьных) объектов: набор объектов с теми же именами объектов, что и у соответствующего определительного набора символьных объектов, но который вызывает отображение символов; он является аппаратно-зависимой версией соответствующего определительного набора объектов.

4.96 Документ: собрание информации, которая обрабатывается как единый модуль. Документ классифицируется по конкретным типам.

ПРИМЕЧАНИЕ. В этом международном стандарте термин практически неизменно означает (без потери точности) документ SGML.

4.97 Архитектура документа: правила для формулировки приложений обработки текста.

ПРИМЕЧАНИЕ. Например, архитектура документа может определять:

a) семантику атрибута для использования в ряде определений элементов;

b) классы элементов, на основании которых элементы имеют атрибуты;

c) структурные правила для определения типов документа в

терминах классов элементов;

d) процессы связи, и то, как на них воздействуют значения атрибутов; и/или

e) информацию, сопровождающую документ в процессе обмена ("профиль документа").

4.98 Набор символов документа: набор символов, используемый для всей разметки в документе SGML, а первоначально (по крайней мере) для данных.

ПРИМЕЧАНИЕ. Когда происходит обмен документом между системами, его набор символов транслируется в набор символов получающей системы.

4.99 Элемент документа: элемент, который является наиболее удаленным элементом экземпляра типа документа; то есть элемент, чьим *родовым идентификатором* является *имя типа документа*.

4.100 Экземпляр документа: экземпляр типа документа.

4.101 Набор экземпляров документа: часть *объекта документа SGML* или *объекта поддокумента SGML* в структуре объекта, которая содержит один или большее число экземпляров типов документа. Он взаимно расширяет элемент базового документа в структуре элемента.

ПРИМЕЧАНИЕ. Когда используется функция параллельных экземпляров, в документе могут существовать множественные экземпляры, при этом данные и разметка могут быть для них общими.

4.102 Тип документа: класс документов, имеющих подобные характеристики; например, журнал, статья, техническое руководство или записка.

4.103 Объявление типа (документа): объявление разметки, которое содержит формальную спецификацию определения типа документа.

4.104 Подмножество объявлений типа документа: наборы элементов, объектов и кратких ссылок, встречающиеся в пределах подмножества объявления типа документа.

ПРИМЕЧАНИЕ. Внешний объект, на который имеется ссылка в объявлении типа документа, рассматривается как часть подмножества объявлений.

4.105 Определение (типа) документа: правила, определенные приложением, которые применяют SGML к разметке документов определенного типа. Определение типа документа включает формальную спецификацию, выраженную в объявлении типа документа, типов элемента, отношений элемента и атрибутов, и ссылки, которые могут быть представлены разметкой. Таким образом, она определяет словарь разметки, для которой SGML определяет синтаксис.

ПРИМЕЧАНИЕ. Определение типа документа может также включать комментарии, которые описывают семантику элементов и атрибутов, и любых условных соглашений приложения.

4.106 Спецификация типа документа: часть тега или ссылки на объект, которая идентифицирует образцы типа документа, в пределах которых будет обрабатываться тег или ссылка на объект.

4.107 ds (разделитель): разделитель объявлений, встречающийся в подмножествах объявлений.

4.108 DTD: определение типа документа.

4.109 Эффективное состояние (отмеченного раздела): ключевое слово

состояния с наивысшим приоритетом, указанное в объявлении отмеченного раздела.

4.110 Элемент: компонент иерархической структуры, определяемый в соответствии с определением типа документа; он идентифицируется в экземпляре документа описательной разметкой, обычно начальным и конечным тегами.

ПРИМЕЧАНИЕ. Элемент классифицируется по конкретному типу.

4.111 Объявление элемента: объявление разметки, которое содержит формальную спецификацию части определения типа элемента, которая рассматривает содержание и минимизацию разметки.

4.112 Набор элементов: набор объявлений элементов, которые используются вместе.

ПРИМЕЧАНИЕ. Набор элементов может быть публичным текстом.

4.113 Структура элемента: организация документа в иерархии элементов, каждая из которых соответствует различному определению типа документа.

4.114 Тип элемента: класс элементов, имеющих подобные характеристики; например, параграф, глава, резюме, сноска или библиография.

4.115 Определение (типа) элемента: специфические для приложения правила, которые применяют SGML к разметке элементов конкретного типа. Определение типа элемента включает формальную спецификацию, выраженную в объявлениях списка определений элементов и атрибутов, содержания, минимизации разметки, и атрибутов, допустимых для указанного типа элемента.

ПРИМЕЧАНИЕ. Определение типа элемента обычно составляет часть определения типа документа.

4.116 Параметр типа элемента: параметр объявления элемента, который идентифицирует тип элемента, к которому применяется определение .

ПРИМЕЧАНИЕ. Спецификация может быть прямая, в форме индивидуального родового идентификатора или элемента группы имен, или косвенная, через ранжированный элемент или элемент ранжированной группы.

4.117 Пустой набор связей: набор связей, в котором все элементы - результаты являются неявными, и никакие атрибуты не определены.

4.118 Пустое отображение: отображение краткой ссылки, в котором все разграничители ни на что не отображаются.

ПРИМЕЧАНИЕ. Пустое отображение не должно (и не может) быть объявлено явно, но на него можно ссылаться по его зарезервированному имени "#EMPTY" в конкретном синтаксисе ссылок.

4.119 Конечный тег: описательная разметка, которая идентифицирует конец элемента.

4.120 Объект: группа символов, на которые можно ссылаться как на единый модуль.

ПРИМЕЧАНИЯ:

1. Такие объекты, как книжные главы, написанные различными авторами, символы команд обработки или фотографии, часто наилучшим образом управляются как индивидуальные объекты.

2. Физическая организация объектов определяется системой и может принимать форму файлов, элементов разбитого на разделы набора данных, компонентов структуры данных или записей в таблице символов.

4.121 Объявление объекта: объявление разметки, которое присваивает объекту имя SGML, чтобы на него можно было сослаться.

4.122 Конец (сигнал) объекта: сигнал от системы об окончании текста замены объекта.

4.123 Менеджер объектов: программа (или часть программы или комбинация программ), такая как файловая система или таблица символов, которая может осуществлять управление и обеспечивать доступ к множественным объектам.

4.124 Ссылка на объект: ссылка, которая заменяется объектом.

ПРИМЕЧАНИЕ. Существует два вида ссылок: поименованная ссылка на объект и краткая ссылка.

4.125 Набор объектов: набор объявлений объектов, которые используются вместе.

ПРИМЕЧАНИЕ. Набор объектов может быть публичным текстом.

4.126 Структура объекта: организация документа в один или большее число отдельных объектов.

ПРИМЕЧАНИЕ. Первым объектом является объект документа SGML: он содержит ссылки на объекты, которые указывают, какие другие объекты ему принадлежат.

4.127 Текст объекта: параметр объявления объекта, который определяет текст замены либо включением его в литерал параметра, либо указывая на него с помощью внешнего идентификатора.

4.128 Строка эквивалентной ссылки: строка символов, состоящая из ссылки на объект и, возможно, ***RE*** и/или ***RS***, которая заменяет краткую ссылку, когда документ конвертируется из конкретного синтаксиса, который поддерживает краткие ссылки, в синтаксис, который этого не делает.

4.129 Escape-последовательность: управляющая последовательность, первым символом которой является символ Escape (ESC).

4.130 Исключения: параметр объявления элемента, который изменяет действие *модели содержания* элемента и моделей содержания элементов, встречающихся в ней, разрешая включения и запрещая изъятия.

4.131 Изъятия: элементы, которые не допускаются где-либо в содержании элемента или его подэлементов, даже если применимая модель содержания или включения разрешили бы это факультативно.

4.132 Явная ссылка на содержание: ссылка на содержание, которая была определена в *спецификации атрибута*.

4.133 Явная связь (определение процесса): определение процесса связи, в котором могут быть определены типы элементов - результатов и их атрибуты и множественные наборы значений атрибутов связи.

4.134 Внешний объект: объект, чей текст не включен непосредственно в объявление объекта; вместо этого определены его системный идентификатор и/или публичный идентификатор.

ПРИМЕЧАНИЕ. Тип документа или объявление типа связи могут включать идентификатор внешнего объекта, содержащий все или часть подмножества объявления; внешний идентификатор служит одновременно и как объявление объекта и как ссылка на объект.

4.135 Внешний идентификатор: параметр, который идентифицирует внешний объект или нотацию содержания данных.

ПРИМЕЧАНИЕ. Имеются два вида внешних идентификаторов: системный идентификатор и публичный идентификатор.

4.136 Фиксированный атрибут: атрибут, чье установленное значение (если таковое существует) должно быть идентично его значению по умолчанию.

4.137 Формальный публичный идентификатор: публичный идентификатор, который построен в соответствии с правилами, определенными в этом международном стандарте, таким образом, чтобы идентификатор его владельца и компоненты его текстового идентификатора могли различаться.

4.138 Ошибка формального публичного идентификатора: ошибка в конструкции или использовании *формального публичного идентификатора*, иная нежели ошибка, которая сделала бы невозможным его существование как действительного *минимального литерала*.

ПРИМЕЧАНИЕ. Ошибка формального публичного идентификатора может возникать, только если в объявлении SGML определено "FORMAL YES". Однако невозможность публичного идентификатора быть минимальным литералом всегда является ошибкой.

4.139 Символ функции: символ разметки, назначенный конкретным синтаксисом, который может выполнять некоторую функцию SGML в дополнение к потенциальному распознаванию в качестве разметки. Если он не распознается как разметка в контексте, в котором допускаются

данные, он обрабатывается как данные (если только язык не предписывает специальную обработку, как в случае символов функций ***RE*** и ***RS***).

4.140 Параметр идентификации символа функции: параметр объявления SGML, который идентифицирует символы, присвоенные функциям ***RE***, ***RS*** и ***SPACE***, и позволяет определять дополнительные функции.

4.141 Набор G0: в расширении кода графического алфавита виртуальный набор символов, который представляет графические символы набора символов документа, номера символов которых менее 128, в их нормальных позициях кодового набора.

4.142 Общий разграничитель (роль): роль разграничителя, иная нежели краткая ссылка.

4.143 Общий объект: объект, на который имеется ссылка в содержании элемента или литерала значения атрибута.

4.144 Ссылка на общий объект: поименованная ссылка на общий объект.

4.145 Родовой идентификатор: имя, которое идентифицирует тип элемента.

4.146 GI: Родовой идентификатор.

4.147 Графический символ: символ, такой как буква, цифра или знак пунктуации, который обычно занимает отдельную позицию при отображении текста.

4.148 Расширение кода графического алфавита: расширение кода, в котором множественные наборы графических символов отображаются на позиции кодового набора документа с использованием функций смещения для порождения виртуальных наборов символов.

4.149 Группа: часть параметра, который ограничена уравновешенной парой разграничителей ***grpo*** и ***grpc*** или разграничителей ***dtgo*** и ***dtgc*** .

ПРИМЕЧАНИЕ. Имеются пять видов групп: группа имен, группа маркеров имен, группа модели, группа тегов данных, и группа шаблонов тегов данных. Группы имен, маркеров имен или шаблонов тегов данных не могут содержать группу, но группа модели может содержать группу модели, а группа тегов данных может содержать группу шаблонов тегов данных.

4.150 ID: уникальный идентификатор.

4.151 Список ссылок на ID: значение атрибута, которое является списком значений ссылок на ID.

4.152 Значение ссылки на ID: значение атрибута, которое является *именем*, определенным как *значение id* в том же экземпляре документа.

4.153 Значение ID: значение атрибута, которое является *именем*, уникально идентифицирующим элемент; т.е. оно не может быть тем же, что и какое-либо другое *значение id* в том же экземпляре документа.

4.154 Возможный атрибут: атрибут, для которого необязательно наличие *спецификации атрибута*, и чье значение определяется приложением, если оно не определено.

4.155 Неявная связь (определение процесса): определение процесса связи, в котором все типы элементов - результатов и их атрибуты подразумеваются приложением, но могут быть определены множественные наборы значений атрибутов связи.

4.156 Символы инертных функций: класс символов, состоящий из символов функций, чья дополнительная SGML "функция" не должна ничего делать.

4.157 Включенный подэлемент: подэлемент, который не разрешается моделью содержащего его элемента, но разрешается исключением включения.

4.158 Включения: элементы, наличие которых допускается где-либо в

содержании элемента или его подэлементов даже при том, что применимая модель их не допускает.

4.159 Собственно факультативный маркер: маркер группы модели, который:

a) Имеет индикатор появления ***opt*** или ***rep***; или

b) Является группой ***or***, один из чьих маркеров собственно факультативен; или

c) Является группой ***and*** или ***seq***, все маркеры которой собственно факультативны.

4.160 Образец (типа документа): данные и разметка для иерархии элементов, который удовлетворяют определению типа документа.

4.161 Интерпретируемый литерал параметра: текст *литерала параметра* за исключением разграничителей литерала, в котором были заменены ссылки на символьные объекты и объекты параметров.

4.162 Идентификатор владельца ИСО: *идентификатор владельца*, состоящий из порядкового номера публикации ИСО или номера регистрации набора символов, который используется, когда *публичный идентификатор* идентифицирует публикацию ИСО или назначается ей, либо идентифицирует зарегистрированный набор символов ИСО.

4.163 Описание текста ИСО: *описание публичного текста*, состоящее из последнего элемента заголовка публикации ИСО (без обозначения части, если таковое имеется), которое используется, когда *публичный идентификатор* идентифицирует публикацию ИСО.

4.164 Ключевое слово: параметр, который является зарезервированным именем, определенным конкретным синтаксисом, в противоположность произвольному тексту.

ПРИМЕЧАНИЕ. В параметрах, где может быть указано ключевое слово или имя, определенное приложением, ключевое слово всегда

предшествует индикатору зарезервированного имени. Поэтому приложение может определять имена безотносительно того, используются ли также эти имена конкретным синтаксисом.

4.165 Атрибут связи: атрибут типа элемента - источника, значимый только в контексте конкретного процесса, который выполняется над экземпляром документа - источника.

4.166 Процесс связи: процесс, который создает новый образец некоторого типа документа (результат) из существующего образца документа того же или другого типа (источник). Процессы могут быть организованы в цепочку таким образом, чтобы результат одного являлся источником для следующего.

ПРИМЕЧАНИЕ. К примерам процессов связи относится редактирование, в котором типы документов источника и результата обычно одинаковы, и форматирование, в котором они обычно различны.

4.167 Определение процесса связи: специфические для приложения правила, которые применяют SGML для описания процесса связи. Определение процесса связи включает формальную спецификацию, выраженную в *объявлении типа связи*, связи между элементами источника и результата, включая определения атрибутов источника, применимых к процессу связи ("атрибуты связи").

ПРИМЕЧАНИЯ:

1. Определение процесса связи может также включать комментарии, которые описывают семантику процесса, включая значение атрибутов связи и их влияние на процесс.
2. Имеются три вида определений процесса связи: простое, неявное и явное.

4.168 Набор связей: поименованный набор сопоставлений, объявленных в *объявлении набора связей*, в котором элементы типа документа - источника связаны с элементами типа документа - результата. Для каждой связи элементов, могут быть определены атрибуты связи источника и атрибуты элемента - результата .

4.169 Объявление набора связей: объявление разметки, которое определяет набор связей.

4.170 Объявление типа связи: объявление разметки, которое содержит формальную спецификацию определения процесса связи.

4.171 Подмножество объявления типа связи: наборы объектов, наборы атрибутов связи, набор связей и объявления использования набора связей, которые возникают в подмножестве объявления типа связи.

ПРИМЕЧАНИЕ. Внешний объект, на который имеется ссылка из объявления типа связи, рассматривается как часть подмножества объявления.

4.172 Смещение с блокировкой: функция смещения, которая применяется до появления другой функции смещения с блокировкой.

4.173 Буквы нижнего регистра: класс символов, состоящий из 26 строчных букв от "a" до "z" без диакритических знаков.

4.174 Символы имен нижнего регистра: класс символов, состоящий из каждого дополнительного *символа имени* нижнего регистра, присвоенного конкретным синтаксисом.

4.175 Начальные символы имен нижнего регистра: класс символов, состоящий из каждого дополнительного *начального символа имени* нижнего регистра, присвоенного конкретным синтаксисом.

4.176 LPD: определение процесса связи.

4.177 Отображение: отображение краткой ссылки.

4.178 Разметить: добавить разметку к документу.

4.179 Отмеченный раздел: раздел документа, который был идентифицирован для специальной цели, например, игнорирования разметки в его пределах.

4.180 Объявление отмеченного раздела: объявление разметки, которое идентифицирует отмеченный раздел и определяет, как он должен обрабатываться.

4.181 Конец отмеченного раздела: закрывающая последовательность разграничителей объявления отмеченного раздела.

4.182 Начало отмеченного раздела: открывающая последовательность разграничителей объявления отмеченного раздела.

4.183 Разметка: текст, который добавляется к данным документа, чтобы передать информацию о нем.

ПРИМЕЧАНИЕ. Имеются четыре вида разметки: описательная разметка (теги), ссылки, объявления разметки и команды обработки.

4.184 Символ разметки: символ SGML, который в зависимости от контекста может интерпретироваться как разметка или как данные.

4.185 Условное соглашение разметки: условное соглашение приложения, управляющее разметкой, такое как правила формулирования имени объекта или привилегированного подмножества допустимых разграничителей кратких ссылок.

4.186 Объявление (разметки): разметка, которая управляет тем, как должна интерпретироваться другая разметка документа.

ПРИМЕЧАНИЕ. Имеются 13 видов объявлений: SGML, объекта, элемента, списка определений атрибутов, нотации, типа документа, типа связи, набора связей, использования связи, отмеченного раздела,

отображение краткой ссылки, использования краткой ссылки и комментария.

4.187 Функция минимизации (разметки): функция SGML, которая позволяет минимизировать разметку, сокращая или опуская теги, или сокращая ссылки на объект.

ПРИМЕЧАНИЕ. Функции минимизации разметки не затрагивают определение типа документа, так что минимизированный документ может быть послан системе, которая не поддерживает эти функции, путем первоначального восстановления пропущенной разметки. Имеются пять видов функций: SHORTTAG, OMITTAG, SHORTREF, DATATAG, и RANK.

4.188 Символы включения разметки: класс символов, состоящий из символов функций, которые восстанавливают распознавание разметки, если оно было подавлено появлением символа выключения разметки.

4.189 Символы выключения разметки: класс символов, состоящий из символов функций, которые подавляют распознавание разметки до появления символа включения разметки или конца объекта.

4.190 Символы подавления разметки: класс символов, состоящий из символов функций, которые подавляют распознавание разметки для символа, следующего непосредственно за ними в том же самом объекте (если таковой имеется).

4.191 Минимальный документ SGML: удовлетворяющий требованиям SGML документ, который повсеместно использует конкретный синтаксис ядра, никаких функций, и набор доступных объемов ссылок.

4.192 Функция минимизации: функция минимизации разметки.

4.193 Модель: модель содержания.

4.194 Группа модели: компонент модели содержания, который определяет порядок появления элементов и строк символов в *содержании* элемента, с учетом изменений за счет *исключений,* определенных в *модели содержания* элемента и моделях содержания других открытых элементов.

4.195 Многокодовый базовый конкретный синтаксис: многокодовый вариант базового конкретного синтаксиса, в котором разметка не распознается, когда используется расширение кода.

4.196 Многокодовый конкретный синтаксис: конкретный синтаксис, который допускает управляющим символам расширения кода быть символами SGML.

4.197 Многокодовый конкретный синтаксис ядра: многокодовый вариант конкретного синтаксиса ядра, в котором разметка не распознается, когда используется расширение кода.

4.198 Имя: маркер имени, чей первый символ является начальным символом имени.

4.199 Символ имени: символ, который может присутствовать в имени: начальные символы имен, цифры и другие символы, назначенные конкретным синтаксисом.

4.200 Группа имен: группа, чьи маркеры должны быть именами.

4.201 Начальный символ имени: символ, с которого может начинаться имя: буквы и другие символы, назначенные конкретным синтаксисом.

4.202 Маркер имени: строка символов, состоящая исключительно из символов имени, чья длина ограничена количеством NAMELEN.

ПРИМЕЧАНИЕ. Маркер имени, который появляется в группе, также является маркером; тот, который появляется как значение атрибута, им не является.

4.203 Группа маркеров имени: группа, чьи маркеры должны быть маркерами имен.

4.204 Поименованная ссылка на символ: ссылка на символ, состоящая из разграниченного *имени функции*.

4.205 Поименованная ссылка на объект: ссылка на объект, состоящая из разграниченного имени общего объекта или объекта параметра (возможно квалифицируемого спецификацией типа документа), которая была объявлена в объявлении объекта.

ПРИМЕЧАНИЕ. Ссылка на общий объект может иметь необъявленное имя, если был объявлен объект по умолчанию.

4.206 Параметр правил наименования: параметр объявления SGML, который идентифицирует добавления к стандартным классам символов алфавита имен и определяет замену регистра.

4.207 Символ, не относящийся к SGML: символ в наборе символов документа, чье кодированное представление никогда не появляется в объекте SGML.

4.208 Объект данных, не относящийся к SGML: объект, чьи символы не интерпретируются в соответствии с этим международным стандартом, и в котором, поэтому, не может быть распознана разметка SGML.

ПРИМЕЧАНИЕ. Интерпретация объекта данных, не относящегося к SGML, управляется нотацией содержания данных, которая может быть определена другим международным стандартом.

4.209 NONSGML: класс не относящихся к SGML символов, определенных набором символов документа.

4.210 Нормализованная длина (списка спецификаций атрибутов): длина, рассчитанная путем игнорирования фактических символов, используемых для разграничивания и отделения компонентов, и вместо этого подсчета дополнительного фиксированного числа на каждый компонент.

4.211 Атрибут нотации: атрибут, чьим значением является *имя нотации*, которое идентифицирует нотацию содержания данных *содержания* элемента.

ПРИМЕЧАНИЕ. Атрибут нотации не применяется, когда имеется явная ссылка на содержание, поскольку содержание элемента будет пусто.

4.212 Объявление нотации: объявление разметки, которое сопоставляет имя с идентификатором нотации.

4.213 Идентификатор нотации: *внешний идентификатор*, который идентифицирует нотацию содержания данных в *объявлении нотации*. Это может быть *публичный идентификатор*, если нотация является публичной, а в противном случае описание или другая информация, достаточная, чтобы вызвать программу для интерпретации нотации.

4.214 Имя нотации: имя, присвоенное нотации содержания данных объявлением нотации.

4.215 Номер: маркер имени, состоящий исключительно из цифр.

4.216 Маркер номера: маркер имени, чьим первым символом является цифра.

ПРИМЕЧАНИЕ. Маркер номера, который появляется в группе, также является маркером; тот, который появляется как значение атрибута, им не является.

4.217 Ссылка на цифровой символ: ссылка на символ, состоящая из разграниченного *номера символа*.

4.218 Доступный объем объекта: предел доступного объема для конкретного вида объекта, такого как определенные объекты или символы текста объекта.

4.219 Параметр минимизации пропущенного тега: параметр объявления элемента, который определяет, должен ли рассматриваться технически действительный пропуск начального тега или конечного тега как сообщаемая ошибка разметки.

4.220 Открытый элемент: *элемент*, чей *начальный тег* появился (или был пропущен посредством минимизации разметки), но чей *конечный тег* еще не возник (или не был пропущен через минимизацию разметки).

4.221 Открытый объект: объект, на который была ссылка, но чей конец еще не появился.

4.222 Открытое объявление отмеченного раздела: объявление отмеченного раздела, чье *начало отмеченного раздела* встретилось, но чей *конец отмеченного раздела* еще не появился.

4.223 Идентификатор владельца: часть публичного идентификатора, которая идентифицирует владельца или создателя публичного текста.

ПРИМЕЧАНИЕ. Имеются три вида идентификаторов: ИСО, зарегистрированный и незарегистрированный.

4.224 Параметр: часть объявления разметки, которая ограничена разделителями параметров (необходимыми или факультативными). Параметр может содержать другие параметры.

4.225 Объект параметра: объект, на который имеется ссылка в параметре объявления разметки.

4.226 Ссылка на объект параметра: поименованная ссылка на объект

параметра.

4.227 Литерал параметра: параметр или маркер, состоящий из разграниченных заменимых данных параметра.

4.228 Анализируемые символьные данные: ноль или большее число символов, которые возникают в контексте, в котором анализируется текст и распознается разметка. Они классифицируются как символы данных, поскольку они не были распознаны как разметка в ходе синтаксического анализа.

4.229 PCDATA: анализируемые символьные данные.

4.230 Объект PI: объект команды обработки.

4.231 Пункт: единица измерения доступного объема, грубый показатель относительных требований к объему памяти.

4.232 Процедура: обработка, определенная приложением для выполнения на элементах конкретного типа.

ПРИМЕЧАНИЯ:

1. Отдельная процедура может быть сопоставлена больше чем одному типу элемента, и/или больше чем одна процедура может выполняться над одним типом элемента в различных точках документа.
2. Процедура обычно является частью набора процедур.

4.233 Набор процедур: процедуры, которые используются совместно для данного прикладного процесса.

ПРИМЕЧАНИЕ. В приложениях SGML набор процедур обычно образует прикладную обработку для определения процесса связи.

4.234 Команда обработки: разметка, состоящая из системно-

специфических данных, которая управляет обработкой документа.

4.235 Объект команды обработки: объект, чей текст обрабатывается как *системные данные команды обработки*, когда на него имеется ссылка.

4.236 Пролог: часть объекта документа SGML или поддокумента SGML, которая содержит объявления типа документа и типа связи.

4.237 Надлежащий подэлемент: подэлемент, который допускается содержащей его моделью элемента.

4.238 ps (разделитель): разделитель параметров, встречающийся в объявлениях разметки.

4.239 Публичный идентификатор: минимальный литерал, который идентифицирует публичный текст.

ПРИМЕЧАНИЯ:

1. Публичные идентификаторы в документе могут факультативно интерпретироваться как формальные публичные идентификаторы.
2. Система отвечает за преобразование публичных идентификаторов в системные идентификаторы.

4.240 Публичный текст: текст, который известен вне контекста отдельного документа или системной среды, и к которому можно обращаться с помощью публичного идентификатора.

ПРИМЕЧАНИЯ:

1. Примерами являются стандартные или зарегистрированные определения типа документа, наборы объектов, наборы элементов, нотации содержания данных и другие конструкции разметки (см. приложение D).
2. Публичный текст не эквивалентен опубликованному тексту; он

не подразумевает неограниченный публичный достуВ частности, владелец публичного текста может по своему выбору продавать его другим лицам или выдавать лицензии на его использование, или ограничить доступ к нему для единственной организации.

3. Публичный текст упрощает доступ к общедоступным конструкциям, сокращает объем текста, которым следует обмениваться и уменьшает вероятность ошибок копирования .

4.241 Класс публичного текста: часть текстового идентификатора, идентифицирующая конструкцию разметки SGML, которой соответствует публичный текст.

4.242 Описание публичного текста: часть текстового идентификатора, которая описывает публичный текст.

4.243 Последовательность, обозначающая публичный текст: часть текстового идентификатора, используемая в том случае, когда публичный текст является набором символов, которая содержит escape-последовательность ИСО 2022, обозначающая набор.

4.244 Версия отображения публичного текста: факультативная часть текстового идентификатора, которая выделяется среди публичного текста, имеющего обычное *описание публичного текста*, описыванием поддерживаемых устройств или используемой схемы кодирования. Если она пропущена, публичный текст аппаратно независим.

4.245 Язык публичных текстов: часть текстового идентификатора, которая определяет естественный язык, на котором написан публичный текст.

ПРИМЕЧАНИЕ. Это может быть язык данных, комментариев и/или определенных имен.

4.246 Количество: числовое ограничение на некоторый аспект разметки, такое как максимальная длина названия имени или максимальный уровень вложенности открытых элементов.

ПРИМЕЧАНИЕ. Количества определяются абстрактным синтаксисом, но конкретные значения назначаются им конкретным синтаксисом.

4.247 Набор количеств: набор назначений числовых значений именам количеств.

4.248 Ранжированный элемент: элемент, чей *родовой идентификатор* составлен из *базы ранга* и *суффикса ранга*. Когда начинается ранжированный элемент, его *суффикс ранга* становится текущим рангом для его *базы ранга*, и для баз ранга в *ранжированной группе* (если таковая существует), элементом которой является *база ранга*.

4.249 Ранжированная группа: группа баз ранга, которые совместно используют один и тот же текущий ранг. Когда начинается любой ранжированный элемент, чья база находится в группе, ее *суффикс ранга* становится текущим рангом для всех баз ранга в группе.

4.250 База ранга: имя, из которого может быть получен родовой идентификатор путем добавления в конец текущего ранга.

4.251 Суффикс ранга: номер, который добавляется в конец базы ранга для формирования *родового идентификатора*.

ПРИМЕЧАНИЕ. Номера обычно являются последовательными, начинающимися с 1, так что получающиеся в результате родовые идентификаторы предполагают относительные ранги своих элементов (например, H1, H2 и H3 для уровней заголовков элементов, где "H" - база ранга).

4.252 Запись: единица объекта SGML, ограниченная символами начала записи и конца записи, обычно соответствующая строке ввода на устройстве ввода текста.

ПРИМЕЧАНИЯ:

1. Она называется "записью", а не "строкой", чтобы отличить ее от строк вывода, создаваемых текстовым форматером.
2. Объект SGML может состоять из многих записей, единственной записи или текста без символов границы записи (который может рассматриваться как часть записи или текст без записей, в зависимости от того, встречаются ли где-либо в документе символы границы записи).

4.253 Граница записи (символ): символ начала записи (***RS***) или конца записи (***RE***).

4.254 Конец записи: *символ функции*, назначенный конкретным синтаксисом, который представляет конец записи.

4.255 Начало записи: *символ функции*, назначенный конкретным синтаксисом, который представляет начало записи.

4.256 Ссылка: разметка, которая заменяется другим текстом, или объект или одиночный символ.

4.257 Набор доступных объемов ссылки: набор доступных объемов, определенный в этом международном стандарте.

4.258 Конкретный синтаксис ссылок: конкретный синтаксис, определенный в этом международном стандарте, который используется во всех объявлениях SGML.

4.259 Набор разграничителей ссылок: набор разграничителей, определенный в этом международном стандарте, который используется в конкретном синтаксисе ссылок.

4.260 Набор количеств ссылок: набор количеств, определенный этим международным стандартом.

4.261 Зарезервированное имя ссылки: зарезервированное имя, определенное этим международным стандартом.

4.262 Зарегистрированный идентификатор владельца: идентификатор владельца, который был создан в соответствии с ИСО 9070. Он уникален среди зарегистрированных идентификаторов владельцев и отличается от идентификаторов владельцев ИСО и незарегистрированных идентификаторов владельцев.

4.263 Заменяемые символьные данные: символьные данные, в которых распознается и заменяется *ссылка на общий объект* или *ссылка на символ*.

ПРИМЕЧАНИЕ. Разметка, которая отменяет *заменяемые символьные данные*, не распознается в тексте замены объектов, на которые она ссылается.

4.264 Заменяемые данные параметра: символьные данные, в которых распознается и заменяется *ссылка на объект параметра* или *ссылка на символ*.

ПРИМЕЧАНИЕ. Разметка, которая отменяет *заменяемые данные параметра*, не распознается в тексте замены объектов, на которые она ссылается.

4.265 Символ замены: символ, который заменяет *ссылку на символ*.

4.266 Текст замены: текст объекта, который заменяет ссылку на объект.

4.267 Сообщаемая ошибка разметки: невозможность документа удовлетворить этому международному стандарту, когда он

анализируется относительно активного документа и типов связи, помимо таких ошибок как:

a) неоднозначная модель содержания;

b) изъятие, которое может изменять необходимое или факультативное состояние группы в модели;

c) превышение предела доступного объема;

d) ошибка в объявлении SGML;

e) появление символа, не относящегося к SGML; или

f) ошибка формального публичного идентификатора.

4.268 Обязательный атрибут: атрибут, для которого всегда должна существовать *спецификация атрибута* для значения атрибута.

4.269 Зарезервированное имя: имя, определенное конкретным синтаксисом, а не приложением, такое как имя объявления разметки.

ПРИМЕЧАНИЕ. Такие имена появляются в этом международном стандарте как синтаксические литералы.

4.270 Параметр использования зарезервированного имени: параметр *объявления SGML*, которое определяет любую замену в объявленном конкретном синтаксисе для зарезервированного имени ссылки.

4.271 Тип документа - результата (связи): тип документа, новый экземпляр которого создан как результат процесса связи.

4.272 Тип элемента - результата (связи): элемент, который определен в объявлении типа документа - результата.

4.273 s (разделитель): разделитель, состоящий из символов разделителя и других непечатаемых символов функций, который встречается в разметке и в *содержании элемента*.

4.274 Удовлетворенный маркер: *маркер содержания*, для которого встретилось соответствующее содержание.

4.275 Объект SDATA: объект специальных символьных данных.

4.276 Разделитель: *s*, *ds*, *ps* или *ts*.

4.277 Символы разделителя: класс символов, состоящий из символов функций, которые допустимы в разделителях и которые будут заменяться символом ***Space*** в тех контекстах, в которых ***RE*** заменяется символом ***Space***.

4.278 SGML: Стандартный Обобщенный Язык Разметки

4.279 Приложение SGML: правила, которые применяют SGML к приложению обработки текста. Приложение SGML включает формальную спецификацию конструкций разметки, используемых в приложении, выраженную в SGML. Оно может также включать не относящееся к SGML определение семантики, условных соглашений приложения, и/или обработки.

ПРИМЕЧАНИЯ:

1. Формальная спецификация приложения SGML обычно включает определения типа документа, нотации содержания данных и наборы объектов, а также возможно конкретный синтаксис или набор доступных объемов. Если обработка определяется приложением, формальная спецификация может также включать определения процессов связи.

2. Формальная спецификация приложения SGML составляет обычные части документов, обработанных приложением. Эти обычные части часто делаются доступными в качестве публичного текста.

3. Формальная спецификация обычно сопровождается комментариями и/или документацией, которая объясняет семантику, условные соглашения приложения и спецификации обработки приложения.

4. Приложение SGML существует независимо от любой реализации. Однако, если обработка определяется приложением, определение, не относящееся к SGML, может включать прикладные процедуры, реализованные на языке программирования или обработки текста.

4.280 Символ SGML: символ, который допускается в объекте SGML.

4.281 Объявление SGML: объявление разметки, которое определяет набор символов, конкретный синтаксис, факультативные функции и требования доступного объема разметки документа. Оно применяется ко всем объектам SGML документа.

4.282 Документ SGML: документ, который представлен как последовательность символов, организованных физически в структуру объекта, а логически в структуру элемента, по существу так, как описано в этом международном стандарте. Документ SGML состоит из символов данных, которые представляют его информационное содержание, и символов разметки, которые представляют структуру данных и другую информацию, полезной для его обработки. В частности, разметка описывает по крайней мере одно определение типа документа и экземпляр структуры, соответствующих определению.

4.283 Объект документа SGML: объект SGML, с которого начинается документ SGML. Он содержит, как минимум, объявление SGML, объявление типа базового документа, и начало и конец (если он не весь) элемента базового документа.

4.284 Объект SGML: объект, чьи символы интерпретируются как разметка или данные в соответствии с этим международным стандартом.

ПРИМЕЧАНИЕ. Имеются три типа объектов SGML: *объект документа SGML, объект поддокумента SGML* и *объект текста SGML*.

4.285 Синтаксический анализатор SGML: программа (или часть программы или комбинации программ), которая распознает разметку в удовлетворяющих требованиям SGML документах.

ПРИМЕЧАНИЕ. Если бы следовало провести аналогию с процессорами языков программирования, синтаксический анализатор SGML выполнял бы функции как лексического анализатора, так и синтаксического анализатора в отношении документов SGML.

4.286 Объект поддокумента SGML: объект SGML, который соответствует объявлению SGML объекта документа SGML, соответствую при этом своему собственному типу документа и объявлениям типа связи. Он содержит, как минимум, объявление типа базового документа, а также начало и конец элемента базового документа.

4.287 Система SGML: система, которая включает синтаксический анализатор SGML, менеджер объектов и оба или любое из следующего:

a) реализация одного или большего числа приложений SGML; и/или

b) средства пользователя для реализации приложений SGML, с доступом к синтаксическому анализатору SGML и менеджеру объектов.

4.288 Объект текста SGML: объект SGML, который соответствует объявлению SGML объекта документа SGML, и объявлениям типа документа и типа связи, которым соответствует объект, в котором на него имеется ссылка.

4.289 Функция смещения: в расширении кода графического алфавита управляющая последовательность или управляющий символ, который вызывает набор графических символов.

ПРИМЕЧАНИЕ. Имеются два вида функций: одиночное смещение и смещение с блокировкой.

4.290 Краткая ссылка: строка краткой ссылки.

4.291 Роль разграничителя краткой ссылки: роль разграничителя, которой конкретным синтаксисом может быть присвоено ноль или большее число строк. Когда распознается строка краткой ссылки, она заменяется общим объектом, на имя которого она отображается в текущем отображении, или обрабатывается как разделитель или данные, если ни на что не отображается.

4.292 Отображение (краткой ссылки): поименованный набор сопоставлений, объявленных *объявлением отображения краткой ссылки*, в котором каждый разграничитель краткой ссылки отображается на имя общего объекта или ни на что.

4.293 Объявление отображения краткой ссылки: объявление разметки, которое определяет отображение краткой ссылки.

4.294 Набор кратких ссылок: набор отображения краткой ссылки, использования краткой ссылки, и объявлений объекта, которые используются совместно.

ПРИМЕЧАНИЕ. Набор кратких ссылок может быть публичным текстом.

4.295 Краткая ссылка (строка): строка символов, назначенная конкретным синтаксисом на роль разграничителя краткой ссылки.

4.296 Объявление использования краткой ссылки: объявление разметки, которое сопоставляет отображение краткой ссылки с одним или большим числом типов элемента, или идентифицирует новое текущее отображение для текущего элемента.

4.297 Избегаемый символ (номер): номер символа, идентифицированный конкретным синтаксисом, который следует избегать в документах, использующих этот синтаксис, поскольку некоторые системы могут ошибочно обрабатывать его как управляющий символ.

4.298 Существенный символ SGML: *символ разметки* или символ *минимальных данных*.

4.299 Простая связь (определение процесса): определение процесса связи, в котором все типы элемента - результата и их атрибуты подразумеваются приложением, и может быть определен только один набор значений атрибутов связи. Документ - источник должен быть базовым.

4.300 Одиночное смещение: функция смещения, которая применяется только к следующему символу.

4.301 Тип документа - источника (связи): тип документа, существующий образец которого - источник процесса связи.

4.302 Тип элемента - источника (связи): тип элемента, который определен в объявлении документа - источника.

4.303 Пробел: *символ функции*, назначенный конкретным синтаксисом, который представляет пробел.

4.304 Объект специальных символьных данных: объект, чей текст обрабатывается как символьные данные, когда на него имеется ссылка. Текст зависит от конкретной системы, устройства или прикладного процесса.

ПРИМЕЧАНИЕ. Объект специальных символьных данных обычно должен переопределяться для различных приложений, систем или устройств вывода.

4.305 Стандартный Обобщенный Язык Разметки: язык представления документа, который формализует разметку и делает ее независимой от системы b обработки.

4.306 Начальный тег: описательная разметка, которая идентифицирует начало элемента и определяет его родовой идентификатор и атрибуты.

4.307 Ключевое слово состояния: параметр объявления отмеченного раздела, который определяет, должен ли отмеченный раздел игнорироваться, и если нет, должен ли он обрабатываться как символьные данные, заменимые символьные данные или обычным образом.

4.308 Строка: строка символов.

4.309 Подэлемент: элемент, который встречается в содержании другого элемента ("содержащий элемент") таким способом, что подэлемент начинается, когда содержащий элемент является текущим элементом.

4.310 Набор символов синтаксиса - ссылок: набор символов, обозначенный конкретным синтаксисом и известный всем потенциальным пользователям синтаксиса, который содержит каждый существенный символ SGML. Он дает возможность конкретному синтаксису быть определенным безотносительно конкретного документа или наборов системных символов, с которыми он может использоваться.

4.311 Набор системных символов: набор символов, используемый в системе SGML.

4.312 Объявление системы: объявление, включенное в документацию для удовлетворяющей требованиям SGML системы, которое определяет функции, набор доступных объемов, конкретные синтаксисы и набор символов, которые поддерживает система, нотации содержания данных, которые она может интерпретировать, и любые услуги проверки действительности, которые она может выполнять.

4.313 Системный идентификатор: системные данные, которые

определяют идентификатор файла, место в памяти, вызов программы, позицию потока данных или другую системно-специфическую информацию, которая определяет местоположение внешнего объекта.

4.314 Тег: описательная разметка.

ПРИМЕЧАНИЕ. Имеются два вида тегов: начальный тег и конечный тег.

4.315 Целевой элемент: элемент, чей родовой идентификатор определяется в группе тегов данных

4.316 Текст: символы.

ПРИМЕЧАНИЕ. Символы могут иметь их нормальное значение набора символов, либо могут интерпретироваться в соответствии с нотацией содержания данных как представление графики, изображений и т.д.

4.317 Текстовый идентификатор: часть публичного идентификатора, которая идентифицирует публичный текст так, чтобы он мог отличаться от любого другого публичного текста с тем же идентификатором владельца.

ПРИМЕЧАНИЕ. Текстовый идентификатор состоит из класса публичного текста, факультативного индикатора недоступного текста, описания публичного текста, языка публичных текстов, и факультативной версии отображения публичного текста.

4.318 Приложение обработки текста: связанный набор процессов, выполняемых над документами соответствующих типов.

ПРИМЕЧАНИЕ. Некоторые примеры:

a) Публикация технических руководств для разработчика программного обеспечения: типы документа включают руководства по инсталляции, эксплуатации и обслуживанию; процессы включают создание, пересмотр, форматирование и макетирование страниц для ряда устройств вывода.

b) Подготовка рукописей независимыми авторами для членов ассоциации издателей: типы документов включают книгу, журнал и статью; создание является единственным определенным процессом, поскольку каждый издатель имеет свои собственные методы форматирования и печати.

c) Офисная корреспонденция: типы документа включают записки, журналы регистрации почтовых сообщений и отчеты; процессы включают создание, пересмотр, простое форматирование, хранение и поиск, обновление журналов регистрации записок и генерацию поколений.

4.319 Маркер: часть группы, включая полную вложенную группу (но не соединитель), который ограничен разделителями маркера (необходимыми или факультативными).

4.320 Полный доступный объем: предел суммы доступных объемов всех объектов.

4.321 ts (разделитель): разделитель маркера, встречающийся в группах.

4.322 Определение типа: определение типа документа.

4.323 Недоступный публичный текст: публичный текст, который доступен только ограниченной части населения, выбранной его владельцем.

4.324 Уникальный идентификатор: имя, которое уникально идентифицирует элемент.

4.325 Незарегистрированный идентификатор владельца: идентификатор владельца, который может отличаться от зарегистрированных идентификаторов владельца и идентификаторов владельца ИСО. Поскольку он не построен в соответствии со стандартом регистрации, он может дублировать другой незарегистрированный идентификатор владельца.

4.326 Буквы верхнего регистра: класс символов, составленный из 26 заглавных букв от "A" до "Z".

4.327 Символы имени верхнего регистра: класс символов, состоящий из форм верхнего регистра соответствующих символов имен нижнего регистра.

4.328 Начальные символы имени верхнего регистра: класс символов, состоящий из форм верхнего регистра соответствующих начальных символов имен нижнего регистра.

4.329 Синтаксический анализатор SGML с проверкой ошибок: удовлетворяющий требованиям SGML синтаксический анализатор, который может находить и сообщать о сообщаемой ошибке разметки если (и только если) она существует.

4.330 Иной конкретный синтаксис: конкретный синтаксис, другой нежели конкретный синтаксис ссылок или конкретный синтаксис ядра.

4.331 Иной документ (удовлетворяющий) SGML: удовлетворяющий требованиям SGML документ, который использует иной конкретный синтаксис.

4.332 Виртуальный набор символов: в расширении кода графического алфавита один из наборов символов, известный как GO, G1, G2 или G3, который представляет отображение реального набора графических символов, обозначаемого escape-последовательностью, на позиции кодового набора документа, предварительно объявленные escape-последовательностью.

5 Нотация

ПРИМЕЧАНИЕ. В этом разделе описывается нотация, используемая в данном международном стандарте для определения SGML. Эта нотация сама по себе не является частью SGML (хотя между ними имеются некоторые подобия), поэтому этот раздел должен рассматриваться как влияющий только на представление этого международного стандарта, но не как его существенная часть.

Абстрактный синтаксис SGML определяется формальными синтаксическими правилами, каждое из которых определяет "синтаксическую переменную". Продукция состоит из номера сноски (в квадратных скобках), имени определяемой синтаксической переменной, знака равенства и выражения, которое составляет определение.

[номер] синтаксическая переменная = выражение

Выражение состоит из одного или большего числа "синтаксических маркеров", вложенных выражений и символов, которые определяют порядок и выбор среди них.

5.1 Синтаксические маркеры

Приведенный ниже список показывает типы синтаксических маркеров с помощью типографские условных соглашений, используемых для них в этом международном стандарте, вместе с источником их определений:

синтаксическая переменная. Синтаксический маркер, который определяется синтаксическим правилом.

"СИНТАКСИЧЕСКИЙ ЛИТЕРАЛ". Синтаксический маркер, состоящий из зарезервированного имени, которое должно быть введено в разметку точно так, как оно показано в синтаксическом правиле, за исключением того, что могут использоваться соответствующие буквы

нижнего регистра, если конкретным синтаксисом определена трансляция верхнего регистра общих имен. Синтаксические литералы определяются всякий раз, когда они встречаются в синтаксическом правиле, и определение применимо только в контексте этого правила.

роль разграничителя. Синтаксический маркер, который представляет строку разграничителя. Роли Разграничителя определены на рис. 3, который также приводит список строк, назначаемые на общие роли разграничителя конкретным синтаксисом ссылок. Строки, назначенные на роль разграничителя ***shortref***, показаны на рис. 4.

ТЕРМИНАЛЬНАЯ ПЕРЕМЕННАЯ. Синтаксический маркер, который представляет класс символов, чьи элементы не обязательно одинаковы во всех документах SGML. Терминальные переменные, чьи элементы назначаются конкретным синтаксисом, определены на Рис. 2. (Переменная ***NONSGML***, чьи члены назначаются набором символов документа, определена в 13.1.2.)

Терминальная Константа. Синтаксический маркер, который представляет либо сигнал конца объекта либо класс символов, чьи члены одинаковы во всех документах SGML. Терминальные константы определены на рис. 1.

5.2 Символы упорядочивания и выбора

Если в выражении присутствуют более чем один синтаксический маркер, их упорядочивание и выбор среди них определяется символами, которые их соединяют, следующим образом:

, Все должны появиться в указанном порядке.

& Все должны появиться в любом порядке.

| Должен появиться один и только один.

Каждый выбранный синтаксический маркер должен появиться один и только один раз, если иное не указано суффиксом, состоящим из одного из следующих символов:

? Факультативный (0 или 1 раз).

+ Обязательный и повторяемый (1 или большее число раз).

* Факультативный и повторяемый (0 или большее число раз).

Сначала применяются суффиксы появления, затем соединители порядка. Для изменения этих приоритетов могут использоваться скобки как в математике.

6 Структура объекта

6.1 Документ SGML

Документ SGML физически организован в виде структуры объектов. Первый объект является *объектом документа SGML*; он содержит объектные ссылки, которые указывают, где находятся другие относящиеся к нему объекты.

[1] Документ SGML = *объект документа SGML,*

(объект поддокумента SGML |

объект текста SGML |

*объект данных, не относящихся к SGML)**

ПРИМЕЧАНИЯ:

1. Этот международный стандарт не ограничивает физическую организацию документа в пределах потока данных, протокола обработки сообщений, файловой системы и т.д., которые его содержат. В частности, отдельные объекты могут встречаться в одном физическом объекте, единый объект может быть разделен между множественными физическими объектами, а физические объекты могут возникать в любом порядке.

2. Этот международный стандарт не накладывает ограничений на символы, которые могут встречаться в потоке данных вне объектов

SGML. Такие символы должны интерпретироваться в соответствии с применимыми стандартами и соглашениями.

3. Формат Обмена Документами SGML (SDIF), стандартизированный в ИСО 9069, позволяет условно управлять документом SGML как единым объектом, сохраняя при этом объектную структуру.

6.2 Объекты SGML

[2] Объект документа SGML =

объявление SGML, s, пролог, s*,*

набор экземпляров документа, ***Ee***

[3] Объект поддокумента SGML = *пролог, s*,*

набор экземпляров документа, ***Ee***

[4] Объект текста SGML = *символ SGML*,* ***Ee***

Символы объекта SGML анализируются в соответствии с этим международным стандартом для распознавания *объявления SGML*, *пролога* и *набора экземпляров документа*, а также их составных частей.

Каждый *символ SGML* анализируется в порядке его появления следующим способом:

a) Производится проверка, не является ли символ частью строки разграничителя (см. 9.6). Если распознается общая строка разграничителя, выполняется соответствующее действие разграничителя. Если распознается *краткая ссылка*, она обрабатывается, как определено в 9.4.6.

b) Если символ не является частью строки разграничителя, или это часть неотображаемой краткой ссылки, производится проверка, не является ли он разделителем, и если так, то он игнорируется.

c) Если символ не является частью разграничителя или разделителем, он обрабатывается как данные.

Если *символ SGML* является *символом функции*, его функция выполняется дополнительно к любой другой обработке, которую он может получить.

6.2.1 S разделитель

[5] s = ***SPACE*** | ***RE*** | ***RS*** | ***SEPCHAR***

Символ SGML не рассматривается как часть s, если он может интерпретироваться как некоторая другая разметка. Если он рассматривается как часть s, он игнорируется.

6.2.2 Конец объекта

Ee является сигналом конца объекта.

ПРИМЕЧАНИЕ. ***Ee*** не является символом и никогда не обрабатывается как данные. Он может появляться, только если разрешен явным образом.

Система может представлять ***Ee*** любым способом, который позволит отличить его от символов SGML.

ПРИМЕЧАНИЕ. Например, ***Ee*** может быть представлен битовой комбинацией символа, не относящегося к SGML, если такой был назначен.

6.2.3 Подразумеваемое объявление SGML

При исключительной обработке документа одной системой, система может подразумевать *объявление SGML*. Однако, объявление должно присутствовать в явном виде, если впоследствии документ передается другой системе.

6.3 Объект данных, не относящихся к SGML

[6] Объект данных, не относящихся к SGML = *символ**, ***Ee***

ПРИМЕЧАНИЕ. *Объект данных, не относящихся к SGML*, объявляется с параметром нотации, который указывает, как должны интерпретироваться данные.

7 Структура элемента

7.1 Пролог

[7] пролог = *другой пролог*,*

объявление типа базового документа,

(объявление типа документа | другой пролог),*

*(объявление типа связи | другой пролог)**

[8] другой пролог = *объявление комментария*|

команда обработки | s

[9] объявление типа базового документа =

объявление типа документа

Объект документа или поддокумента SGML соответствует определениям типа документа и процесса связи, которые определяются соответственно в объявлениях типа документа и типа связи в его *прологе*. Текстовый объект SGML соответствует *прологу*, которому соответствует объект, в котором имеется ссылка на него. Синтаксический анализ объекта SGML производится в отношении тех типов документа и связи, которые считаются активными.

ПРИМЕЧАНИЕ. Обычно система указывает типы активного документа и связи синтаксическому анализатору SGML во время инициализации (см. F.2).

Объявления типа документа в дополнение к базовому разрешаются, только если в *объявлении SGML* определены "CONCUR YES" или "EXPLICIT YES".

7.2 Элемент документа

[10] набор экземпляров документа = *элемент базового документа*

[11] элемент базового документа = *элемент документа*

[12] элемент документа = *элемент*

7.2.1 Пределы

Экземпляры определений типа документа кроме базового разрешаются только в пределах, указанных параметром "CONCUR" *объявления SGML*.

7.3 Элемент

[13] элемент = *начальный тег?, содержание, конечный тег?*

Если *объявленное содержание* элемента имеет значение "EMPTY", или явную ссылку на содержание, *конечный тег* должен быть опушен.

ПРИМЕЧАНИЕ. Это требование не имеет отношения к минимизации разметки.

7.3.1 Минимизация пропущенным тегом

Если в *объявлении SGML* определен параметр "OMITTAG YES", тег может быть пропущен, как предусмотрено в этом пункте, в той мере, насколько пропуск не создает неоднозначность.

ПРИМЕЧАНИЕ. Определение типа документа может рассматривать технически справедливый пропуск ошибкой разметки (см. 11.2.2).

7.3.1.1 Пропуск начального тега

Начальный тег может быть пропущен, если элемент является контекстно обязательным элементом, а любые другие элементы, которые могут встретиться, являются контекстно факультативными элементами, кроме тех случаев, когда:

a) тип элемента имеет необходимый атрибут или *объявленное содержание*; или

b) *содержание* экземпляра элемента пусто.

Пропущенный *начальный тег* обрабатывается как имеющий пустой список спецификаций атрибутов.

7.3.1.2 Пропуск конечного тега

Конечный тег может быть пропущен для *элемента документа*, или для элемента, за которым следует либо

a) *конечный тег* другого открытого *элемента*; либо

b) *элемент* или *символ SGML*, который не допустим в его *содержании*.

ПРИМЕЧАНИЕ. Элемент, который не допустим, поскольку является изъятием, имеет тот же эффект, как и элемент, который не допустим, поскольку для него в группе модели не появляется маркер.

7.3.2 Минимизация тегом данных

Если в *объявлении SGML* определен параметр "DATATAG YES", *конечный тег* может быть пропущен для элемента, который является соответствующим содержанием группы тегов данных.

Содержание данных (целевого) элемента и его подэлементов сканируется для поиска строки, которая соответствует одной из структур тега данных в *шаблоне тега данных* элемента. Эта строка плюс любые последующие символы, которые соответствуют *структуре дополнения тега данных*, рассматриваются как тег данных элемента. Тег данных обрабатывается как *конечный тег* целевого элемента и как *символьные данные* в содержащем его элементе.

ПРИМЕЧАНИЕ. *Родовой идентификатор*, который встречается как

целевой элемент в *группе тегов данных*, может также встречаться в других контекстах как *маркер элемента*. В этих контекстах он не будет сканироваться для поиска тега данных. Он может также появляться в других группах тегов данных, возможно с различными шаблонами тега данных.

Содержание данных целевого элемента сканируется посимвольно. На каждом символе, проверяется самая длинная возможная *структура тега данных* в *шаблоне тега данных*. Если более одного целевого элемента являются открытыми элементами, то на каждом символе первым проверяются структуры последнего открытого целевого элемента. Если совпадения отсутствуют, проверяются структуры предшествующего открытого целевого элемента, и т.д.

ПРИМЕЧАНИЕ. Поэтому тег данных закончит последний открытый целевой элемент, с чьим шаблоном тега данных он совпадет.

Соответствие *шаблона тега данных* содержанию возникает после распознавания разметки и замены ссылок в содержании, но прежде, чем игнорируются какие-либо символы ***RE*** или ***RS***.

ПРИМЕЧАНИЕ. Поэтому шаблон тега данных может содержать поименованные символьные ссылки на ***RE*** и ***RS***.

Тег данных не может распознаваться в любом контексте, в котором не распознаются конечные теги; например, в пределах отмеченного раздела CDATA .

При сопоставлении с *шаблоном дополнения тега данных* символы шаблона могут быть пропущены, использованы однократно или

неопределенно повторены. Последнее (или единственное) использование может укоротить полный шаблон.

7.3.3 Количества

Число открытых элементов не может превышать параметра "TAGLVL".

Длина тега данных не может превышать параметра "DTAGLEN".

7.4 Начальный тег

[14] начальный тег = (***stago***,

спецификация типа документа,

спецификация родового идентификатора,

список спецификаций атрибутов, s,* ***tagc)*** |

минимизированный начальный тег

7.4.1 Минимизация

[15] минимизированный начальный тег = *пустой начальный тег* |

незакрытый начальный тег |

начальный тег, обеспечивающий net

Начальный тег может быть *минимизированным начальным тегом*, только если в *объявлении SGML* определен параметр "SHORTTAG YES" .

7.4.1.1 Пустой начальный тег

[16] пустой начальный тег == ***stago***,

спецификация типа документа, s,* ***tagc***

А *начальный тег* может быть *пустым начальным тегом*, только если *элемент* находится в экземпляре базового документа, и в этом случае *спецификация родового идентификатора* тега предполагается следующей:

a) если в *объявлении SGML* определен параметр "OMITTAG YES", то это *родовой идентификатор* последнего начатого открытого элемента в типе базового документа; или

b) если в *объявлении SGML* определен параметр "OMITTAG NO", то

это *родовой идентификатор* последнего законченного элемента в типе базового документа; или

c) если такие предыдущие элементы отсутствуют, то это *родовой идентификатор элемента документа*.

Пустой начальный тег обрабатывается как имеющий пустой *список спецификаций атрибутов*.

7.4.1.2 Незакрытый начальный тег

[17] незакрытый начальный тег = ***stago***,

спецификация типа документа,

спецификация родового идентификатора,

*список спецификаций атрибутов, s**

А начальный тег может быть *незакрытым начальным тегом*, только если за ним непосредственно следует строка символов, назначенная на роль разграничителя ***stago*** или ***etago***, независимо от того, начинается ли строка с действительной последовательности контекстных разграничителей.

7.4.1.3 Начальный тег, обеспечивающий NET

[18] начальный тег, обеспечивающий net = ***stago***,

спецификация типа документа,

спецификация родового идентификатора,

*список спецификаций атрибутов, s**, ***net***

А *начальный тег* может быть *начальным тегом, обеспечивающим net*, только если его *элемент* находится в экземпляре базового документа.

7.4.2 Количества

Длина *начального тега* до разрешения ссылок в *списке спецификаций атрибутов* и помимо разграничителей ***stago*** и ***tagc*** или ***net*** не может превышать значение параметра "TAGLEN".

7.5 Конечный тег

[19] конечный тег = (***etago***,

спецификация типа документа,

спецификация родового идентификатора, s,* ***tagc***) |

минимизированный конечный тег

Конечный тег заканчивает последний начатый открытый *элемент* в пределах экземпляра типа документа, определенного *спецификацией типа документа*, чей родовой идентификатор определен *спецификацией родового идентификатора*.

7.5.1 Минимизация

[20] минимизированный конечный тег = *пустой конечный тег* |

незакрытый конечный тег | *нулевой конечный тег*

Конечный тег может быть *минимизированным конечным тегом*, только если в *объявлении SGML* определен параметр "SHORTTAG YES".

7.5.1.1 Пустой конечный тег

[21] пустой конечный тег = ***etago****, s*,* ***tagc***

Если *конечный тег* является *пустым конечным тегом*, его *родовым идентификатором* является родовой идентификатор последнего начатого открытого элемента в экземпляре базового документа. Если такого элемента нет, *конечный тег* является ошибкой.

7.5.1.2 Незакрытый конечный тег

[22] незакрытый конечный тег = ***etago***,

спецификация типа документа,

*спецификация родового идентификатора, s**

Конечный тег может быть *незакрытым конечным тегом*, только если за ним непосредственно следует строка символов, назначенная на роль разграничителя ***stago*** или ***etago***, независимо от того, начинается ли строка с действительной последовательности контекстных разграничителей.

7.5.1.3 Нулевой конечный тег

[23] нулевой конечный тег = **net**

net распознается как *нулевой конечный тег*, только если *начальный тег* открытого документа в типе базового документа был *начальным тегом, обеспечивающим net*. Он предполагается конечным тегом такого последнего начатого элемента.

7.6 Содержание

[24] содержание = *смешанное содержание* | *содержание элемента* | *заменяемый символ данных* | *символ данных*

[25] смешанное содержание = (*символ данных*| *элемент* | *другое содержание*)*

[26] содержание элемента = (*элемент* | *другое содержание* | *s*)*

[27] другое содержание = *объявление комментария* | *объявление использования краткой ссылки* | *объявление использования типа связи* | *команда обработки* | ***shortref*** | *ссылка на символ* | *ссылка на общий объект* | *объявление отмеченного раздела* | ***Ee***

Объявленное содержание элемента или *модель содержания* определяют, какое из четырех типов *содержания* он имеет, или является ли он пустым, за исключением того, что *содержание* должно быть пусто, если элемент имеет явную ссылку на содержание.

Содержание элемента, объявленного как *символьные данные* или *заменяемые символьные данные*, заканчивается только контекстным разграничителем ***etago*** (который не должен открывать действительный *конечный тег*) или действительным ***net***. Такое завершение является ошибкой, если бы оно было ошибкой в случае *смешанного содержания*.

ПРИМЕЧАНИЕ. Символы содержания могут классифицироваться как содержание данных по любой из двух причин:

a) объявленные символьные данные.

Полное содержание элемента объявлено как *символьные данные* или *заменяемые символьные данными* параметром *объявленного содержания объявления элемента*.

b) Анализируемые символьные данные.

Элемент объявлен, как имеющий *смешанное содержание*, и *символ SGML* внутри него анализируется как данные, поскольку он не был распознан как разметка.

7.6.1 Границы записи

Если ***RS*** в *содержании* не интерпретируется как разметка, он игнорируется.

RE, остающийся в *содержании* после замены всех ссылок и распознавания разметки, обрабатывается как данные, если только его присутствие не может быть приписано исключительно разметке. Т.е.:

a) первый ***RE*** в элементе игнорируется, если ему не предшествовали ***RS***, данные или надлежащий подэлемент.

b) Последний ***RE*** в элементе игнорируется, если за ним не следуют данные или надлежащий подэлемент .

c) ***RE***, который не следует непосредственно за ***RS*** или ***RE***, игнорируется, если между ними не вмешались данные или надлежащий подэлемент.

При применении этих правил к элементу, содержание подэлемента игнорируется; т. е. надлежащий или включенный подэлемент обрабатывается как единица, которая заканчивается в той же записи, в которой начинается.

ПРИМЕЧАНИЕ.. Например, в

запись 1<outer><sub>
запись 2</sub> </outer>
запись 3

первый ***RE*** в элементе outer находится в конце записи 2. Он обрабатывается как данные, если "sub" является надлежащим элементом "outer", он игнорируется, если "sub" является включенным элементом, поскольку никакие данные или надлежащий элемент не предшествуют ему в элементе outer.

В любом случае первый ***RE*** в подэлементе находится в конце записи 1; он игнорируется, поскольку в подэлементе ему не предшествуют данные или надлежащий подэлемент.

RE считается возникшим непосредственно перед следующими за ним первыми данными или надлежащим подэлементом (т.е., после любого вмешивающегося объявления разметки, команды обработки или включенного подэлемента).

ПРИМЕЧАНИЯ:

1. Объект специальных символьных данных, объект данных, не относящихся к SGML или объект поддокумента SGML обрабатывается как данные, в то время как объект команды обработки не обрабатывается таким образом.
2. Хотя обработка границ записи определена SGML, требование, чтобы документы SGML были организованы в записи, отсутствует.
3. Никакой объект, включая *объект документа SGML* и внешние объекты, не считается начинающимся с ***RS*** или заканчивающимся ***RE***, если только это фактически не имеет место.

7.7 Спецификация типа документа

[28] спецификация типа документа = *группа имен*?

Разметка, содержащая *спецификацию типа документа*, обрабатывается только если:

a) имя в *группе имен* является именем типа активного документа; или

b) отсутствует *группа имен* (т.е., *спецификация типа документа* пуста).

ПРИМЕЧАНИЕ. Влияние этого требования заключается в том, что разметка с пустой спецификацией типа документа (т.е., без *группы имен*) будет применяться ко всем экземплярам документа (или к единственному).

Группа имен может быть определена, только если в *объявлении SGML* определены параметры "CONCUR YES" или "EXPLICIT YES".

7.7.1 Ссылки на общие объекты

Общий объект должно быть объявлен в *объявлении типа документа* каждого типа активного документа, поименованного в *спецификации типа документа*, или, если спецификация была пуста, или объект не был объявлен таким образом, в *объявлении типа* базового *документа*.

ПРИМЕЧАНИЕ. Влияние этого требования заключается в том, что на общий объект, объявленный в типе базового документа (включая объект по умолчанию), может быть ссылка в экземпляре любого типа документа, в котором не был определен общий объект с тем же именем.

7.7.2 Ссылки на объекты параметра

Группа имен не может быть определена для *ссылки на объект*

параметра внутри *объявления типа документа* или *объявления типа связи*. Такой объект должен быть объявлен в том же *объявлении типа документа* или, в случае *объявления типа связи*, в *объявлении типа документа- источника*.

7.8 Спецификация родового идентификатора (GI)

[29] спецификация родового идентификатора =

родовой идентификатор | *база ранга*

[30] родовой идентификатор = *имя*

Родовой идентификатор действителен, только если он был определен как тип элемента в определении типа документа и:

a) был назван в *модели содержания объявления элемента* для элемента, в котором он появляется; или

b) был назван в применимом исключении *включений*; или

c) является *именем типа документа* и появился в *начальном теге* или в *конечном теге* элемента документа.

7.8.1 Функция ранга

Условия настоящего пункта применяются в том случае, если в *объявлении SGML* определен параметр "RANK YES".

7.8.1.1 Полный родовой идентификатор

Если для ранжированного элемента определен *полный родовой идентификатор*, его *суффикс ранга* становится текущим рангом для его *базы ранга* и для баз ранга в любой *ранжированной группе*, элементом которой является *база ранга*.

7.8.1.2 База ранга

Спецификация родового идентификатора может быть *базой ранга*, если это было объявлено через *ранжированный элемент* или элемент *ранжированной группы* в соответствующем *объявлении элемента*.

Определение *базы ранга* эквивалентно определению *родового идентификатора*, который получен путем добавления к нему текущего

ранга. Определение *базы ранга*, если предварительно не появился элемент для установки текущего ранга для этой базы, является ошибкой.

7.9 Список спецификаций атрибутов

[31] список спецификаций атрибутов = *спецификация атрибута**

[32] спецификация атрибута = *s**, (*имя*, *s**, ***vi***, *s**)?,

спецификация значения атрибута

Действительность списка спецификаций атрибутов определяется *списком определений атрибутов*, связанным с элементом. Если не имеется связанного *списка определений атрибутов, список спецификаций атрибутов* должен быть пустым.

Каждый атрибут, для которого имеется *определение атрибута*, иное нежели возможный атрибут, должен быть определен (если только не используется минимизация разметки, как описано в 7.9.1.1).

Для каждого *определения атрибута* может существовать только одна *спецификация атрибута*.

Первое s может быть пропущено только в *спецификации атрибута*, которая следует за разграничителем.

7.9.1 Минимизация

7.9.1.1 Спецификация пропущенного атрибута

Если в *объявлении SGML* определены параметры "SHORTTAG YES" или "OMITTAG YES":

a) *Спецификация атрибута* должна существовать только для необходимого атрибута, а также для текущего атрибута при первом появлении любого элемента, в чьем *списке определений атрибутов* он имеется. Другие атрибуты будут обрабатываться как определенные со *значением атрибута* равным объявленному *значению по умолчанию*.

b) Если для текущего атрибута имеется *спецификация значения атрибута*, определенное *значение атрибута* станет значением по

умолчанию. Новое значение по умолчанию влияет на все элементы, связанные *списком определений атрибутов*, в котором был определен атрибут.

7.9.1.2 Имя пропущенного атрибута

Если в *объявлении SGML* определен параметр "SHORTTAG YES", *имя* и ***vi*** могут быть пропущены, если *спецификация значения атрибута* является неразграниченным *маркером имени*, который является членом группы, определенной в *объявленном значении* для этого атрибута.

ПРИМЕЧАНИЕ. *Маркер имени* может появляться только в одной группе в *списке определений атрибутов* (см. 11.3.3)

7.9.2 Количества

Нормализованная длина *списка спецификаций атрибутов* представляет собой сумму нормализованных длин каждого определенного имени атрибута и значения атрибута, которая не может превышать количество "ATTSPLEN".

Нормализованная длина имени атрибута является количеством "NORMSEP" плюс число символов в имени.

7.9.3 Спецификация значения атрибута

[33] спецификация значения атрибута =

значение атрибута | *литерал значения атрибута*

[34] литерал значения атрибута = (***lit***,

*заменяемые символьные данные**, ***lit***) | (***lita***,

*заменяемые символьные данные **, ***lita***)

Литерал значения атрибута интерпретируется как *значение атрибута* путем замены ссылки, игнорирования ***Ee*** и ***RS*** и замены ***RE*** или ***SEPCHAR*** символом ***SPACE***.

7.9.3.1 Минимизация

Если в *объявлении SGML* определен параметр "SHORTTAG YES", *спецификация значения атрибута* может быть *значением атрибута* (т.е., не *литералом значения атрибута*), при условии что он ничего не содержит кроме символов имени.

7.9.4 Значение атрибута

[35] значение атрибута = *символьные данные* |
имя общего объекта | *значение id*|
значение ссылки на id | *список ссылок на id*| *имя* |
список имен | *маркер имени* | *список маркеров имен* |
имя нотации | *номер* | *список номеров* |
маркер номера | *список маркеров номеров*

[36] значение id == *имя*

[37] список ссылок на id = *список имен*

[38] значение ссылки на id = *имя*

[39] список имен = *имя*, (s+, *имя*)*

[40] список маркеров имен = *маркер имени*, (s+, *маркер имени*)*

[41] имя нотации = *имя*

[42] список номеров = *номер*, (s+, *номер*)*

[43] список маркеров имен = *маркер номера*, (s+, *маркер номера*)*

Параметр *объявленного значения определения атрибута* определяет, какой из четырнадцати типов *значений атрибута* должен быть указан.

7.9.4.1 Синтаксические требования

Значение атрибута должно соответствовать *объявленному значению*.

Если *объявленное значение* включает группу, *значение атрибута* должно быть маркером в этой группе.

Пустой *литерал значения атрибута* может быть указан только для *значения атрибута*, чьим типов являются *символьные данные*.

7.9.4.2 Фиксированный атрибут

Значением атрибута, определенным для фиксированного атрибута, должно быть *значение по умолчанию*.

7.9.4.3 Имя общего объекта

Значение атрибута *имени общего объекта* должно быть именем объекта поддокумента SGML или объект данных, не относящихся к SGML. Если атрибут был определен в *начальном теге*, определение объекта должно применяться к каждому экземпляру документа, в котором встречается тег. Если атрибут представляет собой атрибут связи, определение должно применяться к типу документа - источника.

ПРИМЕЧАНИЕ. Нотация объекта данных, не относящегося к SGML, должна применяться к тем же типам документов, что и объект.

7.9.4.4 Нотация

Значение атрибута нотации должно быть *именем нотации*, которое было объявлено в том же объявлении типа документа, что и элемент.

Определение значения для атрибута нотации, если имеется явная ссылка на содержание, является ошибкой.

ПРИМЕЧАНИЕ. Поскольку *содержание* элемента будет пусто, бессмысленно определять для него нотацию. Даже если ссылка на содержание касается объекту данных, не относящегося к SGML, применимая нотация будет определяться параметром *имени нотации* объявления объекта.

7.9.4.5 Количества

Нормализованная длина *значения атрибута*, определенного непосредственно или интерпретируемого из *литерала значения атрибута*, является количеством "NORMSEP" плюс:

a) Для списка *ссылок на id, списка имен, списка маркеров имен, списка номеров*, или *списка маркеров номеров* (даже если в списке имеется только один маркер), сумма числа символов в каждом маркере списка плюс количество "NORMSEP" для каждого маркера в списке; или

b) для всех прочих, число символов в значении плюс количество "NORMSEP" для каждой ссылки на объект "CDATA" или "SDATA".

Нормализованная длина *значения атрибута* не может превышать количество "LITLEN" .

В единственном *начальном теге* общее число имен в *значении ссылки на id* и значениях атрибутов *списка ссылок на id*, по умолчанию или определенных, не может превышать количество "GRPCNT".

8 Команда обработки

[44] команда обработки = ***pio***, *системные данные,* ***pic***

[45] системные данные = *символьные данные*

Применение команд обработки не приветствуется, поскольку они ухудшают переносимость документа. Если команда обработки должна быть использована, она должна быть определена как объект, чтобы *системные данные* были помещены в *пролог*, где получатель документа может более легко обнаружить и модифицировать их.

Команда обработки, которая возвращает данные, должна быть определена как объект "SDATA" и введена ссылкой на объект. Команда обработки, которая не возвращает данные, должна быть определена как

объект "PI".

В системных данных не распознается никакая разметка кроме разграничителя, который их завершает.

ПРИМЕЧАНИЕ. Символы, допустимые в *системных данных* и их интерпретации, определяются системой. Если желательно допустить использование в *системных данных* символов, не относящихся к SGML, или символов разграничителя ***pic***, должен быть предусмотрен такой альтернативный путь их ввода, чтобы фактические символы не появлялись в документе.

8.1 Количества

Длина *команды обработки* за исключением ее разграничителей, не может превышать количество "PILEN".

9 Общие конструкции

9.1 Заменяемые символьные данные

[46] заменяемые символьные данные =

(*символ данных* | *ссылка на символ* |

ссылка на общий объект | ***Ee***)*

Разметка, которая завершает *заменяемые символьные данные*, не распознается в объекте, на который была ссылка из тех же *заменяемых символьных данных*.

Ee может появляться в *заменяемых символьных данных*, только если ссылка на объект, который она завершает, появилась в тех же *заменяемых символьных данных*.

9.2 Символьные данные

[47] символьные данные = *символ данных**

[48] символ данных = *символ SGML*

[49] символ = *символ SGML* | ***NONSGML***

ПРИМЕЧАНИЕ. Не относящийся к SGML символ может быть введен как символ данных в пределах объекта SGML с помощью *ссылки на символ*.

9.2.1 Символ SGML

[50] символ SGML = *символ разметки* | ***DATACHAR***

[51] символ разметки = *символ имени* |

символ функции | ***DELMCHAR***

[52] символ имени = *начальный символ имени* |

Digit | ***LCNMCHAR*** | ***UCNMCHAR***

[53] начальный символ имени =

LC Letter | ***UC Letter*** | ***LCNMSTRT*** | ***UCNMSTRT***

9.2.2 Символ функции

[54] символ функции= ***RE*** | ***RS*** | ***SPACE*** |

SEPCHAR | ***MSOCHAR*** | ***MSICHAR*** |

MSSCHAR | ***FUNCHAR***

9.3 Имя

[55] имя = *начальный символ имени, символ имени**

[56] номер = ***Digit*** +

[57] маркер имени = *символ имени*+

[58] маркер номера = ***Digit***, *символ имени**

Прописная форма каждого символа в *имени, маркере имени, номере* или *маркере номера*, как определено параметром "NAMECASE" *объявления SGML*, заменяется фактически введенным символом.

9.3.1 Количества

Длина *имени, маркера имени, номера* или *маркера номера* не может

превышать количество "NAMELEN".

9.4 Ссылки на объекты

Текст замены ссылки на объект должен удовлетворять синтаксическим и семантическим требованиям, которые управляют контекстом ссылки. В этом правиле ссылка на *объект поддокумента SGML* или *объект данных, не относящийся к SGML*, обрабатывается подобно ссылке на символ данных.

ПРИМЕЧАНИЕ. Такие объекты могут быть также доступны посредством *атрибута имени общего объекта*.

Ссылка на необъявленный объект является ошибкой, если только это не применимый объект по умолчанию (см. 9.4.4).

Ссылка на объект, на который уже была ссылка, и который еще не закончен, является недействительной (т.е., не могут существовать рекурсивные ссылки).

9.4.1 Количества

Число открытых объектов (за исключением бессылочного *объекта документа SGML*, с которого начинается *документ*), не может превышать количества "ENTLVL".

9.4.2 Пределы

Число открытых объектов поддокументов SGML не может превышать количества, определенного параметром "SUBDOC" *объявления SGML*.

Переменная	Символы	Номера	Описание
Digit	0 – 9	48 – 57	Цифры
Ee	(системный сигнал; не символ)		Сигнал конца объекта
LC Letter	a – z	97 – 122	Строчные буквы
Special	' ( ) + , - . / : = ?	39 – 41 43 –47 58 61 63	Специальные символы
UC Letter	A – Z	65 – 90	Прописные буквы

Рис. 1. Классы символов: абстрактный синтаксис

9.4.3 Запутывающие ссылки на объект

Не приветствуется любое использование ссылок на объекты, которые вносят неясность в разметку.

ПРИМЕЧАНИЕ. Большинство таких неправильных обращений запрещены синтаксисом SGML. Следует соблюдать следующие принципы (те, в которых используется термин "должен", являются повторением правил синтаксиса, формально заявленных в другом месте, которые усиливают принципы):

a) открывающий разграничитель тега, *команды обработки*, объявления, литерала, или другого разграниченного текста, должен находиться в том же объекте, что и закрывающий разграничитель. Объект не должен заканчиваться в разграниченном тексте, если только он не там не начинался, а объект, который там начинается, должен там и закончиться .

b) Содержание элемента или отмеченного раздела, который был объявлен как *символьные данные* или *заменяемые символьные данные*, должно (как следует и другим элементам) начинаться и заканчиваться в одном и том же объекте.

c) *Начальному тегу и конечному тегу* элемента (и *началу отмеченного раздела* и *концу отмеченного раздела*) следует находиться в одном и том же объекте, либо им следует быть текстом замены объектов, чьи ссылки находятся в том же объекте.

d) В объявлении разметки ссылка должна быть заменена нулем или большим числом последовательных полных параметров (с любыми включенными разделителями ps), или, в пределах группы, одним или большим числом последовательных полных маркеров (с любыми включенными разделителями и/или соединителями ts).

9.4.4 Поименованная ссылка на объект

[59] ссылка на общий объект = ***ero***,

спецификация типа документа, имя, конец ссылки

[60] ссылка на объект параметра = ***pero***,

спецификация типа документа, имя, конец ссылки

Имя объекта должно быть объявлено в *объявлении объекта* прежде, чем оно может использоваться в поименованной ссылке на объект, за исключением того, что необъявленное имя может использоваться в *ссылке на общий объект*, чтобы обратиться к объекту по умолчанию, если такой был определен.

9.4.5 Конец ссылки

[61] конец ссылки= (***refc*** | ***RE***)?

ПРИМЕЧАНИЕ. Завершение ссылки с помощью ***RE*** равносильно подавлению конца записи.

refc или ***RE*** могут быть пропущены, только если за ссылкой не следует *символ имени* или символ, который может интерпретироваться как пропущенный *конец ссылки*.

9.4.6 Краткая ссылка

Если *краткая ссылка* отображается на имя общего объекта в текущем отображении, она обрабатывается как разметка и заменяется поименованным объектом. Если *краткая ссылка* ни на что не отображается, каждый символ в строке разграничителя обрабатывается как разделитель s, если он может быть распознан как таковой, а в противном случае обрабатывается как данные.

9.4.6.1 Строка эквивалентной ссылки

Краткая ссылка может быть удалена из документа путем ее замены строкой эквивалентной ссылки, которая содержит поименованную

ссылку на объект. *Имя* объекта должно быть тем, на которое отображается *краткая ссылка* в текущем отображении.

Если краткая ссылка содержит какое-либо количество символов ***RS*** или ***RE***, строка эквивалентной ссылки будет включать одиночный символ ***RS*** или ***RE***, либо оба, как показано в следующем списке:

Краткая ссылка	*Строка эквивалентной ссылки*
Ни ***RS***, ни ***RE***	***ero****, имя,* ***refc***
RS; нет ***RE***	***RS***, ***ero****, имя,* ***refc***
RE; нет ***RS***	***ero****, имя,* ***RE***
И ***RS*** и ***RE***	***RS***, ***ero****, имя,* ***RE***

ПРИМЕЧАНИЯ:

1. Строки эквивалентной ссылки используются, когда документ конвертируется из конкретного синтаксиса, который поддерживает краткие ссылки, в синтаксис, который этого не делает.

2. Одиночные символы ***RS*** и/или ***RE*** сохраняются в строке эквивалентной ссылки для предотвращения объединения записей и возможного превышения ограничений системы на максимальную длину. Они не распознаются как данные: ***RS***, поскольку он никогда ими не является, а ***RE*** поскольку он служит *концом ссылки*.

Переменная	Символы	Номера	Описание
DATACHAR	(неявные)	(неявные)	Выделенные символы данных
DELMCHAR	(неявные)	(неявные)	Символы – разграничители
FUNCHAR	(нет)	(нет)	Символы инертных функций
LCNMCHAR	-.	45 46	Символы имен нижнего регистра
LCNMSTRT	(нет)	(нет)	Начальные символы имен нижнего регистра
MSICHAR	(нет)	(нет)	Символы включения разметки
MSOCHAR	(нет)	(нет)	Символы выключения разметки
MSSCHAR	(нет)	(нет)	Символы подавления разметки
RE		13	Символ конца записи
RS		10	Символ начала записи
SEPCHAR		9	Символы разделителей
SPACE		32	Пробел
UCNMCHAR	-.	45 46	Символы имени верхнего регистра
UCNMSTRT	(нет)	(нет)	Начальные символы имени верхнего регистра

Рис. 2. Классы символов: конкретный синтаксис

9.5 Символьная ссылка

[62] символьная ссылка = ***cro***, (*имя функции* | *номер символа*), *конец ссылки*

[63] имя функции = "***RE***" | "***RS***" | "SPACE" | *имя*

[64] номер символа = *номер*

Имя, указанное как *имя функции*, должно быть было определено как *добавленная функция* в конкретном синтаксисе.

Символьная ссылка должна использоваться в том случае, когда символ не может быть иным привычным образом введен в текст.

Считается, что символ замены находится в том же самом объекте, что и ссылка на него.

Символ замены обрабатывается так, как если бы он был введен непосредственно, за исключением того, что замена для *символьной ссылки* на цифру всегда обрабатывается как данные.

ПРИМЕЧАНИЯ:

1. Система может определять свое собственное внутреннее представление для символа замены. Следует позаботиться о том, чтобы отличить нормальный *символ функции* (введенный непосредственно или как замена для поименованной символьной ссылки), от того, который заменяет ссылку на цифровой символ.

2. Когда документ транслируется в набор символов другого документа, *номер символа* каждой ссылки на цифровой символ должен быть изменен на соответствующий *номер символа* нового набора.

9.6 Распознавание разграничителей

Строка разграничителя распознается как удовлетворяющая роли разграничителя только в рамках конкретного режима (или режимов)

распознавания, в котором роль является значимой и, в некоторых случаях, только если соблюдено контекстное ограничение. Роли, их режимы распознавания и любые контекстные ограничения на их распознавание приведены на рис. 3. На нем также показаны строки, назначенные общим разграничителям в наборе разграничителей ссылок, и номера символов этих строк в *наборе символов трансляции - ссылок* конкретного синтаксиса ссылок. Строки, назначенные как разграничители краткой ссылки в наборе разграничителей ссылок, показаны на рис. 4.

Имя	Строка	Номер	Режим	Ограничение	Описание роли
AND	&	38	GRP		Соединитель И
COM	--	45 45	CXT MD		Начало или конец комментария
CRO	&#	38 35	CON LIT	CREF	Символьная ссылка открыта
DSC	]	93	DS MD		Подмножество объявления закрыто
DSO	[	91	CXT MD		Подмножество объявления открыто
DTGC	]	93	GRP		Группа тегов данных закрыта
DTGO	[	91	GRP		Группа тегов данных открыта
ERO	&	38	CON LIT	NMS	Ссылка на объект открыта
ETAGO	< /	60 47	CON	GI	Конечный тег открыт
GRPC	)	41	GRP		Группа закрыта
GRPO	(	40	CXT GRP MD		Группа открыта
LIT	"	34	GRP LIT MD TAG		Начало или конец литерала
LITA	F	39	GRP LIT MD TAG		Начало или конец литерала (альтернативное)
MDC	>	62	CXT MD		Объявление разметки закрыто
MDO	< !	60 33	CON DS	DCL	Объявление разметки открыто
MINUS	-	45	MD		Минус; изъятие
MSC	]]	93 93	CON DS	MSE	Отмеченный раздел закрыт
NET	/	47	CON TAG	ELEM	Нулевой конечный тег
OPT	?	63	GRP		Индикатор факультативного появления
OR	|	124	GRP		Соединитель или
PERO	%	37	DS GRP MD	NMS	Ссылка на объект параметра открыта
PIC	>	62	PI		Команда обработки закрыта
PIO	< ?	60 63	CON DS		Команда обработки открыта
PLUS	+	43	GRP MD		Факультативный и повторяемый; включение
REFC	;	59	REF		Ссылка закрыта
REP	*	42	GRP		Факультативный и повторяемый
RNI	#	35	GRP MD		Индикатор зарезервированного имени
SEQ	,	44	GRP		Соединитель последовательности
SHORTREF			CON		Краткая ссылка (см. рис. 4)
STAGO	<	60	CON TAG	GI	Начальный тег открыт
TAGC	>	62	CXT TAG		Тег закрыт
VI		61	TAG		Индикатор значения

Рис.3. Набор разграничителей ссылок: общие данные

9.6.1 Режимы распознавания

Существуют следующие режимы распознавания:

Режим ***Значение***

CON Распознается в *содержании* или в *отмеченном разделе* объявлений отмеченных разделов, которые встречаются в *содержании*.

CXT Распознается как часть контекстной последовательности контекстных разграничителей в режимах "CON" или "DS". (См. ниже.)

DS Распознается в подмножестве объявления и в *отмеченном разделе* объявлений отмеченных разделов, которое встречается в подмножестве объявления.

GRP Распознается в группе.

LIT Распознается в литерале.

MD Распознается в *объявлении разметки* (ином нежели группа или подмножество объявления).

PI Распознается в *команде обработки*.

REF Распознается в *ссылке на общий объект*, *ссылке на объект параметра* или *ссылке на символ*.

TAG Распознается в *начальном теге* или *конечном теге*.

9.6.2 Контекстные ограничения

Наиболее общим ограничением является обязательное начало строки разграничителя контекстным разграничителем, в которой за ним следует одна из ниже перечисленных контекстных последовательностей:

CREF *начальное имя символа* или ***Digit***

DCL *начальное имя символа*, ***com***, ***dso*** или ***mdc***

GI *начальный символ имени* или, если в *объявлении SGML* определен параметр "SHORTTAG YES", ***tagc***, или если в *объявлении SGML* определен параметр "CONCUR YES", ***grpo***

MSE ***mdc***

NMS *начальный символ имени*, или если в *объявлении SGML* определен параметр "CONCUR YES", ***grpo***

Другим контекстным ограничением является следующее:

ELEM В режиме "CON", распознается только в пределах элемента, чей *начальный тег* был *начальным тегом, обеспечивающим net*; в режиме "TAG" без ограничений.

9.6.3 Порядок распознавания

Строки разграничителей (включая любые обязательные контекстные последовательности) распознаются в том порядке, в котором они встречаются, без перекрытия.

ПРИМЕЧАНИЕ. Например, если "abc" и "bed" являются строками разграничителей, а документ содержит "abcde", тогда будет распознаваться "abc", и анализ продолжится с "d", поэтому "bed" не будет распознаваться.

Это правило остается в силе, даже если распознаваемый разграничитель семантически некорректен или является краткой ссылкой, которая ни на что не отображается.

ПРИМЕЧАНИЕ. Например, в наборе разграничителей ссылок косая черта (/) будет распознаваться как часть контекстного разграничителя ***etago***, а не как *net*, даже если *конечный тег* GI не был объявлен или не является GI открытого элемента.

9.6.4 Разграничители, начинающиеся с одинаковых символов

Если несколько строк разграничителей начинаются с одинаковых символов, в данном месте документа будет распознаваться только наиболее длинная строка разграничителей или контекстный разграничитель.

ПРИМЕЧАНИЕ. Например, если "ab" и "abc" являются разграничителями, и документ содержит "abed", тогда будет распознаваться "abc", а анализ продолжится с "d", поэтому "ab" распознаваться не будет.

Это правило остается в силе, даже если наиболее длинный разграничитель семантически некорректен или является краткой ссылкой, которая ни на что не отображается.

ПРИМЕЧАНИЕ. Если в предыдущем примере "ab" и "abc" являются разграничителями краткой ссылки, будет распознаваться только краткая ссылка "abc", а краткая ссылка "ab" не будет, даже если краткая ссылка abc" ни на что не отображается в текущем отображении, а краткая ссылка "ab" отображается на объект.

9.6.5 Краткие ссылки с последовательностями пропусков

Если в определении разграничителя краткой ссылки имеется B последовательность, она вызовет распознавание последовательности пропусков в *содержании*. Минимальная длина последовательности пропусков является длиной B последовательности.

ПРИМЕЧАНИЕ. Так, один B означает один или большее число пропусков, два B означают два или большее число пропусков и т.д.

Строка, которая может распознаваться как несколько разграничителей, будет рассматриваться как строка наиболее специфического разграничителя из тех, которым она удовлетворяет.

ПРИМЕЧАНИЕ. Например, символ табуляции будет распознаваться

как "&#TAB:", а не как "B", а три пробела будут распознаваться как "BBB", а не как "BB".

9.6.5.1 Количества

Длина последовательности пропусков, распознаваемой как краткая ссылка, не может превышать количество "BSEQLEN". Если фактическая последовательность пропусков длиннее, только первые символы "BSEQLEN" будут включены в строку краткой ссылки, и синтаксический анализ возобновится со следующего символа.

9.6.6 Символы имени

Если *начальному символу имени* присвоена роль разграничителя, он будет распознаваться как разграничитель (вместо распознавания как *начальный символ имени*), если *маркер имени* еще не начался; если он найден в пределах *маркера имени*, он будет обрабатываться как *символ имени*.

Если параметром "NAMECASE" *объявления SGML* определена общая замена верхнего регистра, то с целью распознавания разграничителя, *символ имени*, которому присвоена роль разграничителя, обрабатывается, как если бы он был в форме верхнего регистра.

9.7 Подавление разметки

MSOCHAR подавляет распознавание разметки до появления ***MSICHAR*** или конца объекта. ***MSSCHAR*** подавляет распознавание разметки для следующего символа в том же объекте (если таковой существует).

ПРИМЕЧАНИЕ. ***MSOCHAR***, встречающийся в *символьных данных* или другом разграниченном тексте, подавит распознавание закрывающего разграничителя. ***MSSCHAR*** может так сделать, если он предшествует разграничителю.

Если распознавание разметки не было подавлено ***MSOCHAR***, ***MSICHAR*** не оказывает никакого воздействия на распознавание разметки, но не является ошибкой.

Если распознавание разметки было подавлено ***MSOCHAR***, последующий ***MSOCHAR*** или ***MSSCHAR*** не оказывает никакого воздействия на распознавание разметки, но не является ошибкой.

MSOCHAR, который следует за ***MSSCHAR***, не оказывает никакого воздействия на распознавание разметки.

9.8 Доступный объем

Размер и сложность документа не должны превышать число пунктов доступного объема, допустимое набором доступных объемов документа для объектов, встречающихся в документе.

Имена полного и индивидуального доступных объемов, вместе со значениями, присвоенными им в наборе доступных объемов ссылки, показаны на рис.5. Публичный идентификатор набора следующий:

ISO 8879 – 1986//CAPACITY Reference//EN

Строка	Номер	Описание
&#TAB;	9	Горизонтальная табуляция
&#***RE***;	13	Конец записи
&#RS;	10	Начало записи
&#***RS***;B	10 66	Начальные пропуски
&#***RS***;&#***RE***;	10 13	Пустая запись
&#***RS***;B&#***RE***;	10 66 13	Запись из пропусков
B&#***RE***;	66 13	Замыкающие пропуски
&#SPACE;	32	Пробел
BB	66 66	Два или более пропуска
“	34	Знак кавычек
#	35	Знак номера
%	37	Знак процента
/	39	Апостроф
(	40	Левая скобка
)	41	Правая скобка
*	42	Звездочка
+	43	Знак плюс
,	44	Запятая
-	45	Дефис
--	45 45	Два дефиса
:	58	Двоеточие
;	59	Точка с запятой
=	61	Знак равно
@	64	Коммерческое "at"
[	91	Левая квадратная скобка
]	93	Правая квадратная скобка
/\	94	Диакритический знак
_	95	Подчеркивание
{	123	Левая фигурная скобка
|	124	Вертикальная линия
}	125	Правая фигурная скобка
~	126	Тильда

Рис. 4. Набор разграничителей ссылок: краткие ссылки

Сумма пунктов для каждого типа объекта не может превышать значение индивидуального доступного объема для этого объекта, а сумма пунктов для всех объектов не может превышать значение "TOTALCAP".

Значения доступного объема должны быть достаточны для наибольшего требования доступного объема среди возможных наборов одновременных экземпляров или цепочек процессов связи, которые могут обрабатываться одновременно.

Пункты рассчитываются для объекта документа SGML и объектов

текста SGML, на которые он ссылается, плюс набор возможных открытых объектов поддокумента и объектов текста SGML, на которые они ссылаются, которые потребуют максимального доступного объема.

ПРИМЕЧАНИЕ. В качестве примера расчета доступного объема, когда используется конкретный синтаксис с 32 краткими ссылками и значением "NAMELEN" равным 8, потребуется доступный объем 30818 или более для размещения документа, состоящего из 100 объектов в среднем по 70 символов (800+7000), 200 типов элементов (1600) с 2000 маркеров в моделях содержания (16000) и 25 групп изъятий с общим числом имен 50 (200+400), 50 атрибутов со значениями по умолчанию в среднем по 20 символов (400+1000) и 100 маркерами имен атрибутов (800), 5 нотаций содержания данных с идентификаторами в среднем по 50 символов (40+250), 50 ID и 50 атрибутов IDREF (400+400), 5 отображений кратких ссылок (5x(8+(8x32)) =1320), и одного неявного объявления типа связи с 4 наборами связей, каждый из которых содержит 5 имен элементов – источников (40+168).

10 Объявления разметки: общие сведения

ПРИМЕЧАНИЕ. Имена объявлений и ключевые слова в синтаксических правилах представляют собой зарезервированные имена ссылок, которые могут быть переопределены для иного конкретного синтаксиса с параметром *использования зарезервированного имени объявления SGML*.

10.1 Части объявлений

10.1.1 Разделитель параметров

[65] ps == s | ***Ee*** | *ссылка на объект параметра* | *комментарий*

Ссылка на объект параметра может использоваться везде, где может встретиться параметр. Объект должен состоять из нуля или большего числа последовательных полных параметров, которые следуют за *ps*, в котором встречается ссылка, вместе с любыми включенными разделителями *ps*. Объект должен заканчиваться в пределах того же объявления.

Ee может возникать в *ps*, только если ссылка на объект, который он заканчивает, встречается в *ps* того же объявления.

Обязательный *ps*, который является соседним с разграничителем или другим *ps*, может быть опущен, если при этом не возникает неоднозначность.

ps должен начинаться с *s*, если его пропуск создает неоднозначность.

10.1.2 Литерал параметра

[66] литерал параметра = ***(lit,***

заменяемые данные параметра, ***lit)*** |

(lita*, заменяемые данные параметра,* ***lita)***

[67] заменяемые данные параметра =

(символ данных | *символьная ссылка* |

ссылка на объект параметра | ***Ee)****

Литерал параметра интерпретируется как параметр (или маркер) путем замены ссылок в ходе обработки объявление.

За исключением ссылок на объекты параметра и на символы в *литерале параметра* не распознается никакая разметка кроме оконечных ***lit*** или ***lita,*** а последние не распознаются в пределах текста замены ссылки.

ПРИМЕЧАНИЕ. Если литерал находится в параметре *текста объекта объявления объекта,* символы разметки в его тексте могут

распознаваться при ссылке на объект.

Ee может встречаться в *заменяемых данных параметра*, только если ссылка на объект, который он заканчивает, встречается в тех же *заменяемых данных параметра*.

10.1.2.1 Количества

Длина интерпретируемого *литерала параметра* не может превышать количества "LITLEN" (если только в контексте, в котором он используется, не применяется какое-либо иное ограничение).

10.1.3 Группа

[68] группа маркеров имен = ***grpo,*** *ts*, маркер имени,*

(ts, соединитель, ts*, маркер имени)*, ts**, ***grpc***

[69] группа имен = ***grpo****, ts*, имя,*

(ts, соединитель, ts*, имя)*, ts**, ***grpc***

[70] ts = *s* | ***Ee*** | *ссылка на объект параметра*

В одной *группе имен* или *группе маркеров имен* должен использоваться только один тип *соединителя.*

ПРИМЕЧАНИЕ. Никакой специальный тип *соединителя* не является обязательным, поэтому на группу, определенную в объекте, может ссылаться как *группа модели* (где специальный соединитель является значимым), так и *группа имен* (где это не так).

В одной *группе имен* или *группе маркеров имен* маркер может встречаться только один раз.

Ссылка на объект параметра может использоваться в любом месте группы, в которой может встретиться маркер. Объект должен состоять из одного или большего числа последовательных полных маркеров, которые следуют за *ts*, в котором ссылка встречается в той же группе (т. е., в том

же уровне вложения), вместе с любыми окружающими или включенными разделителями *ts* и любыми включенными соединителями. Объект должен заканчиваться в пределах той же группы.

Ee может встретиться в *ts*, только если:

— *ts* следует за маркером (в отличие от *соединителя,* ***grpo*** или ***dtgo),*** и

— ссылка на объект, который заканчивает ***Ee,*** встречается в той же группе (т.е. в том же уровне вложения).

10.1.3.1 Количества

Число маркеров в группе не может превышать количества "GRPCNT".

10.1.4 Разделитель объявлений

[71] ds = *s* | ***Ee*** | *ссылка на объект параметра* |
объявление комментария | *команда обработки* |
объявление отмеченного раздела

Ссылка на объект параметра в *ds* должна ссылаться на объект, который состоит из нуля или большего числа полных объявлений разметки и/или разделителей *ds*.

Ee может встретиться в *ds*, только если ссылка на объект, который он заканчивает, встречается в *ds* в том же параметре.

10.1.5 Тип связанного элемента

[72] тип связанного элемента =
родовой идентификатор | *группа имен*

Каждое *имя* в *группе имен* должно быть *родовым идентификатором.*

10.1.6 Внешний идентификатор

[73] внешний идентификатор =
("SYSTEM" | ("PUBLIC", *ps*+,
публичный идентификатор)),

(ps+, системный идентификатор)?

[74] публичный идентификатор = *минимальный литерал*

[75] системный идентификатор =

(lit, *системные данные,* ***lit)*** **|**

(lita, *системные данные,* ***lita)***

Системный идентификатор может быть пропущен, если система может сгенерировать его из *публичного идентификатора* и/или другой доступной ей информации.

Если в *объявлении SGML* определен параметр "FORMAL YES", *публичный идентификатор* интерпретируется как *формальный публичный идентификатор (*см. 10.2), и может возникнуть ошибка формального публичного идентификатора.

ПРИМЕЧАНИЕ. Он по-прежнему является *минимальным литералом,* и к нему применяются все требования, относящиеся к минимальным литералам.

10.1.6.1 Количества

Длина *системного идентификатора* без разграничителей не может превышать количество "LITLEN".

10.1.6.2 Доступные объемы

Число символов *текста объекта,* просчитанного в отношении доступного объема ENTCHCAP для *внешнего идентификатора,* представляет собой компонент его *системного идентификатора*, определенный или сгенерированный (за исключением разграничителей).

10.1.7 Минимальный литерал

[76] минимальный литерал **= *(lit,*** *минимальные данные,* ***lit)*** **|**

(lita, *минимальные данные,* ***lita)***

[77] минимальные данные = *символ минимальных данных**

[78] символ минимальных данных =

RS **|** ***RE*** **|** ***SPACE*** **|** **LC** ***Letter*** **|** ***UC Letter*** **|** ***Digit*** **|** ***Special***

Минимальный литерал интерпретируется путем игнорирования ***RS*** и заменой последовательности одного или большего числа символов ***RE*** и/или ***SPACE*** одиночным символом ***SPACE.***

10.1.7.1 Количества

Длина интерпретируемого *минимального литерала* без разграничителей не может превышать количества "LITLEN" в наборе количеств ссылок независимо от используемого конкретного синтаксиса.

10.2 Формальный публичный идентификатор

[79] формальный публичный идентификатор =

идентификатор владельца,

"//", текстовый идентификатор

Формальный публичный идентификатор не может содержать последовательные косые черты ("//"), если только это не разрешено явно в данном пункте.

ПРИМЕЧАНИЕ. Поскольку ***публичный идентификатор*** является *минимальным литералом,* символы ***RS*** удаляются, а последовательности из одного или большего числа символов ***RE*** и/или ***SPACE*** заменяются одиночным символом ***SPACE*** до интерпретации в качестве *формального публичного идентификатора.* Ограничение длины *минимального литерала* применяется к интерпретируемому тексту (см. 10.1.7.1).

10.2.1 Идентификатор владельца

[80] идентификатор владельца =

идентификатор владельца ИСО |

зарегистрированный идентификатор владельца |

незарегистрированный идентификатор владельца

ПРИМЕЧАНИЕ. При формулировании *идентификатора владельца* может быть полезен стандарт ИСО 3166.

10.2.1.1 Идентификатор владельца ИСО

[81] идентификатор владельца ИСО = *минимальные данные*

Обычная форма *идентификатора владельца ИСО* может использоваться, когда *публичный идентификатор* идентифицирует публикацию ИСО или назначается внутри него. Он состоит из номера публикации ИСО без суффикса языка.

ПРИМЕЧАНИЕ. Например, *идентификатором владельца ИСО* для публичного текста, определенного в настоящем документе, является "ISO 8879-1986" во всех переводах. Если публичный текст переводится, этот факт указывается определением соответствующего *языка публичного текста* в *текстовом идентификаторе.*

Специальная форма *идентификатора владельца ИСО* может использоваться, только когда публичный текст является зарегистрированным набором символов ИСО, а *классом публичного текста* является "CHARSET". Он состоит из строки "ISO Registration Number" (Регистрационный номер ИСО), за которой следует регистрационный номер набора символов.

10.2.1.2 Зарегистрированный идентификатор владельца

[82] зарегистрированный идентификатор владельца =

"+//", *минимальные данные*

ПРИМЕЧАНИЕ. Зарегистрированный идентификатор владельца может быть цитатой национального или отраслевого стандарта, либо

каким-либо иным уникальным идентификатором, назначенным в соответствии с ИСО 9070.

10.2.1.3 Незарегистрированный идентификатор владельца

[83] незарегистрированный идентификатор владельца =

"-//", *минимальные данные*

ПРИМЕЧАНИЕ. Незарегистрированный идентификатор владельца может быть обозначением (которое предполагается уникальным), созданным торговой организацией, иным сообществом пользователей или отдельным лицом.

10.2.2 Текстовый идентификатор

[84] текстовый идентификатор = *класс публичного текста*, ***SPACE***,

индикатор недоступного текста?,

описание публичного текста,

"//", (язык публичного текста |

последовательность, обозначающая публичный текст),

("//", версия отображения публичного текста)?

[85] индикатор недоступного текста = "-//"

Если имеется *индикатор недоступного текста*, текст является недоступным публичным текстом; в противном случае это доступный публичный текст.

Если *классом публичного текста* является "CHARSET", *текстовый идентификатор* включает *последовательность, обозначающую публичный текст;* в противном случае он включает *язык публичного текста.*

Текстовый идентификатор не может совпадать с другим *текстовым идентификатором* в *формальном публичном*

идентификаторе, который имеет тот же *идентификатор владельца.*

ПРИМЕЧАНИЕ. Если два публичных текста с одним владельцем имеют одинаковое описание публичного текста, они должны принадлежать разным классам, версиям и т.д.

10.2.2.1 Класс публичного текста

[86] класс публичного текста = *имя*

Именем должно быть одно из имен следующего списка, которое идентифицирует конструкцию SGML:

Имя	*Конструкция SGML*
CAPACITY	*набор доступных объемов*
CHARSET	*Символьные данные*
DOCUMENT	*Документ SGML*
DTD	*Подмножество объявления типа документа*
ELEMENTS	*набор элементов*
ENTITIES	*набор объектов*
LPD	*Подмножество объявления типа связи*
NONSGML	*объект данных, не относящихся к SGML*
NOTATION	*Символьные данные*
SHORTREF	*набор кратких ссылок*
SUBDOC	*объект поддокумента SGML*
SYNTAX	*Конкретный синтаксис*
TEXT	*объект текста SGML*

Имя должно вводиться прописными буквами.

ПРИМЕЧАНИЕ. Когда это уместно, система может использовать *класс публичного текста* для определения стратегии конвертации публичного текста из формы, в которой происходит обмен, в ссылочный

объект, который использует набор символов и конкретный синтаксис системы.

10.2.2.2 Описание публичного текста

[87] описание публичного текста =

описание текста ИСО | *минимальные данные*

Описание текста ИСО может использоваться, только когда *публичный идентификатор* идентифицирует публикацию ИСО. Он состоит из последнего элемента заглавия публикации без обозначения номера части (если таковое имеется).

ПРИМЕЧАНИЕ. Например, *описание текста ИСО* для ИСО 8632/4 – "Clear text encoding".

10.2.2.3 Язык публичного текста

[88] язык публичного текста = *имя*

Языком публичного текста должно быть двухсимвольное *имя,* введенное прописными буквами. *Имя* должно быть кодом языка из ИСО 639, который идентифицирует главный естественный язык, используемый в публичном тексте.

ПРИМЕЧАНИЯ:

1. Естественный язык влияет на возможность использования некоторых классов публичного текста по сравнению с другими.
2. К частям текста, которые с наибольшей вероятностью подвержены влиянию естественного языка, относятся данные, определенные имена и комментарии.
3. Система может использовать *язык публичного текста* для облегчения автоматического перевода.

10.2.2.4 Последовательность, обозначающая публичный текст

[89] последовательность, обозначающая публичный текст =

минимальный литерал

Минимальный литерал должен быть внешней формой обозначающей escape-последовательности, предписанной ИСО 2022 для набора символов, на который ссылается *публичный идентификатор.* Если публичный тест является зарегистрированным набором символов ИСО, обозначающая escape-последовательность должна быть зарегистрированной escape-последовательностью для данного набора.

ПРИМЕЧАНИЯ:

1. Например, внешней формой зарегистрированной обозначающей последовательности GO для графических символов ИСО 646 IRV (зарегистрированный набор символов 002) является:
ESC 2/8 4/0

2. Для зарегистрированных наборов символов *последовательность, обозначающая публичный текст,* уникально идентифицирует публичный текст. Для других наборов символов он уникально идентифицирует публичный текст безотносительно конкретного *идентификатора владельца.*

10.2.2.5 Версия отображения публичного текста

[90] версия отображения публичного текста = *минимальные данные*

Версия отображения публичного текста должна быть пропущена, если *классом публичного текста* является "CAPACITY", "CHARSET", "NOTATION" или "SYNTAX". Если публичный текст является аппаратно-независимым, для других классов текстовый идентификатор должен включать *версию отображения публичного текста,* которая описывает поддерживаемые устройства или используемую схему кодирования.

Когда система обращается к публичному тексту, для которого может быть определена, но не была определена *версия отображения публичного текста,* она может подставить наилучшую из доступных аппаратно-зависимых версий для используемого устройства отображения. Если такая версия отсутствует, подстановка не происходит.

ПРИМЕЧАНИЕ. Этот механизм часто используется с наборами символьных объектов.

10.3 Объявление комментария

[91] объявление комментария =

mdo, *(комментарий, (s | комментарий)*)?,* ***mdc***

[92] комментарий = ***com,*** *символ SGML*,* ***com***

В *комментарии* не распознается никакая разметка кроме разграничителя ***com,*** который его завершает.

10.4 Объявление отмеченного раздела

[93] объявление отмеченного раздела =

начало отмеченного раздела,

спецификация ключевого слова состояния, ***dso,***

отмеченный раздел, конец отмеченного раздела

[94] начало отмеченного раздела == ***mdo, dso***

[95] конец отмеченного раздела = ***msc, mdc***

[96] отмеченный раздел = *символ SGML**

Отмеченный раздел должен удовлетворять синтаксическим и семантическим требованиям, которые управляют контекстом, в котором встречается *объявление отмеченного раздела.*

Конец отмеченного раздела, который появляется за пределами *объявления отмеченного раздела,* является ошибкой.

10.4.1 Количества

Число открытых объявлений отмеченных разделов не может превышать количества "TAGLVL".

10.4.2 Спецификация ключевого слова состояния

[97] спецификация ключевого слова состояния = *(ps+,*
разрешенное ключевое слово состояния |
ключевое слово состояния | "TEMP")*,ps*

[98] разрешенное ключевое слово состояния =
квалификатор ключевого слова состояния,
ключевое слово состояния

[99] квалификатор ключевого слова состояния = *группа имен*

[100] ключевое слово состояния = "CDATA" | "IGNORE" |
"INCLUDE" | "RCDATA"

где

IGNORE определяет, что раздел обрабатывается так, как если бы в *отмеченном разделе* не было символов, за исключением того, что вложенное *объявление отмеченного раздела* распознается таким образом, что может быть обнаружено правильное окончание, однако его *спецификация ключевого слова состояния* игнорируется.

INCLUDE определяет, что *отмеченный раздел* не должен игнорироваться.

CDATA определяет, что *отмеченный раздел* обрабатывается как *символьные данные.*

RCDATA определяет, что *отмеченный раздел* обрабатывается как *заменяемые символьные данные.*

TEMP Идентифицирует раздел как временную часть документа, который позднее может быть необходимо удалить.

В случае конфликтующей спецификации ключевые слова состояний имеют следующий приоритет (в порядке убывания):

"IGNORE"

"CDATA"

"RCDATA"

"INCLUDE"

Если ни одно из них не указано, предполагается "INCLUDE".

Разрешенное ключевое слово состояния игнорируется, если только *группа имен* не включает имя типа активного документа или типа активной связи.

Разрешенное ключевое слово состояния может быть определено, только если в *объявлении SGML* определены параметры "CONCUR YES" или "EXPLICIT YES".

Если эффективным состоянием является "CDATA" или "RCDATA", вложенное *объявление отмеченного раздела* в пределах *отмеченного раздела* в том же объекте не допускается; *объявление отмеченного раздела* завершается первым *концом отмеченного раздела.*

Если эффективным состоянием является "IGNORE", ***Ее*** не допускается в *отмеченном разделе.*

ПРИМЕЧАНИЕ. Просмотр отмеченного раздела "IGNORE", "CDATA" или "RCDATA" виртуально игнорирует всю разметку кроме концов отмеченного раздела. В результате команды обработки, значения атрибутов, литералы, элементы символьных данных и комментарии таковыми не распознаются, поэтому их символы также просматриваются. Это может вызывать ошибочные результаты, если символы напоминают отмеченные разделы. В большинстве случаев проблем можно избежать, вводя такие символы со ссылками вместо непосредственного ввода.

10.5 Объявление объекта

[101] объявление объекта = ***mdo.*** "ENTITY", ps+,

*имя объекта, ps +, текст объекта, ps**, ***mdc***

10.5.1 Имя объекта

[102] имя объекта = *имя общего объекта* | *имя объекта параметра*

[103] имя общего объекта = *имя* | ***(rni,*** "DEFAULT")

[104] имя объекта параметра = ***pero****, ps+, имя*

где

DEFAULT означает, что объект является объектом по умолчанию.

Попытка переопределить объект игнорируется, но не является ошибкой.

ПРИМЕЧАНИЕ. Это требование позволяет объявлению объекта в подмножестве объявления типа документа иметь преимущество перед более поздним объявлением этого же объекта в определении типа публичного документа.

Наличие *ps* в *имени объекта параметра* обязательно, даже если он следует за разграничителем.

10.5.1.1 Количества

Имя в *имени объекта параметра* должно быть по крайней мере на один символ короче количества "NAMELEN".

10.5.1.2 Доступные объемы

Пункты считаются в отношении доступного объема "ENTCAP" для объекта по умолчанию, и для каждого уникального имени объекта, которое принято по умолчанию в одной или большем числе ссылок.

Пункты считаются в отношении доступного объема "ENTCHCAP"

для объекта по умолчанию. Они считаются для объектов по умолчанию, только если генерируется системный идентификатор.

10.5.2 Текст объекта

[105] текст объекта = *литерал параметра* | *текст данных* |
текст, заключенный в скобки |
спецификация внешнего объекта

Если только *литерал параметра* определен в качестве *текста объекта,* интерпретируемый *литерал параметра* является заменяемым текстом объекта.

Ee считается присутствующим в конце текста замены, он не вводится явно.

10.5.3 Текст данных

Текст данных обрабатывается при ссылке как *символьные данные* независимо от контекста, в котором встречается ссылка на объект. Он определяется как *литерал параметра,* чьи символы после разрешения ссылок обычным способом будут содержать *текст объекта.*

[106] текст данных = ("CDATA" | "SDATA" | "PI"), ps +,
литерал параметра

где

CDATA означает, что интерпретируемый *литерал параметра* является текстом замены объекта символьных данных.

SDATA означает, что интерпретируемый *литерал параметра* является текстом замены специального объекта символьных данных.

PI означает, что интерпретируемый *литерал параметра* является текстом замены объекта данных команды обработки.

"CDATA" или "SDATA" могут быть определены, только если *именем объекта* является *имя общего объекта.*

Команда обработки, которая возвращает данные, может быть определена как объект "SDATA".

ПРИМЕЧАНИЯ:

1. Ссылки на объекты "CDATA" or "SDATA" должны быть в контексте, в котором могут встретиться *символьные данные,* а ссылки на объекты "PI" должны быть в контексте, в котором может встретиться *команда обработки.*

2. "SDATA" обычно указывается, если объект должен быть переопределен для других приложений, систем или устройств вывода; например, если данные, содержащие команды обработки или символы, не присутствуют в *наборе символов трансляции - ссылок.*

10.5.4 Текст, заключенный в скобки

[107] текст, заключенный в скобки =

("STARTTAG" | "ENDTAG" | "MS" | "MD"), ps+,

литерал параметра

где ключевые слова означают, что объект состоит из интерпретируемого *литерала параметра,* заключенного между разграничителями следующим образом:

STARTTAG	означает, что предшествует ***stago***, а заключает ***tagc.***
ENDTAG	означает, что предшествует ***etago***, а заключает ***tagc.***
MS	означает, что предшествует *начало отмеченного раздела*, а заключает *конец отмеченного раздела.*
MD	означает, что предшествует ***mdo***, а заключает ***mdc.***

ПРИМЕЧАНИЕ. Текст, заключенный в скобки, является простым текстом с символами разграничителя; требования к объектам в

отношении формирования действительных начальных тегов или иных конструкций отсутствуют. Как обычно, действительность объекта определяется контекстом, в котором имеется ссылка на объект.

10.5.4.1 Количества

Длина интерпретируемого *литерала параметра* в *тексте, заключенном в скобки,* не может превышать количество "LITLEN" за вычетом длины ограничивающих разграничителей.

10.5.5 Спецификация внешнего объекта

[108] спецификация внешнего объекта =

внешний идентификатор, (ps +, тип объекта)?

[109] тип объекта = "SUBDOC" | ("NDATA", ps+, *имя нотации}*

где

SUBDOC означает, что объект является *объектом поддокумента SGML.*

NDATA означает, что объект является *объектом данных, не относящимся к SGML.*

Тип объекта может быть определен, только если *имя объекта* является *именем общего объекта.*

Если *тип объекта пропущен*, объект является *объектом текста SGML.*

Имя нотации должно быть объявлено в том же определении типа документа, что и объект.

ПРИМЕЧАНИЕ. Оно не должно быть определено до этого объявления, однако должно быть определено до ссылки на объект.

Объект данных, не относящийся к SGML, может ссылаться (в своей собственной нотации) на другие объекты данных, не относящиеся к

SGML, и объекты поддокументов SGML. Такие объекты должны быть объявлены в том же определении типа документа, что и первоначальный объект.

"SUBDOC" может быть определен, только если в *объявлении SGML* определен параметр "SUBDOC YES".

11 Объявления разметки: определение типа документа

11.1 Объявление типа документа

[110] объявление типа документа = ***mdo,***"DOCTYPE", ps+,
имя типа документа, (ps+, внешний идентификатор)?,
(ps+, **dso,** *подмножество объявления типа документа,* dsc)?,
ps*, ***mdc***

[111] имя типа документа = *родовой идентификатор*

[112] подмножество объявления типа документа =
*(набор объектов | набор элементов | набор кратких ссылок)**

[113] набор объектов = *(объявление объекта | ds)**

[114] набор элементов = *(объявление элемента |*
объявление списка определений атрибутов |
*объявление нотации | ds)**

[115] набор кратких ссылок == *(объявление объекта |*
объявление отображения кратких ссылок |
*объявление использования кратких ссылок | ds)**

Имя типа документа должно быть *родовым идентификатором*, который не встречается как *имя типа документа* или *имя типа связи* в том же п*рологе.*

Внешний идентификатор указывает на объект, на который имеется ссылка в конце подмножества объявления и который рассматривается

как его часть (или целое). Эффективное определение типа документа представляет собой комбинацию объявлений, введенных в подмножество, и внешних объявлений.

ПРИМЕЧАНИЕ. Объявление объекта параметра в подмножестве будет иметь приоритет перед другим объявлением для этого объекта во внешнем объекте, поскольку внешний объект анализируется позже.

Объявление типа документа должно включать *объявление элемента* для *имени типа документа.*

Набор кратких ссылок разрешен только в объявлении типа базового документа.

11.2 Объявление элемента

[116] объявление элемента = ***mdo,*** "ELEMENT", *ps+,*

тип элемента, {ps+, минимизация пропущенным тегом)?, ps+,

(объявленное содержание | модель содержания), ps,* ***mdc***

Порядок, в котором элементы и символы возникают в пределах элемента в экземпляре документа, должен соответствовать определению типа элемента, указанному в объявлении элемента.

Параметр минимизации пропущенным тегом и предшествующий ему ps могут быть пропущены, только если в *объявлении SGML* определен параметр " OMITTAG NO" .

11.2.1 Тип элемента

[117] тип элемента = *родовой идентификатор | группа имен |*

ранжированный элемент | ранжированная группа

В рамках определения типа документа *родовой идентификатор* может быть указан только один раз в параметре *типа элемента*, прямо или косвенно.

Если *тип элемента* является группой, члены группы определяются

в порядке появления их имен, а определение применяется к каждому из них.

Если *тип элемента* является *группой имен,* каждое имя является *родовым идентификатором.*

11.2.1.1 Ранжированный элемент

[118] ранжированный элемент = *база ранга, ps+, суффикс ранга*

[119] ранжированная группа = ***grpo,*** *ts*, база ранга, (ts* соединитель, ts*, база ранга)**, *ts**, ***grpc,*** ps+, *суффикс ранга*

[120] база ранга = *имя*

[121] суффикс ранга = *номер*

Родовой идентификатор, указанный *ранжированным элементом* или членом *ранжированной группы,* является *базой ранга* с добавленным *суффиксом ранга.*

11.2.1.2 Количества

Длина *родового идентификатора* не может превышать количества "NAMELEN".

11.2.2 Минимизация пропущенным тегом

[122] минимизация пропущенным тегом =
минимизация начальным тегом, ps +,
минимизация конечным тегом

[123] минимизация начальным тегом = "O" | ***minus***

[124] минимизация конечным тегом = "O" | ***minus***

где

O означает, что пропуск тега согласно условиям, определенным в 7.3.1, не является ошибкой разметки.

minus означает, что пропуск тега согласно условиям, определенным в 7.3.1, является ошибкой разметки.

minus должен быть указан для минимизации начальным тегом, если пропуск запрещен согласно 7.3.1.

"O" должен быть указан для минимизации конечным тегом, если элемент имеет объявленное значение "EMPTY".

ПРИМЕЧАНИЕ. Указание "0" служит напоминанием, что пустые элементы не имеют конечных тегов (хотя это не имеет ничего общего с минимизацией разметки).

11.2.3 Объявленное содержание

[125] объявленное содержание = "CDATA" | "RCDATA" | "EMPTY"

где

RCDATA означает, что *содержание* представляет собой *заменяемые символьные данные.*

CDATA означает, что *содержание* представляет собой *символьные данные*.

EMPTY означает, что *содержание* пусто.

11.2.4 Модель содержания

[126] модель содержания = *(группа модели* | "ANY"),
(ps+, *исключения)?*

[127] группа модели = ***grpo***, *ts*, маркер содержания, (ts*, соединитель, ts*, маркер содержания)*, ts*,* ***grpc,*** *индикатор появления?*

[128] маркер содержания = *примитивный маркер содержания* | *группа модели*

[129] примитивный маркер содержания = ***(rnl,*** "PCDATA") | *маркер элемента* | *группа тегов данных*

[130] маркер элемента = *родовой идентификатор, индикатор появления?*

где

ANY означает, что *содержание* представляет собой *смешанное содержание,* в котором допускаются *анализируемые символьные данные* и любые элементы, определенные в том же определении типа документа.

PCDATA означает, что допускаются *анализируемые символьные данные*.

ПРИМЕЧАНИЕ. *rni* отличает это ключевое слово от *маркера элемента* "PCDATA".

Если в *группе модели* присутствует "#PCDATA" или *группа тегов данных, содержанием* элемента является *смешанное содержание;* если это не так, то это *содержание элемента.* В любом случае элементы и символы данных *содержания* должны соответствовать *модели содержания* путем удовлетворения маркеров *группы модели* и *исключений* в следующем порядке приоритетов:

a) повторение последнего удовлетворенного маркера, если он имеет индикатор появления ***rep*** или ***plus***; или

b) некоторого другого маркера в *группе модели,* возможно модифицированного *исключениями* изъятий (*см.* 11.2.5);

или

c) маркера в группе *исключений* включений (см. 11.2.5.1).

ПРИМЕЧАНИЕ. Например, в экземпляре следующего элемента <!элемент e (a+ | b)+> последовательные элементы "a" будут удовлетворять повторения маркера элемента, а не повторения группы модели.

Все символы данных, встречающиеся между последовательными тегами, считаются удовлетворяющими единственный маркер

"#PCDATA", даже если некоторые из них были объявлены как символьные данные объявлением отмеченного раздела.

11.2.4.1 Соединитель

[131] соединитель = ***and*** | ***or*** | ***seq***

Если в *группе модели* имеется более одного *маркера соединителя,* порядок и выбор среди них соответствующего содержания определяется *соединителем* следующим образом:

seq	Все должны появиться в порядке ввода.
and	Все должны появиться в любом порядке.
or	Должен появиться один и только один.

В одной *группе модели* может встретиться только одни вид *соединителя* (однако вложенная в нее *группа модели* может содержать другой *соединитель).*

11.2.4.2 Индикатор появления

[132] индикатор появления = ***opt*** | **p*lus*** | ***rep***

Соответствующее содержание каждого выбранного *маркера содержания* должно встречаться однажды и только однажды, если только иное не обозначено *индикатором появления* маркера следующим образом:

opt	Факультативный (0 или 1 раз).
plus	Обязательный и повторяющийся (1 или большее число раз).
rep	Факультативный и повторяющийся (0 или большее число раз).

Маркер содержания "#PCDATA" считается имеющим *индикатор появления* ***rep***.

Собственно факультативный маркер обрабатывается с индикатором появления ***opt,*** если не определен никакой индикатор, или с индикатором появления ***rep,*** если указан ***plus***.

11.2.4.3 Неоднозначная модель содержания

Модель содержания не может быть неоднозначной; т.е. элемент или строка символов, которые возникают в экземпляре документа, должны быть способны удовлетворить только один *примитивный маркер содержания* без анализа правого контекста.

ПРИМЕЧАНИЕ. Например, модель содержания в <!элемент e ((a, b?), b)> является неоднозначной, поскольку после появления элемента "a", элемент "b" может удовлетворить любой из оставшихся маркеров. Неоднозначности можно избежать путем использования промежуточных элементов, как в:

<!элемент e (f, b)>

<!элемент f (a, b?)>

Здесь, маркер, удовлетворенный элементом "b", определяется однозначно тем, заканчивается ли элемент "f" до появления элемента "b". (Теоретические основы моделей содержания обсуждаются в приложении H.)

11.2.4.4 Группа тегов данных

[133] группа тегов данных = ***dtgo,*** *ts*, родовой идентификатор,*
*ts**, ***seq,*** *ts*, шаблон тега данных, ts**, ***dtgc***

[134] шаблон тега данных = (*группа образцов тегов данных* |
шаблон тега данных), (ts*, seq, ts*,
шаблон дополнения тега данных)?

[135] группа образцов тегов данных = ***grpo,*** *ts*,*
*шаблон тега данных, (ts**, ***or,*** *ts*,*
шаблон тега данных), ts**, ***grpc***

[136] шаблон тега данных = *литерал параметра*

[137] шаблон дополнения тега данных = *литерал параметра*

Группа тегов данных интерпретируется как группа ***seq*** с двумя маркерами: элемент GI, за которым следует "#PCDATA".

ПРИМЕЧАНИЕ. Например, с набором разграничителей ссылок *группа модели*

([часы, (":" | ":"). " "], минуты)

обрабатывается так, если бы она имела вид

((часы, #PCDATA), минуты)

Группа тегов данных может присутствовать только в *объявлении типа базового документа.*

Литерал параметра в *шаблоне тега данных* интерпретируется обычным образом, за исключением того, что цифровая символьная ссылка на символ, не относящийся к SGML, или *символ функции* запрещена.

11.2.4.5 Количества

Уровень вложенности моделей содержания не может превышать количества "GRPLVL".

Общая сумма маркеров всех уровней группы модели не может превышать количество "GRPGTCNT".

Длина интерпретируемого *литерала параметра* в *шаблоне тега данных* не может превышать количество "DTEMPLEN".

11.2.5 Исключения

[138] исключения = *(изъятия, (ps+, включения)?)* | *включения*

Исключения применяются в любом месте экземпляра документа, включая подэлементы, чье *содержание* является *смешанным содержанием* или *содержанием элемента.*

В любом месте экземпляра документа если элемент может быть как

включением, так и изъятием, он обрабатывается как изъятие.

11.2.5.1 Включения

[139] включения = ***plus****, группа имен*

Включения изменяют действия групп моделей, к которым они применяются, как показано в следующем примере: пусть "Q" – родовой идентификатор или группа в группе модели, "x" – его индикатор появления (или пустое место, если индикатора появления нет), а "R1" - "Rn" – применимые включения, тогда маркер

Qx

обрабатывается так, как если бы он имел вид

(R1 | R2 |...| Rn)*, (Q, (Rl | R2 |...| Rn)*)x

Считается, что элемент, который может удовлетворить маркер элемента в модели содержания, делает это, даже если этот элемент также является включением.

ПРИМЕЧАНИЯ:

1. Включения не должны использоваться для контекстных подэлементов. Они должны использоваться только для элементов, которые не являются логической частью содержания в точке, где они возникают в документе, такой как индексные входы или плавающие рисунки.

2. ***RE***, который следует за включением, обычно игнорируется, тогда как тот, который следует за надлежащим подэлементом, будет обрабатываться как данные (см. 7.6.1).

11.2.5.2 Изъятия

[140] изъятия = ***minus,*** *группа имен*

Изъятия изменяют воздействие групп моделей, к которым они применяются, путем запрета опций, которые были бы иначе доступны

(также, как если бы пользователь принял решение оставить факультативные элементы за пределами документа).

Если изъятие пытается изменять воздействие группы модели любым другим способом, это является ошибкой. В частности, ошибкой являются следующие случаи:

a) изъятие применяется к маркерам, не входящим во *включения,* имеющим индикатор появления ***opt*** или ***rep***, или являющимся членами групп ***or***; либо

b) изъятие пытается изменить обязательное или факультативное состояние маркера.

ПРИМЕЧАНИЕ. Например, запрещается исключать все члены обязательной *группы модели,* поскольку после этого группа перестанет быть обязательной.

11.3 Объявление списка определений атрибутов

[141] объявление списка определений атрибутов = ***mdo,***
"ATTLIST", *ps +, тип связанного элемента,*
*ps+, список определений атрибутов, ps**, ***mdc***

[142] список определений атрибутов = *определение атрибута,*
*(ps+, определение атрибута)**

[143] определение атрибута = *имя атрибута, ps+,*
объявленное значение, ps+, значение по умолчанию

Тип связанного элемента не может быть связан с другим списком определений атрибутов в том же подмножестве объявления, в котором этот список встречается.

11.3.1 Количества

Общее число имен атрибутов и маркеров имен в *списке определений атрибутов* не может превышать количество "ATTCNT".

11.3.2 Имя атрибута

[144] имя атрибута = *имя*

Имя атрибута может быть определено только однажды в том же *списке определений атрибутов.*

11.3.3 Объявленное значение

[145] объявленное значение = "CDATA" | "ENTITY" | "ID" | "IDREF" | "IDREFS" | "NAME" | "NAMES" | "NMTOKEN" | "NMTOKENS" | "NUMBER" | "NUMBERS" |"NUTOKEN" | "NUTOKENS" | *нотация* | *группа маркеров имен*

[146] нотация = "NOTATION", ps+, *группа имен*

где

CDATA означает, что *значением атрибута* являются *символьные данные.*

ENTITY означает, что *значением атрибута* является *имя общего объекта.*

ID означает, что *значением атрибута* является *значение id.*

IDREF означает, что *значением атрибута* является *значение ссылки на id.*

IDREFS означает, что *значением атрибута* является *список ссылок на id.*

NAME означает, что *значением атрибута* является *имя.*

NAMES означает, что *значением атрибута* является *список имен.*

NMTOKEN означает, что *значением атрибута* является *маркер имени.*

NMTOKENS означает, что *значением атрибута* является *список маркеров имен.*

NOTATION означает, что *значением атрибута* является *имя*

нотации, которое определяет нотацию содержания данных *содержания* элемента. *Группа имен* определяет допустимые имена нотаций.

NUMBER означает, что *значением атрибута* является *номер.*

NUMBERS означает, что *значением атрибута* является *список номеров.*

NUTOKEN означает, что *значением атрибута* является *маркер номера.*

NUTOKENS означает, что *значением атрибута* является *список маркеров номеров.*

"ID" и "NOTATION" каждый могут быть объявлены только однажды в *списке определений атрибутов.*

Маркер не может появляться более одного раза в *списке определений атрибутов,* даже в различных группах.

"NOTATION" не может быть объявлен для элемента, чьим *объявленным содержанием* является "EMPTY".

11.3.4 Значение по умолчанию

[147] значение по умолчанию = ((***rni,*** "FIXED", ps+)?,
спецификация значения атрибута) |
(***rni,*** ("REQUIRED" | "CURRENT" | "CONREF" | "IMPLIED"))

где

FIXED означает, что атрибут является фиксированным атрибутом.

REQUIRED означает, что атрибут является обязательным атрибутом.

CURRENT означает, что атрибут является текущим атрибутом.

CONREF означает, что атрибут является атрибутом ссылки на содержание.

IMPLIED означает, что атрибут является возможным атрибутом.

ПРИМЕЧАНИЕ. Указание пустого литерала не равнозначно указанию "IMPLIED".

Если в этом параметре определено значение атрибута, оно должно удовлетворять синтаксическим требованиям, определенным в 7.9.4.1.

ПРИМЕЧАНИЕ. Последующая проверка значений имени общего объекта и нотации выполняется в том случае, когда в спецификации атрибута используется значение по умолчанию.

Если объявленным значением является "ID", значением по умолчанию должно быть "IMPLIED" или "REQUIRED".

"CONREF" не может быть объявлено для элемента, чьим *объявленным значением* является "EMPTY".

11.3.4.1 Количества

Нормализованные длины "CONREF", "REQUIRED" и "IMPLIED" равны нулю.

11.3.4.2 Доступные объемы

При расчете требований "ATTCHCAP" значением по умолчанию текущего атрибута принимается длина самого длинного значения, указанного для атрибута в документе.

11.4 Объявление нотации

[148] объявление нотации = ***mdo,*** "NOTATION", *ps+*,

имя нотации, ps+, идентификатор нотации, ps,* ***mdc***

[149] идентификатор нотации = *внешний идентификатор*

ПРИМЕЧАНИЕ. *Идентификатор нотации* должен содержать достаточную информацию, которая позволит вызвать интерпретатор нотации с надлежащими параметрами.

Имя нотации не может быть определено в другом *объявлении нотации* в том же определении типа документа.

Если *идентификатор нотации* включает *публичный идентификатор,* а в *объявлении SGML* определен параметр "FORMAL YES", *классом публичного текста* должен быть "NOTATION".

11.5 Объявление отображения краткой ссылки

[150] объявление отображения краткой ссылки =

mdo, "SHORTREF", ps+, *имя отображения, (ps+, литерал параметра, ps+, имя)* +, ps*, ***mdc***

[151] имя отображения = *имя*

Имя отображения не может быть определено в другом *объявлении отображения краткой ссылки* в том же определении типа документа.

Интерпретируемый *литерал параметра* является разграничителем *краткой ссылки*, которая отображается на *имя* общего объекта, которое определено в том же определении типа документа.

ПРИМЕЧАНИЕ. Общий объект является обязательным, поскольку краткая ссылка будет заменена поименованной ссылкой на объект, если документ передается в систему, которая не поддерживает краткие ссылки, и ссылки на объекты параметра в *содержании* не разрешаются.

Разграничитель *краткой ссылки* может быть отображен только один раз в *объявлении отображения краткой ссылки.*

Если разграничитель *краткой ссылки* не определен, он рассматривается как ни на что не отображающийся.

11.6 Объявление использования краткой ссылки

[152] объявление использования краткой ссылки =

mdo, "USEMAP", ps+, *спецификация отображения, (ps+, тип связанного элемента*)?, ps*, ***mdc***

[153] спецификация отображения =

имя отображения | ***(rni,*** "EMPTY")

где

EMPTY означает, что отображение является пустым отображением.

11.6.1 Использование в объявлении типа документа

Если объявление возникает в *объявлении типа документа,* должен быть определен *тип связанного элемента*. Поименованное отображение становится текущим отображением всякий раз, когда элемент связанного типа становится текущим элементом.

Имя отображения должно быть определено в *объявлении отображения краткой ссылки* в том же *объявлении типа документа.*

ПРИМЕЧАНИЕ. Оно не должно быть определено до этого объявления, однако должно быть определено до того, как отображение становится текущим.

Указание типа связанного элемента, который уже сопоставлен с отображением, не является ошибкой, но игнорируется.

11.6.2 Использование в экземпляре документа

Если объявление встречается в экземпляре документа, *тип связанного элемента* не может быть определен. Отображение становится текущим отображением для этого экземпляра текущего элемента.

Имя отображения должно быть определено в *объявлении отображения краткой ссылки* в определении типа документа, которому соответствует экземпляр.

11.6.3 Текущее отображение

Отображение является текущим отображением, пока его связанный элемент является текущим элементом. Оно может уступать свой статус

для экземпляра элемента: либо временно подэлементу, становящемуся текущим элементом, либо постоянно *объявлением использования краткой ссылки*, встречающимся в экземпляре элемента.

Если тип элемента не имеет сопоставленного отображения кратких ссылок, текущим отображением для экземпляра элемента является отображение, которая является текущим при начале экземпляра. Если элемент является элементом документа, текущим отображением будет пустое отображение.

12 Объявления разметки: определение процессов связи

12.1 Определение типа связи

[154] определение типа связи = ***mdo,*** "LINKTYPE", ps+,
имя типа связи, (спецификация простой связи |
спецификация неявной связи | спецификация явной связи),
(ps+, внешний идентификатор)?, (ps+,
dso, *подмножество объявления типа связи, dsc)?, ps*,* ***mdc***

[155] имя типа связи = *имя*

Имя типа связи должно отличаться от любого другого *имени типа связи* или *имени типа документа* в том же *прологе.*

Внешний идентификатор указывает на объект, на который имеется ссылка в конце подмножества объявления и рассматривается как его часть (или целое). Эффективное определение процесса связи представляет собой комбинацию объявлений, введенных в подмножество, и внешних объявлений.

ПРИМЕЧАНИЕ. Объявление объекта параметра в подмножестве будет иметь приоритет перед другим объявлением для того объекта во внешнем объекте, поскольку внешний объект анализируется позже.

12.1.1 Спецификация простой связи

[156] спецификация простой связи =

rni, "SIMPLE", ***rni,*** "IMPLIED"

где

SIMPLE означает, что связь является простой связью.

IMPLIED означает, что *имя типа документа - результата* подразумевается приложением.

Если определена простая связь, в объявлении SGML должен быть определен параметр функций типа связи "SIMPLE YES".

Типом документа - источника является тип базового документа.

12.1.2 Спецификация неявной связи

[157] спецификация неявной связи =

имя типа документа – источника , ***rni,*** "IMPLIED"

где

IMPLIED означает, что *имя типа документа – результата* подразумевается приложением.

Если определена неявная связь, в объявлении SGML должен быть определен параметр функций типа связи "IMPLICIT YES".

Именем типа документа – источника должно быть имя типа базового документа.

12.1.3 Спецификация внешней связи

[158] спецификация внешней связи =

имя типа документа – источника,

имя типа документа – результата

[159] имя типа документа – источника = *имя типа документа*

[160] имя типа документа – результата = *имя типа документа*

Если определена внешняя связь, в объявлении SGML должен быть определен параметр *функций типа связи* "EXPLICIT YES".

Именем типа документа – источника должен быть тип базового

документа либо другой тип документа, который является типом документа – результата в цепочке процессов.

Каждое *имя типа документа* должно быть предварительно определено в *объявлении типа документа* в том же *прологе.*

12.1.3.1 Предельные значения

Число процессов связи в самой длинной цепочке не может превышать количество, указанное для "EXPLICIT" в параметре *функций типа связи объявления SGML.*

12.1.4 Подмножество объявления типа связи

[161] подмножество объявления типа связи =

набор атрибутов связи?, (объявление набора связей |

*объявление использования набора связей | набор объектов)**

[162] набор атрибутов связи =

*(объявление списка определений атрибутов | набор объектов)**

12.1.4.1 Объекты параметров

Объявления объектов в подмножестве объявления должны определять объекты параметров. Когда этот тип связи активен, объявления объектов обрабатываются, как если бы они встречались в конце подмножества объявления типа документа – источника .

Объявление типа связи может содержать ссылки на объект параметра в отношении объектов, определенных в объявлении типа документа - источника, также как в его собственном подмножестве объявления.

12.1.4.2 Атрибуты связи

Типом связанного элемента списка определений атрибутов должен быть тип элемента - источника.

Объявленное значение атрибута связи не может быть "ID", "IDREF", "IDREFS" или "NOTATION".

"CONREF" не может быть определен для атрибута связи.

12.1.4.3 Простая связь

Если объявление определяет простую связь, подмножество объявления должно состоять исключительно из *набора атрибутов связей*, который содержит не больше, чем одно *объявление списка определений атрибутов.* Список должен быть сопоставлен типу элемента базового документа, и может определять только фиксированные атрибуты.

12.2 Объявление набора связей

[163] объявление набора связей = ***mdo,*** "LlNK", ps+,

имя набора связей,

(ps+, спецификация элемента – источника,

ps+, спецификация элемента – результата)+, ps,* ***mdc***

[164] имя набора связей = *имя*

Имя набора связей не может быть определено в другом *объявлении набора связей* в том же определении типа связи.

Элемент - источник может быть привязан к данному *элементу - результату* или к "#IMPLIED" только один раз в наборе связей.

ПРИМЕЧАНИЕ. Т.е., данное сопоставление пары источник/результат должно быть уникальным для набора связей.

12.2.1 Спецификация элемента – источника

[165] спецификация элемента – источника =

тип связанного элемента,

спецификация атрибутов связи?

[166] спецификация атрибутов связей == ps+, ***dso***,

список спецификаций атрибутов, ps,* ***dsc***

Тип связанного элемента в *спецификации элемента- источника* должен быть определен в *объявлении типа документа - источника.*

Действительность *списка спецификаций атрибутов связи* определяется списком определений атрибутов, сопоставленным типу элемента - источника в подмножестве объявления типа связи. Все типы элементов, сопоставленные спецификации атрибута, должны быть сопоставлены тому же определению.

Спецификация атрибутов связи должна быть пропущена, если ее список спецификаций атрибутов пуст.

12.2.2 Спецификация элемента – результата

[167] спецификация элемента – результата =

(*rni,* "IMPLIED") | *(родовой идентификатор,*

спецификация атрибута результата?)

[168] спецификация атрибута результата =

ps+, ***dso****, список спецификаций атрибутов, ps*,* ***dsc***

где

IMPLIED означает, что *элемент – результат* подразумевается приложением.

Родовой идентификатор элемента – результата должен быть определен в *объявлении типа документа – результата.*

Действительность *списка спецификаций атрибутов* результата определяется *списком определений атрибутов,* сопоставленному элементу – результату в объявлении типа документа элемента – результата.

Спецификация атрибута результата должна быть пропущена, если его *список спецификаций атрибутов* пуст.

12.3 Объявление использования набора связей

[169] объявление использования набора связей =

mdo, "USELINK", *ps+*,

спецификация набора связей, ps+,

(тип связанного элемента | имя типа связи), ps,* ***mdc***

[170] спецификация набора связей =

имя набора связей | ***(rni***, "EMPTY")

где

EMPTY означает, что набор связей является пустым набором связей.

12.3.1 Использование в объявлении типа связи

Если объявление встречается в *объявлении типа связи,* должен быть определен *тип связанного элемента.* Поименованный набор связей становится текущим набором связей всякий раз, когда элемент связанного типа становится текущим элементом.

Имя набора связей должно быть определено в *объявлении набора связей* в том же *объявлении типа связи.*

ПРИМЕЧАНИЕ. Оно не должно быть определено до этого объявления, но должно быть определено до того, как набор связей становится текущим.

Типом связанного элемента должен быть тип элемента, определенный в *объявлении типа документа - источника.* Если он уже сопоставлен набору связей в этом *объявлении типа связи,* спецификация игнорируется, но не является ошибкой.

12.3.2 Использование в экземпляре документа

Если объявление возникает в экземпляре документа, должно быть определено *имя типа связи*. Набор связей становится текущим набором связей для этого образца текущего элемента.

Имя типа связи должно быть именем *объявления типа связи*, в котором определен набор связей.

12.3.3 Текущий набор связей

Набор связей является текущим набором связей, пока его элемент

является текущим элементом. Он может уступить статус для экземпляра элемента: либо временно подэлементу, становящимся текущим элементом, или постоянно *объявлением использования набора связей*, встречающимся в экземпляре элемента.

Если тип элемента не имеет сопоставленного набора связей, текущим набором связей для экземпляра элемента является набор связей, который был текущим при начале экземпляра. Если элемент является элементом документа, текущим набором связей - является пустой набор связей.

13 Объявление SGML

[171] объявление SGML = ***mdo,*** "SGML", *ps+*, "IS08879-1986",*ps+*,
набор символов документа, ps+,
набор доступных объемов, ps+,
область конкретного синтаксиса, ps+,
конкретный синтаксис, ps+, использование функций, ps+.
специфичная для приложения информация, ps,* ***mdc***

В *объявлении* SGML должен использоваться конкретный синтаксис ссылок независимо от конкретного синтаксиса, используемого в остальной части документа.

В параметрах и комментариях могут использоваться только символы разметки (в конкретном синтаксисе ссылок) и символы *минимальных данных*, хотя текст замены символьной ссылки может быть символом SGML, отличным от символа разметки или символа минимальных данных.

ПРИМЕЧАНИЯ:

1. *Объявление* SGML предназначено для использования человеком

(в печатной форме!) наряду с автоматизированной обработкой, поскольку оно дает получателю документа возможность определить, может ли система обрабатывать его "как есть", необходима ли трансляция символов или другое алгоритмическое преобразование (например, если использовались функции разметки документа или иной набор разграничителей), или необходимо ли преобразование, которое может потребовать ручного вмешательства (например, если использовались функции типа документа или иной набор количеств).

2. Ссылка на символ типа "Þ" действительна, поскольку ссылка состоит исключительно из разметки и символов минимальных данных, даже если это не касается текста замены.

3. Никакие ссылки на объект не могут появляться в объявлении SGML (поскольку не могут быть объявлены никакие объекты).

13.1 Набор символов документа

[172] набор символов документа = "CHARSET", *ps+*,

описание набора символов

Набор символов документа должен включать кодированное представление как единственную битовую комбинацию для каждого существенного символа SGML.

ПРИМЕЧАНИЕ. Если документ использует два конкретных синтаксиса, символы разметки обоих являются существенными символами SGML.

Как часть трансляции документа в новый набор символов должны быть изменены номера символов в этом параметре и любых ссылках на цифровой символ в документе.

ПРИМЕЧАНИЕ. Признается, что получатель документа должен быть способен транслировать его в свой системный набор символов, прежде чем документ сможет быть обработан машиной. Существуют два основных подхода к передаче этой информации:

a) Если набор символов является стандартным, зарегистрированным или иным образом способным вызываться по ссылке на идентифицирующее название или номер, этот идентификатор может быть передан получателю документа. Сообщение обязательно должно произойти вне документа; например, в поле потока данных обмена документа, или через другой (возможно не электронный) носитель.

b) Для других наборов символов достаточную информацию обеспечит удобочитаемая для человека копия объявления SGML.

13.1.1 Описание набора символов

[173] описание набор символов = (*основной набор символов, ps+,*
часть описываемого набора символов) +

Части описываемого набора символов должны вместе описывать каждый номер символа в описываемом наборе символов один и только один раз.

13.1.1.1 Основной набор символов

[174] основной набор символов = "BASESET", *ps+,*
публичный идентификатор

Публичный идентификатор является удобочитаемым для человека идентификатором *основного набора символов.*

ПРИМЕЧАНИЕ. Например, стандартное или зарегистрированное имя или номер или другое обозначение, которое будет понято ожидаемыми получателями документа.

Если в параметре *других функций* определено значение "FORMAL YES", *публичный идентификатор* должен быть *формальным публичным идентификатором* с *классом публичного текста* "CHARSET".

13.1.1.2 Часть описываемого набора символов

[175] часть описываемого набора символов =
"DESCSET", (*ps+*, *описание символа*)+

[176] описание символа = *номер описываемого набора символов, ps+, число символов, ps+, (номер основного набора символов | минимальный литерал* | "UNUSED")

[177] номер описываемого набора символов = *номер символа*

[178] номер основного набора символов = *номер символа*

[179] число символов = *номер*

где

UNUSED означает, что указанным номерам символов описываемого набора не присвоено значение.

Указанному *числу символов* в описываемом наборе символов, начинающихся с указанного *номера описываемого набора символов,* присваиваются значения следующим образом:

a) Если определен *номер основного набора символов*, значениями являются значения соответствующих символов в *основном наборе символов,* начинающихся с указанного *номера основного набора символов.*

ПРИМЕЧАНИЕ. Если номер основного набора символов не используется, не соответствующему номеру описываемого набора символов не присваивается никакое значение.

b) Если определен *минимальный литерал*, значение или значения описываются литералом.

ПРИМЕЧАНИЕ. *Минимальный литерал* должен быть определен, только если ни один символ в *основном наборе символов* не имеет необходимое значение.

c) Если определен "UNUSED", не присваиваются никакие значения.

13.1.2 Идентификация символов, не относящихся к SGML

Каждый *номер символа*, которому не присвоено никакое значение *описанием набора символов*, добавляется к ***NONSGML,*** идентифицируя таким образом его как не относящийся к SGML символ.

ПРИМЕЧАНИЕ. После получения и трансляции документа, не относящиеся к SGML символы могут быть измениться, поскольку набор символов нового документа может отображать управляющие символы на другие кодированные представления.

Избегаемый символ должен быть идентифицирован как не относящийся к SGML символ, если только это не существенный символ SGML.

ПРИМЕЧАНИЯ:

1. Например, на рис. 8, символы с номерами 9, 10 и 13, которые являются избегаемыми символами, однако не обозначены как не относящиеся к SGML символы, поскольку они являются символами функций.
2. Если документ использует два конкретных синтаксиса, избегаемые символы обоих подчиняются этому требованию.

13.2 Набор доступных объемов

[180] набор доступных объемов = "CAPACITY", *ps+,*

(("PUBLIC", *ps+, публичный идентификатор)* |

("SGMLREF", (*ps+, имя, ps+, номер*) +))

Указанное *имя* является именем, присвоенным доступному объему на рис.5. Доступному объему присваивается значение, обозначенное указанным *номером*.

Значение набора доступных объемов ссылки используется для любого доступного объема, для которого этим параметром не назначена замена.

ПРИМЕЧАНИЕ. Ключевое слово "SGMLREF", которое обязательно (и поэтому избыточно), когда не используется публичный идентификатор, является напоминанием об этом правиле для читателей объявления SGML.

Значения доступного объема должны выражать пределы, которые не превышаются документом. Они должны быть достаточны для наибольшего требования доступного объема среди возможных наборов параллельных экземпляров или цепочек процессов связи, которые могут обрабатываться одновременно.

Значение, назначенное параметру "TOTALCAP", должно быть равно или превышать наибольший индивидуальный доступный объем.

Если в параметре *других функций* определено значение "FORMAL YES", *публичный идентификатор* должен быть *формальным публичным идентификатором* с *классом публичного текста* "CAPACITY".

13.3 Область конкретного синтаксиса

Этот параметр определяет, должен ли объявленный конкретный синтаксис использоваться для всего документа, или может ли конкретный синтаксис ссылок использоваться в прологах.

[181] область конкретного синтаксиса = "SCOPE", *ps+*,

("DOCUMENT" | "INSTANCE")

где

DOCUMENT означает, что объявленный конкретный синтаксис используется во всем документе.

INSTANCE означает, что конкретный синтаксис ссылок используется в прологах, а объявленный конкретный синтаксис используется в наборах экземпляров документов.

Если определен "INSTANCE", объявленный конкретный синтаксис должен выполнять следующие требования:

a) набор символов синтаксиса - ссылок должен быть тем же, что и набор символов конкретного синтаксиса ссылок;

b) существенные символы SGML должны быть такими, чтобы начало набора экземпляров документа всегда было различимо с концом его пролога; и

c) значения набора количеств должны равняться или превышать значения набора количеств ссылок.

Имя	Значение	Пункты	*Объекты, для которых считаются пункты доступного объема*
TOTALCAP	**35000**	**(всего)**	Суммарное общее количество пунктов индивидуальных доступных объемов.
ENTCAP	**35000**	**NAMELEN**	Определенный объект.
ENTCHCAP	**35000**	**1**	Символ текста объекта
ELEMCAP	**35000**	**NAMELEN**	Определенный элемент
GRPCAP	**35000**	**NAMELEN**	Маркер любого уровня модели содержания.
EXGRPCAP	**35000**	**NAMELEN**	Группа исключений изъятий или включений.
EXNMCAP	**35000**	**NAMELEN**	Имя в группе исключений изъятий или включений.
ATTCAP	**35000**	**NAMELEN**	Определенный атрибут.
ATTCHCAP	**35000**	**1**	Символ значения атрибута по умолчанию (ключевое слово считается как нуль).
AVGRPCAP	**35000**	**NAMELEN**	Маркер, определенный в группе имен значений атрибутов или группе маркеров имен.
NOTCAP	**35000**	**NAMELEN**	Определенная нотация содержания данных.
NOTCHCAP	**35000**	**1**	Символ в идентификаторе нотации.
IDCAP	**35000**	**NAMELEN**	Указанный атрибут ID (явно или по умолчанию).
IDREFCAP	**35000**	**NAMELEN**	Указанный атрибут IDREF (явно или по умолчанию).
MAPCAP	**35000**	**NAMELEN**	(плюс NAMELEN для каждого разграничителя краткой ссылки в конкретном синтаксисе) Объявленное отображение краткой ссылки.
LKSETCAP	**35000**	**NAMELEN**	Определенные типы связей или наборы связей.
LKNMCAP	**35000**	**NAMELEN**	Тип документа или элемент в типе связи или объявлении набора связей.

Рис. 5. Набор доступных объемов ссылок

13.4 Конкретный синтаксис

[182] конкретный синтаксис =

"SYNTAX", *ps+*, (*публичный конкретный синтаксис* |
(*идентификация номера избегаемого символа, ps+,*
набор символов синтаксиса - ссылок, ps+,
идентификация символа функции, ps+,
правила наименования, ps+, набор разграничителей, ps+,
использование зарезервированного имени, ps+,
набор количеств))

Должны использоваться конкретный синтаксис ссылок или конкретный синтаксис ядра, если только не требуется иной конкретный синтаксис в соответствии с такими требованиями, как возможности клавиатуры, дисплея или характеристики национального языка.

13.4.1 Публичный конкретный синтаксис

[183] публичный конкретный синтаксис = "PUBLIC", *ps+*,
публичный идентификатор, (ps+, "SWITCHES", (*ps+*,
номер символа, ps+, номер символа) +)?

где

SWITCHES означает, что символы разметки в указанном конкретном синтаксисе были переключены.

Пары номеров символов находятся в *наборе символов синтаксиса - ссылок публичного конкретного синтаксиса.* Первый из каждой пары - символ разметки в идентифицированном конкретном синтаксисе, а второй - символ, который подставляется вместо него в каждом экземпляре, в котором использовался первый символ.

ПРИМЕЧАНИЕ. Конкретный синтаксис, который возникает в результате переключений, должен выполнять все обычные требования так же, как если бы он был объявлен явно.

Если в параметре *других функций* определено значение "FORMAL YES", *публичный идентификатор* должен быть *формальным публичным идентификатором* с *классом публичного текста* "SYNTAX".

13.4.2 Идентификация номера избегаемого символа

[184] идентификация номера избегаемого символа =
"SHUNCHAR", *ps+*,
("NONE" | (("CONTROLS" | *номер символа*),
(*ps+, номер символа*)*))

где

NONE означает, что номера избегаемых символов отсутствуют.

CONTROLS означает, что любой номер символа, который набор

символов документа рассматривает как кодированное представление управляющего символа, а не графического символа, является избегаемым символом.

Каждый указанный *номер символа* идентифицируется как номер избегаемого символа.

ПРИМЕЧАНИЕ. Номера символов в этом параметре не должны (и не могут) изменяться, когда документ транслируется в другой набор символов.

13.4.3 Набор символов синтаксиса – ссылок

[185] набор символов синтаксиса – ссылок = *описание набора символов*

Набор символов синтаксиса - ссылок должен включать кодированное представление каждого существенного символа SGML как единственную битовую комбинацию.

13.4.4 Идентификация символа функции

[186] идентификация символа функции =

"FUNCTION", *ps+*, "RE", *ps+*,

номер символа, ps+, " RS", *ps+*,

номер символа, ps+, "SPACE", ps+,

номер символа, (ps+, добавленная функция, ps+,

*класс функции, ps+, номер символа)**

[187] добавленная функция = *имя*

[188] класс функции = "FUNCHAR" | "MSICHAR" |

"MSOCHAR" | "MSSCHAR" | "SEPCHAR"

где ключевые слова идентифицируют *добавленную функцию* следующим образом:

FUNCHAR означает символ инертной функции.

SEPCHAR означает символ разделителя.

MSOCHAR означает символ выключения разметки.

MSICHAR означает символ включения разметки.

MSSCHAR означает символ подавления разметки.

Символ с указанным *номером символа* в *наборе символов синтаксиса - ссылок* присваивается функции.

Символ может быть присвоен только одной функции.

Добавленная функция не может быть "RE", "RS", "SPACE" или другой *добавленной функцией.*

"MSICHAR" должен быть определен по крайней мере для одной *добавленной функции*, если "MSOCHAR" определен для *добавленной функции.*

ПРИМЕЧАНИЕ. Когда используется расширение кода, функциям подавления разметки могут быть присвоены символы смещения с целью избежать ложного распознавания разграничителей, но только жертвуя возможностью использовать ссылки на объект для достижения аппаратной независимости (см. E.3).

13.4.5 Правила наименования

[189] правила наименования = "NAMING", *ps+*,
"LCNMSTRT", ps+, *литерал параметра, ps+,*
"UCNMSTRT", *ps+*, *литерал параметра, ps+,*
"LCNMCHAR", *ps+*, *литерал параметра, ps+,*
"UCNMCHAR", *ps+*, *литерал параметра, ps+,*
"NAMECASE", *ps+*, "GENERAL", *ps+*, ("NO" | "YES"), *ps+*,
"ENTITY", *ps+*, ("NO" | "YES")

где

LCNMSTRT означает, что каждый *символ* в литерале (если имеется) добавляется к ***LCNMSTRT.***

UCNMSTRT каждый *символ* в литерале (если имеется) добавляется к ***UCNMSTRT*** как связанная форма верхнего регистра символа в соответствующей позиции ***LCNMSTRT.***

LCNMCHAR означает, что каждый *символ* в литерале (если имеется) добавляется к ***LCNMCHAR.***

UCNMCHAR каждый *символ* в литерале (если имеется) добавляется к ***UCNMCHAR*** как связанная форма верхнего регистра символа в соответствующей позиции ***LCNMCHAR.***

NAMECASE определяет, должна ли выполняться подстановка верхнего регистра для объектных ссылок и имен объектов ("ENTITY") и/или для всех других имен, маркеров имен, маркеров номеров и строк разграничителей ("GENERAL").

YES означает, что ***LC Letter*** будет заменяться соответствующей ***UC Letter***, а *символ* в ***LCNMSTRT*** или ***LCNMCHAR*** будет заменяться соответствующей формой верхнего регистра.

NO означает, что подстановка верхнего регистра выполняться не будет.

Форма верхнего регистра символа имени может быть той же, что и форма нижнего регистра.

Символ, присвоенный ***LCNMCHAR, UCNMCHAR, LCNMSTRT,*** или ***UCNMSTRT,*** не может быть ***LC Letter, UC Letter, Digit, RE, RS, SPACE,*** или ***SEPCHAR.***

Символ, присвоенный ***LCNMCHAR*** или ***UCNMCHAR,*** *не может* быть присвоен ***LCNMSTRT*** или ***UCNMSTRT.***

UCNMCHAR должен иметь то же число символов что и

LCNMCHAR, UCNMSTRT должен иметь то же число символов что и ***LCNMSTRT.***

13.4.6 Набор разграничителей

[190] набор разграничителей = "DELIM", *ps+*,

общие разграничители, ps+, разграничители краткой ссылки

13.4.6.1 Общие разграничители

Разграничитель или контекстный разграничитель должны отличиться от любого другого разграничителя и контекстного разграничителя, которые могут распознаваться в том же режиме.

Использование *начального символа имени* или ***Digit*** в строке разграничителя не приветствуется.

[191] общие разграничители = "GENERAL", *ps+*, "SGMLREF",

(*ps+*, *имя, ps+, литерал параметра)**

Указанное имя является именем, присвоенным роли общего разграничителя на рис. 3. Роли присваивается интерпретируемый литерал параметра.

Роли общих разграничителей, не назначенные этим параметром, назначаются как в наборе разграничителей ссылок.

ПРИМЕЧАНИЕ. Ключевое слово "SGMLREF", которое является обязательным (и поэтому избыточно), служит напоминанием об этом правиле для читателей объявления SGML.

Общая строка разграничителя не может состоять исключительно из символов функций. Общая строка разграничителя, которая содержит такие символы в комбинации с другими, разрешается, но не приветствуется.

13.4.6.2 Разграничители кратких ссылок

[192] разграничители кратких ссылок = "SHORTREF", *ps+*,

("SGMLREF" | "NONE"), (*ps+, литерал параметра)**

где

SGMLREF означает, что краткие ссылки, назначенные набором разграничителей ссылок, включены в этот набор разграничителей.

NONE означает, что помимо разграничителей *кратких ссылок,* назначенных этим параметром, .другие не назначены.

Интерпретируемый литерал параметра назначается как строка разграничителей краткой ссылки.

Литерал параметра может иметь единственную В последовательность, которой не может предшествовать или которая не может завершаться последовательностью пропусков или ссылкой на символ, который может появиться в последовательности пропусков.

Строка *краткой ссылки* длиннее одного символа не приветствуется, если только эта строка не является обычным условным соглашением ввода или кодовой последовательностью.

Строка *краткой ссылки* не приветствуется, если:

a) она содержит полностью или начальную часть разграничителя или контекстного разграничителя, который распознается в режиме CON; и

b) она может создать впечатление, что разграничитель был ошибочно проигнорирован.

ПРИМЕЧАНИЕ. При применении этого требования следует помнить, что краткая ссылка распознается как разграничитель, даже когда она не отображается на объект. Поэтому, общий разграничитель внутри нее никогда не будет распознаваться как таковой.

13.4.7 Использование зарезервированного имени

[193] использование зарезервированного имени = "NAMES", *ps+*,
"SGMLREF", (*ps+*, *имя, ps+*, *имя*)*

Первое из каждой пары имен является зарезервированным именем ссылки, а второе - именем, которое должно заменить его в объявленном конкретном синтаксисе.

ПРИМЕЧАНИЕ. Зарезервированные имена, которые возникают только в *объявлении SGML,* включая роль разграничителя, количество, и имена доступного объема, не могут быть заменены, поскольку *объявление SGML* всегда находится в конкретном синтаксисе ссылок.

Зарезервированное имя ссылки используется для любого зарезервированного имени, для которого этим параметром не назначена замена.

ПРИМЕЧАНИЕ. Ключевое слово "SGMLREF", которое обязательно (и поэтому избыточно), является напоминанием об этом правиле для читателей объявления SGML.

Заменой для зарезервированного имени ссылки не может быть другое зарезервированное имя ссылки, или замена для него.

13.4.8 Набор количеств

[194] набор количеств = "QUANTITY", *ps+*,
"SGMLREF", (*ps+*, *имя, ps+*, *имя*)*

Указанное *имя* является именем, данным количеству на рис. 6, на котором также показаны присвоения значений, которые составляют набор количеств ссылки. Обозначенному количеству присваивается значение, определенное указанным *номером.*

Значение набора количества ссылки используется для любого количества, для которого этим параметром не назначена замена.

Имя	**Значение**	***Описание количества***
ATTCNT	**40**	Число имен атрибутов и маркеров имен в *определениях атрибутов* элемента.
ATTSPLEN	**960**	Нормализованная длина *спецификаций атрибутов* начального тега.
BSEQLEN	**960**	Длина последовательности пропусков в строке краткой ссылки.
DTAGLEN	**16**	Длина тега данных.
DTEMPLEN	**16**	Длина шаблона тега данных или шаблона образца (не разграничен).
ENTLVL	**16**	Уровень вложенности объектов (кроме первичных).
GRPCNT	**32**	Число маркеров в группе.
GRPGTCNT	**96**	Суммарное число маркеров всех уровней группы модели.
GRPLVL	**16**	Уровень вложенности групп модели (включая первый уровень).
LITLEN	**240**	Длина литерала или разграниченного *значения атрибута* (не разграничен).
NAMELEN	**8**	Длина *имени, маркера имени, номера,* и т.д..
NORMSEP	**2**	Используется вместо учета разделителей при расчете нормализованных длин.
PILEN	**240**	Длина *команда обработки* (не разграничен).
TAGLEN	**960**	Длина *начального тега* (не разграничен).
TAGLVL	**24**	Уровень вложенности открытых элементов.

Рис. 6. Набор количеств ссылки

ПРИМЕЧАНИЕ. Ключевое слово "SGMLRRF", которое обязательно (и поэтому избыточно), является напоминанием об этом правиле читателю объявления SGML.

13.5 Использование функции

[195] использование функции = "FEATURES", *ps+*,

функции минимизации разметки, ps+,

функции типа связи, ps+, другие функции

13.5.1 Функции минимизации разметки

[196] функции минимизации разметки = "MINIMIZE" *ps+*,

"DATATAG", *ps+*,("NO" | "YES"), *ps+*,

"OMITTAG", *ps+*, ("NO" | "YES"), *ps+*,

"RANK", *ps+*, ("NO" | "YES"), *ps+*,

"SHORTTAG", *ps*+, ("NO" | "YES")

где

NO	означает, что функция не используется.
YES	означает, что функция используется.
DATATAG	означает, что символы данных могут одновременно служить в качестве тегов.
OMITTAG	означает, что некоторые теги могут быть пропущены все вместе.
RANK	означает, что ранги элементов могут быть опущены из тегов.
SHORTTAG	означает, что могут использоваться краткие теги с пропущенными разграничителями, спецификациями атрибутов или родовыми идентификаторами.

ПРИМЕЧАНИЕ. Этим параметром не определяется использование кратких ссылок, поскольку оно определено параметром "SHORTREF".

13.5.2 Функции типа связи

[197] функции типа связи == "LINK", *ps*+,
"SIMPLE", *ps*+, ("NO" | "YES"), *ps*+,
"IMPLICIT", *ps*+, ("NO" | "YES"), *ps*+,
"EXPLICIT", *ps*+, ("NO" | ("YES", *ps*+, номер))

где

NO	означает, что функция не используется.
YES	означает, что функция используется.
EXPLICIT	означает, что могут использоваться определения явных процессов связи, а наиболее длинная цепочка процессов связи имеет указанное число связей (1 или более).

IMPLICIT означает, что могут использоваться определения неявных процессов связи.

SIMPLE означает, что могут использоваться определения простых процессов связи.

13.5.3 Другие функции

[198] другие функции = "OTHER", *ps*+, "CONCUR", *ps*+,
("NO" | ("YES", *ps*+, *номер*)), *ps*+, "SUBDOC", *ps*+,
("NO" | ("YES", *ps*+, *номер}), ps+,*
"FORMAL", *p*s+, ("NO" | "YES")

где

NO означает, что функция не используется.

YES означает, что функция используется.

CONCUR означает, что экземпляры указанного числа типов документа (1 или более) могут появляться одновременно с экземпляром типа базового документа.

SUBDOC означает, что может быть открыто одновременно указанное число объектов поддокументов SGML (1 или более).

FORMAL означает, что публичные идентификаторы интерпретируются как формальные публичные идентификаторы.

13.6 Специфичная для приложения информация

[199] специфичная для приложения информация=
"APPINFO", *ps*+,
("NONE" | *минимальный литерал)*

где

NONE означает, что не указана никакая специфичная для приложения информация.

Минимальный литерал определяет специфичную для приложения

информацию, которая применима к документу.

14 Конкретные синтаксисы ссылок и ядра

Конкретный синтаксис ссылок определяется параметром *конкретного синтаксиса* объявления SGML, показанным на рис. 7. Его *публичный идентификатор* следующий:

"ISO 8879-1986//SYNTAX Reference//EN"

Конкретный синтаксис ядра аналогичен конкретному синтаксису ссылок, за исключением того что для параметра "SHORTREF" определено "NONE". Его *публичный идентификатор* следующий:

"ISO 8879-1986//SYNTAX Core//EN"

ПРИМЕЧАНИЕ. *Набор символов синтаксиса - ссылок* конкретного синтаксиса ссылок - ИСО 646 IRV. Этот набор состоит из символов с номерами 0 - 127, которые соответствуют аналогично пронумерованным символам в ИСО 4873 и ИСО 6937. Набор был выбран, поскольку это простейший стандартный набор символов, который содержит все существенные символы SGML, используемые в конкретном синтаксисе ссылок. Этот выбор не ограничивает ни наборы символов документа, которые могут использоваться, ни их размер.

15 Соответствие

15.1 Документ, удовлетворяющий требованиям SGML

Если документ SGML соответствует всем условиям этого международного стандарта, он является документом, удовлетворяющим требованиям SGML.

15.1.1 Основной документ SGML

Если документ, удовлетворяющий требованиям SGML, повсеместно использует конкретный синтаксис ссылок, набор доступных объемов ссылок и только функции SHORTTAG и OMITTAG, он является основным документом SGML.

ПРИМЕЧАНИЕ. Типичное объявление SGML для основного документа SGML показано на рис. 8. При переходе от одного основного документа SGML к другом различаться может только параметр *набора символов документа*.

15.1.2 Минимальный документ SGML

Если документ, удовлетворяющий требованиям SGML, использует конкретный синтаксис ядра, набор доступных объемов ссылок и не использует функции, он является минимальным документом SGML.

15.1.3 Иной документ, удовлетворяющий требованиям SGML

Если документ, удовлетворяющий требованиям SGML, использует иной конкретный синтаксис, он является иным документом, удовлетворяющим требованиям SGML.

15.2 Приложение, удовлетворяющее требованиям SGML

Если приложение SGML удовлетворяет всем требованиям данного раздела, оно является приложением, удовлетворяющим требованиям SGML.

15.2.1 Условные соглашения приложения

Условные соглашения приложения, удовлетворяющих требованиям SGML, могут оказывать воздействие только на те области, которые остаются открытыми для спецификации приложениями.

ПРИМЕЧАНИЕ. К некоторым примерам относятся: условные

соглашения по наименованию элементов и объектов, или условные соглашения содержания о том, что символы данных, не входящие в набор символов трансляции – ссылок, всегда должны вводиться с помощью ссылок, а не непосредственно.

15.2.2 Соответствие документов

Приложение, удовлетворяющее требованиям SGML, должно требовать, чтобы его документов были документами, удовлетворяющими требованиям SGML, и не должно запрещать какую-либо разметку, которую этот международный стандарт допустил бы в таких документах.

ПРИМЕЧАНИЕ. Например, условное соглашение разметки приложения может рекомендовать, чтобы использовались только отдельные функции минимизации, но не может запрещать использование других функций, если они допускаются формальной спецификацией.

15.2.3 Соответствие документации

Документация приложения, удовлетворяющего требованиям SGML, должна выполнять требования этого международного стандарта (см. 15.5).

15.3 Система, удовлетворяющая требованиям SGML

Если система SGML выполняет требования этого раздела, она является системой, удовлетворяющей требованиям SGML.

ПРИМЕЧАНИЕ. Функцией настоящего раздела является требование, чтобы система, удовлетворяющая требованиям SGML, была способна обработать минимальный документ SGML.

```
                SYNTAX

SHUNCHAR        CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
                     18 19 20 21 22 2324 25 26 27 28 29 30 31 127 255
BASESET         "ISO 646-1983//CHARSET
                 International Reference Version (IRV) // ESC 2/5 4/0"
DESCSET         0 128 0
FUNCTION        RE                              13
                RS                              10
                SPACE                           32
                TAB             SEPCHAR          9
NAMING          LCNMSTRT        " "
                UCNMSTRT        " "
                LCNMCHAR        "-. "   -- Дефис и точка одинаковы              --
                UCNMCHAR        "-. "   -- в нижнем и верхнем регистрах (45 46). --
                NAMECASE        GENERAL YES
                                ENTITY  NO
DELIM           GENERAL         SGMLREF
                SHORTREF        SGMLREF
NAMES           SGMLREF
QUANTITY        SGMLREF
```

Рис. 7. Конкретный синтаксис ссылок

15.3.1 Соответствие документации

Документация системы, удовлетворяющей требованиям SGML, должна выполнять требования этого международного стандарта (см. 15.5).

15.3.2 Соответствие объявлению системы

Система, удовлетворяющая требованиям SGML, должна быть способна обработать любой документ, удовлетворяющий требованиям SGML, который не является несовместимым с *объявлением системы* (см. 15.6).

ПРИМЕЧАНИЕ. Поскольку этот международный стандарт не определяет нотации содержания данных или системные данные, невозможность системы обработать такой текст не является основанием для определения, удовлетворяет ли система требованиям SGML.

15.3.3 Поддержка конкретного синтаксиса ссылок

Система, удовлетворяющая требованиям SGML, должна быть способна анализировать документы в конкретном синтаксисе ссылок в дополнение к любому иному конкретному синтаксису, который она может поддерживать.

ПРИМЕЧАНИЕ. Это требование может быть удовлетворено путем конвертирования конкретного синтаксиса ссылки в системный конкретный синтаксис при получении документа.

Система, удовлетворяющая требованиям SGML, которая может создавать или исправлять документы SGML, должна быть способна делать это для документов SGML, которые используют конкретный синтаксис ссылок.

ПРИМЕЧАНИЕ. Это требование может быть удовлетворено путем конвертирования конкретного синтаксиса ссылки в системный конкретный синтаксис, когда документ должен экспортироваться.

Если система, удовлетворяющая требованиям SGML, позволяет пользователю непосредственно редактировать разметку SGML, она должна также позволить конкретному синтаксису ссылок непосредственно редактироваться.

Если система SGML не поддерживает краткие ссылки в любом синтаксисе, вместо конкретного синтаксиса ссылок может использоваться конкретный синтаксис ядра.

ПРИМЕЧАНИЯ:

1. Система может удовлетворить требование поддержки

конкретного синтаксиса ссылок, используя отдельные программы или модули.

2. Это требование не должно интерпретироваться, как требование ограничения обмена конкретным синтаксисом ссылок; обмен документами может также происходить в иных конкретных синтаксисах.

```
                        <!SGML "ISO 8879-1986"
                -- Этот документ является основным документом SGML. --

                                CHARSET
                --8-битовый набор символов документа, чьи первые 128 символов
                  аналогичны символам набора символов синтаксиса - ссылок.--
BASESET     "ISO 646-1983//CHARSET
             International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET     0 9         UNUSED
            9 2         9
            11 2        UNUSED
            13 1        13
            14 18       UNUSED
            32 95       32
            127 1       UNUSED
BASESET     "ISO Registration Number 109//CHARSET
            ECMA-94 Right Part of Latin Alphabet Nr. 3//ESC 2/13 4/3"
DESCSET     128 32      UNUSED
            160 5       32
            165 1       "SGML User's Group logo"
            166 88      38          -- Включает 5 неиспользуемых для NONSGML --
            254 1       127         -- Перемещает 127 на неиспользуемую позицию, т.к.--
            255 1       UNUSED      -- 255 является номером избегаемого символа --

CAPACITY    PUBLIC      "ISO 8879-1986//CAPACITY Reference//EN"
SCOPE       DOCUMENT
SYNTAX      PUBLIC      "ISO 8879-1986//SYNTAX Reference//EN"

                        FEATURES
MINIMIZE    DATATAG     NO    OMITTAG   YES   RANK      NO    SHORTTAG  YES
LINK        SIMPLE      NO    IMPLICIT  NO    EXPLICIT  NO
OTHER       CONCUR      NO    SUBDOC    NO    FORMAL    NO

                        APPINFO NONE>
```

Рис. 8. Типичное объявление SGML для основного документа SGML

15.3.4 Поддержка набора доступных объемов ссылок

Система, удовлетворяющая требованиям SGML, должна быть способна анализировать документы, чьи доступные объемы не превышают таковые набора доступных объемов ссылки. Если документы SGML могут быть созданы системой, система должна быть способна создавать документы, чьи доступные объемы не превышают таковые набора доступных объемов ссылки.

15.3.5 Совместимость анализа

Система, удовлетворяющая требованиям SGML, должна одинаково анализировать один и тот же документ для всех приложений и процессов, которые с ним работают.

ПРИМЕЧАНИЯ:

1. Прикладная программа, использующая нормальные интерфейсы с синтаксическим анализатором SGML, не должна быть способна повлиять на состояние синтаксического анализа, генерируя текст и побуждая анализировать его, как если бы он был частью документа. Документация для разработчиков приложений должна извещать их об этом требовании.
2. Это требование дает системе возможность проверяться на соответствие без проверки каждого приложения.

15.3.6 Условные соглашения приложения

Система, удовлетворяющая требованиям SGML, не должна предписывать условные соглашения приложения, как если бы они являлись требованиями этого международного стандарта.

ПРИМЕЧАНИЕ. Предупреждения о нарушении условных соглашений приложения могут даваться, но они должны отличаться от сообщений ошибок разметки.

15.4 Синтаксический анализатор SGML с проверкой ошибок

Если синтаксический анализатор SGML в удовлетворяющей требованиям SGML системе выполняет требования этого раздела, он является синтаксический анализатор SGML с проверкой ошибок.

ПРИМЕЧАНИЕ. Система, удовлетворяющая требованиям SGML, не должна обязательно содержать синтаксический анализатор SGML с проверкой ошибок. Поэтому разработчик может решать, вносить ли в данную систему надстройку проверки данных. Например, пользователь, чья система редактирования текста позволяет проверять данные и исправлять документы SGML, не стал бы требовать повторения процесса проверки данных при обработке документов системой форматирования.

15.4.1 Распознавание ошибок

Синтаксический анализатор SGML с проверкой ошибок должен находить и извещать о сообщаемой ошибке разметки, если таковая существует, и не должен сообщать о ошибке, когда ошибок нет.

Синтаксический анализатор SGML с проверкой ошибок может дополнительно сообщать о следующих фактах:

a) неоднозначная модель содержания;

b) изъятие, которое может изменять обязательное или факультативное состояние группы в модели;

c) неудача в соблюдении предела доступного объема;

d) ошибка в объявлении SGML;

e) появление не относящегося к SGML символа; или

f) ошибка формального публичного идентификатора.

ПРИМЕЧАНИЕ. Этот международный стандарт не определяет, как должна обрабатываться ошибка разметки, помимо требования сообщения о ней. В частности, он не определяет, должен ли ошибочный текст

обрабатываться как данные, и/или должна ли быть сделана попытка продолжить обработку после того, как ошибка найдена.

15.4.2 Идентификация сообщений SGML

Сообщения об ошибках разметки SGML, включая факультативные сообщения, должны идентифицироваться как сообщения SGML таким способом, чтобы ясно отличить их от всех других сообщений.

15.4.3 Содержание сообщений SGML

Сообщение об ошибке разметки SGML, включая факультативное сообщение, должно достаточно подробно указывать характер и местоположение ошибки, чтобы допустить ее исправление.

ПРИМЕЧАНИЕ. Это требование сформулировано, чтобы обеспечить разработчиков максимальной гибкостью при соблюдении их пользовательских и системных требований. Более точные предложения приведены в F.4.

15.5 Требования к документации

Цели этот международного стандарта будут выполнены наиболее эффективно, если пользователи на всех уровнях будут осведомлены о том, что документы SGML соответствуют международному стандарту, который является независимым от любого приложения или синтаксического анализатора. Документация удовлетворяющей требованиям SGML системы или приложения должна способствовать такому пониманию.

ПРИМЕЧАНИЕ. Эти требования предназначены для того, чтобы помочь пользователям применить знания, полученное в одной системе SGML, при использования других систем, не запрещая произвольный и дружественный стиль письма.

15.5.1 Стандартная идентификация

Стандартная идентификация должна быть на национальном языке документации.

Текст стандартной идентификации должен быть заметно отображен

a) на видном месте впереди всех публикаций (обычно на титульном листе и на обложке);

b) на всех идентифицирующих экранах программ; и

c) во всех рекламных и обучающих материалах.

Для приложений текст идентификации является следующим:

Приложение SGML, удовлетворяющее международному стандарту ИСО 8879 *Стандартный Обобщенный Язык Разметки*

Для систем текст идентификации является следующим:

Система SGML, удовлетворяющая международному стандарту ИСО 8879 –Стандартный Обобщенный Язык Разметки.

Документация для системы, удовлетворяющей требованиям SGML, должна включать объявление системы (см. 15.6).

15.5.2 Идентификация конструкций SGML

Документация должна отличать конструкции SGML от условных соглашений приложения и системных функций, и должна идентифицировать конструкции SGML как части Стандартного Обобщенного Языка Разметки.

ПРИМЕЧАНИЕ. Целью этого требования является извещение пользователя о том, какие конструкции являются общими для всех систем SGML, а какие являются уникальными для данной системы. Это сократит время изучения опытным пользователем новой системы или приложения.

Этот международный стандарт должен цитироваться как ссылка для

поддерживаемых конструкций SGML, которые специально не документированы для системы или приложения. Например, если для простоты представлено только подмножество некоторой функции (например, с пропуском некоторых опций объявления объекта), должно быть ясно заявлено, что другие опции существуют и могут быть найдены в этот международные стандарте.

15.5.3 Терминология

Все конструкции SGML должны быть представлены с помощью терминологии этот международного стандарта, переведенной на национальный язык, используемый публикацией или программой.

Такая стандартная терминология должна использоваться во всей документации. Если, тем не менее, для стандартного термина используется ненормативный эквивалент, это должно быть представлено в контексте и не должно находиться в противоречии с любыми стандартными терминами SGML, включая термины для неподдерживаемых или неописанных конструкций.

15.5.4 Иной конкретный синтаксис

Если используется иной конкретный синтаксис, этот факт должен быть ясно донесен до пользователя. Правила этого синтаксиса не должны приписываться SGML.

15.6 Объявление системы

[200] объявление системы = ***mdo,*** "SYSTEM", *ps+*,

набор доступных объемов, ps+, использование функций, ps+,

область конкретного синтаксиса, ps+,

поддерживаемые конкретные синтаксисы, ps+,

услуги проверки данных, ps,* ***mdc***

Объявление системы должно соблюдать те же требования синтаксиса, что и *объявление SGML,* в отношении используемого конкретного синтаксиса, допустимых символов данных и т.д.

Параметр *набора доступных объемов* определяется как в *объявлении SGML*, за исключением того, что описывается доступный объем системы, а не требования доступного объема документа.

Параметр *использования функции* определяется как в *объявлении SGML*, за исключением того, что описывается способность системы поддерживать функцию, а не характеристики документа, который использует функцию.

Параметр *области конкретного синтаксиса* определяется как в *объявлении SGML*, за исключением того, что описывается способность системы поддерживать два синтаксиса одновременно, а не использование двух синтаксисов документом.

ПРИМЕЧАНИЕ. *Объявление системы* должен включать комментарии для указания, какие нотации содержания данных и типы *системных данных* может поддерживать система.

15.6.1 Поддерживаемые конкретные синтаксисы

Этот параметр определяет конкретные синтаксисы, которые может анализировать системный синтаксический анализатор SGML, трансляцию их символов разметки в набор системных символов, и любые допустимые отклонения.

[201] поддерживаемые конкретные синтаксисы =

(ps+, *конкретный синтаксис*,

(*ps+, изменения конкретного синтаксиса*)?,

(*ps+, трансляция набора символов*)?)+

Параметр *конкретного синтаксиса* определяется как в *объявлении SGML* для каждого конкретного синтаксиса, который может анализировать система. Одним из указанных конкретных синтаксисов должен быть либо конкретный синтаксис ссылок, если краткие ссылки

поддерживаются каким-либо конкретным синтаксисом, либо конкретный синтаксис ядра, если они не поддерживаются.

15.6.1.1 Изменения конкретного синтаксиса

Этот параметр описывает конкретные синтаксисы, которые может анализировать система, и которые являются незначительными модификациями указанного *конкретного синтаксиса*. Ключевые слова определяют характер и степень допустимых изменений.

[202] изменения конкретного синтаксиса = "CHANGES", *ps+*,
("SWITCHES" | ("DELIMLEN", *ps+*, *номер, ps+*,
"SEQUENCE", ps+, ("YES" | "NO"), *ps+*, "SRCNT", *ps+*, *номер*))

где

SWITCHES означает, что символы разметки в указанном конкретном синтаксисе могут быть переключены, при условии что каждая замена замещает ее оригинальный символ в каждом экземпляре, в котором этот символ используется.

DELIMLEN означает, что новым строкам, которые не превышают указанное число символов (1 или больше) могут быть присвоены роли разграничителей.

SEQUENCE указывает, может ли использовать последовательность пропусков в разграничителях краткой ссылки. Если да, считается, что ее длина равна 1 символу.

SRCNT означает, что могут быть назначены различные разграничители кратких ссылок, пока они не превышают указанное число (0 или более).

15.6.1.2 Трансляция набора символов

Параметр *трансляции набора символов* определяет в той же форме, что и в *объявлении SGML*, трансляцию указанного *конкретного синтаксиса* в системный набор символов из *набора символов трансляции – ссылок*.

Если несколько конкретных синтаксисов имеют один *набор символов трансляции - ссылок,* этот параметр должен быть определен только для одного из них, а применяться будет ко всем.

15.6.2 Услуги проверки данных

Параметр *услуг проверки данных* определяет, имеет ли система синтаксический анализатор SGML с проверкой данных, и какие дополнительные услуги проверки данных, если такие имеются, он предоставляет.

[203] услуги проверки данных = "VALIDATE", *ps+*,
"GENERAL", *ps+*, ("NO" | "YES"), *ps+*,
"MODEL", *ps+*, ("NO" | "YES"), *ps+*,
"EXCLUDE", *ps+*, ("NO" | "YES"), *ps+*,
"CAPACITY", *ps+*, ("NO" | "YES"), *ps+*,
"NONSGML", *ps+*,("NO" | "YES"), *ps+*,
"SGML", *ps+*, ("NO" | "YES"), *ps+*,
"FORMAL", *ps+*, ("NO" | "YES")

где:

NO — Означает, что услуги не предоставляются.

YES — Означает, что услуги не предоставляются.

GENERAL — Означает, что будет найдена и указана сообщаемая ошибка разметки.

MODEL — Означает, что будет сообщаться о неоднозначной модели содержания.

EXCLUDE — Означает, что будет сообщаться об изъятиях, которые могут изменить обязательный и факультативный статус группы в модели.

CAPACITY — Означает, что будет сообщаться о превышении предела доступного объема.

NONSGML Означает, что будет сообщаться о появлении по крайней мере одного символа, не относящегося к SGML, но не обязательно обо всех.

SGML Означает, что будет сообщаться об ошибке в объявлении SGML.

FORMAL Означает, что будет сообщаться об ошибке формального публичного идентификатора.

ПРИЛОЖЕНИЕ А

ВВЕДЕНИЕ В ОБОБЩЕННУЮ РАЗМЕТКУ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

А.1 Процесс разметки

Системы обработки текстов обычно требуют, чтобы дополнительная информация распределялась среди естественного текста обрабатываемого документа. Эта добавленная информация, называемая "разметкой", служит двум целям:

a) выделение логических элементов документа; и

b) определение функций обработки, которые должны быть выполнены над этими элементами.

В издательских системах, где форматирование может быть достаточно сложным, разметка обычно делается непосредственно пользователем, который был специально обучен для этой задачи. В текстовых процессорах, форматеры обычно имеют меньшее количество функций, поэтому (более ограниченная) разметка может быть сгенерирована пользователем без сознательных усилий. Однако, по мере того, как более функциональные принтеры становятся доступными по более низкой стоимости, офисная рабочая станция должна будет обеспечить больший объем функциональных возможностей издательской системы, и "бессознательная" разметка станет возможна только для части офисной обработки текстов.

Поэтому важно рассмотреть, как пользователь высокофункциональной системы размечает документ. Существуют три

Адаптированный текст статьи Goldfarb, Charles F., "A Generalized Approach to Document Markup", *SIGPLAN Notices,* June 1981, с разрешения автора и Ассоциации Вычислительной Техники.

различных шага, хотя он может их не воспринимать как таковые.

a) Сначала он анализирует информационную структуру и другие атрибуты документа; т.е. он идентифицирует каждый значимый отдельный элемент и характеризует его как параграф, заголовок, упорядоченный список, сноску или некоторый другой тип элемента.

b) Затем он определяет, из памяти или книги стилей, команды обработки ("средства управления"), которые обеспечат формат, необходимый для этого типа элемента.

c) Наконец, он вставляет выбранные средства управления в текст.

Здесь показано, как выглядит начало этой статьи, после того как оно размечено средствами управления на типичном языке форматирования системы обработок текста:

```
.SK 1
Системы обработки текстов обычно требуют, чтобы
дополнительная информация распределялась среди
естественного текста обрабатываемого документа.
```

Эта добавленная информация, называемая "разметкой", служит двум целям:

```
.TB 4
.OF 4
.SK 1
1. выделение логических элементов документа; и
.OF 4
.SK 1
2. определение функций обработки, которые должны быть
выполнены над этими элементами.
.OF 0
.SK 1
```

Средства управления .SK, .TB и .OF соответственно, вызывают пропуск вертикального пробела, установку позиции табулятора и

смещение, или "выступ", стиль форматирования. (Знак "не" (¬) в каждом элементе списка представляет код позиции табуляции, который иначе не был бы видим.)

Процедурная разметка, подобная этой, однако, имеет множество недостатков. С одной стороны, информация относительно атрибутов документа обычно теряется. Если пользователь решит, например, при форматировании выровнять по центру и заголовки и названия рисунка, средство управления "центрирование" не будет указывать, является ли текст, над которым оно совершает операцию, заголовком или названием. Поэтому, если он желает использовать документ в приложении информационного поиска, программы поиска будут неспособны отличить заголовки — которые могут быть очень существенны в информационном содержании — от какого-либо иного текста, который был центрирован.

Процедурная разметка также не является гибкой. Если пользователь решит изменить стиль своего документа (возможно, поскольку он использует иное устройство вывода), он должен будет повторить процесс разметки, чтобы отразить изменения. Это помешает ему, например, создавать черновые экземпляры с двойным промежутком на недорогом компьютерном строчном принтере при получении высококачественного окончательного экземпляра на дорогой фотонаборной машине. И если он пожелает искать конкурентоспособные предложения для набора своего документа, он будет ограничен теми поставщиками, которые используют идентичную систему обработки текста, если только он не пожелает оплачивать стоимость повторения процесса разметки.

Кроме того, разметка с управляющими словами может отнимать много времени, быть подверженной ошибкам и требовать высокой степени обучения оператора, особенно когда необходимы комплексные типографские результаты. Это справедливо (хотя в меньшей степени), даже когда система позволяет выполнять определенные процедуры ("макрокоманды"), поскольку они должны быть добавлены к

пользовательскому словарю примитивных средства управления. Например, изящная и мощная система TeX (2), которая широко используется для математического набора, включает в своем базовом исполнении приблизительно 300 примитивных средств управления и макрокоманд.

Этих недостатков процедурной разметки помогает избежать схема разметки, разработанная C. F. Goldfarb, E. J. Mosher и R. A. Lorie (3, 4). Она называется "обобщенной разметкой", потому что она не ограничивает документы на единственным приложением, стилем форматирования или системой обработки. Обобщенная разметка основана на двух новых аксиомах:

a) Разметка должна описывать структуру документа и другие атрибуты, а не определять обработку, которая должна с ним выполняться, так как описательная разметка должна делаться только однажды и будет достаточна для всей последующей обработки.

b) Разметка должна быть настолько строга, чтобы методы, доступные для обработки строго-определенных объектов, подобных программам и базам данных, могли бы использоваться также и для обработки документов.

Эти аксиомы будут развиты интуитивно при изучении свойств этого типа разметки.

A.2 Описательная разметка

С обобщенной разметкой процесс разметки останавливается на первом шаге: пользователь обнаруживает каждый существенный элемент документа и отмечает его мнемоническим именем ("родовым идентификатором"), который, как ему кажется, наилучшим образом характеризует этот элемент. Система обработки сопоставляет разметку с

командами обработки способом, который будет коротко описан.

Нотация для обобщенной разметки, известная как Стандартный Обобщенный Язык Разметки (SGML), была разработана Рабочей Группой Международной Организации для Стандартизации (ИСО). При разметке в SGML начало этой статьи могло бы выглядеть следующим образом:

<p>

Системы обработки текстов обычно требуют, чтобы дополнительная информация распределялась среди естественного текста обрабатываемого документа.

Эта добавленная информация, называемая <q>разметкой</q>, служит двум целям:

выделение логических элементов документа; и

определение функций обработки, которые должны быть выполнены над этими элементами.

Каждый родовой идентификатор (GI) разграничен символом "меньше" (<), если он находится в начале элемента, или "меньше", за которым следует косая черта (< /), если он находится в конце. Символ "больше" (>) отделяет GI от любого текста, который следует за ним[2]. Мнемоники P, Q, OL и LI обозначают, соответственно, следующие типы элементов: параграф, цитата, упорядоченный список и элемент списка. Комбинация GI и его разграничителей называется "начальным тегом" или "конечным тегом", в зависимости от того, идентифицирует ли она начало или конец элемента.

Этот пример имеет некоторые интересные свойства:

а) В тексте отсутствуют кавычки; обработка элемента цитаты генерирует их и обеспечит различие между открывающими и

[2] Фактически эти символы являются символами по умолчанию. SGML разрешает выбор символов разграничителей.

закрывающими кавычками, если это допускается устройством вывода.

b) Запятая, которая следует за элементом цитаты, фактически не является ее частью. Здесь, она также была оставлена вне кавычек в ходе форматирования, но она могла также легко быть включена внутрь, если бы этот стиль был предпочтителен.

c) Для элементов упорядоченного списка отсутствуют порядковые номера; они генерируются в ходе форматирования.

Другими словами, исходный текст содержит только информацию; символы, чья роль заключается только в расширении представления, генерируются в ходе форматирования.

Если, как утверждается, описательная разметка, подобная этой, достаточна для всей обработки, за этим должно следовать, что обработка документа является функцией его атрибутов. Способ, которым сочиняется текст, дает интуитивную поддержку этому предположению. Такие методы, как начало глав на новой странице, выделение курсивом особо важных фраз и выравнивание списков, используются, чтобы помочь пониманию читателя путем подчеркивания структурных атрибутов документа и его элементов.

На основании этого анализа может быть создана модель с 3 шагами обработки документов :

a) Распознавание: распознается атрибут документа, например, элемент с родовым идентификатором "сноска".

b) Отображение: атрибут связывается с функцией обработки. Сноска GI, например, может быть связана с процедурой, которая печатает сноски в нижней части страницы или другой, которая собирает их в конце главы.

c) Обработка: выполняется выбранная функция обработки.

Текстовые программы форматирования соответствуют этой модели. Они распознают такие элементы, как слова и предложения,

интерпретируя, прежде всего, пробелы и знаки пунктуации как неявную разметку. Отображение происходит обычно через таблицу переходов. Типичная обработка для слов включает определение ширины слова и проверку на превышение ограничения строки; обработка для предложений может вызывать вставку пробела между ними[3].

В случае элементов нижнего уровня, таких как слова и предложения, пользователю обычно предоставляется небольшой контроль над обработкой, и почти никакого над распознаванием. Некоторые форматеры предлагают большую гибкость в отношении элементов высокого уровня, таких как параграфы, тогда как форматеры с мощными макроязыками идут настолько далеко, что могут поддерживать описательную разметку. В терминах модели обработки документов преимущество описательной разметки состоит в том, что она разрешает пользователю определять атрибуты — и поэтому типы элемент — не известные форматеру и определять их обработку.

Например, описанный только что отрывок SGML включает такие типы элемента, как "упорядоченный список" и "элемент списка", в дополнение к более общему "параграфу". Встроенное распознавание и обработка таких элементов маловероятны. Вместо этого, каждый из них будет распознаваться его явной разметкой и отображаться на процедуру, связанную с ним для частного выполнения обработки. И сама процедура, и сопоставление с GI будут выражены в макроязыке системы. При другом выполнении обработки, или в разные моменты одного выполнения, сопоставление может быть изменено. Элементы списка, например, могут быть пронумерованы в основном тексте книги, но обозначены буквами в приложении.

Таким образом обсуждение касалось только единственного атрибута, родового идентификатора, чье значение характеризует

[3] Эта модель может не отражаться архитектурой программы; например, обработка слов может быть встроена в основной цикл распознавания для улучшения производительности.

семантическую роль или назначение элемента. Некоторые схемы описательной разметки обращаются к разметке как к "родовому кодированию", потому что GI является единственным атрибутом, который они признают (5). В схемах родового кодирования распознавание, отображение и обработка могут быть выполнены одновременно простым устройством использования GI как имен управляющих процедур. Затем из одной разметки могут быть получены различные форматы путем вызова различных наборов омонимичных процедур. Этот подход настолько эффективен, что одна известная реализация, система SCRIBE, способна полностью запретить процедурную разметку (1).

Родовое кодирование дает значительное преимущество перед процедурной разметкой в практическом использовании, но концептуально оно недостаточно. Документы являются сложными объектами, и они имеют другие атрибуты, описать которые должен быть способен язык разметки. Например, предположим, что пользователь решит, что его документ должен включать тип элементов, именуемый "figure" (рисунок), и что должна быть возможность обращаться к отдельному рисунку по имени. Разметка для конкретного рисунка под названием "angelfig" могла начинаться со следующего начального тега:

<fig id = angelfig>

"Fig", конечно, обозначает "figure", значение атрибута родового идентификатора. GI идентифицирует элемент как член набора элементов, имеющих одинаковую роль. Напротив, атрибут "уникального идентификатора" (ID) отличает элемент от всех других, даже с тем же самым GI. (Не нужно сообщать "GI=fig", как это было сделано для ID, потому что в SGML понимается, что первая часть разметки для элемента является значением его GI).

Атрибуты GI и ID называются "первичными", поскольку их может иметь каждый элемент. Имеются также "вторичные" атрибуты, которыми

обладают только некоторые типы элементов. Например, если пользователь хочет, чтобы некоторые из рисунков в его документе содержали иллюстрации, которые должны выполняться художником и добавляться к выводу после обработки, он может определить тип элемента "artwork" (художественное произведение). Поскольку важен размер сгенерированных извне художественных произведений, он может определить элементы художественных произведений с вторичным атрибутом, "depth" (разрешение)[4]. Это приведет к появлению следующего начального тега для части художественного произведения размером 24 пики:

<artwork depth =24p>

Разметка для рисунка должна также описывать его содержание. "Содержание", конечно, является первичным атрибутом, т.е. тем, которое описывают вторичные атрибуты элемента. Содержание состоит из организации других элементов, каждый из которых в свою очередь может иметь другие элементы в своем содержании, и так далее, пока дальнейшее деление станет невозможно[5]. Одно из свойств, которыми SGML отличается от схем родового кодирования, заключается в концептуальных и нотационных средствах, которые он предусматривает для работы с этой иерархической структурой. Они основаны на второй гипотезе обобщенной разметки о том, что разметка может быть строгой.

A.3 Строгая разметка

Предположим, что содержание рисунка "angelfig" состоит из двух элементов, корпуса рисунка (figure body) и названия рисунка (figure

[4] "Depth=" не является просто эквивалентом управляющего слова вертикального расстояния. Хотя программа составления полных страниц может создать фактическое расстояние, форматер гранок может напечатать сообщение с инструкцией для наборщика оставить его пустым. Программа выборки может просто индексировать рисунок и полностью игнорировать размер.

[5] Поэтому можно говорить о документах и элементах практически как об одном и том же: документ просто является элементом, который расположен на вершине иерархии для данного выполнения обработки. Технический отчет, например, может форматироваться и как документ сам по себе, и как элемент журнала.

caption). Корпус рисунка в свою очередь содержит элемент художественного произведения, в то время как содержание названия - текстовые символы без явной разметки. Разметка для этого рисунка могла бы выглядеть следующим образом:[6]

```
<fig id=angelfig>
<figbody>
<artwork depth=24p >
</artwork>
</figbody>
<figcap>Три танцующих ангела
</figcap>
</fig>
```

Разметка строго выражает иерархию, идентифицируя начало и конец каждого элемента в классическом левом порядке списка. Для интерпретации структуры не требуется никакая дополнительная информация, и реализация поддержки возможна с помощью обсужденной ранее простой схемы вызова макрокоманды. Цена этой простоты, однако, состоит в том, что для каждого элемента должен присутствовать конечный тег .

Эта цена была бы абсолютно неприемлема, если бы пользователь должен был вводить все теги самостоятельно. Он знает, например, что начало параграфа заканчивает предыдущий параграф, поэтому он бы сопротивлялся необходимости идти на неприятности и издержки, связанные с вводом очевидного конечного тега для каждого отдельного параграфа только для того, чтобы поделиться своими знаниями с системой. Аналогичные сильные чувства он ощущал бы в отношении других типов элемента, которые он мог бы для себя определить, если бы они встречались с достаточно большой частотой.

[6] Подобно "GI=" "содержание=" может быть легко опущен. Он необязателен, когда содержание генерируется извне, он понимается, когда содержание состоит исключительно из тегированных элементов, а для символов данных он подразумевается разграничителем (>), который заканчивает начальный тег.

Однако, с SGML возможно опустить большую часть разметки, уведомляя систему о структуре и атрибутах любого типа элемента, который определяет пользователь. Это делается путем создания "определения типа документа", используя конструкцию языка, называемого "объявление элемента". Тогда как разметка в документе состоит из описаний отдельных элементов, определение типа документа определяет набор всей возможной действительной разметки типа элемента.

Объявление элемента включает описание допустимого содержания, обычно выражаемое в варианте регулярной нотации выражения. Предположим, например, что пользователь расширяет свое определение "рисунка" с тем, чтобы разрешить корпусу рисунка содержать либо художественные произведения, либо некоторые виды текстовых элементов. Объявление элемента могло бы выглядеть следующим образом:[7]

```
<!-- МИН.     СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ) -->
<!ELEMENT   fig   - -   (figbody, figcap?)>
<!ELEMENT   figbody   - 0   (artwork | (p | ol | ul)+)>
<!ELEMENT   artwork   - 0   EMPTY>
<!ELEMENT   figcap    - 0   (#PCDATA)>
```

Первое объявление означает, что рисунок содержит корпус рисунка и, факультативно, может содержать название рисунка после корпуса рисунка. (Дефисы вскоре будут объясняться.)

Второе объявление говорит о том, что корпус может содержать либо художественные произведения, либо смешанное собрание параграфов, упорядоченных списков и неупорядоченных списков. "0" в поле минимизации разметки ("MIN") указывает на то, что конечный тег

[7] Вопросительный знак (?) означает, что элемент является факультативным, запятая (,) – что он следует за предшествующим элементов в последовательности, звездочка (*) - что элемент может появиться 0 или большее число раз, а плюс (+)- что он должен появиться 1 или большее число раз. Вертикальная черта (|) используется для разделения альтернатив. Скобки используются для группирования как в математике.

корпуса может быть опущен, если он однозначно подразумевается началом следующего элемента. Предшествующий дефис означает, что начальный тег не может быть опущен.

Объявление для художественного произведения определяет его как имеющее пустое содержание, поскольку произведение будет сгенерировано извне и вставлено. Поскольку в документе отсутствует содержание, отсутствует необходимость в наличии завершающей разметки.

Заключительное объявление определяет содержание названия рисунка как 0 или большее число символов. Символ является конечным, неспособным к дальнейшему делению. "0" в поле "MIN" указывает на то, что конечный тег названия может быть опущен. В дополнение к уже приведенным причинам, пропуск возможен, когда конечный тег однозначно подразумевается конечным тегом элемента, который содержит название.

Предполагается, что p, ol и ul определены в других объявлениях элемента. С этим формальным определением доступных элементов рисунка, теперь приемлема следующая разметка для "angelfig" :

<fig id= angelfig>
<figbody>
<artwork depth=24p>
<figcap>Три танцующих ангела
</fig>

Имеет место сокращение разметки на 40%, так как для трех элементов конечные теги больше не нужны.

— Поскольку объявление элемента определило название рисунка как часть содержания рисунка, завершение рисунок автоматически завершает название.

— Так как само название рисунка находится на том же уровне, что и корпус рисунка, начальный тег <figcap> неявно завершает корпус рисунка.

— Элемент художественного произведения является самозавершающимся, поскольку объявление элемента определило его содержание пустым.[8]

Определение типа документа также содержит "объявление списка определений атрибутов" для каждого элемента, который имеет атрибуты. Определения включают возможные значения, которые может иметь атрибут, а также значение по умолчанию, если атрибут факультативен и не определен в документе.

Ниже приведены объявления списка атрибутов для "figure" и "artwork":

```
<!--ЭЛЕМЕНТЫ ИМЯ ЗНАЧЕНИЕ ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
    <!ATTLIST fig id ID #IMPLIED>
    <!ATTLIST artwork depth CDATA #REQUIRED>
```

Объявление для рисунка указывает, что он может иметь атрибут ID, чье значение должно быть уникальным именем идентификатора. Атрибут является факультативным и не имеет значения по умолчанию, если он не определен.

Напротив, атрибут размера элемента художественного произведения является обязательным. Его значением может быть любая строка символов.

Определения типа документа используются в дополнение к минимизации разметки[9]. Они могут использоваться для проверки правильности разметки в документе перед переходом к обработке, или управлять диалогами подсказок для пользователей, незнакомых с типом документа. Например, приложение ввода документа может читать описание элемента рисунка и вызывать процедуры для каждого типа элемента. Процедуры могут выдавать на терминал сообщения, запрашивающие пользователя ввести ID рисунка, размер

[8] Фактически SGML позволяет сокращать разметку даже в большей степени.
[9] Некоторые полные, практичные определения типа документа можно найти в (4), хотя они не кодированы в SGML.

художественного произведения и текст названия. Процедуры могут также ввести разметку непосредственно в создаваемый документ.

Определение типа документа дает возможность SGML минимизировать усилия пользователя по вводу текста не полагаясь на "смышленую" программу редактирования или текстовый процессор. Это максимизирует переносимость документа, потому что он может быть понят и исправлен людьми, использующими любую из миллионов существующих "немых" клавиатур. Тем не менее, определение типов и размеченный документ совместно остаются строго описанным документом, который требует автоматизированной обработки.

A.4 Заключение

Независимо от степени точности и гибкости описания документа, которое делает возможным разметка, по-прежнему остается беспокойство пользователя, который готовит документы для публикации: может ли Стандартный Обобщенный Язык Разметки или любая описательная схема разметки достигнуть типографских результатов, сопоставимых с процедурной разметкой? Недавняя публикация Prentice-Hall International (6) представляет эмпирическое подтверждение гипотез обобщенной разметки в контексте этого насущного практического вопроса.

Это учебник по программированию, содержащий сотни формул в символической нотации, изобретенной автором. Несмотря на типографскую сложность материала (многие строк, например, имеют дюжину или большее число смен шрифтов), процедурная разметка не понадобилась нигде в тексте книги. Она была размечена с помощью языка, который твердо придерживался принципов обобщенной разметки, но был менее гибок и полон, чем SGML (4).

Доступные процедуры поддерживали только компьютерные устройства вывода, которые были адекватны для предварительных

версий книги, использовавшихся как классные заметки. Набору не уделялось внимание, пока книга не была принята для публикации, и когда ее автор задумался о времени и усилиях, необходимых для повторного ввода и корректуры приблизительно 350 сложных страниц. Он начал искать альтернативу в то же самое время, когда автор этой статьи искал экспериментальный объект для подтверждения возможности применения обобщенной разметки к коммерческой публикации.

Оба поиска оказались успешны, и был начат необычный проект. Поскольку процессор автора не поддерживал непосредственно фотонаборные машины, были написаны процедуры, которые создавали исходный файл с процедурной разметкой для отдельной программы типографского набора. Спецификации форматирования обеспечивались издателем, и не потребовалось никаких уступок, чтобы включить использование обобщенной разметки, несмотря на то, что размеченный документ существовал до спецификаций.[10]

Эксперимент был закончен вовремя, и издатель считает его полным успехом (7).[11] Процедуры, с некоторой модификацией относительно стиля форматирования, нашли дополнительное использование при создании ряда внутренних публикаций.

Поэтому обобщенная разметка имеет и практические, и академические преимущества. В издательской среде она снижает стоимость разметки, сокращает время при изготовлении книг и предлагает максимальную независимость от текстовой базы данных. В офисе она разрешает обмен между различными видами текстовых процессоров с изменением функциональных способностей, и позволяет создавать автоматически вспомогательные "документы", такие как

[10] Напротив, издатель использовал преимущества обобщенной разметки, изменив некоторые спецификации, после того как он увидел проверку страниц.

[11] Несмотря на некоторые географические сложности: издатель находился в Лондоне, автор книги в Брюсселе, а автор настоящей статьи в Калифорнии. Почти все общение велось по международной компьютерной сети, и проект был практически завершен до того, как все его участники встретились в первый раз.

записи журнала регистрации почты, из соответствующих элементов основного документа, таких как записка.

В то же время, строгая описательная разметка SGML делает текст более доступным для компьютерного анализа. В то время как процедурная разметка (или вообще отсутствие разметки) оставляет документ как строку символов, которая не имеет никакой формы кроме той, которая может быть выведена из анализа значения документа, обобщенная разметка сокращает документ до регулярного выражения в известной грамматике. Это разрешает применять существующие методы вычислительной лингвистики и разработки компиляторов к обработке естественных языков и другим приложениям обработки документов.

A.5 Благодарности

Автор благодарен E. J. Mosher, R. A. Lorie, T. Я. Peterson и A. J. Symonds — его коллегам на ранних этапах разработки обобщенной разметки — за их большой вклад в идеи, представленные в этой статье, N. R. Eisenberg за его сотрудничество в создании и разработке процедур, используемых для проверки применимости обобщенной разметки к коммерческим публикациям, а также C. B. Jones и Ron Decent за готовность рискнуть их любимой книгой ради нескольких новых идей.

A.6 Библиография

1. B. K. Reid, "The Scribe Document Specification Language and its Compiler", *Proceedings of the International Conference on Research and Trends in Document Preparation Systems*, 59-62 (1981).

2. Donald E. Knuth, *TAU* EPS/LOW *CHI, a system for technical text,* American Mathematical Society, Providence, 1979.

3. C. F. Goldfarb, E. J. Mosher, and T. I. Peterson, "An Online System for Integrated Text Processing", *Proceedings of the American Society for*

Information Science, 7,147-150 (1970).

4. Charles F. Goldfarb, *Document Composition Facility Generalized Markup Language: Concepts and Design Guide*, Form No. SH20-9188-1, IBM Corporation, White Plains, 1984.

5. Charles Lightfoot, *Generic Textual Element Identification—A Primer*, Graphic Communications Computer Association, Arlington, 1979.

6. C. B. Jones, *Software Development: A Rigorous Approach*. Prentice-Hall International, London, 1980.

7. Ron Decent, *personal communication to the author* (September 7, 1979).

ПРИЛОЖЕНИЕ В

ОСНОВНЫЕ КОНЦЕПЦИИ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Это приложение описывает некоторые из основных концепций Стандартного Обобщенного Языка Разметки (SGML). До его изучения читателю следует ознакомиться с приложением А, чтобы приобрести первоначальное знакомство с родовым кодированием и обобщенной разметкой.

ПРИМЕЧАНИЕ. Читатель должен знать, что это приложение не охватывает ни все основные конструкции SGML, ни все подробности рассмотренных конструкций, при этом тонкие детали часто игнорируются ради представления ясного обзора.

B.1 Документы, определения типа документа и процедуры

Фундаментальной концепцией обобщенной разметки являются соотношения между документами, определениями типа документа и процедурами.

B.1.1 Документы

В обобщенной разметке термин "документ" не относится к физической конструкции, такой как файл или набор напечатанных страниц. Вместо этого документ представляет собой логическую конструкцию, которая содержит *элемент документа*, высший узел дерева элементов, которые составляют *содержание* документа. Книга, например, может содержать элементы "главы", которые в свою очередь

содержат элементы "параграфы" и "изображения".

В конечном счете, достигаются предельные узлы этого дерева документа и обнаруживаются фактические символы или иные данные. Если бы, например, параграфы являлись предельными узлами, их содержание было бы символами, а не другими элементами. Если бы предельными были фотографии, они не содержали бы ни элементы, ни символы, но некоторые несимвольные данные, который представляют образ.

Элементы различаются друг от друга с помощью дополнительной информации, называемой *разметкой*, которая добавляется к содержанию данных. Документ, таким образом, состоит из двух видов информации: данные и разметка.

B.1.2 Определения типа документа

Разметка элемента состоит из *начального тега* в начале элемента и *конечного тега* в его конце. Теги описывают характерные качества элемента.

Одной из таких характеристик является *родовой идентификатор*, который идентифицирует "тип" элемента (справочник, параграф, рисунок, список и т.д.). Кроме того, могут иметься другие характеристики, называемые "атрибутами", которые более подробно характеризуют родовой идентификатор.

Теги разметки отдельного документа описывают его структуру элементов. Т.е. они указывают, какие элементы и в каком порядке появляются в содержании документа. Эта структура должна соответствовать правилам, которые определяют разрешенные структуры для всех документов относительно данного типа; т.е. тех документов, которые имеют тот же родовой идентификатор.

Правила, которые определяют возможные структуры, являются частью *определения типа документа* для этого типа документа. Определение типа документа определяет:

a) Родовые идентификаторы (GI) элементов, которые являются допустимыми в документе этого типа.

b) Для каждого GI возможные атрибуты, диапазон их значений и значения по умолчанию.

c) Для каждого GI структуру его содержания, включая

i) какие GI подэлементов и в каком порядке могут появляться;

ii) могут ли встречаться текстовые символы;

iii) могут ли встречаться несимвольные данные.

Определение типа документа *не* определяет:

a) Разграничители, которые используются для указания разметки.

b) Способы, которыми документ может быть отформатирован или обработан иным образом.

B.1.3 Процедуры

Теги разметки описывают структуру элементов документа; они не говорят о том, как обработать эту структуру. Возможны многие виды обработки, один из которых должен *форматировать* текст.

Форматирование может рассматриваться как отображение структуры элемента на бумагу или экран дисплея с учетом условных соглашений полиграфии. Например, элемент "параграф" может быть отображаться путем отделения текст элемента от окружающего текста пустыми строками. При другом способе он может отображаться отступом для его первой строки.

Обработка совершается *процедурами*, которые написаны на языке форматера или другой системы обработки. В процессе обработки документа процедура сопоставляется с каждым родовым идентификатором (т.е. с каждым типом элемента). Затем процедура

обрабатывает содержание этого элемента. Например, в случае форматирования, процедура совершает действия, которые превращают элемент в печатный текст или другую форму отображения.

Таким образом, производство документа начинается, когда пользователь создает текст, размечая его как конкретный тип документа. Одним из средств, которые могут обрабатывать этот документ, является форматер, который может иметь более чем один набор доступных процедур.

Например, документ, называемый "моя книга", размеченный как тип документа "ТехРуководство", может быть отформатирован несколькими способами, каждый раз используя различный *набор процедур*. Один набор может осуществлять вывод в стиле одиночного столбца дисплея, другой набор - в стиле двухколоночного печатного сообщения, а третий набор - в каком-либо ином стиле.

Тогда при разработке совершенного нового текстового приложения проектировщику следует создавать определения типа документа, используя Стандартный Обобщенный Язык Разметки. Вероятно, он также должен реализовать один или несколько наборов процедур, используя языки систем, которые должны обрабатывать документы.

B.2 Разметка

Разметка представляет собой текст, который добавляется к данным документа, чтобы перенести информацию о нем. В SGML разметка в документе делится на четыре категория:

а) Описательная разметка ("теги")

Теги наиболее частый и наиболее важный вид разметки. Они определяют структуру документа, как описано выше.

b) Ссылка на объект

В пределах системы, одиночный документ может быть сохранен в нескольких частях, каждая в отдельном модуле системной памяти, называемом *объектом*. (В зависимости от системы объектом может быть файл, набор данных, переменная, объект потока данных, библиотечный элемент и т.д.).

Отдельные объекты соединяются ссылками на объект, которые могут появляться в разметке документа. Ссылка на объект представляет собой запрос на вставку текста — объекта — в документ в месте ссылки. Объект может быть определен либо ранее внутри документа, либо вне его.

Возможность ссылки на объект включает функции, обычно называемые подстановкой символа и вставкой файла.

c) Объявление Разметки

Объявления представляют собой инструкции, которые управляют интерпретацией разметки. Они могут использоваться для определения объектов и создания определений типа документа.

d) Команды обработки

Это команды системе обработки, написанные на ее собственном языке, предпринять некоторое определенное действие. В отличие от других видов разметки, команды обработки зависят от системы и обычно также от приложения. Они обычно должны быть заменены, если документ обрабатывается иным образом (например, форматируется в другом стиле), или в другой системе.

Система SGML должна распознавать эти четыре вида разметки и обрабатывать их надлежащим образом: т.е. она должна иметь "синтаксический анализатор SGML". Синтаксический анализатор не должен обязательно быть специализированной программой; если система может выполнять процесс синтаксического анализа, может

считаться, что она имеет синтаксический анализатор SGML.

Разметка появляется в документе в соответствии с жестким набором правил. Некоторые из правил диктуются SGML; они применяются ко всем типам документов. Другие правила определяются определением типа документа для типа обрабатываемого документа.

Используя правила SGML, синтаксический анализатор разметки должен:

a) Просмотреть текст содержания каждого элемента, чтобы отличить четыре вида разметки друг от друга и от данных. (Несимвольные данные содержания синтаксическим анализатором не просматриваются.)

b) Заменить ссылки на объект их объектами.

c) Интерпретировать объявления разметки.

d) Передать управление системе обработки, чтобы выполнить команды обработки.

e) Интерпретировать теги описательной разметки, чтобы распознать родовые идентификаторы ("GI") и атрибуты, и, в соответствии с правилами типа документа:

i) Определить, действительны ли каждый GI и его атрибуты.

ii) Проследить местоположение в структуре документа.

f) Передать управление системе обработки, чтобы выполнить процедуру, сопоставленную с GI. (Еще раз повторим, что не существуют какие-либо фактические требования для отдельных программ. "Передача управления" означает только то, что обеспечение обработки не определено этим международным стандартом.)

B.3 Различение разметки и текста

Разметка, рассматриваемая в этом разделе, применяется ко всем типам документа. Используемые символы - разграничители представляют собой набор разграничителей *конкретного синтаксиса ссылок*. (Они будут рассматриваться, как если бы имелся только один конкретный синтаксис, хотя SGML позволяет определять иные конкретные синтаксисы.)

B.3.1 Теги Описательной разметки

Теги описательной разметки идентифицируют начало и конец элементов. Имеются три специальных строки символов, которые являются важными (см. рис. 9):

STAGO Начальный тег открыт

Это разграничитель, который указывает начало начального тега. На рисунке "<" обозначает ***stago***.

TAGC Тег закрыт

Строка от ***stago*** до ***tagc*** называется начальным тегом. В нем приводятся родовой идентификатор ("GI") и все атрибуты. На рисунке "кавычка" обозначает GI, а ">" используется для ***tagc***.

ETAGO Конечный тег открыт

Это двухсимвольный разграничитель, который указывает начало конечного тега. На рисунке "</" обозначает ***etago***. Между ***tagc*** начального тега и ***etago*** конечного тега находится *содержание* элемента, которое может включать символы данных и подчиненные элементы. (Несимвольные данные хранятся отдельно; это будет обсуждаться позже.) Конечный тег содержит повтор GI, чтобы сделать разметку более легкой для чтения.

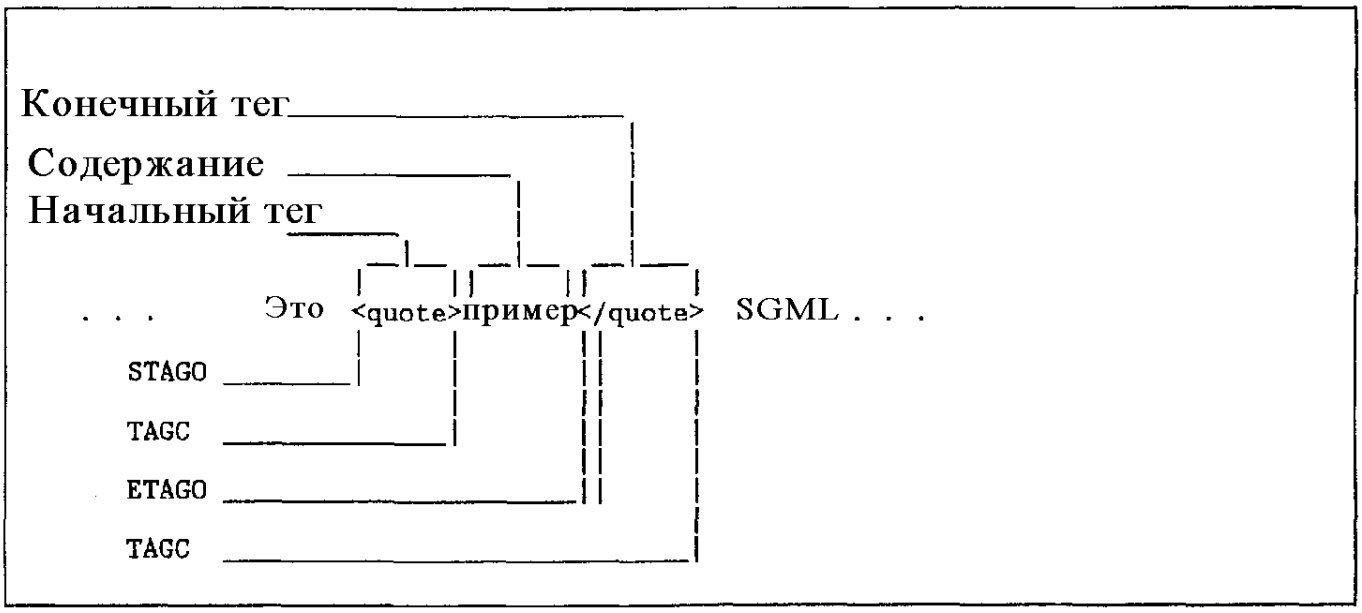

Рис. 9. Разметка элемента

Описанная схема представляет собой наиболее общий способ разграничивания элемента. SGML, тем не менее, допускает применение нескольких методов *минимизации разметки*, которые позволяют сокращать теги и даже пропускать их. Эти методы, которые доступны как факультативные функции, будут описаны позже.

B.3.2 Другая разметка

Ссылка на объект начинается с разграничителя "ссылка на объект открыта" (***ero***) и завершается разграничителем "ссылка закрыта" ***(refc)***. В следующем примере ими являются, соответственно, амперсанд и точка с запятой:

&SGML; поддержка издательских и офисных систем.

Объявление разметки разграничивается разграничителями "объявление разметки открыто" ***(mdo)*** и "объявление разметки закрыто" (**mdc**), а команды обработки разграничителями "команда обработки открыта" (***pio***) и "команда обработки закрыта" (pic).

это пример вставленного <!объявления разметки>

и пример вставленной <?команды обработки>

Таким образом:

Строка	**Имя**	***Значение***
&	**ERO**	Открывает поименованную ссылку на объект.
T	**REFC**	Закрывает ссылку.
<!	**MDO**	Открывает объявление разметки.
>	**MDC**	Закрывает объявление разметки.
<?	**PIO**	Открывает команду обработки.
>	**PIC**	Закрывает команду обработки.

B.3.3 Границы записи

Не каждая система обработки текста разбивает свои объекты хранения на записи. В тех системах, которые это делают, границы записей представлены символами функций, известными как "начало записи" ***(RS)*** и "конец записи" ***(RE)***. Границы записи могут использоваться как общие символы - разграничители в ином конкретном синтаксисе, или как специальная форма разграничителя, который служит как "краткая ссылка на объект". Если граница записи не является разграничителем, ее обработка зависит от того, где она встречается.

B. 3.3.1 Границы записи в данных

В значениях атрибутов (будут рассмотрены позже) и в содержании данных элемента начала записи игнорируются. Тем не менее, концы записи обрабатываются как часть данных, поскольку они могут быть существенны для системы обработки документа. Форматер, например, обычно интерпретирует конец записи как пробел.

Однако, концы записи игнорируются, когда они вызваны разметкой. Таким образом:

— Концы записи игнорируются в начале или конце содержания. Например,

<p>

Краткие данные параграфа.

аналогично

Краткие данные параграфа.

— Концы записи игнорируются после записи, которая содержит только команды обработки или объявления разметки. В результате,

Начальные данные

<?команда 1>

<?команда 2>

конечные данные.

и

Начальные данные

<?команда 1><?команда 2>конечные данные.

эквивалентны. Другими словами, насколько это касается потока символов данных, объявления и команды обработки просто игнорируются.

B.3.3.2 Границы записи в разметке

В тегах или объявлениях начала записи и концы записи обрабатываются как пробелы. Они служат как внутренние разделители внутри разметки (также, кстати, как и горизонтальные позиции табуляции).

Значение границ записи в пределах команд обработки зависит от системы обработки.

B.4 Структура документа

Теги SGML служат двум целям:

а) Они показывают структурные отношения между элементами

документа.

b) Они идентифицируют родовой идентификатор (GI) и атрибуты каждого элемента .

Правила для определения структуры и атрибутов определены SGML для всех документов. (Некоторые правила уже были представлены; другие будут обсуждаться позже.) Однако, *специальные* элементы и атрибуты, допускаемые в документе, определяются определением типа этого документа.

B.4.1 Определения типа документа

Родовой идентификатор (GI) идентифицирует элемент как элемент класса, или "типа". *Определение типа документа* представляет собой набор объявлений разметки, которые обращаются к всем документам определенного типа.

Три наиболее важных вида объявлений, которые могут встречаться в определении типа документа:

a) *Объявление элемента, определяющее GI, которые могут встречаться в каждом элементе, и их порядок.*

b) Объявление списка определений атрибутов, определяющее атрибуты которые могут быть определены для элемента, и их возможные значения.

c) Объявление объекта, определяющее объекты, на которые могут быть ссылки в документах этого типа. Например, ссылки на объект могут упростить набор с клавиатуры часто используемых длинных фраз:

<!ENTITY SGML " Стандартный Обобщенный Язык Разметки">

Чтобы избегать повторного набора, определение типа документа обычно сохраняется как отдельный объект. Затем оно включается в каждый документ посредством *объявления типа документа*, которое идентифицирует тип документа и служит ссылкой на

внешний объект.

B.4.2 Объявления элемента

Элементы могут появляться в документе только согласно правилам определения типа документа. Например, если только определение не разрешает параграфу находиться внутри рисунка, поместить туда параграф будет ошибкой. Объявление элемента используется для определения этих правил.

B.4.2.1 Модели содержания

Для каждого элемента в документе, разработчик приложения определяет два параметра объявления элемента: GI элемента и его *модель содержания*. Параметр модели определяет, какие подэлементы и строки символов могут появляться в содержании.

Например, объявление для учебника может выглядеть следующим образом:

<!ELEMENT учебник (начало, корпус, конец)>

Здесь, "учебник" - это GI, чье содержание определяется, а "(начало, корпус, конец)" является моделью, которая его определяет. Пример говорит о том, что учебник содержит идентификаторы "начало", "корпус" и "конец". (Идентификаторы вероятно замещают "объект начала", "объект корпуса" и "объект конца", однако это интересно только людям, но не синтаксическому анализатору SGML.)

Модель - это своего рода *группа*, которая является собранием связанных членов, называемых *маркерами*. Группа ограничена круглыми скобками и разграничителями "группа открыта" (***grpo***) и "группа закрыта" (***grpc***). Круглые скобки требуются даже тогда, когда группа модели содержит единственный маркер.

Маркеры в группе модели являются GI. Имеются также разграничители, называемые *соединителями*, которые упорядочивают GI, и другие разграничители, называемые *индикаторами появления*,

которые показывают, сколько раз каждый может встречаться каждый GI.

B.4.2.2 Соединители и индикаторы появления

Соединитель используется *между* GI, чтобы показать, как они связаны. Модель учебника использует соединитель "последовательности" (***seq***), запятую. Это означает, что элементы должны следовать друг за другом в документе в той же последовательности, что и их GI встречаются в модели.

Индикатор появления используется *после* GI, к которому он применяется. Их нет в примере учебника: каждый элемент должен поэтому встречаться один и только один раз. Чтобы сделать объект начала объект конца факультативными, следует использовать вопросительный знак, который является индикатором факультативного появления (opt).

<!ELEMENT учебник (начало?, корпус, конец?)>

Пока определен только верхний уровень учебника. Определение типа должно также иметь объявления структуры для "начала", "корпуса" и "конца", затем для элементов, содержащихся в "начале", "корпусе" и "конце", и так далее до символов данных.

Для упрощения примера можно принять, что корпус представляет собой только последовательностью многих параграфов. Т.е. единственным GI, разрешенным в корпусе является "p" (для параграфа), но он может встречаться много раз. Чтобы указать множественные параграфы, GI сопровождается знаком плюс, который является индикатором появления "обязательным и повторяемым" (plus).

<!ELEMENT корпус (p+)>

Знак плюс указывает, что должен иметься по крайней мере один параграф в корпусе. Если, по каким-либо причинам, желательно позволить корпусу вообще не иметь параграфов, добавленным символ будет звездочка, которая является индикатором появления

"факультативной и повторяемый" (rep).

<!ELEMENT корпус (p*)>

Предположим, что учебник может иметь в корпусе примеры наряду с параграфами. Если GI для примера "xmp", "много параграфов или примеров" будут обозначены следующим образом:

<!ELEMENT корпус (p | xmp)+>

Вертикальная черта является соединителем "или" (***or***). Выражение "p | xmp" означает "или параграф, или пример".

Хотя группа модели содержит маркеры, сама по себе она является единичным маркером, к которому может применяться индикатор появления. Группирование в примере необходимо, поскольку индикаторы появления ("?", "+" и "*") имеют более высокий приоритет, чем соединители ("," и "|").

Поэтому, не будет эффективно говорить

<!ELEMENT корпус (p+ | xmp+)>

поскольку это означало бы "или много параграфов или много примеров", а не требуемое "много перемешанных параграфов или примеров".

Имеется еще один вид соединителя для рассмотрения. Предположим, что объект начала учебника имел титульный лист, который содержит заголовок, имя автора и издателя, но в *любом* порядке. Соединитель ***seq*** не может использоваться, поскольку он требует определенного порядка. Не может использоваться и соединитель ***or***, поскольку он выбирает только один из этих трех элементов.

Вместо этого должен использоваться соединитель амперсант ("&"), который является соединителем "и" (and). Он указывает, что все GI в группе модели должны появиться, но в любом порядке.

<!ELEMENT титульный лист (заголовок & автор & издатель)>

B.4.2.3 Ссылки на объекты в моделях

Предположим, что два элемента имеют почти одинаковые, но не идентичные модели содержания.

<!ELEMENT корпус (p | xmp | h1 | h2 | h3 | h4) + >

<!ELEMENT конец (p | h1 | h2 | h3 | h4) + >

Здесь, объекты корпуса и конца оба имеют параграфы и заголовки, но только корпус может иметь примеры.

Повторяющиеся части параметров объявления разметки могут обрабатываться со ссылками на объект, подобно повторяющемуся тексту в документе. Единственное различие заключается в том, что объектные ссылки, используемые в параметрах объявления ("ссылки на объект параметра"), начинаются со специального символа, знака процента ("%"), называемого разграничитель "ссылка на объект параметра открыта" (***pero***). ***pero*** должен также использоваться в объявлении объекта, чтобы показать, что определяемый объект является объектом параметра (но как отдельный параметр, так что он не будет ошибочно интерпретирован как ссылка).

<!ENTITY %hlto4 "h1 | h2 | h3 | h4" >

<!ELEMENT корпус (p | xmp | %hlto4;) + >

<!ELEMENT конец (p | %hlto4;) + >

B.4.2.4 Группы имен

Существует также другое использование групОбъекты корпуса и конца могут иметь одинаковое объявление структуры.

<!ELEMENT корпус (p | xmp)+>

<!ELEMENT конец (p | xmp)+>

Объем работы с клавиатурой может быть уменьшен, а подобие подчеркнуто, если позволить объявлению применяться к группе элементов.

<!ELEMENT (корпус | конец) (p | xmp)+>

B.4.2.5 Символы данных

До сих пор все рассмотренные элементы содержали только другие элементы. Однако, в конечном счете, приложение должно иметь дело с фактическими данными, т.е. местом, где больше нет тегов.

Данные могут быть указаны в модели с помощью зарезервированного имени "#PCDATA", что означает "нуль или большее число анализируемых символов данных". Поскольку оно имеет встроенный индикатор появления rep, ничто не может быть добавлено к нему явным образом. Объявление

<!ELEMENT p (#PCDATA)>

говорит о том, что параграф является строкой, состоящей из нуля или большего количества символов (включая концы записи и пробелы).

Кстати, причиной, по которой "#PCDATA" записано в верхнем регистре, является просто напоминание, что оно определено синтаксическим анализатором разметки. В конкретном синтаксисе ссылок, синтаксический анализатор разметки игнорирует регистр всех имен, кроме имен объектов.

В большинстве документов, элементы, которые содержат символы данных, могут также содержать другие тегированные элементы. Такие элементы обычно включают короткие цитирования, сноски и ссылки на рисунки, различные типы выделенных фраз и специализированные ссылки или цитаты, определенные для конкретного типа документа.

Например, структура параграфа может быть определена как:

<!ENTITY %фраза "кавычка | цитата | ссылка">

<!ELEMENT p (#PCDATA | %фраза;)*>

Эти объявления указывают, что параграф содержит символы, смешанные с элементами "фразы". Их может быть много или ни одного.

Кстати, "#" является разграничителем, называемым "индикатор зарезервированного имени" (***rni***), который используется всякий раз,

когда зарезервированное имя определено в контексте, где может также появиться созданное пользователем имя. ***rni*** показывает, что "#PCDATA" не является GI.

B.4.2.6 Пустое содержание

Может быть определен элемент, для которого пользователь никогда не вводит содержание: например, ссылка на рисунок, для которого процедура всегда будет генерировать содержание в ходе обработки. Чтобы показывать пустой элемент, для объявления содержания используется ключевое слово вместо обычной группы модели в скобках:

<!ELEMENT ссылка_на_рисунок EMPTY>

Элемент, который объявлен пустым, не может иметь конечный тег. (Он и не является необходимым, поскольку отсутствует содержание, которое должно им заканчиваться.)

B.4.2.7 Данные, не относящиеся к SGML

Если корпус учебника содержит фотографии, объявление элемента может иметь вид

<!ELEMENT корпус (p | фото)+>

Фотография обычно представляется строкой битов, которые замещают цвета в различных точках изображения. Эти битовые комбинации не имеют то же значение, что символы в тексте и в разметке; они должны интерпретироваться как уникальная нотация, которая не определяется SGML.

Поскольку синтаксический анализатор разметки не просматривает данные, не относящиеся к SGML, они должны сохраняться в отдельном объекте, чье имя задается специальным атрибутом (будет рассмотрен позже). Содержание элемента будет пусто.

<!ELEMENT фото EMPTY>

B.4.2.8 Сводка разграничителей модели

В приведенном ниже списке сведены воедино разграничители, используемые в моделях, и их строки символов в наборе разграничителей ссылок:

—Группирование:

Строка	**Имя**	***Значение***
(	**GRPO**	Открывает группу. Выражение внутри группы обрабатывается как модуль для других операций.
)	**GRPC**	Закрывает группу.

—Индикаторы появления:

Строка	**Имя**	***Значение***
?	**OPT**	Факультативный: может появляться 0 или 1 раз.
+	**PLUS**	Обязательный и повторяемый: должен появляться 1 или большее число раз.
*****	**REP**	Факультативный и повторяемый: может появляться 0 или большее число раз.

—Соединители:

Строка	**Имя**	***Значение***
,	**SEQ**	Все соединенные элементы должны появиться в документе в той же последовательности, что и в группе модели.
|	**OR**	Должен появиться один и только один из соединенных элементов.
&	**AND**	Все соединенные элементы должны появиться в

документе, он в любом порядке.

—Прочие:

Строка	**Имя**	***Значение***
#	**RNI**	Идентифицирует зарезервированное имя для его различения с именем, определенным пользователем.

B.5 Атрибуты

Описательный тег обычно включает родовой идентификатор (GI) элемента и может включать также атрибуты. GI обычно существительное; атрибуты - существительные или прилагательные, которые описывают существенные характеристики GI. (Использование глаголов или параметров форматирования, которые являются процедурными, нежели описательными, категорически осуждается, поскольку противоречит цели обобщенной разметки.)

Специальные атрибуты, допустимые для данного элемента, определяются определением типа, которое также определяет диапазон значений, которые может иметь атрибут, и значения по умолчанию.

B.5.1 Определение атрибутов

Пример тега с двумя атрибутами показан на рис. 10.

Атрибуты следуют за GI в начальном теге. Каждый атрибут определяется присвоением ему имени, разграничителя индикатора значения (***vi***) и *значения* атрибута. В приведенном примере, атрибуту, названному "защита", было присвоено значение "Внутреннее использование", а атрибуту "отправитель" было присвоено значение "LTG".

B.5.1.1 Имена

Имена атрибутов, такие как "защита" и "отправитель" в приведенном примере, совместно с другими видами имен языка разметки используют некоторые уже встречавшиеся свойства, такие как имена и GI объектов. Имя должно состоять только из тех символов, которые были обозначены как "символы имен", и должно начинаться с одного из подмножества символов имен, называемого символы "начала имен".

Обычно (т.е. в конкретном синтаксисе ссылок), символами имен могут быть буквы, цифры, точка и дефис, а начальными символами имен могут быть только буквы. Буквы нижнего регистра в имени обычно обрабатываются, как если бы они были буквами верхнего регистра, так что не имеет значения, в каком регистре было набрано имя. В именах объектов, тем не менее, регистр существенен.

Имя обычно имеет максимальную длину восемь символов, но система может определять иной *набор количеств*, в котором длина имени и другие количественные характеристики языка могут отличиться от конкретного синтаксиса ссылок.

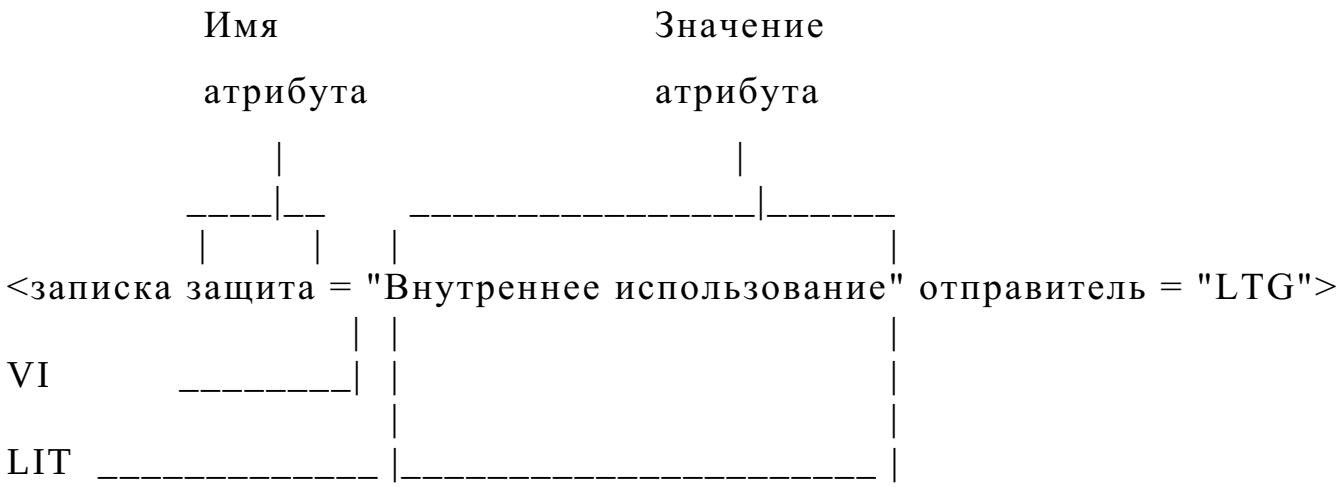

Рис. 10. Начальный тег с двумя атрибутами

B.5.1.2 Значения атрибута

Значение атрибута состоит из символов данных и ссылок на объект, ограниченных разграничителями, называемыми "разграничители литерала" ***(lit),*** которые обяыно являются кавычками ("). Альтернативные разграничители литерала ***(lita),*** обычно апострофы ('), также могут использоваться, но два типа не могут быть соединены друг с другом в паре.

Пустое значение атрибута указывается двумя последовательными разграничителями lit:

<список имя=Джоунз телефон="555-1234" другой_телефон="">

Концы записи и позиции табуляции в значении атрибута заменяются пробелами. Символы начала записи игнорируются (т.е. удаляются).

В приведенном ниже списке сведены воедино роли разграничителя и их назначения строк в наборе разграничителей ссылок:

Строка	**Имя**	***Значение***
=	**VI**	Значение индикатор
"	**LIT**	Разграничитель литерала
'	**LITA**	Разграничитель литерала (альтернативный)

B.5.2 Объявление атрибутов

Для каждого элемента определение типа документа должно содержать информацию, которая устанавливает атрибуты элемента. Это делается с помощью объявления списка определений атрибутов. Если список определений атрибутов для элемента отсутствует, то элемент просто не имеет атрибутов.

B.5.2.1 Синтаксис определения атрибута

Объявление списка определений атрибутов начинается с *типа связанного элемента*, к которому применяется список определений атрибутов. Объявление может определить единственный GI или их группу, при этом одинаковые атрибуты применяются ко всем членам группы.

Для каждого атрибута объявление также включает определение атрибута, состоящее из

a) имя атрибута,

b) его допустимые значения (*объявленное значение*), и

c) значение по умолчанию, которое будет использоваться, если атрибут не определен в документе, и действует минимизация разметки опущенным тегом (будет описана позже).

Список атрибутов имеет большее количество параметров, чем объявления, которые рассматривались до сих пор. При работе с длинными объявлениями к ним полезно добавлять пояснительные комментарии. В объявлении разметки комментарии могут появляться между любыми параметрами, или объявление может состоять исключительно из комментариев.

Комментарии начинаются и заканчиваются двумя дефисами, "разграничителем комментария" (зерно). Один из видов использования для комментариев - помещение заголовка над параметрами:

<! -- ЭЛЕМЕНТЫ ИМЯ ЗНАЧЕНИЕЗНАЧЕНИЕ ПО УМОЛЧАНИЮ -- >

Список определений атрибутов требует, чтобы некоторые из параметров были повторены для каждого атрибута элемента. Поскольку объявления могут охватывать границы записей, при желании может использоваться табличная форма ввода. Ниже приведены объявления в табличной форме для элемента записки на рис. 10:

```
                        <! -- СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT записка       (от кого, кому, тема, текст, подпись, копия?)>
<!--      ЭЛЕМЕНТЫ      ИМЯ         ЗНАЧЕНИЕ      ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST записка       статус      (оконч|проект) "оконч"
                        защита      CDATA          ≠REQUIRED
                        версия      NUMBER         "01"
                        отправитель NAME           ≠IMPLIED
>
```

Значение списка определений атрибутов следующее:

статус Атрибут "статус" должен иметь значение "оконч" или "проект". Поскольку атрибут не был определен на рис. 10, синтаксический анализатор будет действовать, как если бы было введено значение по умолчанию "оконч".

защита Атрибут "защита" имеет в качестве значения строку из нуля или большего числа символов. Ключевое слово "REQUIRED" в заданном по умолчанию параметре указывает, что атрибут всегда должен определяться в документе.

версия Значением атрибута "версия" должна быть строка из 1 или большего числа цифр. Значение по умолчанию - "01" строка.

отправитель Атрибут "отправитель" должен иметь в качестве значения синтаксически действительное имя SGML. Ключевое слово "IMPLIED" указывает, что атрибут является факультативным, и что значение будет даваться приложением, если атрибут не определен в документе. (На рис. 10 оно было определено как "LTG".)

Обратите внимание, что разграничители могут быть опущены в значении по умолчанию, когда оно составлено полностью символов из имен. (Ключевые слова параметра значения по умолчанию начинаются с ***rni****, поэтому* они не могут быть перепутаны с именами, определяемыми приложением.) В вышеупомянутом примере значения по умолчанию "оконч" и "01" могли быть введены без разграничителей.

Поскольку была затронута тема комментариев, следует также отметить, что пустое объявление разметки (" <! > ") также является комментарием. Оно может использоваться, чтобы отделить части документа - источника без опасности неправильной интерпретации пустой записи как использования функции "краткой ссылки" (будет обсуждаться позже), каковой она может быть.

B.5.2.2 Комплексные значения атрибутов

Значение атрибута может логически состоять из нескольких элементов. Например, иллюстрация вывода, отображаемого компьютером, может иметь два цвета, связанные с ней: зеленый и черный. Поскольку атрибут может быть определен в теге только один раз,

<отображение цвет=черный цвет=зеленый>

не может быть введен, поскольку это будет ошибка.

Однако, объявляя значение параметра "NAMES" для атрибута "цвет", оба цвета могут быть определены в единственном значении атрибута:

<!ELEMENT отображение (p +) >

<!ATTLIST отображение цвет NAMES "белый черный">

<!>

<отображение цвет= "черный зеленый">

Ключевое слово "NAMES" означает "список из одного или большего числа имен SGML, разделенных пробелами, концами записи,

началами записи, или символами горизонтальной табуляции". Некоторые другие объявленные ключевые слова значений, которые разрешают списки, приведены ниже:

NUMBERS список из одного или большего числа номеров.

NMTOKENS список из одного или большего числа маркеров имен (подобны именам, но они не должны обязательно начинаться с начального символа имени, например, " -abc 123 12.3 123a.123").

NUTOKENS список из одного или большего числа маркеров номеров (подобны маркерам имени, но первым символом должна быть цифра: например, " 123 12.3 123a 0.123", но не ".123 456").

Сингулярные формы "NMTOKEN" и "NUTOKEN" могут использоваться, только когда разрешен единственный маркер. (Обратите внимание, что ***rni*** не является обязательным для объявленных ключевых слов параметра значения, поскольку определяемое пользователем имя не может быть определено для этого параметра.)

Комплексного атрибута можно избежать в этом случае, определяя два отдельных атрибута:

<!ELEMENT отображение (p+)>

<!ATTLIST отображение цвет_фона NAME "белый" цвет_текста NAME "зеленый">

<!>

<отображение цвет_фона=черный цвет_текста=зеленый>

B.5.2.3 Группы маркеров имени

Значение атрибута может быть сокращено до элемента группы уникальных имен или маркеров имен, называемой *группой маркеров имени*:

```
                                <! -- СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT записка          (от кого, кому, тема, текст, подпись, копия?)>
<!--          ЭЛЕМЕНТЫ  ИМЯ        ЗНАЧЕНИЕ      ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST  записка       статус    (оконч|проект)  "оконч"
>
```

С учетом приведенного выше объявления, может быть определен либо

<записка статус="проект">

либо

<записка статус="оконч">

Любое другое значение для атрибута "статус" будет некорректным.

Маркер имени может появляться только однажды в списке определений атрибутов, который применяется к единственному типу элемента. Он транслируется в верхний регистр таким же образом, как и имя.

B.5.2.4 Изменение значений по умолчанию

Если значение по умолчанию определено как "CURRENT", оно автоматически станет последним указанным значением. Это позволяет значению атрибута быть "унаследованным" по умолчанию от предыдущего элемента того же типа. (Этот эффект должен быть учтен при добавлении или удалении тегов, которые определяют текущий атрибут.)

B.6 Объекты

Ссылки на объект и ссылки на объект параметра активно использовались в некоторых из уже появившихся примеров. Хотя объект обладает поверхностным сходством с переменной языка программирования, фактически он является частью документа, и в таком качестве он представляет собой константу. Ссылки разрешают использование нескольких полезных методов:

a) Может использоваться краткое имя для обращения к длинной или текстовой строке, или к строке, которая не может быть легко введена с помощью доступной клавиатурой.

b) Могут быть вставлены части документа, которые хранятся в отдельных системных файлах.

c) Может легче осуществляться обмен документами между различными системами, поскольку ссылки на системно-специфические объекты (такие как символы, которые не могут быть введены непосредственно) могут представляться в форме ссылок на объект, которые разрешаются получающей системой.

d) Результат динамически выполняемой команды обработки (такой как команда выборки текущей даты) может быть вставлен как часть документа.

B.6.1 Синтаксис ссылок на объект

Имеются два вида поименованной ссылки на объект. Ссылка на общий объект может использоваться где-либо в содержании элементов и значениях разграниченных атрибутов. Ссылки на объект параметра могут использоваться в параметрах объявления разметки, которые разграничены разграничителями ***lita*** или ***lit***. Они могут также использоваться, чтобы ссылаться на последовательные полные параметры или маркеры группы с вставленными разделителями.

Ссылка на общий объект представляет собой имя, разграниченное разграничителями "ссылка на объект открыта" (***ero***), в качестве которого обычно используется амперсанд, и "ссылка закрыта" (***refc***), в качестве которого обычно используется точка с запятой:

напечатано в &место; сего

Если за ссылкой на объект следует пробел или конец записи, refc может быть опущен:

напечатано в &место сего

Ссылка на объект параметра такая же, за исключением того, что она начинается с разграничителя "ссылка на объект параметра открыта" (pero), в качестве которой обычно используется знак процента. (Кстати, "обычно", в контексте строк разграничителя означает "в наборе разграничителей ссылок".)

Различие между общими объектами и объектами параметра организовано таким образом, чтобы разработчик документ мог составлять имена объектов, не зная, использовались ли те же имена персоналом поддержки, который создавал объявления разметки для типа документа.

В приведенном ниже списке сведены воедино разграничители, используемые в ссылках на объект и их строках символов в конкретном синтаксисе ссылок:

Строка	**Имя**	***Значение***
&	**ERO**	Ссылка на объект открыта
%	**PERO**	Ссылка на объект параметра открыта
;	**REFC**	Ссылка закрыта

B.6.2 Объявление объектов

Прежде, чем на объект можно будет сослаться, он должен быть объявлен с помощью объявления объекта. Существуют два основных параметра: имя объекта и текст объекта. Объявление

<!ENTITY uta "United Typothetae of America">

означает, что ссылка на имя "uta" (т.е. "&uta;") в документе будет эквивалентна вводу текста "United Typothetae of America". Текст объекта разграничен разграничителями lit (или lita) (подобно значению атрибута), и называется "литерал параметра".

Объект не обязательно начинается началом записи или заканчивается концом записи. Если требуется, чтобы границы записи окружали объект, они должны быть помещены вокруг ссылки на объект.

Источник

The &uta; - издательская организация.

разрешается в

The United Typothetae of America - издательская организация.

тогда как источник

Печатающие организации:

&uta;

Общество Научных, Технических и Медицинских Издателей

разрешается в

Печатающие организации:

United Typothetae of America

Общество Научных, Технических и Медицинских Издателей

Однако, для завершения ссылки на объект используется конец записи, а не чем refc, к объекту присоединяется следующая запись.

&uta

, Inc.

разрешается в

United Typothetae of America, Inc.

B.6.2.1 Команды Обработки

Команда обработки может быть сохранена как объект. Она будет игнорироваться при создании объекта, но выполняться, когда встречается ссылка на объект.

<!ENTITY страница PI "новая_страница; пробел 3" >

Ключевое слово "PI" указывает, что объект будет интерпретироваться при ссылке как команда обработки.

B.6.2.2 Объекты со ссылками на объект

Объект параметра объявляется указанием ***pero*** в качестве первого

параметра перед именем объекта

<!ENTITY % маркер "o" >

На объекты параметра можно ссылаться внутри литерала параметра:

<!ENTITY префикс "%маркер; " >

Ссылка на "%маркер; " разрешается, когда объявляется объект "префикс". Она не разрешается в каждой ссылке на объект "префиксный".

B.6.2.3 Внешние объекты

Во многих системах обработки текста существуют множественные классы хранения, такие как файлы, библиотечные элементы, макроопределения и символы для текстовых строк. Такие системные зависимости могут храниться вне тела документа, путем ссылок на внешние объекты памяти как на объекты SGML:

<!ENTITY часть2 SYSTEM>

Если имени объекта не достаточно, чтобы дать возможность системе идентифицировать объект памяти, может быть определена дополнительная информация (называемая "системный идентификатор"):

<!ENTITY часть2 SYSTEM "user.sectionX3.textfile">

Системный идентификатор разграничивается тем же способом, что и литерал параметра. Характер и синтаксис системного идентификатора зависят от компонента системы SGML, который называется *менеджером объектов*, чья работа заключается в преобразовании ссылок на объекты в реальные системные адреса.

B.6.2.4 Публичные объекты

Внешний объект, который известен вне контекста отдельного документа или системной среды, называется "публичный объект". Ему присваивается "публичный идентификатор" международным, национальным или отраслевым стандартом или просто сообществом

пользователей, которые желают его совместно использовать.

Одним из применений публичных объектов являются определения типа общедоступного документа. Другим могут быть общедоступные "наборы объектов" объявлений объекта, которые поддерживают графические символы и терминологию специализированных областей, таких как математика или химия.

Публичные объекты объявляются так же, как и другим внешние объекты, за исключением того, что "спецификация публичного идентификатора " заменяет ключевое слово "SYSTEM":

<!ENTITY % ISOgrkI

PUBLIC "ИСО 8879-1986 //ENTITIES Греческие буквы //EN" >

Спецификация состоит из ключевого слова "PUBLIC", публичного идентификатора, который разграничен как литерал, и факультативного системного идентификатора (в примере опущен). Публичный идентификатор может содержать только буквы, цифры, пробелы, концы и начала записи, и некоторые специальные символы; все вместе они известны как символы "минимальных данных".

B.7 Символы

Каждый символ в документе занимает позицию в *наборе символов* документа. Общее количество позиций зависит от размера *кодового набора*; т.е. числа двоичных цифр ("биты"), используемых для представления каждого символа.

Например, набор символов, известный как Международная Эталонная Версия ИСО 646 (ИСО 646 IRV) представляет собой 7 - разрядный набор. Существуют 128 допустимых *битовых комбинаций* с 7 битами, в диапазоне десятичных значений от 0 до 127. В 8-разрядных наборах возможно 256 битовых комбинаций. Номер позиции, или *номер*

символа, является целым десятичным эквивалентом битовой комбинации, которая представляет символ.

Возможно также использовать "методы расширения кода", в которых битовая комбинация может представлять больше чем один символ. Использование таких методов с SGML обсуждается в приложении E.3.

B.7.1 Классификация символов

Многие наборов символов были определены, чтобы включить множество национальных алфавитов, научных нотаций, клавиатур, устройств отображения и систем обработки. В каждом из них последовательность разрядных комбинаций отображается на различные алфавиты цифр, букв и других символов. Любой набор символов, достаточно большой, чтобы представлять символы разметки (символы имен, разграничители и символы функций) и символы минимальных данных, может использоваться в документе SGML.

SGML классифицирует символы следующим образом:

символы функций Конец записи и начало записи, которые уже разъяснялись, являются символами функций, как и пробел. Конкретный синтаксис ссылок добавляет символ горизонтальную табуляции (***TAB***), который используется, чтобы отделить маркеры в объявлениях разметки и тегах (наряду с символами пробела, ***RS*** и ***RE***). Символы функций могут также служить как символы данных и могут появляться в разграничителях краткой ссылки (будут рассматриваться позже).

символы имен Это символы, которые могут использоваться в имени. Они всегда включают прописные и строчные буквы от A до Z и цифры от 0 до 9; конкретный синтаксис ссылок

добавляет точку и дефис. Каждый символ имени (не являющийся цифрой и прописной буквой) имеет связанную форму верхнего регистра, которая используется для замены регистра в именах (за исключением имен объектов в конкретном синтаксисе ссылок). Подмножество символов имен, называемых *начальными символами имен*, состоит из строчных и прописных букв, плюс любых других символов, которыми конкретный синтаксис позволяет начинать имя.

набор разграничителей

Эти символы в различных контекстах предписывают рассматривать текст как разметка, а не как данные.

не относящиеся к SGML символы

Это символы, которые набор символов документа идентифицирует символы, не встречающиеся в объектах SGML, выбранные частично из кандидатов, указанных конкретным синтаксисом. Конкретный синтаксис ссылок, например, классифицирует таким образом управляющие символы (за исключением некоторых, которые используются как символы функций). Не относящиеся к SGML символы встречаются в не относящихся к SGML данных (таких как изображения), которые находятся во внешних объектах, и в "оболочке" файловых систем, потоков данных и т.д., которые содержат или передают документ. Система может также использовать их для своих собственных целей, таких как дополнение или разграничивание в ходе обработки, так как не имеется возможности перепутать их с символами разметки или данных.

символы данных Все другие символы, такие как знаки пунктуации и математические символы, которые не используются как разграничители, являются символами данных. (Символы разметки также могут быть данными, когда они встречаются в контексте, в котором они не распознаются как разметка. Такие данные называются "анализируемые символьные данные".)

B.7.2 Ссылки на символ

Редко бывает удобно или даже возможно вводить каждый символ непосредственно:

a) На устройстве ввода может отсутствовать соответствующая клавиша.

b) Он может не отображаться.

c) Он может быть не относящимся к SGML символом, который не может встречаться непосредственно, как символ данных или разметка.

d) Это может быть символ функции, который Вы хотите обрабатывать как данные, а не использовать его SGML функцию.

Для таких ситуаций существует метод, именуемый "ссылка на символ", который позволяет косвенно ввести символ. (Для наглядности в приведенных ниже иллюстрациях будет использоваться символ, который может быть введен с клавиатуры, дефис.) Два объявления в следующем примере эквивалентны в любом наборе символов, в котором дефис является номером символа 45:

```
<!ENTITY дефис "-" >
<!ENTITY дефис "&#45;" >
```

В объявлениях объектов, проиллюстрированных ранее, литералы содержали только *символьные данных* как текст объекта. В показанном выше примере, второе объявление содержит ссылку на символ. Ссылка

на символ начинается с разграничителя "ссылка на символ открыта" (&#) и заканчивается разграничителем ***refc***, но вместо имени она содержит номер символа.

Литерал может содержать как символьные данные, так и ссылки на символ. Следующие объявления также эквивалентны:

<!ENTITY конец "-|-">

<!ENTITY конец "-|-">

Ссылки на символ могут также быть введены непосредственно в значениях атрибутов и содержании данных, также как и ссылки на объект.

На символы функций можно ссылаться поименованными ссылками на символ: &#RS;, &#RE;, &#SPACE; и &#TAB;. Эта форма ссылки используется, когда требуется функция символа; форма номера символа используется, чтобы обойти функцию и ввести символ как данные.

В следующем списке сведены воедино разграничители, используемые в ссылках на символы и их строки символов в конкретном синтаксисе ссылок:

Строка	**Имя**	***Значение***
&#	***CRO***	Ссылка на символ открыта
;	***REFC***	Ссылка закрыта

B.7.3 Использование символов - разделителей как данных

Стандартный Обобщенный Язык Разметки не обязывает использовать конкретные символы как разграничители. Напротив, он определяет *роли* разграничителей как часть "абстрактного синтаксиса" и позволяет назначать им строки символов как часть определения конкретного синтаксиса. Хотя существует много таких ролей, возможность пользователя свободно вводить символы данных по существу не ограничена, поскольку:

— Большинство разграничителей встречаются только в пределах

объявлений разметки или тегов; только некоторые из них встречаются в содержании элементов.

— Одни и те же символы используются больше чем для одной роли.

— Роли разграничителя, значимые в содержании, являются контекстными; они распознаются, только когда сопровождается соответствующей контекстной последовательностью (или допускаются иным образом). Например, ***ero*** распознается только тогда, когда за ним следует начальный символ имени.

— Большинство разграничителей являются многосимвольными строками, которые уменьшают вероятность их появления как данных.

— Многосимвольный разграничитель или "контекстный разграничитель" не будет распознаваться, если в его пределах начинается или кончается объект.

Благодаря последнего пункту всегда можно избежать неоднозначности, используя ссылку для ввода символа данных. Для большинства ситуаций в наборе разграничителей ссылок необходимы только два объекта, и еще три будут использоваться в частных случаях:

<!ENTITY амперсант "&">

<!ENTITY меньше_чем "<">

Ссылки на объекты "&амперсант;" и "&меньше_чем;" могут свободно использоваться свободно в тексте всякий раз, требуется использовать амперсант или знак "меньше чем", поскольку использование ссылки прекращает контекстный разграничитель.

<!ENTITY правая_квадратная_скобка "]">

Объект правой квадратной скобки может использоваться аналогичным образом, чтобы избежать распознавания конца отмеченного раздела (будет рассматриваться позже).

<!ENTITY косая_черта CDATA "/" >

Косая черта является действительным разграничителем, только когда используется функция SHORTTAG (будет объяснена позже), и даже тогда она должна быть специально разрешена для элемента. Поскольку разграничитель является одиночным символом и не требует никакой контекстной последовательности, сама ссылка на объект не достаточна, чтобы обеспечить, его обработку как данные. Поэтому определено ключевое слово "CDATA"; оно заставляет обрабатывать текст объекта как символьные данные, даже если он может напоминать разграничитель.

<! ENTITY кавычки '"' >

Объект "&кавычки;" (кавычки) необходим только в редких случаях, когда оба разграничителя ***lit*** и ***lita*** встречаются как данные в одном и том же литерале. Обычно, если литерал содержит символ ***lit***, он разграничен с помощью разграничителей ***lita***, и наоборот.

B.8 Отмеченные разделы

Отмеченный раздел представляет собой раздел документа, который вводится как параметр объявления *отмеченного раздела*, чтобы маркировать его для специальной цели, такой как выключение в нем разграничителей или игнорирование раздела в ходе выполнения некоторых видов обработки.

B.8.1 Игнорирование отмеченного раздела

Если документ обрабатывается двумя различными системами, команда обработки, которая применяется к одной из них, не будет понята другой. Синтаксический анализатор разметки может быть сделан таким образом, чтобы игнорировать одну из команд, если раздел документа, который ее содержит, отмечен как раздел, который нужно

игнорировать:

<![IGNORE [<?команда для системы A>]]>

Объявление отмеченного раздела в примере состоит из пяти частей:

a) *начало отмеченного раздела,* которое состоит из разграничителя ***mdo,*** за которым следует разграничитель "подмножество объявления открыто" (***dso***), обычно левая квадратная скобка;

<![

b) *ключевые слова,* в данном случае единственное ключевое слово "IGNORE";

IGNORE

c) другой разграничитель " подмножество объявления открыто" (***dso***) для указания начала содержания отмеченного раздела;

d) содержание;

<?команда для системы A>

e) и конец томеченного раздела, состоящий из разграничителя "отмеченный раздел закрыт" ***(msc),*** обычно две правых квадратных скобки (для уравнивания двух разграничителей ***dso***), за которым следует ***mdc***.

]]>

Команда обработки для другой системы может быть помечена как подлежащая включению:

<![INCLUDE [

<? команда для системы B>

]]>

Передача документа системе A требует обмена ключевыми словами состояния для двух разделов. Наиболее просто это может быть сделано, используя два объекта параметра, по одному для каждой системы обработки. Один объект должен содержать строку символов "IGNORE", а другой - "INCLUDE":

<!ENTITY %система_a "IGNORE" >

<!ENTITY %система_b "INCLUDE" >

Ключевое слово "IGNORE" не будет использоваться непосредственно в любом объявлении отмеченного раздела. Вместо этого, будет ссылка на один из двух системно - зависимых объектов. Учитывая предыдущие объявления, команда для "системы А" в следующем примере будет игнорироваться, в то время как для "системы В" она будет выполняться:

<![%система_a;[<?команда для системы А>]]>

<![%система_b;[<? команда для системы В>]]>

Каждый другой отмеченный раздел документа, который ссылается на "%система_a;" или "%система_b;", будет обрабатываться аналогичным образом.

Теперь, если два объявления объектов будут зарезервирован следующим образом

<!ENTITY %система_a "INCLUDE">

<!ENTITY %система_b "IGNORE">

каждый отмеченный раздел документа, который ссылается на "%система_b;" будет игнорироваться.

Обратите внимание, что даже при том, что содержание раздела игнорируется, осуществляется достаточный синтаксический анализ, чтобы распознать начала и концы вложенных отмеченных разделов, так что для раздела будет использоваться правильный конец.

B.8.2 Версии одиночного документа

Документ может издаваться в множественных версиях, чьи тексты слегка отличаются. SGML позволяет обрабатывать все версии такого документа без дублирования текст общих частей. Текст, зависящий от версии, вводится в отмеченные разделы, используя описанный выше способ, тогда как общий текст остается вне их:

<![%v1; [текст для версии 1]]>

<![%v2; [текст для 2-й версии]]>

общий текст для обеих версий

<![%v1; [дополнительный текст для версии 1]]>

<![%v2; [дополнительный текст для 2-й версии]]>

Теперь если будут определены следующие объявления объектов:

<!ENTITY % vl "INCLUDE" >

<!ENTITY % v2 "IGNORE" >

версия 1 будет обрабатываться следующим образом:

текст для версии 1

общий текст для обеих версий

дополнительный текст для версии 1

Если зарезервированы объявления объектов:

<!ENTITY % vl "IGNORE" >

<!ENTITY % v2 "INCLUDE" >

версия 2 будет обрабатываться следующим образом:

текст для 2-й версии

общий текст для обеих версий

дополнительный текст для 2-й версии

B.8.3 Не анализируемые разделы

Отмеченный раздел может быть маркирован как не подлежащий синтаксическому анализу:

<![CDATA [

<?команда>

Параграф со ссылкой &объект_x;

которая не распознается.

<?команда>

]]>

Содержание раздела обрабатывается как символьные данные,

поэтому две команды обработки и ссылка на объект как таковые не будут распознаваться или обрабатываться.

Если существует необходимость разрешения ссылок на объекты и ссылок на символы, игнорируя другие разграничители, содержание может рассматриваться как заменяемые символьные данные подобно следующему примеру:

```
<! [ RCDATA [
<?команда>
<p>Параграф со ссылкой &объект_x;
которая не распознается.</p>
<?команда>
]]>
```

Отмеченные разделы CDATA и RCDATA не являются вкладываемыми, как отмеченные разделы IGNORE. Первый конец отмеченного раздела завершает их.

B.8.4 Временные разделы

Для легкой идентификации и удаления отмеченный раздел может быть маркирован как временный:

```
<![TEMP[<?новая_страница>]]>
```

Такие разделы полезны для временных "фиксаций", когда процессор работает не совсем верно.

B.8.5 Спецификации ключевых слов

Чтобы облегчить ввод ключевых слов со ссылками на объект, одновременно могут использоваться четыре ключевых слова состояния и "TEMP" могут использоваться, при этом разрешаются дублирования. Это позволяет использовать множественные ссылки на объект в одиночном объявлении отмеченного раздела без возможности ошибки, вызываемой их разрешением в конфликтующие или дублированные ключевые слова.

Ключевые слова состояния расположены по приоритетам следующим образом:

IGNORE

CDATA

RCDATA

INCLUDE

Если ключевое слово не указано, предполагается "INCLUDE".

B.8.6 Определение отмеченного раздела как объекта

Отмеченный раздел может быть определен как объект, поэтому он может быть встроен во многие места документа:

<!ENTITY фраза1 MS

"RCDATA[

повторная фраза с

примером <tag>

]">

С учетом этого объявления, строка ввода

Это &phrase1; в ней.

будет разрешаться следующим образом

Это <![RCDATA[

повторная фраза с

примером <tag>

]]> в ней.

"<tag>" не распознается как разметка из-за ключевого слова "RCDATA" в объявлении отмеченного раздела.

Обратите внимание, что перед отмеченным разделом нет начала записи, как и конца записи после него. Следующий пример приведет к их появлению:

Это

&фраза1;

в ней.

Отмеченный раздел как таковой не анализируется или не обрабатывается, если определен объект. Начало отмеченного раздела и конец отмеченного раздела добавляются автоматически.

B.9 Атрибуты уникального идентификатора

Уникальный идентификатор ("ID") - атрибут, который именует элемент, чтобы отличать его от всех других элементов. Синтаксический анализатор разметки SGML может выполнять обычную обработку для атрибутов ID, таким образом минимизируя работу по реализации процедур.

Назначение ID состоит в том, чтобы позволить одному элементу ссылаться на другой: например, чтобы позволить ссылке на рисунок ссылаться на рисунок. Обычно, процедура для рисунка должна сопоставить некоторые данные с ID (таким как номер рисунка). Процедура ссылки на рисунок производит выборку данных и их печать.

Хотя синтаксический анализатор разметки обычно не осведомлен о значении специфических атрибутов, ему можно сообщить, когда атрибут является уникальным идентификатором:

```
<!--          ЭЛЕМЕНТ        СОДЕРЖАНИЕ          -->
<!ELEMENT     figure         (figbody, figcap)>
<!--          ЭЛЕМЕНТ  ИМЯ   ЗНАЧЕНИЕ            ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST     figure   id    ID                   #IMPLIED>
```

Для элемента допускается только один атрибут ID.

Значение атрибута уникального идентификатора должно быть именем, которое отличается от значения атрибута любого другого ID в документе. Как и для большинства имен используется форма верхнего регистра независимо от того, как оно было введено.

Элемент, который делает ссылку, должен иметь атрибут "ссылки уникального идентификатора ", обозначенный в объявляемом значении ключевым словом "IDREF":

```
<!--          ЭЛЕМЕНТ       СОДЕРЖАНИЕ          -->
<!ELEMENT     figref                EMPTY>
<!--          ELEMENT  ИМЯ  ЗНАЧЕНИЕ            ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ -->
<!ATTLIST     figref   Refid IDREF              #IMPLIED>
```

Значение атрибута ссылки ID должно быть именем которое совпадает с именем атрибута ID какого-либо элемента. Как обычно, используется форма верхнего регистра.

B.10 Атрибуты ссылок на содержание

Некоторые документы форматируются и печатаются как отдельные главы. Отдельное форматирование предотвращает автоматические ссылки на рисунки рисунка через границы глав, хотя они по-прежнему возможны в пределах главы. Определение элемента может поддерживать два типа ссылки на рисунки, определяя, что иногда содержание может быть пусто (случай ссылки внутри главы), а в других случаях оно будет вводиться явно (случай ссылки между главами).

Условие пустоты - спецификация атрибута, который обозначен в определении атрибута как атрибут "ссылки на содержание". (Все остальное с этим атрибутом подчиняется обычному порядку.) Обозначение осуществляется путем ввода ключевого слова "#CONREF" как значения по умолчанию:

```
<!--          ЭЛЕМЕНТ              СОДЕРЖАНИЕ          -->
<!ELEMENT     figref               (fignum, #PCDATA)   >
<!--          ELEMENT  ИМЯ         ЗНАЧЕНИЕ            ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ -->
<!ATTLIST     figref   refid       IDREF               #CONREF>
```

Ключевое слово означает, что атрибут является атрибутом ссылки на содержание, а также что это подразумеваемый атрибут (такой же, как если бы значением по умолчанию было "#IMPLIED").

В следующем примере первая "figref" (ссылка на рисунок) имеет пустое содержание, тогда как вторая имеет как "fignum" (номер рисунка), так и строку символов:

Это текст с генерируемой ссылкой на рисунок
<figref refid=figdavis>, а также введенной пользователем
ссылкой на<figref><fignum>A-l</fignum> в
Приложении</figref>.

Первая "figref" не имеет (и не может иметь) конечный тег, поскольку элемент был идентифицирован как пустой явной ссылкой на содержание.

Элемент, который оказывается пустым только потому, что его модель содержания факультативна и не введено никакое содержание, не обрабатывается таким образом. Такой случайно пустой элемент должен иметь конечный тег (если только не используется минимизация разметки, как будет объяснено позже), поскольку невозможно сказать по начальному тегу, является ли экземпляр элемента фактически пустым.

Было бы бессмысленно (и поэтому запрещено) определять атрибут ссылки на содержание для элемента, который объявлен пустым.

Суммируя вышесказанное (и слегка расширяя): атрибуты ссылки на содержание могут быть обозначены для любого типа элемента, который не объявлен пустым. Когда один или большее число таких атрибутов имеет явное значение, указанное в начальном теге, этот экземпляр элемента рассматривается пустым, и для него конечный тег не разрешается.

B.11 Исключения в модели содержания

Модель содержания определяет элементы, которые возникают на верхнем уровне содержания (подэлементы). Однако, существуют случаи, когда может быть необходимо сослаться на элементы, которые находятся на нижних уровнях. Для этой цели используется

факультативный параметр объявления элемента, называемый параметром "исключений".

B.11.1 Включенные элементы

В индексированной книге элементы записей для индекса могут быть рассеяны где-либо в источнике. В ходе форматирования процедуры будут их собирать, сортировать и объединять наряду с их номерами страниц, и печатать индекс.

Громоздко выражать соотношение книги с ее индексными записями обычным способом, поскольку записи должны быть включены почти в каждую группу модели. Вместо этого может использоваться параметр исключений:

```
<!-- СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)? -->
<!ELEMENT учебник (начало?, корпус, конец?) +(запись)>
```

Знак "плюс" является разграничителем plus . Здесь это означает, что индексные записи могут появляться в любом месте учебника, даже среди символов данных. Эта часть параметра исключений называется группой "включений".

(Кстати, вопросительный знак в комментарии заголовка является напоминанием о том, что параметр исключений является факультативным.)

B.11.2 Выключенные элементы

Может быть желательно предотвратить обнаружение некоторого элемента на любом уровне определяемого элемента. Например, рисунки могут защищаться от вкладывания следующим объявлением:

```
<!--          СОДЕРЖАНИЕ  ЭЛЕМЕНТОВ                 -->
                          (ИСКЛЮЧЕНИЯ)?
<!ELEMENT     fig         (figbody, figcap?) -(fig)  >
<!ELEMENT     figbody     (artwork | p+)             >
<!ELEMENT     p           (#PCDATA | fig)            >
>
```

Модель содержания ясно не позволяет рисунку появляться на

верхнем уровне другого рисунка, но тело рисунка может содержать параграф, который может содержать рисунок. Параметр исключений с префиксом ***minus*** (дефис) предотвращает такие появления; это называется группой "изъятий".

GI может быть в группе изъятий, только если его маркер во всех затрагиваемых группах модели имеет индикатор появления ***opt*** или ***rep***, или находится в группе ***or*** или соответствующей группе включений. Другими словами, Вы не можете исключать что - либо с помощью группы изъятий, если группа модели требует наличия этого в документе. Из этого следует, что Вы не можете использовать группу изъятий, чтобы изменить обязательный или факультативный статус маркера; например, Вы не можете исключить всех членов обязательной группы ***or***, тем самым превращая их в необязательные.

Возможно определить и группы изъятий и группы включений в том порядке.

<! -- СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)? -- >

<! ЭЛЕМЕНТ fig (figbody, figcap?) - (fig | xmp) + (gloss) >

Если тот же самый GI находится и в группе изъятии и в группе включений, как в одной модели содержания или в моделях содержания различных открытых элементов, его присутствие в группе изъятий является определяющим.

B.12 Объявление типа документа

Тип документа SGML идентифицируется "объявлением типа документа", которое появляется в "прологе" документа перед любыми данными. Элемент, список атрибутов и другие объявления, которые обсуждались ранее и которые составляют определение типа документа, сгруппированы в параметре называемом "подмножество объявления":

<!DOCTYPE руководство [

<!ELEMENT руководство (начало?, корпус, конец?) +(запись)>

<!--Далее следует напоминание об объявлениях,

составляющих определение типа документа. -->]>

Левые и правые квадратные скобки являются соответственно разграничителями "подмножество объявления открыто" (***dso***) и "подмножество объявления закрыто" (***dsc***).

Часть документа SGML, которая появляется после пролога, и которая содержит данные и описательную разметку, называется "экземпляром" типа документа (или просто "экземпляром документа").

Если, как обычно, несколько документов удовлетворяют одному определению типа документа, во внешнем объекте может храниться единственная копия определения, на которую будут ссылки из документов. Объект может одновременно объявляться и на него могут иметься ссылки путем определения внешнего идентификатора в объявлении типа документа:

<!DOCTYPE руководство PUBLIC "-//Cave Press//DTD Manual//EN">

Если все определение является внешним, как в приведенном выше примере, подмножество объявления не требуется. Обычно, внешнее определение и подмножество используются вместе, с подмножеством, содержащим вещи подобные объявлениям объекта, которые применяются только к одному документу.

<!DOCTYPE руководство PUBLIC "-//Stutely Press//DTD Manual//EN" [

<!ENTITY название "AIBOHPHOBIA: Fear of Palindromes">

<!—Далее следует напоминание о местных объявлениях,

поддерживающих определение типа документа. -->

]>

Внешний объект технически рассматривается как часть подмножества, как если бы перед dsc появилась на него ссылка. Это позволяет в первую очередь выполнять те объявления в документе, что дает им приоритет перед объявлениями во внешнем объекте.

B.13 Содержание данных

Содержание данных документа - это часть, которая не относится к разметке. Оно имеет две главных характеристики:

a) Его *представление* определяет, может ли его просматривать синтаксический анализатор разметки.

Существуют два основных класса: символьные данные, в которых битовые комбинации представляют символы, и битовые данные, в который битовые комбинации (по одиночке или все вместе) обычно представляют двоичные значения.

b) Его *нотация* определяет, как символ или строки битов будут интерпретироваться процедурами.

Например, в нотации естественного языка строка символов "дельта" может интерпретироваться как английское слово, тогда как в научной нотации она может интерпретироваться как одиночный графический символ, греческая буква.

B.13.1 Представления содержания данных

Стандарт признает два представления данных: символьные данные, который соответствует стандарту, и не относящиеся к SGML символьные или битовые данные, который не соответствуют.

B.13.1.1 Символьные данные (PCDATA, CDATA и RCDATA)

В символьных данных каждая битовая комбинация представляет символ в наборе символов документа.

Прежде, чем символ может рассматриваться как символьные

данные, он обычно должен анализироваться, чтобы определить, не является ли он разметкой. Такие символы представлены в группе модели ключевым словом "PCDATA".

В параграфе со следующей структурой, символ может быть данные либо частью тега элемента "фраза" или "кавычка" :

<!ELEMENT p (//PCDATA | фраза | кавычка)*>

Символ может также быть частью объявления разметки, команды обработки, ссылки на объект или другой разметки, которая допустима в содержании элемента. Только, если он не является тегом или другой разметкой, он будет обрабатываться как данные и передаваться в процедуру для обработки.

Символ, используемый для разметки, обычно не передается в процедуру, но когда используется факультативная функция "DATATAG" (будет обсуждаться позже), символ может быть как разметкой, так и данными. Пробел, например, может служить как конечный тег для слова и при этом быть частью данных предложения, в котором встречается слово.

Возможно также вводить символьные данные непосредственно, без их анализа для разметки обычным способом. Объявление элемента может объявлять, что его содержание содержит символьные данных, определяя ключевое слово вместо модели содержания:

<!ELEMENT формула CDATA >

Обратите внимание, что ***rni*** не требуется, поскольку определяемое пользователем имя не может быть указано для этого параметра (кроме как в пределах разграничителей группы модели).

Если элемент содержит объявленные символьные данных, он не может содержать что-либо еще. Синтаксический анализатор разметки просматривает его, только чтобы обнаружить ***etago*** или ***net***; другая разметка игнорируется. Только правильный конечный тег (или

конечный тег элемента, в который этот элемент вложен) будет распознан.

Вариант CDATA, называемый *заменяемыми символьными данными*, определяется ключевым словом "RCDATA". Оно аналогично CDATA за исключением того, что распознаются ссылки на объект и ссылки на символ.

B.13.1.2 Данные, не относящегося к SGML (NDATA)

Данные, не относящегося к SGML, это данные, который не могут быть проанализированы в соответствии с этим стандартом. Это либо данные в неопределенном наборе символов, битовые данные или некоторое их смешение. В неопределенных данных набора символов битовые комбинации представляют символы, но не в наборе символов документа. В битовых данных битовые комбинации, хотя они и могут управляться как символы, не представляют символьный алфавит обычным способом.

Поскольку не относящегося к SGML данные не могут просматриваться синтаксическим анализатором разметки и могут содержать битовые комбинации, которые могут, например, ошибочно вынуждать операционную систему заканчивать файл, они должны храниться во внешнем объекте. Их расположение становится известно синтаксическому анализатору обычным способом через объявление объекта, за исключением дополнительных параметров, идентифицирующих их как данные, не относящиеся к SGML:

<!ENTITY япония86 SYSTEM NDATA кана>

Данные, не относящиеся к SGML, могут быть введены только в общих объектах, чье объявление определяет ключевое слово NDATA и имя нотации содержания данных (будет объяснено позже). На такой объект можно ссылаться с помощью ссылки на общий объект где-либо в содержании, где распознается ссылка и может появляться символ

данных.

Если желательно сопоставить атрибуты с данными, не относящимися к SGML, для этого может быть определен специализированный тип элемента. Элемент NDATA должен иметь пустое содержание и определение атрибута для атрибута "ENTITY" ("общее имя объекта") атрибутом, чье значение - имя внешнего объекта.

<!ELEMENT Японский EMPTY>

<!ATTLIST Японский файл ENTITY #REQUIRED

предмет (поэзия | проза) проза

>

Разметка для элемента "японский" включает имя объекта как значение атрибута "файл":

<японский файл = "япония86">

Атрибут внешнего объекта может быть определен для любого элемента, даже с моделью содержания. Приложение определяет, как и со всеми атрибутами, как его значение относится к содержанию элемента для выполняемой обработки.

B.13.2 Нотации содержания данных

Различие между разными типами данных, не относящихся к SGML (или разными видами символьных данных), заключается в нотации содержания данных. Нотация не существенна для синтаксического анализатора разметки, но весьма существенна для людей и процедур.

Возможно огромное число нотаций содержания данных. Однако, они имеют тенденцию попадать в небольшое число различных классов.

B.13.2.1 Нотации для символьных данных

Некоторыми общими классами нотаций для символьных данных являются естественные языки, научные нотации и форматированный текст. Когда используется нотация естественного языка, процедуры интерпретируют символы как неявно отмеченные текстовые элементы.

Например, последовательность символов, связанная пробелами между словами, может распознаваться как элемент "слово", последовательность слов, ограниченная подходящей пунктуацией как элемент "фраза" или "предложение", и т.д. В таких нотациях ***RE*** обычно интерпретируется как пробел между словами.

Для дополнительной гибкости нотация текста естественного языка может быть дополнена явной разметкой для специализированных вариантов элементов, которые обычно являются неявными (например, цитируемые фразы, программируемые ключевые слова, подчеркнутые фразы).

В обычных системах обработки текста, неявные условные соглашения разметки и обмены между явной и неявной разметкой являются встроенными. В SGML, функция минимизации разметки тегом данных (будет обсуждаться позже) позволяет пользователю определять степень, в которой такие элементы будут распознаваться синтаксическим анализатором разметки, и степень, в которой они будут интерпретироваться процедурами как часть нотации содержания данных.

Нотации естественного языка появляются в таких элементах, как параграфы, элементы списка и заголовки. Содержание обычно является *смешанным содержанием*, в котором символы данных смешаны с подэлементами.

Научные нотации:

Эти нотации напоминают нотации естественного языка, но слова и фразы значимы для приложения. Например, в математической нотации, фраза "3 над 4" может интерпретироваться как дробь "3/4".

Научные нотации появляются в таких элементах, как формулы, уравнения, химические структуры, формальные грамматики, музыка и т.д. Содержание элемента - обычно CDATA или RCDATA, позволяющие

системам, которые не могут интерпретировать научную нотацию, обрабатывать элемент, как если бы использовалась нотация естественного языка.

Форматированный текст:

Нотации форматированного текста подобны нотациям естественного языка, и особенности обеих могут быть включены в единую нотацию. Их цель состоит в том, чтобы идентифицировать элементы "структуры расположения", созданной предыдущим приложением форматирования. Нотации форматированного текста включают символьные последовательности, которые идентифицируют окончания форматированных строк, пробелы и дефисы, введенные для выравнивания, коды смены типа шрифта, верхние и нижние индексы и т.д.

Эти нотации обнаруживаются в тех же типах элементов, что и нотации естественного языка (параграфы, заголовки и т.д.), а также, когда используется факультативная функция параллельного типа документа (будет обсуждаться позже), в элементах структуры расположения форматированного документа. Содержание обычно смешанное.

B.13.2.2 Нотации для данных, не относящихся к SGML

В объектах NDATA битовые комбинации (одиночно или все вместе) могут представлять неопределенные символы, двоичные значения или смесь двоичных значений и символов.

Неопределенные символы:

Нотации естественного языка, научные нотации и нотации форматированного текста обнаруживаются в NDATA также, как и в CDATA, но в другом наборе, нежели набор символов документа. Данные могут, например, использовать метод расширения кода, который назначает множественные битовые комбинации одиночному символу, тогда как набор символов документа этого не делает. Такая

схема используется для языков, которые требуют большего числа символов, чем имеется разрядных комбинаций в наборе символов документа.

Двоичные значения:

Двоичные значения могут интерпретироваться как шкала яркости или значения цвета для точек изображения, цифрового звука, музыкальных мелодий или как другие наборы числовых или логических значений. Основным приложением двоичных нотаций в обработке текста является представление элементов иллюстрации: автотипии, фотографии и т.д.

Границы записи часто игнорируются в двоичных системах обозначений, но это не следует делать.

Смешанные двоичные значения и символы:

Научная нотация или нотация форматированного текста может использовать смесь символов и двоичных полей. Такие нотации обрабатываются как не анализируемые нотации NDATA, поскольку разрядные комбинации в двоичных полях могут ошибочно интерпретироваться как разграничители разметки.

B.13.2.3 Определение нотаций содержания данных

Набор нотаций содержания данных, используемых в документе, должен быть объявлен с помощью *объявлений нотации*. Каждое объявление определяет имя нотации содержания данных, используемой в документе, вместе с описательной информацией о ней:

```
<!NOTATION  eqn     PUBLIC  "-//local//NOTATION EQN Formula//EN">
<!NOTATION  tex     PUBLIC  "-//local//NOTATION TeX Formula//EN">
<!NOTATION  lowres  SYSTEM  "SCAN.MODULE" – Сканирование с низким разрешением-->
```

Нотация содержания данных элемента определяется атрибутом нотации. Объявленным значением атрибута является ключевое слово "NOTATION", сопровождаемое дополнительным параметром, группой имен:

```
<! --          СОДЕРЖАНИЕ       ЭЛЕМЕНТОВ              -->
<!ELEMENT      формула                RCDATA>
<!--           ЭЛЕМЕНТЫ   ИМЯ   ЗНАЧЕНИЕ              ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST      формула    данные NOTATION(eqn | tex)  #REQUIRED>
<!>
<формула данные="eqn">3 над 4</формула>
```

Группа имен содержит действительные значения атрибута.

B.14 Настройка

Многие характеристики Стандартного Обобщенного Языка Разметки могут быть приспособлены под специальные требования.

B.14.1 Объявление SGML

Приспосабливание документа описывается в его *объявлении SGML,* объявлении разметки, которое первым появляется в документе. Объявление SGML обычно обеспечивается автоматически синтаксическим анализатором разметки, но если приспособление документа отличается от того, что ожидает система обработки, объявление SGML должно быть введено явным образом.

Имеются две основных категории приспосабливания: использование факультативных функций SGML и определения иного конкретного синтаксиса.

B.14.1.1 Факультативные функции

Существует одиннадцать факультативных функций, которыми можно пользоваться:

SHORTREF	Разграничители краткой ссылки на объект
CONCUR	Параллельные экземпляры типа документа
DATATAG	Минимизация тегом данных
OMITTAG	Минимизация пропущенным тегом
RANK	Минимизация пропущенным суффиксом ранга
SHORTTAG	Минимизация кратким тегом

SUBDOC Вложенные поддокументы

FORMAL Формальные публичные идентификаторы

SIMPLE Простые процессы связи

IMPLICIT Неявные процессы связи

EXPLICIT Явные процессы связи

Они описаны в приложении C.

B.14.1.2 Иной конкретный синтаксис

Основой SGML является "абстрактный синтаксис", определяющий способ, использования таких конструкций разметки, как родовые идентификаторы, атрибуты и ссылки на объект. Символы - разграничители, имена объявлений и ключевые слова, ограничения длины и т.д., которые обсуждались в этом приложении, образуют конкретное отображение абстрактного синтаксиса на реальный набору символов и количеств, известный как "конкретный синтаксис ссылок".

SGML позволяет документу использовать иной конкретный синтаксис, чтобы удовлетворить потребности системных сред, национальных языков, клавиатур и т.д. Характеристики конкретного синтаксиса объявляются в объявлении SGML в следующих категориях:

a) Назначения разграничителей, включая разграничители кратких ссылок на объект.

b) Использование символов, включая идентификацию символов функций и кандидатов на символы, не относящиеся к SGML.

c) Правила наименования, включая алфавит символов имен и трансляцию регистра.

d) Определение имен объявлений подмены, ключевых слов и других зарезервированных имен.

e) Количественные характеристики, такие как максимальная длина имен и значения атрибутов.

B.14.2 Воздействие настройки

Документ SGML, который использует конкретный синтаксис ссылок и не использует функции (известный как "минимальный документ SGML"), может быть предметом обмена между всеми системами SGML. Однако, как ожидается, через какое-то время в широкое использование войдут другие комбинации функций и иных конкретных синтаксисов. Некоторые возможности:

a) Документы, которые вводятся непосредственно людьми для издательских приложений, будут вероятно использовать функции SHORTREF, SHORTTAG и OMITTAG. (Те документы, которые используют только эти функции и конкретный синтаксис ссылок, известны как "базовые документы SGML".)

b) Документы, которые будут использоваться для лингвистического анализа или в связи со структурными базами данных, будут вероятно использовать функцию DATATAG.

c) Документы, созданные интеллектуальными текстовыми процессорами, будут вероятно использовать небольшую минимизацию. Такие системы также будут использовать функцию параллельного типа документа, чтобы неформатированные "логические структуры" и форматированные "структуры расположения" могли быть представлены одновременно.

d) Организации пользователей определят меню функций для выполнения своих специальных требований.

B.15 Соответствие

Документ, который выполняет этот международный стандарт во всех аспектах, известен как "удовлетворяющий требованиям SGML документ". Система, которая может обрабатывать такой документ,

известна как "удовлетворяющая требованиям SGML система". Этот международный стандарт не устанавливает требования к архитектуре, методы реализации или обработки ошибок разметки, используемые соответствующими системами.

ПРИЛОЖЕНИЕ С

ДОПОЛНИТЕЛЬНЫЕ КОНЦЕПЦИИ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Данное приложение представляет факультативные функции, которые могут использоваться в документе SGML. Имеются три категории: минимизация разметки, тип связи, и другие функции.

— Функции минимизации разметки

Эти особенности позволяют минимизировать разметку, сокращая или опуская теги или сокращая ссылки на объект. Функции минимизации разметки не затрагивают определение типа документа, поэтому минимизированный документ может быть передан системе, которая не поддерживает эти функции, восстанавливая опущенную разметку.

К функциям относятся SHORTTAG, OMITTAG, SHORTREF, DATATAG и RANK.

— Функции типа связи

Эти функции позволяют использовать "определения процесса связи", которые определяют, как должен обрабатываться документ - источник, чтобы произвести документ - результат другого типа (например, форматированный документ).

Процессы связи определяются в *объявлениях типа связи*, которые независимы от объявлений типа документа или любой другой разметки. Однако, они должны быть удалены до отправки документа системе, которая не поддерживает эти функции.

К функциям типа связи относятся SIMPLE, IMPLICIT и

EXPLICIT, которые различаются по степени управления, которое может осуществляться при определении документа - результата.

— Другие функции

Эти функции позволяют переопределять элементы и объекты для различных частей документа, и интерпретировать публичные идентификаторы для автоматической обработки. Они затрагивают определение типа документа, так что документ, использующий эти функции, может требовать модификации до его отправки системе, которая не поддерживает их.

К функциям относятся CONCUR, SUBDOC и FORMAL.

Использование факультативных функций указывается параметром использования функции объявления SGML (за исключением SHORTREF, чей использование указывается конкретным синтаксисом и его собственными объявлениями "отображения краткой ссылки").

С 1 Функции минимизации разметки

К функциям минимизации разметки относятся следующие:

SHORTTAG означает, что могут использоваться краткие теги с опущенными разграничителями, спецификациями атрибута или родовыми идентификаторами.

OMITTAG означает, что некоторые теги могут быть опущены полностью.

SHORTREF означает, что вместо полных ссылок на объект могут использоваться разграничители кратких ссылок.

DATATAG означает, что символы данных могут служить

одновременно как теги.

RANG — означает, что в тегах могут быть опущены ранги элементов.

C 1.1 SHORTTAG: Теги с опущенной разметкой

Краткий тег - тег, в котором опущена вся или часть обычной разметки.

C.1.1.1 Незакрытые краткие теги

tagc может быть опущен в теге, за которым непосредственно следует другой тег.

Например, разметка:

<глава> Короткая глава. </глава>

может быть сокращена следующим образом:

<глава Короткая глава.</p</глава>

опуская tagc из начального тега "глава" и конечного тега "p".

C.1.1.2 Пустые теги

Пустой краткий тег - это тег, в котором не определены GI или атрибуты; тег состоит исключительно из своих разграничителей. Для пустого конечного тега синтаксический анализатор предполагает, что GI тот же, что и для последнего открытого элемента. Например, следующие записи эквивалентны благодаря минимизации кратким тегом:

Это <q>цитированное</q> слово.

Это <q>цитированное</> слово.

Для пустых начальных тегов, когда функция опущенного тега также не допускается, синтаксический анализатор использует GI последнего законченного элемент и значения по умолчанию атрибутов. С помощью пустых начального и конечного тегов, список может быть размечен следующим образом:

<!ELEMENT – ИСКЛЮЧЕНИЯ СОДЕРЖАНИЯ? --

1 список (позиция+)

2 позиция (p | список)* >

<список>

<позиция>Это первая позиция (какая еще?)</>

<>Это вторая позиция.</>

<>Это третья и последняя позиция.</>

</список>

(См. С.1.2.6 по вопросу использования функции краткого тега, когда разрешена функция опущенного тега.)

Разметка может быть сокращена еще более путем использования односимвольного разграничителя "нулевой конечный тег" (***net***) (обычно "/"), который разрешается функцией краткого тега. net интерпретируется как пустой конечный тег только для элемента, в котором он также использовался как разграничитель tagc (т.е. вместо него).

Этот параграф содержит
<q/цитата/ в тексте и
косую черту, (/) которая является данными.

В приведенном ниже списке сведены воедино разграничители, используемые с функцией краткого тега и их символьные строки в наборе разграничителей ссылок:

Строка	**Имя**	***Значение***
/	**NET**	Нулевой конечный тег

С.1.1.3 Минимизация атрибута

Весь или часть списка спецификации атрибута может быть опущена в следующих обстоятельствах:

Разграничители значений:

Разграничители могут быть опущены из спецификации значения атрибута, если значение ограничено символами имен.

<стандарт защита=публичный>

Обратите внимание, что в таких значениях атрибутах ссылки на объект не разрешаются.

Значения по умолчанию:

Если атрибут был объявлен с фактическим значением по умолчанию или ключевыми словами "^IMPLIED" или "#CONREF", полная спецификация для него может быть опущена. Атрибут будет обрабатываться, как если бы для него было определено значение по умолчанию. (Это правило также применяется к текущим атрибутам после того, как они были однажды определены.)

Имена:

Имя и разграничитель ***vi*** атрибута могут быть опущены, если его объявленное значение включено в *группу имен* или *группу маркеров имен.*

Эта форма минимизации полезна, когда значения атрибута подразумевают имя атрибута. Например, записка может иметь атрибут "статус", чье значение будет "проект" или "оконч".

```
                                <! – СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT   Записка          (от кого, кому, тема, текст, подпись, копия?)>
<!--        ЭЛЕМЕНТЫ  ИМЯ    ЗНАЧЕНИЕ        ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST   Записка   Статус (оконч|проект)  "оконч"
```

Обычная разметка для атрибута:

<записка статус="проект">

будет в данном случае громоздка, поскольку "проект" подразумевает "статус". Однако, с помощью минимизации SHORTTAG в документе можно записать:

<записка статус="проект">

либо

<записка статус=проект>

либо

<записка проект>

Попуск имен атрибутов ведет к появлению неоднозначности в разметке документа. Этот эффект может быть уменьшен, если размер групп сохраняется небольшим, а маркеры выбранных имен являются описательными прилагательными, которые подразумевают имя атрибута (например, "НОВЫЙ | ПЕРЕСМОТРЕННЫЙ", "СЕКРЕТНЫЙ | ВНУТРЕННИЙ | ПУБЛИЧНЫЙ"). В приведенном ниже примере атрибут "компактный" с этой точки зрения лучше, чем "выделение":

```
                                <! – СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT   список                          (позиция*)>
<!--        ЭЛЕМЕНТЫ    ИМЯ         ЗНАЧЕНИЕ        ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST   список      компактный  (компактный)    #IMPLIED
                        выделение   (0|1|2|3)       0
```

C.1.2 OMITTAG: теги могут быть опущены

Определение типа устанавливает возможные изменения структуры среди документов этого типа. Поэтому отдельный документ не должен содержать такого количества разметки.

C.1.2.1 Концепция пропуска тега

Предположим, что класс документов "журнальная статья" с GI "статья" имеет следующее определение типа:

```
<! – СОДЕРЖАНИЕ   ЭЛЕМЕНТОВ --                  >
<!ELEMENT  статья       (заголовок, корпус)     >
<!ELEMENT  заголовок    (#PCDATA)               >
<!ELEMENT  корпус       (p*)                    >
<!ELEMENT  p            (#PCDATA | список)*     >
<!ELEMENT  список       (позиция+)              >
<!ELEMENT  позиция      (#PCDATA, (p | список)*)>
```

Полная разметка для экземпляра статьи может выглядеть следующим образом:

```
<статья>
<заголовок>Кошка</заголовок>
<корпус>
<p>Кошка может:
<список>
```

<позиция>прыгать</позиция>

<позиция>мяукать</позиция>

</список>

У нее есть 9 жизней.

</корпус>

</статья>

Обратите внимание, что определение типов говорит о том, что за параграфом не может содержать непосредственно другой параграф; он может содержать только текстовые символы и списки. Аналогичным образом, позиция списка не может непосредственно содержать другую позицию списка (даже при том, что она может содержать другой список). В результате, разметка может быть минимизирована путем пропуска многих конечных тегов:

<статья>

<заголовок>Кошка</заголовок>

<корпус>

Кошка может:

<список>

<позиция>прыгать

<позиция>мяукать

</список>

У нее есть 9 жизней.

</статья>

Возможно опустить конечный тег позиции: появление другой позиции указывает на конец предыдущей (поскольку позиция не может содержать позицию). Конечный тег параграфа может быть опущен по тем же причинам (параграф не может содержать другой параграф).

Наконец, логично, что конец элемента должен также быть концом всего, что он содержит. Таким образом, конечный тег статьи заканчивает корпус и последний параграф, также как и статью.

(Разметка могла быть минимизирована еще в большей степени, что читатель без сомнения обнаружит.)

C 1.2.2 Специфицирование минимизации

Минимизация разметки - хорошая вещь, но не в том случае, если она затрудняет обнаружение ошибок разметки. В последнем примере, если бы был опущен конечный тег списка, он подразумевался бы конечным тегом статьи, также как и конечным тегом корпуса.

<статья>
<заголовок>Кошка</заголовок>
<корпус>
<p>Кошка может:
<список>
<позиция>прыгать
<позиция>мяукать
<p>У нее есть 9 жизней.
</статья>

Это, однако, не входило в намерения автора, поскольку последний параграф становился бы ошибочно частью последней позиции списка.

Чтобы предотвратить такие неверные истолкования, в объявлении элемента имеются два параметра, которые определяют минимизацию опущенным тегом, допустимую для элемента. Когда в объявлении SGML определен "OMITTAG YES", разрешается функция опущенного тега, и тогда требуются два параметра минимизации тогда во всех объявлениях элементов.

C.1.2.3 Пропуск конечного тега: внедрение начального тега

Конечный тег может быть опущен, если содержание его элемента

сопровождается начальным тегом элемента, который не может появиться внутри него. Это имело место для параграфа и конечных тегов позиций в примере статьи. Объявления элемента могут выглядеть следующим образом:

```
<!--          МИН.     СОДЕРЖАНИЕ   ЭЛЕМЕНТОВ              >
<!ELEMENT     p        - 0          (#PCDATA | список)*    >
<!ELEMENT     список   - -          (позиция+)             >
<!ELEMENT     позиция  - 0          (#PCDATA, (p | список)*) >
```

и иметь следующие значения:

a) Заголовок "MIN" идентифицирует пару параметров для минимизации начальным тегом и конечным тегом. Оба параметра должны быть определены.

b) "0" указывает, что минимизация опущенным тегом разрешается для конечного тега элементов "p" и "позиция".

c) Дефис - разграничитель ***minus***; он указывает, что для начальных тегов минимизация не допускается.

Вспомните, что конечный тег должен опускаться всегда, когда элемент имеет пустое содержание, поскольку все, что следует за начальным тегом должно быть частью содержащего элемента. Хотя это правило не имеет никакого отношения к минимизации разметки, полезно отметить "0" как напоминание, что в документе не могут быть обнаружены конечные теги.

```
<!ELEMENT ссылка_рис - 0 EMPTY>
```

C 1.2.4 Пропуск конечного тега: конечный тег содержащего элемента

Конечный тег содержащего элемента может быть опущен, когда он сопровождается конечным тегом элемента, который его содержит.

```
<!--          МИН.     СОДЕРЖАНИЕ   ЭЛЕМЕНТОВ     >
<!ELEMENT     список   - -          (позиция+)     >
<!ELEMENT     позиция  - 0          (#PCDATA)      >
```

Вышеупомянутое объявление позволяет опускать конечный тег

третьей позиции опущенным из следующего списка, поскольку он подразумевается конечным тегом списка:

<список>

<позиция>Эта первая позиция(какая еще?)</позиция>

<позиция>Это вторая позиция.</позиция>

<позиция>Это третья и последняя позиция.

</список>

С.1.2.5 Пропуск начального тега: контекстно обязательный элемент

Начальный тег может быть опущен, если тип элемента является контекстно обязательным и любые другие типы элемента, которые могут встретиться, контекстно факультативны.

В комбинации с другой минимизацией, рассмотренной выше, объявление элемента будет выглядеть следующим образом:

```
<!--          МИН.      СОДЕРЖАНИЕ      ЭЛЕМЕНТОВ     >
<!ELEMENT     список    - -             (позиция+)    >
<!ELEMENT     позиция   0 0             (#PCDATA)     >
```

А список может быть размечен следующим образом:

<список>

Эта первая позиция(какая еще?)

<позиция>Это вторая позиция.

<позиция>Это третья и последняя позиция.

</список>

Начальный тег мог быть опущен для первой позиции, потому что она обязательна; он не мог быть опущен для последующих позиций, потому что они были факультативны.

Даже когда элемент контекстно обязателен, его начальный тег не может быть опущен, если тип элемента имеет обязательные атрибуты или объявленное содержание, или если экземпляр элемента пуст.

С.1.2.6 Комбинация с минимизацией кратким тегом

Когда разрешены обе минимизации кратким тегом и опущенным

тегом, пустые начальный и конечный теги обрабатываются единообразно: им дается GI последнего открытого элемента. Две формы минимизации могут использоваться вместе, как в следующем примере:

```
<!--          МИН.      СОДЕРЖАНИЕ   ЭЛЕМЕНТОВ                  >
<!ELEMENT     р         - 0          (#PCDATA | список)*        >
<!ELEMENT     список    - -          (позиция+)                 >
<!ELEMENT     позиция   - 0          (#PCDATA, (р | список)*)    >
<список>
<позиция>Эта первая позиция(какая еще?)
<>Это вторая позиция.
<>Это третья и последняя позиция.
</список>
```

Обратите внимание, что в конце необходимо сказать "</список>" вместо "</>", поскольку последний означал бы "конечный тег позиции", а не "конечный тег списка". Альтернативно, может быть введена строка "</> </>" для обозначения "конец позиции, затем конец списка".

Теперь рассмотрим, что происходит, когда разметка минимизируется далее путем удаления начального тега первой позиции:

```
<список>
Эта первая позиция(какая еще?)
<>Это вторая позиция.
<>Это третья и последняя позиция.
</список>
```

Достигаются идентичные результаты, поскольку когда встречается первый пустой тег, открытым элементом была "позиция", а не "список", даже при том, что "позиция" подразумевалась "0" минимизацией, а не была введена явно.

C 1.2.7 Соображения по минимизации разметки

Минимизация кратким тегом требует от пользователя (и синтаксического анализатора разметки) знать текущее расположение в структуре документа, и понимать объявления списка атрибутов. Однако, минимизация опущенным тегом, также требует знания

объявлений элемента.

Это различие должно учитываться при решении о том, от какого рода минимизации пользователь может ожидать надлежащей обработки, поскольку исключение тега может приводить к появлению такого документа, который будучи свободен от ошибок SGML, тем не менее не отвечает ожиданиям пользователя.

C 1.3 SHORTREF: разграничители краткой ссылки могут заменять полные ссылки на объект

Краткие ссылки - одиночные символы или краткие строки, которые могут заменять полные разграниченные ссылки на объекты. Они могут использоваться для имитации обычного набора на пишущей машинке и упрощения ввода элементов с повторными структурами.

C 1.3.1 Набор на пишущей машинке: обобщенный режим WYSIWYG

Некоторые текстовые процессоры предлагают операторам ввода текста интерфейс, подобный интерфейсу пишущих машинок, на которых многие операторы обучались. На пишущих машинках каждая функциональная клавиша при ударе производит немедленный результат форматирования: перевод каретки начинает новую строку, клавиша табуляции генерирует горизонтальный пробел и т.д. Текстовые процессоры, которые имитируют эту характеристику пишущих машинок, иногда известны как системы "WYSIWYG" — "что Вы видите, то и получаете".

Символы, создаваемые функциональными клавишами, являются конкретными командами обработки. Подобно всем командам обработки они ограничивают документ единым стилем форматирования и выполняются только на машинах, которые понимают команды.

SGML, тем не менее, может предлагать преимущество по сравнению с привычным набором на пишущей машинке, сохраняя при этом общность документа. С помощью кратких ссылок символы

функциональных клавиш пишущей машинки могут интерпретироваться как описательная разметка. Например, объявления

```
<!ENTITY ptag STARTTAG "p" >
<!SHORTREF wysiwyg "&#TAB," ptag
"&#RS;&#RE," ptag >
```

отображают и позицию табуляции и перевод строки, последовательность перевода каретки (пустая запись) на начальный тег для элемента параграфа. Фактическое форматирование параграфа зависит, как всегда, от прикладной процедуры, как если бы начальный тег был введен явно. Но устройство ввода по-прежнему производит табуляцию или ввод дополнительной строки, вызываемой функциональными символами, таким образом давая пользователю непосредственную визуальную обратная связь, которая является таким важным аспектом систем WYSIWYG.

Объявления разметки эффективно создали "обобщенный режим WYSIWYG", в котором функциональные символы пишущей машинки интерпретируются как обобщенная разметка при сохранении их визуального эффекта. Кроме того, пользователь может свободно смешивать обобщенный режим WYSIWYG с явными тегами, используя тот, который наиболее удобен для данной части документа. Эта возможность может быть особенно полезна для сложных элементов, таких как многостраничные таблицы с выполняющимися заголовками, где функции пишущей машинки не обеспечивают удобные и общепринятые соглашения ввода.

C 1.3.2 Пример набора на пишущей машинке: определение отображения краткой ссылки

С помощью кратких ссылок пользователь может создавать документы SGML с обычными соглашениями ввода систем обработки текстов, а не с сознательным вводом разметки. Большое число строк символов определены как разграничители краткой ссылки в конкретном

синтаксисе ссылок. Те из них, которые содержат "невидимые" функциональные символы и кавычки особенно полезны для поддержки обычного набора; они:

Строка	*Описание*
&#TAB;	Горизонтальная табуляция
&#RE;	Конец записи
&#RS;	Начало записи
&#RS;B	Ведущие пропуски (начало записи, один или большее число пробелов и/или позиций табуляции)
&#RS;&#RE;	Пустая запись (начало записи, конец записи)
&#RS;B&#RE;	Запись с пропуском (начало записи, одно или большее число пробелов и/или позиций табуляции, конец записи)
B&#RE;	Завершающие пропуски (один или большее число пробелов и/или позиций табуляции, конец записи)
&#SPACE;	Пробел
BB	Два или больше пропуска (два или больше пробела и/или позиции табуляции)
"	Знак кавычек

Каждый разграничитель краткой ссылки может быть связан с именем объекта в таблице, называемой "отображение краткой ссылки". Разграничитель, который не "отображается" на объект, обрабатывается как данные. Определение объекта делается обычным способом с объявлением объекта. Отображение делается с помощью "объявления отображения краткой ссылки". Следующий пример определяет пустую запись как ссылку на начальный тег параграфа:

```
<!ENTITY ptag "<p>">
<!SHORTREF mapl "&#RS;&#RE;" ptag>
```

Объявление "SHORTREF" определяет отображение под именем

"map1". Отображение содержит только одно явное отображение: краткая ссылка пустой записи отображается на объект "ptag". Всякий раз, когда это отображение является текущим (будет объяснено позже), пустая запись будет заменяться ссылкой на объект. Например,

Текст последнего параграфа.
Текст следующего параграфа.
будет интерпретироваться как

Текст последнего параграфа.
&ptag;
Текст следующего параграфа.
который будет моментально после разрешения ссылки на объект интерпретироваться как

Текст последнего параграфа.
<p>
Текст следующего параграфа.
Поскольку никакие другие строки краткой ссылки не были отображены, они будут обрабатываться как данные, пока это отображение остается текущим.

C.1.3.3 Пример набора на пишущей машинке: активизация отображения краткой ссылки

Обычно, отображение становится текущим путем сопоставления его имени с типом элемента в объявлении "использования краткой ссылки". В следующем примере "map1" становится текущим отображением всякий раз, когда начинается "глава" :

<!USEMAP mapl глава >
Всякий раз, когда начинается глава, map1 становится текущим и остается текущим, кроме как в пределах любых вложенных подэлементов, которые самостоятельно сопоставляются с отображениями.

Отображение краткой ссылки не должно быть связано с каждым типом элемента. Элемент, который не имеет отображения, будет просто использовать отображение, являющееся текущим в его начале. В

следующем примере, элементы кавычки и глава имеют отображения, но элемент параграфа не имеет:

```
<!ENTITY      ptag       "<p>"              -- начальные тег параграфа -- >
<!ENTITY      qtag       "<кавычки>" >
<!ENTITY      qendtag    "</кавычки>" >
<!SHORTREF    chapmap    "&#RS;&#RE;"       ptag
                         ’ ” ’              qtag >
<!SHORTREF    qmap       ’ ” ’              qendtag>
<!USEMAP      chapmap    Глава>
<!USEMAP      qmap       q>
<!ELEMENT     chapter    (p*)               >
<!ELEMENT     p          (q|#PCDATA)*       >
<!ELEMENT     q          (#PCDATA)          >
```

Когда начинается элемент параграфа "chapmap" остается текущим; он отображает знак кавычек на объект "qtag", который содержит начальный тег кавычек. Как только начинается элемент кавычек, "qmap" становится текущим отображением, и знак кавычек будет заменен ссылкой на объект "qendtag", который содержит конечный тег кавычек. В конце кавычек, восстанавливается элемент параграфа, и с ним отображение краткой ссылки "chapmap".

Разметка

<глава Здесь имеется "цитата" в тексте.

теперь эквивалентна

<глава Здесь имеется <кавычки>цитата</кавычки> в тексте.

Объявления позволяют кавычкам служить как начальный тег и конечный тег элементов цитирования, а не обрабатываться как данные. Процедуры форматирования могут, поэтому, использовать открывающие и закрывающие изогнутые кавычки, чтобы отличить начало и конец цитаты, что иначе было бы невозможно с обычными типографскими соглашениями.

Кстати, при объявлении объекта, чей текст замены является тегом, могут использоваться следующие формы объявления объекта:

<!ENTITY qtag STARTTAG "кавычки" >
<!ENTITY qendtag ENDTAG "кавычки" >

Ключевые слова "STARTTAG" и "ENDTAG" позволяют опускать разграничители тега. Они также позволяют системе оптимизировать обработку объекта, поскольку она знает, что он будет наиболее вероятно использоваться как тег, а не как данные.

Краткие ссылки на объект могут использоваться только в элементах, чье содержание было определено моделью (т.е. не в CDATA или RCDATA). Они не могут использоваться в значениях атрибута или параметрах объявления.

C.1.3.4 Пример с таблицами

Многие печатаемые символы определены как разграничители краткой ссылки в конкретном синтаксисе ссылок (см. рис. 4). Эти символы могут использоваться как подстановки более длинных ссылок на общие объекты, который позволяют создать более краткий и визуальный стиль разметки, когда это уместно, как в таблицах:

```
<!ENTITY        строка          "<строка><столбец>" >
<!ENTITY        столбец         "</столбец><столбец>" >
<!ENTITY        конец_строки    "</столбец></строка>" >
<!SHORTREF      tablemap        "(" строка
                                "|" столбец
                                ")" конец_строки >
<!USEMAP        tablemap        таблица>
<!ELEMENT       таблица         (строка*)>
<!ATTLIST       таблица         столбцы ЧИСЛО #REQUIRED>
<!ELEMENT       строка          (столбец+)>
<!ELEMENT       столбец         (#PCDATA)>
<!>
<таблица столбцы=3>
(строка1,столбец1|строка1,столбец2|строка1,столбец3)
(строка2,со11|строка2,со12|строка2,со13)
(строка3,столбец1|строка3,столбец2|строка3,столбец3)
(строка4,столбец1|строка4,столбец2|строка4,столбец3)
```

</таблица
>
<!>

Пример позволяет использовать круглые скобки и вертикальную линию (|) как ссылки на объект. (Поскольку краткие ссылки распознаются только в содержании, нет опасности толкования вертикальной линии как соединителя). Текст разрешится в следующее:

```
< таблица столбцы=3>
<строка><столбец>строка1,столбец1</столбец><столбец>строка1,столбец2</столбец
><столбец>строка1,столбец3</столбец></строка>
<строка><столбец>строка2,столбец1</столбец><столбец>строка2,столбец2</столбе
ц><столбец>строка2,столбец3</столбец></строка>
<строка><столбец>строка3,столбец1</столбец><столбец>строка3,столбец2</столбе
ц><столбец>строка3,столбец3</столбец></строка>
<строка><столбец>строка4,столбец1</столбец><столбец>строка4,столбец2</столбе
ц><столбец>строка4,столбец3</столбец></строка>
</таблица>
```

"tablemap" является текущим только в границах таблицы, так что в остальной части документа разграничители краткой ссылки могут использоваться как данные.

C.1.3.5 Специальные требования

Хотя отображения краткой ссылки обычно связываются с определенными типами элемента, возможно использовать объявление использования краткой ссылки, чтобы выбрать тип произвольно:

```
<рисунок>
Начальный текст рисунка (использует нормальное отображение рисунка).
<!USEMAP графика – Разрешает краткие ссылки на символы графики -->
Остальной текст рисунка (использует отображение "графика").
</рисунок>
```

Отображение, поименованное в объявлении (здесь "графика"), заменяет текущее отображение для последнего начатого элемента (здесь "рисунок"), так как если бы оно было названо в объявлении элемента "рисунок". Однако, это применяется только к этому экземпляру рисунка, но не к какому-либо другому. Если позже в этом рисунке

требуется нормальное отображение рисунка, оно должно быть активизировано объявлением использования краткой ссылки, но для более поздних рисунков оно станет текущим автоматически.

Возможно отменить все отображения, используя зарезервированное имя отображения "#EMPTY" вместо нормального имени отображения. "Пустое" отображение, которое вызывает обработку всех разграничителей кратких ссылок как данных (или разделителей), будет текущим при тех же самых обстоятельствах, что и нормальное отображение.

Краткие ссылки хотя и являются мощным средством, не адекватны для интерпретации произвольных существующих документов, как если бы они были размечены с помощью SGML. Они предназначены, чтобы позволить пользователю дополнять обычную разметку SGML знакомыми соглашениями ввода систем обработки текстов. Вместе с другими методами минимизации разметки краткие ссылки дают возможность устранить наиболее осознанную разметку даже без "интеллектуального" текстового процессора.

C.1.4 DATATAG: данные могут также быть тегом

Функция минимизации тегом данных позволяет модели содержания определять символы данных, которые будут интерпретироваться как теги, оставаясь частью содержания данных.

Обсуждаемые до сих пор методы минимизации разметки проходят долгий путь к созданию текста, который кажется почти свободным от разметки. Следующий список, например, не содержит в себе видимой явной разметки:

```
<!ENTITY itemtag        STARTTAG позиция --JOBITEM начальный
                        тег-->
<!SHORTREF listmap      "&#RS;" itemtag >
<!USEMAP listmap        список>
<!—МИНИМАЛЬНОЕ          СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
```

```
<!ELEMENT список      - -    (позиция+)>
<!ELEMENT позиция     - 0    (#PCDATA)>
<joblist>
Шэрон Адлер,                Вице-председатель
Лэрри Бек,                  Секретаь
Андерс Берглунд,            Издатель
Эрон Бигман,                Бывший член
Джим Кокс,                  Бывший член
Билл Дэвис,                 Председатель
Джо Гангеми,                Член
Чарльз Голдфарб,            Редактор
Майк Гудфеллоу,             Консультант
Рэнди Гроувз,               Член
Чарльз Лайтфут,             Бывший член
Сперлинг Мартин,            Бывший член
Бетти          МакДэйвид    Консультант
Мэйсон,
Джим Мэйсон,                Член
Линн Прайс,                 Теоретик
Стэнли Рис,                 Исследователь
Норма Шарпф,                Обозреватель
Крэйг Смит,                 Теоретик
Джоан Смит,                 Публицист
Ед Смура,                   Член
Билл Танниклифф,            Член
Кит фон Сак,                Член
</список>
```

Конечный тег для каждой "позиции" кроме последний может быть опущен, поскольку за ним следует начальный тег позиции. Последний подразумевается концом списка. Начальные теги присутствуют явно, но не видимы, потому что они находятся в объектах, на которые ссылается непечатаемый разграничитель краткой ссылки (начало записи). Таким образом список практически полностью размечен только двумя тегами, сознательно введенными пользователем, но можно сделать и больше.

Объявление элемента заявляет, что "позиция" - это просто последовательность символов. Читатель, однако, видит, что она содержит отдельные элементы "имя" и "должность", поскольку запятая

и пробелы идентифицируют название должности так же ясно, как это делал бы начальный тег. Такая ситуация, в которой соглашения, используемые в данных, могут однозначно идентифицировать элементы в пределах данных, называется *минимизация тегом данных*.

Функция тега данных SGML позволяет выражать эти соглашения в объявлении элемента с помощью *шаблона тега данных*:

```
<!--            МИНИМАЛЬНОЕ      СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT позиция          - 0   ([имя, ", ",″ ″], должность) >
<!ELEMENT имя              0 0   (#PCDATA)>
<!ELEMENT должность        0 0   (#PCDATA)>
```

Объявление заявляет (среди прочего), что:

a) Начальный тег для "имени" может быть опущен, поскольку оно должно быть первым в "позиции".

b) Конечный тег "имени" может быть опущен, когда он встречается в "позиции", поскольку он подразумевается запятой и пробелами шаблона тега данных.

c) Начальный тег "должности" может быть опущен, поскольку элемент "должность" не допускается в "имени".

d) Конечный тег "должности" может быть опущен, поскольку он подразумевается концом "позиции".

Полное объявление для "списка" выглядит следующим образом:

```
<!ENTITY itemtag          STARTTAG позиция --JOBITEM начальный тег-->
<!SHORTREF listmap        "&#RS;" itemtag >
<!USEMAP listmap          Список>
<!—МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT позиция         - 0 ([имя, ", ", " "], должность) >
<!ELEMENT имя             0 0    (#PCDATA)>
<!ELEMENT должность       0 0    (#PCDATA)>
```

Символы - это обычно либо данные, либо разметка, но не то и другое. Сущность минимизации тегом данных заключается в том, что символы одновременно являются данными и тегом. Тег данных - это конечный тег, который соответствует *шаблону тега данных*,

следующему за элементом тега данных в *группе тега данных* в модели ее содержащего элемента. В вышеупомянутом примере, группа тега данных следующая:

[имя, ", ", " "]

Группа является группой ***seq***, заключенной в разграничители "группа тега данных открыта" (***dtgo***) и "группа тега данных закрыта" (***dtgc***) (обычно "[" и "]"), вместо обычных разграничителей группы, чтобы указать на ее особый характер. В группе имеются три маркера:

a) GI минимизируемого элемента:

имя

b) *Шаблон тега данных,* который в данном случае является одиночным литералом, но может также быть группой **or**, составленной из литералов:

", "

Литерал означает "запятая, сопровождаемая пробелом".

c) *Шаблон добавления тега данных* (который факультативен):

" "

Он означает, что вслед за обязательной запятой и пробелом может следовать ноль или большее число пробелов для образования тега данных.

Таким образом, любая строка, состоящая из запятой и одного или большего числа пробелов, которые появляются после того, как элемент "имя" начинается в "позиции", будет рассматриваться как конечный тег "имени". Он, однако, будет также рассматриваться как часть данных "позиции".

Обратите внимание на различия между тегом данных и краткой ссылкой, чьим объектом является конечный тег:

a) Строка краткой ссылки является разметкой, не данными.

b) Краткая ссылка распознается в любом элементе, чье

отображение отображает ее на объект. Тег данных распознается только тогда, когда его элемент открыт, а содержащий элемент - тот, в чьей модели появилась группа тега данных.

c) Краткая ссылка - постоянная строка; тегом данных может быть любая из ряда строк, которые соответствуют шаблону тега данных.

Минимизация тегом данных полезна для приложений, которые анализируют текст. Следующий пример использует теги данных для идентификации предложений и слов в пределах параграфа:

```
<! ENTITY % стоп '(             ".&#RE;"  | ". "  | ".)&#RE;" | ".) "  |
                                "?&#RE;" | "? " | "?)&#RE;" | "?) " |
                                "!&#RE;"  | "! " | "!)&#RE;" | "!) "  )′ >
<! ENTITY % пауза '(            " "   |"&#RE;"   | ", "   | ",&#RE;"   |
                                "; "  |";&#RE;"  | ": "   | ":&#RE;"  |
                                ") "  |")&#RE,"  | ",) " | ",)&#RE;" |
                                ";) " |";)&#RE;" | ":) " | ":)&#RE;" |
                                "), " |"),&#RE;" | "); " | ");&#RE;" |
                                "): " |"):&#RE;" )' >
<!ELEMENT p                     - 0 ([sentence,%stop;]+)>
<!ELEMENT предложение           0 0 ([word, %pause;, " "]+)>
<!ELEMENT слово                 0 0 (#PCDATA)>
<p>Первое предложение закачивается здесь.
Второе предложение заканчивается
здесь.
Это третье предложение.
Четвертое предложение заканчивается, не здесь!, но здесь!
</p>
```

В этом примере, конечный тег слова - строка символов "пауза", за которой следует ноль или большее число пробелов. Все слова кроме последнего в каждом предложении имеют тег данных, который соответствует этому шаблону.

Конечный тег предложения - строка символов "стоп", которая заканчивается или концом записи или двумя пробелами. Когда распознается тег, опущенный конечный тег для последнего слова подразумевается обычным правилом "конца содержащего элемента".

Следует проявлять осмотрительность при вводе с использованием

минимизации тегом данных. В следующем примере сокращение будет ошибочно обрабатываться как конечный тег предложения:

Интересно, будет ли г-жа Г.
читать это.

В приведенном ниже списке сведены воедино разграничители, допускаемые функцией тега данных, и их строки символов в наборе разграничителей ссылок:

Символ	**Имя**	***Значение***
[	**DTGO**	Группа тега данных открыта.
]	**DTGC**	Группа тега данных закрыта.

С.1.5 РАНГ: ранги могут опускаться в тегах

Рангом элемента является уровень его вложения. Во многих типах документов, ранг только подразумевается началом и окончанием вложенных элементов, таких как списки, и никогда не определяется явно в разметке документа.

Тем не менее, некоторые разработчики разметки предпочитают использовать явные обозначения ранга для некоторых элементов, таких как заголовки и параграфы (например, p1, p2). Такие элементы имеют тенденцию иметь объявления, подобные следующему:

```
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT  p1          - 0      (#PCDATA, p2*)>
<!ELEMENT  P2          - 0      (#PCDATA, p3*)>
<!ELEMENT  p3          - 0      (#PCDATA, p4*)>
<!ELEMENT  p4          - 0      (#PCDATA)>
```

а разметку документов такую:

<pl>Текст параграфа 1-го уровня.
<pl>Другой параграф 1-го уровня.
<p2>Вложенный параграф 2-го уровня.
<p2>Другой параграф 2-го уровня.
<pl>Назад к 1-му уровню.

Функция RANK SGML предлагает более удобный путь явного определения ранга. Элемент может быть обозначен как *ранжированный элемент*, путем разделения его GI на *базу ранга* и *суффикс ранга,*

который должен быть *номером.*

```
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT  p  1          - 0          (#PCDATA, p2*)>
<!ELEMENT  p  2          - 0          (#PCDATA, p3*)>
<!ELEMENT  p  3          - 0          (#PCDATA, p4*)>
<!ELEMENT  p  4          - 0          (#PCDATA)>
```

Когда используется функция ранга, вместо полного GI в тег может вводиться база ранга. Полный GI будет выводиться из базы и последнего суффикса ранга, указанного для элемента с этой базой.

```
<pl>Текст параграфа 1-го уровня.
<p>Другой параграф 1-го уровня.
<p2>Вложенный параграф 2-го уровня.
<p>Другой параграф 2-го уровня.
<pl>Назад к 1-му уровню.
```

Группа типов элементов может совместно использовать тот же самый ранг, если они имеют идентичные модели содержания. Такая группа называется *ранжированной группой.* Например, если документ имеет нормальные параграфы, пронумерованные параграфы, и маркированные параграфы, каждый может включать любые из трех нижних уровней. Объявление

```
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ-->
<!ELEMENT  (p | n | b) 1    - 0          (#PCDATA, (p2 | n2 | b2)*)>
<!ELEMENT  (p | n | b) 2    - 0          (#PCDATA, (p3 | n3 | b3)*)>
<!ELEMENT  (p | n | b) 3    - 0          (#PCDATA, (p4 | n4 | b4)*)>
<!ELEMENT  (p | n | b) 4    - 0          (#PCDATA)>
```

допускает следующее содержание:

```
<pl>Текст параграфа 1-го уровня.
<n>Нумерованный параграф 1-го уровня.
<p2>Вложенный параграф 2-го уровня.
<b>Маркированный параграф 2-го уровня.
<pl>Назад к 1-му уровню.
```

C.2 Функции СВЯЗИ: ПРОСТАЯ, НЕЯВНАЯ и ЯВНАЯ

До настоящего времени обсуждение ограничивалось разметкой для

единственного типа документа: логической, или абстрактной структуры документа - источника, который подлежит обработке. Однако, результат обработки документа также является документом, хотя последний может иметь совершенно иное определение типа документа.

Например, документ, который соответствует одномерному типу документа - источника с логической структурой, составленной из глав, разделов и параграфов, после форматирования будет также соответствовать двумерному типу документа - результата со структурой "верстки", чьи элементы - страницы, столбцы, текстовые корпусы и строки.

Эти два типа документа совпадут в некоторых пунктах, конечно, на самом высоком уровне (документов) и возможно на других. Глава в логической структуре, например, может соответствовать "набору страниц " в структуре верстки.

SGML поддерживает множественные типы документов двумя различными способами:

a) Определения процесса связи: определения типа документа могут быть связаны, чтобы определить, как документ может быть преобразован от одного типа ("источник") к другому ("результат"); например, как разметка, которая описывает данные в терминах структуры верстки результата, должна генерироваться из исходной логической разметки.

b) Параллельные экземпляры документа: разметка для экземпляров множественных типов документа может существовать одновременно в едином документе.

C.2.1 Определения процесса связи

Объявления набора связей могут использоваться в таких приложениях, как форматирование, чтобы определить, как логические элементы в источнике (например, параграфы или элементы списка)

должны быть преобразованы в элементы верстки в результате (например, текстовые корпусы).

Например, если элементы "параграф" и "позиция" были определены в источнике, а элемент "корпус" был определен в результате с атрибутом "отступ", то запись

<!LINK docset параграф корпус [отступ=3]

позииция корпус [отступ=5]

>

вызвала бы форматирование параграфа как текстового корпуса с отступом 3. Позиции списка были бы также отформатированы как текстовые корпусы, но с отступом 5. Исходный текст:

<позиция>Текст позиции списка. </позиция>

Стал бы текстом результата:

< корпус отступ=5 >Текст позиции списка.</корпус>

Для всех других случаев (предполагая минимизацию SHORTTAG), все другие атрибуты корпуса имели бы их значения по умолчанию, как определено в объявлении элемента корпуса.

Функции СВЯЗИ описаны подробно в разделе 12.

C.3 Другие функции

К оставшимся функциям относятся:

CONCUR — означает, что экземпляры указанного числа типов документа (1 или больше) могут появляться одновременно с типом базового документа.

SUBDOC — означает, что указанное число открытых объектов поддокумента (1 или больше) может быть вложено в документ SGML.

FORMAL означает, что публичные идентификаторы должны интерпретироваться формально.

C.3.1 CONCUR: экземпляры документа могут появляться одновременно

Иногда полезно сохранять информацию о типах документа - источника и документа - результата одновременно в одном документе, как в текстовых процессорах WYSIWYG. Там пользователь взаимодействует с форматированным выводом, но редакционные изменения фактически делаются в источнике, который затем переформатируется для отображения.

Имеются также случаи, когда могут быть полезны даже больше чем два таких "представления" документа, например при обеспечении множественных результатов форматирования для мгновенного вывода на ЭЛТ и принтер, при сохранении логического типа документа, доступного для других приложений.

Функция параллельных экземпляров документа поддерживает множественные параллельные структурные представления в дополнение к абстрактному представлению. Это позволяет пользователю сопоставлять объявления элемента, объекта и нотации со специфическими именами типа документа, через множественные объявления типа документа.

При этом имена типа документа являются префиксами к тегам и ссылкам на объект, таким образом разрешая использовать в документе множественные альтернативные наборы начальных тегов, конечных тегов и объектов в дополнение к основному набору. Другая разметка и данные могут быть связаны с экземпляром конкретного типа документа посредством отмеченных разделов, чей статус "INCLUDE" или "IGNORE" устанавливается разрешенным ключевым словом, связанным с этим типом.

В предыдущем примере, если бы тип документа верстки назывался "верстка", параллельная разметка для форматированной позиции списка выглядела бы (без минимизации разметки) следующим образом:

<позиция>

<(верстка)корпус отступ=5>Текст позиции списка.

</позиция>

</(верстка) корпус>

C.3.2 SUBDOC: могут появляться вложенные объекты поддокументов

Документ в SGML состоит из одного или большего числа объектов. Тот объект, в котором документ начинается, называется "объект документа SGML"; это объект, который содержит объявление SGML, идентифицирующее особенности и конкретный синтаксис, используемый по всему документу. Объект документа SGML также содержит одно или большее число определений типа документа, и размечен так, чтобы соответствовать одному или большему числу этих определений.

Объект, который соответствует разметке SGML, называется "объект SGML". Объект SGML может содержать ссылки на "объекты данных, не относящихся к SGML", которые содержат только те данные, которые не могут анализироваться, или на другие объекты SGML, которые не содержат своих собственных определений типа документа, называемые "объекты текста SGML".

Когда используется функция поддокумента, объект SGML может также содержать ссылки на "объекты поддокумента SGML", которые содержат их собственные определения типа документа. Объект поддокумента SGML должен соответствовать объявлению SGML объекта документа SGML, но в других отношениях он устанавливает свою собственную среду. Объявления типов документа и объектов в

объекте, ссылающемся на поддокумент приостанавливаются, пока объект поддокумента открыт, и восстанавливаются, когда он заканчивается. Текущий ранг начинается в поддокументе заново, также как и уникальные идентификаторы, поэтому не могут существовать ссылки на ID между документом SGML и его поддокументами.

Ссылки на поддокументы делаются тем же способом, что и на объекты данных, не относящихся к SGML. Ссылка на общий объект может использоваться в любом месте содержания, где она может быть распознана, и где может появиться символ данных (то есть в смешанном содержании или в заменяемых символьных данных). Как альтернатива, элемент может быть объявлен как имеющий атрибут имени общего объекта, который указывает место объекта поддокумента:

<!ENTITY art1 SYSTEM SUBDOC)

<!ELEMENT статья - - EMPTY>

<!ATTLIST статья файл ENTITY #REQUIRED>

<!>

 Эта тема рассмотрена в следующей статье.

< статья файл=artl >

Обратите внимание, что тип элемента не должен быть тем же, что и тип документа объекта поддокумента. Последний определен присутствующим в нем объявлением типа документа.

Эта функция позволяет включать отдельно созданные документы различных типов в единый документ, такой как антология.

C.3.3 FORMAL: публичные идентификаторы формальны

Когда используется эта функция, публичные идентификаторы имеют формальную структуру с несколькими компонентами:

а) Идентификатор владельца, которым может быть порядковый номер издания ИСО, идентификатор который соответствует стандарту регистрации для гарантии уникальности или частный

("незарегистрированный") идентификатор.

b) Класс публичного текста, который идентифицирует вид регистрируемого текста: документ SGML, набор объектов, определение типа документа, и так далее.

c) Идентификатор публичного текста, который именует зарегистрированный текст,

d) Обозначение используемого естественного языка в форме стандартного двухсимвольного имени.

e) Факультативная версия, которая идентифицирует поддерживаемые устройства вывода для аппаратно-зависимого публичного текста, версии которого доступны больше чем для одного устройства. Система может автоматически подставлять лучшую версию для используемого устройство в данном процессе.

ПРИЛОЖЕНИЕ D

ПУБЛИЧНЫЙ ТЕКСТ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

SGML определяет, как распознается такая разметка, как родовые идентификаторы, атрибуты и ссылки на объект, но *конкретные* GI или другие имена не являются частью языка. Словарь составляется пользователями в соответствии с их потребностями, и определяется в определениях типов документа и процессов связи.

Существенное преимущество может быть получено в результате индивидуального использования SGML (как это имело место в случае с ранними проектами родового кодирования и обобщенного языка разметки). Однако, торговые организации, технические общества и подобные им группы, желающие обмениваться документами, могут извлекать дополнительные преимущества, совместно используя определения типов документов и другие конструкции разметки.

В связи с этим, SGML включает синтаксис для ссылок на текст с публичными идентификаторами. Это приложение описывает некоторые применения для публичного текста и определяет некоторые публичные наборы объектов для непосредственного использования.

D.1 Наборы элементов

Наборы объявлений элементов могут создаваться для тех элементов, которые обычно не существуют как независимые документы, но которые могут возникать в различных типах документов.

D.1.1 Обычные типы элементов

Существуют некоторые элементы, имеющие хорошо узнаваемые общие формы, которые могут появляться в ряде документов. Некоторые

из них также включают специализированные нотации содержания данных (см. ниже).

К некоторым примерам из Соединенных Штатов с источниками существующих спецификаций относятся:

a) Юридические цитаты (Гарвардский юридический обзор)

b) Математические формулы (Ассоциация Американских Издателей)

c) Химические формулы (Американское Химическое Общество)

d) Библиографические статьи (Библиотека Конгресса)

e) Имя и адрес (организации почтовых рассылок, издатели справочников, телефонные компании)

D.1.2 Формальные типы элементов

Существует множество конфигураций элементов, которые встречаются в ряде документов и элементов, но обычно с различиями в GI или другими изменениями. Публичные *формальные* описания таких конфигураций могут служить руководством для построения конкретных описаний, которые могут потребоваться пользователю.

Примеры таких элементов:

a) Параграф (для типичных приложений форматирования)

b) Параграф (для детального синтаксического или другого анализа)

c) Таблицы

d) Вложенные ранжированные структуры параграфа

e) Списки

D.2 Нотации содержания данных

Существуют общепринятые нотации содержания данных, которые могут быть адаптированы для использования с SGML и данными

публичными идентификаторами.

Например, математическая нотация EQN может быть адаптирована следующим образом:

a) Команды, которые указывают начало и конец инструкции EQN, не нужны, поскольку теги для элемента формулы выполнили бы ту функцию.

b) Оставшаяся часть EQN может использоваться без изменений как нотация содержания данных (возможно дополненная описательной разметкой для некоторых математических элементов).

формула,

<q><формула нотация=EQN> E равняется m

умножить на c в квадрате</формула></q>

должна обеспечить быстрое просвещение.

D.3 Иные конкретные синтаксисы

Чтобы максимизировать шанс успешного обмена с самым широким возможным набором систем SGML были выбраны параметры *идентификации избегаемого символа* и *идентификации символов функций*. В отличие от специализированных текстовых процессоров, приложения обработки текста часто находятся в среде операционных систем, над которыми они не имеют никакого управления, и которые не обязательно соответствуют стандартам ИСО, определяющим архитектуру и связи системы. В таких случаях, битовая комбинация, которую SGML будет обрабатывать как данные, может быть интерпретирована операционной системой как управляющий символ, что приведет к аварийному поведению. Для предотвращения таких ошибок, конкретный синтаксис ссылок запрещает прямое появление в документе любых битовых комбинаций, которые могут рассматриваться как средства управления, за исключением четырех универсально распознаваемых символов функций.

Однако, для некоторых национальных языков необходимы дополнительные символы функций, чтобы можно было использовать методы расширения кода, позволяющие клавиатурам и дисплеям реагировать на изменения в символьном алфавите, по мере ввода или исправления текста. Этот раздел определяет два публичных иных конкретных синтаксиса, которые позволяют символам функций для расширения кода появляться в объектах SGML таким способом, который предотвращает ошибочную интерпретацию их или дополнительных графических символов как разметку.

D.3.1 Многокодовые конкретные синтаксисы

Многокодовый базовый конкретный синтаксис описывается параметром *конкретного синтаксиса* объявления SGML, показанного на рис. 11. Его *публичный идентификатор*:

"ISO 8879-1986//SYNTAX Multicode Basic//EN"

Многокодовый конкретный синтаксис ядра - тот же, что и многокодовый базовый конкретный синтаксис, за исключением того, что для параметра "SHORTREF" определено "NONE". Его *публичный идентификатор*:

"ISO 8879-1986//SYNTAX Multicode Core//EN"

Эти синтаксисы позволяют распознавать разметку только в наборе G0, потому что смещение к G1, G2 или G3 начинается с символа функции, который подавляет распознавание разметки.

Функция LS0 восстанавливает распознавание разметки при обратном смещении к G0. Она должна вводиться после того, как появится escape-последовательность, чтобы позволить дальнейшее распознавание разметки.

ПРИМЕЧАНИЕ. Методы для аппаратно-независимого расширения кода, которые позволяют смешанное использование устройств, как

соответствующих ИСО 2022, так и иных, обсуждаются в E.3.

D.4 Наборы объектов

При публикации текста используются десятки тысяч графических символов, из которых относительно немногие включены в наборы стандартных кодированных символов. Однако, даже если существуют стандартные кодированные представления, могут возникать ситуации, в которых они не могут удобно вводиться с клавиатуры, или в которых не возможно отобразить желательное визуальное изображение символов.

Чтобы помочь в преодолении этих барьеров для успешного обмена документами SGML, этот раздел определяет наборы символьных объектов для некоторых широко используемых специальных графических символов. Алфавиты объектов основаны на применимых изданных и предложенных международных стандартах для наборов кодированных символов и текущей практике производственного и профессионального сообщества.

```
                SYNTAX

SHUNCHAR        CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
                   18 19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
BASESET         "ISO 646-1983//CHARSET
                 International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET           0    14 0
                 14     1 "LS0 in ISO 2022"
                 15     1 "LS1 in ISO 2022"
                 16   112 16
                128    14 UNUSED
                142     1 "SS2 in ISO 2022"
                143     1 "SS3 in ISO 2022"
                144   112 UNUSED
FUNCTION        RE                          13
                RS                          10
                SPACE                       32
                TAB             SEPCHAR      9
                 -- Функции для расширения кода графического алфавита --
                 -- Разметка распознается только в наборе G0. --
                ESC             MSOCHAR     27 -- Escape --
                LSO             MSICHAR     15 -- Смещение с блокировкой ноль
                                               (набор G0) --
                LSI             MSOCHAR     14 -- Смещение с блокировкой один
                                               (набор Gl) --
                                               -- LS1R, LS2, LS2R, LS3 и LS3R являются
                                               ESC последовательностями--
                SS2             MSSCHAR    142 -- Одиночное смещение два (набор G2) --
                SS3             MSSCHAR    143 -- Одиночное смещение три (набор G3) --
NAMING          LCNMSTRT        ""
                UCNMSTRT        ""
                LCNMCHAR        "–."        -- Дефис и точка в нижнем регистре --
                UCNMCHAR        "–."        -- такие же, как и верхнем регистре(45 46).
                NAMECASE        GENERAL     YES
                                ENTITY      NO
DELIM           GENERAL         SGMLREF
                SHORTREF        SGMLREF
NAMES           SGMLREF
QUANTITY        SGMLREF
```

Рис. 11. Многокодовый базовый конкретный синтаксис

ПРИМЕЧАНИЕ. Алфавиты объектов обязательно больше и более повторяющиеся, чем наборы символов, поскольку вообще они имеют дело с конструкциями более высокого уровня. Например, для каждого латинского буквенного символа с диакритическим знаком определены

уникальные объекты, тогда как набор символов мог бы представлять такие символы как комбинации букв и символов диакритического знака. Поэтому эти публичные наборы объектов не должны рассматриваться как требования для новых наборов стандартных кодированных символов.

D.4.1 Общие соображения

В этом пункте рассматриваются критерии разработки, применимые к публичным наборам объектов, включенных в данное приложение.

D.4.1.1 Формат объявлений

Опубликованные здесь наборы объектов являются определительными; текст объекта просто состоит из имени объекта в квадратных скобках и комментария, описывающего в большей мере символ, нежели (возможно) его аппаратно-зависимое кодированное представление:

<!ENTITY frac78 SDATA "[frac78]"--=дробь семь восьмых-->

Если, как в приведенном выше примере, комментарий содержит знак равенства, описание по существу является тем, что дано для символа в ИСО 6937, который также содержит визуальное изображение символа.

Если, как в следующем примере, комментарий включает имя (любой длины), которому предшествует косая черта, имя является идентификатором визуального описания символа в MathSci, an expansion of mathfile, от 26 апреля 1985, изданное Американским Математическим Обществом (American Mathematical Society), 201 Charles St., Providence, Rl 02940, U.S.A..

<!ENTITY frown SDATA "[frown]"--/frown R: кривая вниз-->

Комментарий может включать описание ISO 6937, один или несколько идентификаторов MathSci, либо ни одного или все.

ПРИМЕЧАНИЕ. В документе MathSci идентификатору предшествует обратная наклонная черта вместо простой наклонной черты

Комментарий может включать одиночную прописную букву, за которой следует двоеточие, как в предыдущем примере. Буква указывает на то, что при традиционном математическом наборе символ рассматривается иначе, нежели обычный символ, как указано ниже:

Буква	Рассматривается как:
A	Отношение (стрелка)
B	Двойной оператор
C	Закрывающий разграничитель
L	Большой оператор
N	Отношение (отрицательное)
O	Открывающий разграничитель
P	Пунктуация
R	Отношение

D.4.1.2 Соответствующие отображающие наборы объектов

Система будет должна обеспечить соответствующие отображающие наборы объектов для устройств вывода, которые она поддерживает, в которых текст объекта заменяется командами обработки или символьными последовательностями, которые производят желательное визуальное изображение. Имя объекта и описательный комментарий, конечно, остаются теми же. Например, объявление

<!ENTITY frac78 SDATA "7/8"--=дробь семь восьмых-->

может быть использовано в отображающем наборе символьных объектов для устройств вывода, которые не поддерживают ИСО 6937/2, тогда как

<!ENTITY frac78 SDATA "ß"--=дробь семь восьмых-->

может быть использовано в наборе объектов для устройств с 8-разрядным кодированием, которые поддерживают ИСО 6937/2. Для текстового форматера, управляющего фотонаборной машиной, может быть использовано объявление, подобное следующему:

<!ENTITY frac78 SDATA "?bf pi;&#14;?pf"--=дробь 7/8-->

ПРИМЕЧАНИЕ. Все объявления объектов используют ключевое слово "SDATA" как напоминание о том, что текст объекта может быть системно - специфическими символьными данными, которые могут требовать изменений для различных выходных устройств и приложений.

D.4.1.3 Имена объектов

Имена объектов основаны на словах английского языка. Они выбирались исходя из максимального мнемонического значения, и не противоречат логическому и систематическому использованию сокращений.

ПРИМЕЧАНИЕ. Для других языков может потребоваться перевод.

Регистр в именах объектах является существенным, так регистр букв имени может идентифицировать регистр символа, указывать на дублирование линии или использоваться каким-либо иным образом.

Длина имен объектов ограничена шестью символами, в них могут использоваться только буквы и цифры, поэтому они могут использоваться во множестве конкретных синтаксисов.

ПРИМЕЧАНИЕ. Если для часто встречающихся объектов желательны более краткие имена, они могут быть определены в

документах, где имеет место их частое использование.

Некоторые символы имеют разные семантические коннотации в разных прикладных контекстах. Для некоторых из них были определены .множественные объекты

ПРИМЕЧАНИЕ. Если в контексте конкретного документа иное имя будет более выразительным, объект может быть переопределен внутри этого документа.

D.4.1.4 Организация наборов объектов

Наборы объектов были организованы преимущественно так, чтобы отразить структуру наборов символов ИСО 6937 или сгруппировать большие количества подобных символов. Эта организация вряд ли будет оптимальна для большинства приложений, которые будут обычно требовать смешения объектов из нескольких наборов. Предоставляется разрешение копировать в любой форме все или часть публичных наборов объектов этого пункта для использования с удовлетворяющими требованиям SGML системами и приложениями, при условии, что объявление об авторском праве ИСО (включая текст разрешения на копирование) включено во все копии. В частности, могут быть скопированы объекты из нескольких публичных наборов для того, чтобы сформировать новый набор, при условии, что объявление об авторском праве ИСО будет включено в новый набор.

ПРИМЕЧАНИЕ. Если одно имя объекта присутствует больше чем в одном публичном наборе, и оба необходимы в документе, для одного из них внутри документа должен быть объявлен объект с другим именем.

D.4.2 Алфавитные символы

Эта группа наборов символьных объектов использует непротиворечивую схему наименования, в который символ или его транслитерация сопровождается сокращением для диакритического знака и/или указателя нелатинского алфавита. Символ напечатан прописными буквами в имени объекта, когда объект представляет его прописную форму.

D.4.2.1 Латиница

Этот набор объектов состоит из символов латинского алфавита, используемых в западноевропейских языках, отличных от ***UC Letter*** и ***LC Letter***.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для
     использования с удовлетворяющими SGML системами и
     приложениями, как определено в ИСО 8879, при условии, что это
     объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN">
     %IS01atl;
-->
<!ENTITY aacute  SDATA "[aacute  ]"--=строчная a, акут-->
<!ENTITY Aacute  SDATA "[Aacute ]"--=прописная A, акут-->
<!ENTITY acirc   SDATA "[acirc   ]"--=строчная a, циркумфлекс-->
<!ENTITY Acirc   SDATA "[Acirc   ]"--=прописная A, циркумфлекс-->
<!ENTITY agrave  SDATA "[agrave  ]"--=строчная a, грав-->
<!ENTITY Agrave  SDATA "[Agrave ]"--=прописная A, грав-->
```

```
<!ENTITY aring    SDATA "[aring   ]"--=строчная a, кольцо-->
<!ENTITY Aring    SDATA "[Aring   ]"--=прописная A, кольцо-->
<!ENTITY atilde   SDATA "[atilde  ]"--=строчная a, тильда-->
<!ENTITY Atilde   SDATA "[Atilde  ]"--=прописная A, тильда-->
<!ENTITY aumi     SDATA "[aumi    ]"--=строчная a, трема или знак
                                      умляут-->
<!ENTITY Aumi     SDATA "[Aumi    ]"--=прописная A, трема или знак
                                      умляут-->
<!ENTITY aelig    SDATA "[aelig   ]"--=строчная ae дифтонг
                                      (лигатура)-->
<!ENTITY AElig    SDATA "[Aelig   ]"--=прописная AE дифтонг
                                      (лигатура)-->
<!ENTITY ccedil   SDATA "[ccedil  ]"--=строчная c, седиль-->
<!ENTITY Ccedil   SDATA "[Ccedil  ]"--=прописная C, седиль-->
<!ENTITY eth      SDATA "[eth     ]"--=строчная eth, исландская-->
<!ENTITY ETH      SDATA "[ETH     ]"--=прописная Eth, исландская-->
<!ENTITY eacute   SDATA "[eacute  ]"--=строчная e, акут-->
<!ENTITY Eacute   SDATA "[Eacute  ]"--=прописная E, акут-->
<!ENTITY ecirc    SDATA "[ecirc   ]"--=строчная e, циркумфлекс-->
<!ENTITY Ecirc    SDATA "[Ecirc   ]"--=прописная E, циркумфлекс-->
<!ENTITY egrave   SDATA "[egrave  ]"--=строчная e, грав-->
<!ENTITY Egrave   SDATA "[Egrave  ]"--=прописная E, грав-->
<!ENTITY euml     SDATA "[euml    ]"--=строчная e, трема или знак
                                      умляут-->
<!ENTITY Euml     SDATA "[Euml    ]"--=прописная E, трема или знак
                                      умляут-->
<!ENTITY iacute   SDATA "[iacute  ]"--=строчная i, акут-->
<!ENTITY Iacute   SDATA "[Iacute  ]"--=прописная I, акут-->
```

```
<!ENTITY icirc    SDATA "[icirc    ]"--=строчная i, циркумфлекс-->
<!ENTITY Icirc    SDATA "[Icirc    ]"--=прописная I, циркумфлекс-->
<!ENTITY igrave   SDATA "[igrave   ]"--=строчная i, грав-->
<!ENTITY Igrave   SDATA "[Igrave   ]"--=прописная I, грав-->
<!ENTITY iuml     SDATA "[iuml     ]"--=строчная i, трема или знак
                                       умляут-->
<!ENTITY Iuml     SDATA "[Iuml     ]"--=прописная I, трема или знак
                                       умляут-->
<!ENTITY ntilde   SDATA "[ntilde   ]"--=строчная n, тильда-->
<!ENTITY Ntilde   SDATA "[Ntilde   ]"--=прописная N, тильда-->
<!ENTITY oacute   SDATA "[oacute   ]"--=строчная o, акут-->
<!ENTITY Oacute   SDATA "[Oacute   ]"--=прописная O, акут-->
<!ENTITY ocirc    SDATA "[ocirc    ]"--=строчная o, циркумфлекс-->
<!ENTITY Ocirc    SDATA "[Ocirc    ]"--=прописная O, циркумфлекс-->
<!ENTITY ograve   SDATA "[ograve   ]"--=строчная o, грав-->
<!ENTITY Ograve   SDATA "[Ograve   ]"--=прописная O, грав-->
<!ENTITY oslash   SDATA "[oslash   ]"--=строчная o, перечеркнутая-->
<!ENTITY Oslash   SDATA "[Oslash   ]"--=прописная O, перечеркнутая-->
<!ENTITY otilde   SDATA "[otilde   ]"--=строчная o, тильда-->
<!ENTITY Otilde   SDATA "[Otilde   ]"--=прописная O, тильда-->
<!ENTITY ouml     SDATA "[ouml     ]"--=строчная o, трема или знак
                                       умляут-->
<!ENTITY Ouml     SDATA "[Ouml     ]"--=прописная O, трема или знак
                                       умляут-->
<!ENTITY szlig    SDATA "[szlig    ]"--=строчная острая s, немецкая
                                       (лигатура sz)-->
<!ENTITY thorn    SDATA "[thorn    ]"--=строчная thorn, исландская-->
<!ENTITY THOR     SDATA "[THOR     ]"--=прописная THORN,
```

```
            N                N        исландская--^
<!ENTITY uacute  SDATA "[uacute  ]"--=строчная u, акут-->
<!ENTITY Uacute  SDATA "[Uacute  ]"--=прописная U, акут-->
<!ENTITY ucirc   SDATA "[ucirc   ]"--=строчная u, циркумфлекс-->
<!ENTITY Ucirc   SDATA "[Ucirc   ]"--=прописная U, циркумфлекс-->
<!ENTITY ugrave  SDATA "[ugrave  ]"--=строчная u, грав-->
<!ENTITY Ugrave  SDATA "[Ugrave  ]"--=прописная U, грав-->
<!ENTITY uuml    SDATA "[uuml    ]"--=строчная u, трема или знак
                                     умляут-->
<!ENTITY Uuml    SDATA "[Uuml    ]"--=прописная U, трема или знак
                                     умляут-->
<!ENTITY yacute  SDATA "[yacute  ]"--=строчная y, акут-->
<!ENTITY Yacute  SDATA "[Yacute  ]"--=прописная Y, акут-->
<!ENTITY yuml    SDATA "[yumi    ]"--=строчная y, трема или знак
                                     умляут-->
```

Этот набор объектов содержит дополнительные символы латинского алфавита.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для
     использования с
     удовлетворяющими SGML системами и приложениями, как
     определено в
     ИСО 8879, при условии, что это объявление будет включено во все
     копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOlat2 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 2//EN">
     %IS01at2;
-->

<!ENTITY  abreve   SDATA  "[abreve   ]"--=строчная a, краткая-->
<!ENTITY  Abreve   SDATA  "[Abreve   ]"--=прописная A, краткая-->
<!ENTITY  amacr    SDATA  "[amacr    ]"--=строчная a, удлиненная-->
<!ENTITY  Amacr    SDATA  "[Amacr    ]"--=прописная A, удлиненная-->
<!ENTITY  aogon    SDATA  "[aogon    ]"--=строчная a, огонек-->
```

```
<!ENTITY  Aogon   SDATA  "[Aogon   ]"--=прописная A, огонек-->
<!ENTITY  cacute  SDATA  "[cacute  ]"--=строчная c, акут-->
<!ENTITY  Cacute  SDATA  "[Cacute  ]"--=прописная C, акут-->
<!ENTITY  ccaron  SDATA  "[ccaron  ]"--=строчная c, карон-->
<!ENTITY  Ccaron  SDATA  "[Ccaron  ]"--=прописная C, карон-->
<!ENTITY  ccirc   SDATA  "[ccirc   ]"--=строчная c, циркумфлекс-->
<!ENTITY  Ccirc   SDATA  "[Ccirc   ]"--=прописная C, циркумфлекс-->
<!ENTITY  cdot    SDATA  "[cdot    ]"--=строчная c, точка сверху-->
<!ENTITY  Cdot    SDATA  "[Cdot    ]"--=прописная C, точка сверху-->
<!ENTITY  dcaron  SDATA  "[dcaron  ]"--=строчная d, карон-->
<!ENTITY  Dcaron  SDATA  "[Dcaron  ]"--=прописная D, карон-->
<!ENTITY  dstrok  SDATA  "[dstrok  ]"--=строчная d, штрих-->
<!ENTITY  Dstrok  SDATA  "[Dstrok  ]"--=прописная D, штрих-->
<!ENTITY  ecaron  SDATA  "[ecaron  ]"--=строчная e, карон-->
<!ENTITY  Ecaron  SDATA  "[Ecaron  ]"--=прописная E, карон-->
<!ENTITY  edot    SDATA  "[edot    ]"--=строчная e, точка сверху-->
<!ENTITY  Edot    SDATA  "[Edot    ]"--=прописная E, точка сверху-->
<!ENTITY  emacr   SDATA  "[emacr   ]"--=строчная e, удлиненная-->
<!ENTITY  Emacr   SDATA  "[Emacr   ]"--=прописная E, удлиненная-->
<!ENTITY  eogon   SDATA  "[eogon   ]"--=строчная e, огонек-->
<!ENTITY  Eogon   SDATA  "[Eogon   ]"--=прописная E, огонек-->
<!ENTITY  gacute  SDATA  "[gacute  ]"--=строчная g, акут-->
<!ENTITY  gbreve  SDATA  "[gbreve  ]"--=строчная g, краткая-->
<!ENTITY  Gbreve  SDATA  "[Gbreve  ]"--=прописная G, краткая-->
<!ENTITY  Gcedil  SDATA  "[Gcedil  ]"--=прописная G, седиль-->
<!ENTITY  gcirc   SDATA  "[gcirc   ]"--=строчная g, циркумфлекс-->
<!ENTITY  Gcirc   SDATA  "[Gcirc   ]"--=прописная G, циркумфлекс-->
<!ENTITY  gdot    SDATA  "[gdot    ]"--=строчная g, точка сверху-->
<!ENTITY  Gdot    SDATA  "[Gdot    ]"--=прописная G, точка сверху-->
<!ENTITY  hcirc   SDATA  "[hcirc   ]"--=строчная h, циркумфлекс-->
<!ENTITY  Hcirc   SDATA  "[Hcirc   ]"--=прописная H, циркумфлекс-->
<!ENTITY  hstrok  SDATA  "[hstrok  ]"--=строчная h, штрих-->
<!ENTITY  Hstrok  SDATA  "[Hstrok  ]"--=прописная H, штрих-->
<!ENTITY  Idot    SDATA  "[Idot    ]"--=прописная I, точка сверху-->
<!ENTITY  Imacr   SDATA  "[Imacr   ]"--=прописная I, удлиненная-->
<!ENTITY  imacr   SDATA  "[imacr   ]"--=строчная i, удлиненная-->
<!ENTITY  ijlig   SDATA  "[ijlig   ]"--=строчная ij лигатура-->
<!ENTITY  IJlig   SDATA  "[Ijlig   ]"--=прописная IJ лигатура-->
<!ENTITY  inodot  SDATA  "[inodot  ]"--=строчная i без точки-->
<!ENTITY  iogon   SDATA  "[iogon   ]"--=строчная i, огонек-->
<!ENTITY  Iogon   SDATA  "[Iogon   ]"--=прописная I, огонек-->
<!ENTITY  itilde  SDATA  "[itilde  ]"--=строчная i, тильда-->
<!ENTITY  Itilde  SDATA  "[Itilde  ]"--=прописная I, тильда-->
<!ENTITY  jcirc   SDATA  "[jcirc   ]"--=строчная j, циркумфлекс-->
<!ENTITY  Jcirc   SDATA  "[Jcirc   ]"--=прописная J, циркумфлекс-->
<!ENTITY  kcedil  SDATA  "[kcedil  ]"--=строчная k, седиль-->
<!ENTITY  Kcedil  SDATA  "[Kcedil  ]"--=прописная K, седиль-->
<!ENTITY  kgreen  SDATA  "[kgreen  ]"--=строчная k, гренландская-->
<!ENTITY  lacute  SDATA  "[lacute  ]"--=строчная l, акут-->
<!ENTITY  Lacute  SDATA  "[Lacute  ]"--=прописная L, акут-->
```

```
<!ENTITY  lcaron    SDATA  "[lcaron    ]"--=строчная 1, карон-->
<!ENTITY  Lcaron    SDATA  "[Lcaron    ]"--=прописная L, карон-->
<!ENTITY  lcedil    SDATA  "[lcedil    ]"--=строчная 1, седиль-->
<!ENTITY  Lcedil    SDATA  "[Lcedil    ]"--=прописная L, седиль-->
<!ENTITY  lmidot    SDATA  "[lmidot    ]"--=строчная 1, точка в центре-->
<!ENTITY  Lmidot    SDATA  "[Lmidot    ]"--=прописная L, точка в центре-->
<!ENTITY  Istrok    SDATA  "[lstrok    ]"--=строчная 1, штрих-->
<!ENTITY  Lstrok    SDATA  "[Lstrok    ]"--=прописная L, штрих-->
<!ENTITY  nacute    SDATA  "[nacute    ]"--=строчная n, акут-->
<!ENTITY  Nacute    SDATA  "[Nacute    ]"--=прописная N, акут-->
<!ENTITY  eng       SDATA  "[eng       ]"--=строчная eng, Lapp-->
<!ENTITY  ENG       SDATA  "[ENG       ]"--=прописная ENG, Lapp-->
<!ENTITY  napos     SDATA  "[napos     ]"--=строчная n, апостроф-->
<!ENTITY  ncaron    SDATA  "[ncaron    ]"--=строчная n, карон-->
<!ENTITY  Ncaron    SDATA  "[Ncaron    ]"--=прописная N, карон-->
<!ENTITY  ncedil    SDATA  "[ncedil    ]"--=строчная n, седиль-->
<!ENTITY  Ncedil    SDATA  "[Ncedil    ]"--=прописная N, седиль-->
<!ENTITY  odblac    SDATA  "[odblac    ]"--=строчная o, двойной акут-->
<!ENTITY  Odblac    SDATA  "[Odblac    ]"--=прописная O, двойной акут-->
<!ENTITY  Omacr     SDATA  "[Omacr     ]"--=прописная O, удлиненная-->
<!ENTITY  omacr     SDATA  "[omacr     ]"--=строчная O, удлиненная-->
<!ENTITY  oelig     SDATA  "[oelig     ]"--=строчная oe лигатура-->
<!ENTITY  OElig     SDATA  "[OElig     ]"--=прописная OE лигатура-->
<!ENTITY  racute    SDATA  "[racute    ]"--=строчная r, акут-->
<!ENTITY  Racute    SDATA  "[Racute    ]"--=прописная R, акут-->
<!ENTITY  rcaron    SDATA  "[rcaron    ]"--=строчная r, карон-->
<!ENTITY  Rcaron    SDATA  "[Rcaron    ]"--=прописная R, карон-->
<!ENTITY  rcedil    SDATA  "[rcedil    ]"--=строчная r, седиль-->
<!ENTITY  Rcedil    SDATA  "[Rcedil    ]"--=прописная R, седиль-->
<!ENTITY  sacute    SDATA  "[sacute    ]"--=строчная s, акут-->
<!ENTITY  Sacute    SDATA  "[Sacute    ]"--=прописная S, акут-->
<!ENTITY  scaron    SDATA  "[scaron    ]"--=строчная s, карон-->
<!ENTITY  Scaron    SDATA  "[Scaron    ]"--=прописная S, карон-->
<!ENTITY  scedil    SDATA  "[scedil    ]"--=строчная s, седиль-->
<!ENTITY  Scedil    SDATA  "[Scedil    ]"--=прописная S, седиль-->
<!ENTITY  scirc     SDATA  "[scirc     ]"--=строчная s, циркумфлекс-->
<!ENTITY  Scirc     SDATA  "[Scirc     ]"--=прописная S, циркумфлекс-->
<!ENTITY  tcaron    SDATA  "[tcaron    ]"--=строчная t, карон-->
<!ENTITY  Tcaron    SDATA  "[Tcaron    ]"--=прописная T, карон-->
<!ENTITY  tcedil    SDATA  "[tcedil    ]"--=строчная t, седиль-->
<!ENTITY  Tcedil    SDATA  "[Tcedil    ]"--=прописная T, седиль-->
<!ENTITY  tstrok    SDATA  "[tstrok    ]"--=строчная t, штрих-->
<!ENTITY  Tstrok    SDATA  "[Tstrok    ]"--=прописная T, штрих-->
<!ENTITY  ubreve    SDATA  "[ubreve    ]"--=строчная u, краткая-->
<!ENTITY  Ubreve    SDATA  "[Ubreve    ]"--=прописная U, краткая-->
<!ENTITY  udblac    SDATA  "[udblac    ]"--=строчная u, двойной акут-->
<!ENTITY  Udblac    SDATA  "[Udblac    ]"--=прописная U, двойной акут-->
<!ENTITY  umacr     SDATA  "[umacr     ]"--=строчная u, удлиненная-->
<!ENTITY  Umacr     SDATA  "[Umacr     ]"--=прописная U, удлиненная-->
<!ENTITY  uogon     SDATA  "[uogon     ]"--=строчная u, огонек-->
```

```
<!ENTITY  Uogon    SDATA  "[Uogon    ]"--=прописная U, огонек-->
<!ENTITY  uring    SDATA  "[uring    ]"--=строчная u, кольцо-->
<!ENTITY  Uring    SDATA  "[Uring    ]"--=прописная U, кольцо-->
<!ENTITY  Utilde   SDATA  "[utilde   ]"--=строчная u, тильда-->
<!ENTITY  Utilde   SDATA  "[Utilde   ]"--=прописная U, тильда-->
<!ENTITY  Wcirc    SDATA  "[wcirc    ]"--=строчная w, циркумфлекс-->
<!ENTITY  Wcirc    SDATA  "[Wcirc    ]"--=прописная W, циркумфлекс-->
<!ENTITY  Ycirc    SDATA  "[ycirc    ]"--=строчная y, циркумфлекс-->
<!ENTITY  Ycirc    SDATA  "[Ycirc    ]"--=прописная Y, циркумфлекс-->
<!ENTITY  Yumi     SDATA  "[Yumi     ]"--=прописная Y, трема или знак умляут-->
<!ENTITY  Zacute   SDATA  "[zacute   ]"--=строчная z, акут-->
<!ENTITY  Zacute   SDATA  "[Zacute   ]"--=прописная Z, акут-->
<!ENTITY  Zcaron   SDATA  "[zcaron   ]"--=строчная z, карон-->
<!ENTITY  Zcaron   SDATA  "[Zcaron   ]"--=прописная Z, карон-->
<!ENTITY  Zdot     SDATA  "[zdot     ]"--=строчная z, точка сверху-->
<!ENTITY  Zdot     SDATA  "[Zdot     ]"--=прописная Z, точка сверху-->
```

D.4.2.2 Символы греческого алфавита

Этот набор объектов состоит из букв греческого алфавита. Имена объектов отражают их предполагаемое использование как символов языка, а не как элементов формул. (Греческие символьные объекты для технического использования определены ниже.)

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
      удовлетворяющими SGML системами и приложениями, как
      определено в
      ИСО 8879, при условии, что это объявление будет включено во все
      копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOgrk1 PUBLIC
        "ISO 8879-1986//ENTITIES Greek Letters//EN">
      %ISOgrk1;
-->
```

```
<!ENTITY   agr      SDATA  "[agr      ]"--=строчная альфа, греческая-->
<!ENTITY   Agr      SDATA  "[Agr      ]"--=прописная Альфа, греческая-->
<!ENTITY   bgr      SDATA  "[bgr      ]"--=строчная бета, греческая-->
<!ENTITY   Bgr      SDATA  "[Bgr      ]"--=прописная Бета, греческая-->
<!ENTITY   ggr      SDATA  "[ggr      ]"--=строчная гамма, греческая-->
<!ENTITY   Ggr      SDATA  "[Ggr      ]"--=прописная Гамма, греческая-->
<!ENTITY   dgr      SDATA  "[dgr      ]"--=строчная дельта, греческая-->
<!ENTITY   Dgr      SDATA  "[Dgr      ]"--=прописная Дельта, греческая-->
<tENTITY   egr      SDATA  "[egr      ]"--=строчная эпсилон, греческая-->
<!ENTITY   Egr      SDATA  "[Egr      ]"--=прописная Эпсилон, греческая-->
<!ENTITY   zgr      SDATA  "[zgr      ]"--=строчная зета, греческая-->
<!ENTITY   Zgr      SDATA  "[Zgr      ]"--=прописная Зета, греческая-->
<!ENTITY   eegr     SDATA  "[eegr     ]"--=строчная эта, греческая-->
<!ENTITY   Eegr     SDATA  "[Eegr     ]"--=прописная Эта, греческая-->
<!ENTITY   thgr     SDATA  "[thgr     ]"--=строчная тета, греческая-->
<!ENTITY   THgr     SDATA  "[THgr     ]"--=прописная Тета, греческая-->
<!ENTITY   igr      SDATA  "[igr      ]"--=строчная йота, греческая-->
<!ENTITY   Igr      SDATA  "[Igr      ]"—прописная Йота, греческая-->
<!ENTITY   kgr      SDATA  "[kgr      ]"--=строчная каппа, греческая-->
<!ENTITY   Kgr      SDATA  "[Kgr      ]"--=прописная Каппа, греческая-->
<!ENTITY   lgr      SDATA  "[lgr      ]"--=строчная лямбда, греческая-->
<!ENTITY   Lgr      SDATA  "[Lgr      ]"--=прописная Лямбда, греческая-->
<!ENTITY   mgr      SDATA  "[mgr      ]"--=строчная мю, греческая-->
<!ENTITY   Mgr      SDATA  "[Mgr      ]"--=прописная Мю, греческая-->
<!ENTITY   ngr      SDATA  "[ngr      ]"--=строчная ню, греческая-->
<!ENTITY   Ngr      SDATA  "[Ngr      ]"--=прописная Ню, греческая-->
<!ENTITY   xgr      SDATA  "[xgr      ]"--=строчная кси, греческая-->
<!ENTITY   Xgr      SDATA  "[Xgr      ]"--=прописная Кси, греческая-->
<!ENTITY   ogr      SDATA  "[ogr      ]"--=строчная омикрон, греческая-->
<!ENTITY   Ogr      SDATA  "[Ogr      ]"--=прописная Омикрон, греческая-->
<!ENTITY   pgr      SDATA  "[pgr      ]"--=строчная пи, греческая-->
<!ENTITY   Pgr      SDATA  "[Pgr      ]"--=прописная Пи, греческая-->
<!ENTITY   rgr      SDATA  "[rgr      ]"--=строчная ро, греческая-->
<!ENTITY   Rgr      SDATA  "[Rgr      ]"--=прописная Ро, греческая-->
<!ENTITY   sgr      SDATA  "[sgr      ]"--=строчная сигма, греческая-->
<!ENTITY   Sgr      SDATA  "[Sgr      ]"--=прописная Сигма, греческая-->
<!ENTITY   sfgr     SDATA  "[sfgr     ]"--=конечная строчная сигма, греческая-->
<!ENTITY   tgr      SDATA  "[tgr      ]"--=строчная тау, греческая-->
<!ENTITY   Tgr      SDATA  "[Tgr      ]"--=прописная Тау, греческая-->
<!ENTITY   ugr      SDATA  "[ugr      ]"--=строчная юпсилон, греческая-->
<!ENTITY   Ugr      SDATA  "[Ugr      ]"--=прописная Юпсилон, греческая-->
<!ENTITY   phgr     SDATA  "[phgr     ]"--=строчная фи, греческая-->
<!ENTITY   PHgr     SDATA  "[PHgr     ]"--=прописная Фи, греческая-->
<!ENTITY   khgr     SDATA  "[khgr     ]"--=строчная хи, греческая-->
<!ENTITY   KHgr     SDATA  "[KHgr     ]"--=прописная Хи, греческая-->
<!ENTITY   psgr     SDATA  "[psgr     ]"--=строчная пси, греческая-->
<!ENTITY   PSgr     SDATA  "[PSgr     ]"--=прописная Пси, греческая-->
<!ENTITY   ohgr     SDATA  "[ohgr     ]"--=строчная омега, греческая-->
<!ENTITY   Ohgr     SDATA  "[Ohgr     ]"--=прописная Омега, греческая-->
```

Этот набор объектов содержит дополнительные символы, необходимые для греческого шрифта Monotoniko.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOgrk2 PUBLIC
       "ISO 8879-1986//ENTITIES Monotoniko Greek//EN">
     %ISOgrk2;
-->
<!ENTITY  Aacgr   SDATA  "[aacgr     ]"--=строчная альфа, ударение, греческая-->
<!ENTITY  Aacgr   SDATA  "[Aacgr     ]"--=прописная Альфа, ударение, греческая-->
<!ENTITY  Eacgr   SDATA  "[eacgr     ]"--=строчная эпсилон, ударение, греческая-->
<!ENTITY  Eacgr   SDATA  "[Eacgr     ]"--=прописная Эпсилон, ударение, греческая-->
<!ENTITY  Eeacgr  SDATA  "[eeacgr    ]"--=строчная эта, ударение, греческая-->
<!ENTITY  Eeacgr  SDATA  "[EEacgrr   ]"--=прописная Эта, ударение, греческая-->
<!ENTITY  Idigr   SDATA  "[idigr     ]"--=строчная йота, трема, греческая-->
<!ENTITY  Idigr   SDATA  "[Idigr     ]"--=прописная Йота, трема, греческая-->
<!ENTITY  Iacgr   SDATA  "[iacgr     ]"--=строчная йота, ударение, греческая-->
<!ENTITY  Lacgr   SDATA  "[Iacgr     ]"--=прописная Йота, ударение, греческая-->
<!ENTITY  Idiagr  SDATA  "[idiagr    ]"--=строчная йота, трема, ударение, греческая-->
<!ENTITY  Oacgr   SDATA  "[oacgr     ]"--=строчная омикрон, ударение, греческая-->
<!ENTITY  Oacgr   SDATA  "[Oacgr     ]"--=прописная Омикрон, ударение, греческая-->
<!ENTITY  Udigr   SDATA  "[udigr     ]"--=строчная юпсилон, трема, греческая-->
<!ENTITY  Udigr   SDATA  "[Udigr     ]"--=прописная Юпсилон, трема, греческая-->
<!ENTITY  Uacgr   SDATA  "[uacgr     ]"--=строчная юпсилон, ударение, греческая-->
<!ENTITY  Uacgr   SDATA  "[Uacgr     ]"--=прописная Юпсилон, ударение, греческая-->
<!ENTITY  udiagr  SDATA  "[udiagr    ]"--=строчная юпсилон, трема, ударение, греческая-->
<!ENTITY  ohacgr  SDATA  "[ohacgr    ]"--=строчная омега, ударение, греческая-->
<!ENTITY  OHacgr  SDATA  "[OHacgr    ]"--=прописная Омега, ударение, греческая-->
```

D.4.2.3 Символы кириллических алфавитов

Этот набор объектов содержит символы кириллицы, используемые в русском языке.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOcyrl PUBLIC
       "ISO 8879-1986//ENTITIES Russian Cyrillic//EN">
     %ISOcyrl;
-->
<!ENTITY  acy     SDATA  "[acy     ]"--=строчная а, кириллица-->
<!ENTITY  Acy     SDATA  "[Acy     ]"--=прописная А, кириллица-->
<!ENTITY  bey     SDATA  "[bey     ]"--=строчная бэ кириллица-->
<!ENTITY  Bey     SDATA  "[Bey     ]"--=прописная БЭ, кириллица-->
<!ENTITY  vcy     SDATA  "[vcy     ]"--=строчная вэ, кириллица-->
<!ENTITY  Vcy     SDATA  "[Vcy     ]"--=прописная ВЭ, кириллица-->
<!ENTITY  gcy     SDATA  "[gcy     ]"--=строчная гэ, кириллица-->
<!ENTITY  Gcy     SDATA  "[Gcy     ]"--=прописная ГЭ, кириллица-->
<!ENTITY  dcy     SDATA  "[dcy     ]"--=строчная дэ, кириллица-->
<!ENTITY  Dcy     SDATA  "[Dcy     ]"--=прописная ДЭ, кириллица-->
<!ENTITY  iecy    SDATA  "[iecy    ]"--=строчная е, кириллица-->
<!ENTITY  IEcy    SDATA  "[Iecy    ]"--=прописная Е, кириллица-->
<!ENTITY  iocy    SDATA  "[iocy    ]"--=строчная ё, Russian-->
<!ENTITY  IOcy    SDATA  "[IOcy    ]"--=прописная Ё, Russian-->
<!ENTITY  zhcy    SDATA  "[zhcy    ]"--=строчная же, кириллица-->
<!ENTITY  ZHcy    SDATA  "[ZHcy    ]"--=прописная ЖЕ, кириллица-->
<!ENTITY  zcy     SDATA  "[zcy     ]"--=строчная зэ, кириллица-->
<!ENTITY  Zcy     SDATA  "[Zcy     ]"--=прописная ЗЭ, кириллица-->
<!ENTITY  icy     SDATA  "[icy     ]"--=строчная и, кириллица-->
<!ENTITY  Icy     SDATA  "[Icy     ]"--=прописная И, кириллица-->
<!ENTITY  jcy     SDATA  "[jcy     ]"--=строчная и краткое, кириллица-->
<!ENTITY  Jcy     SDATA  "[Jcy     ]"--=прописная И краткое, кириллица-->
```

```
<!ENTITY  key      SDATA  "[kcy      ]"--=строчная ка, кириллица-->
<!ENTITY  Key      SDATA  "[Kcy      ]"--=прописная КА, кириллица -->
<!ENTITY  lcy      SDATA  "[lcy      ]"--=строчная эль, кириллица-->
<!ENTITY  Ley      SDATA  "[Lcy      ]"--=прописная ЭЛЬ, кириллица-->
<!ENTITY  mcy      SDATA  "[mcy      ]"--==строчная эм, кириллица-->
<!ENTITY  Mcy      SDATA  "[Mcy      ]"--=прописная ЭМ, кириллица-->
<!ENTITY  ncy      SDATA  "[ncy      ]"--=строчная эн, кириллица-->
<!ENTITY  Ncy      SDATA  ″[Ncy      ]"--=прописная ЭН, кириллица-->
<!ENTITY  ocy      SDATA  "[ocy      ]"--=строчная о, кириллица-->
<!ENTITY  Ocy      SDATA  "[Ocy      ]"--=прописная О, кириллица-->
<!ENTITY  pcy      SDATA  "[Pcy      ]"--=строчная пэ, кириллица-->
<!ENTITY  Pcy      SDATA  "[Pcy      ]"--=прописная ПЭ, кириллица-->
<!ENTITY  rcy      SDATA  "[rcy      ]"--=строчная эр, кириллица -->
<!ENTITY  Rcy      SDATA  "[Rcy      ]"--=прописная ЭР, кириллица-->
<!ENTITY  scy      SDATA  "[scy      ]"--=строчная эс, кириллица-->
<!ENTITY  Scy      SDATA  "[Scy      ]"--=прописная ЭС, кириллица-->
<!ENTITY  tcy      SDATA  "[tcy      ]"--=строчная тэ, кириллица-->
<!ENTITY  Tcy      SDATA  "[Tcy      ]"--=прописная ТЭ, кириллица-->
<!ENTITY  ucy      SDATA  "[ucy      ]"--=строчная у, кириллица-->
<!ENTITY  Ucy      SDATA  "[Ucy      ]"--=прописная У, кириллица-->
<!ENTITY  fey      SDATA  "[fcy      ]"--=строчная эф, кириллица-->
<!ENTITY  Fey      SDATA  "[Fcy      ]"--=прописная ЭФ, Cyrlllic-->
<!ENTITY  khcy     SDATA  "[khcy     ]"--=строчная ха, кириллица-->
<!ENTITY  KHcy     SDATA  "[KHcy     ]"--=прописная ХА, кириллица-->
<!ENTITY  tscy     SDATA  "[tscy     ]"--=строчная цэ, кириллица-->
<!ENTITY  TScy     SDATA  "[TScy     ]"--=прописная ЦЭ, кириллица-->
<!ENTITY  chcy     SDATA  "[chcy     ]"--=строчная че, кириллица-->
<!ENTITY  CHcy     SDATA  "[CHcy     ]"--=прописная ЧЕ, кириллица-->
<!ENTITY  shcy     SDATA  "[shcy     ]"--=строчная ша, кириллица-->
<!ENTITY  SHcy     SDATA  "[SHcy     ]"--=прописная ША, кириллица-->
<!ENTITY  shchcy   SDATA  "[shchcy   ]"--=строчная ща, кириллица-->
<!ENTITY  SHCHcy   SDATA  "[SHCHcy   ]"--=прописная ЩА, кириллица-->
<!ENTITY  hardcy   SDATA  "[hardcy   ]"--=строчная твердый знак, кириллица-->
<!ENTITY  HARDcy   SDATA  "[HARDcy   ]"--=прописная ТВЕРДЫЙ знак, кириллица-->
<!ENTITY  ycy      SDATA  "[ycy      ]"--=строчная ы, кириллица-->
<!ENTITY  Ycy      SDATA  "[Ycy      ]"--=прописная Ы, кириллица-->
<!ENTITY  softcy   SDATA  "[softcy   ]"--=строчная мягкий знак, кириллица-->
<!ENTITY  SOFTcy   SDATA  "[SOFTcy   ]"--=прописная МЯГКИЙ знак, кириллица-->
<!ENTITY  ecy      SDATA  "[ecy      ]"--=строчная э, кириллица-->
<!ENTITY  Ecy      SDATA  "[Ecy      ]"--=прописная Э, кириллица-->
<!ENTITY  yucy     SDATA  "[yucy     ]"--=строчная ю, кириллица-->
<!ENTITY  YUcy     SDATA  "[YUcy     ]"--=прописная Ю, кириллица-->
<!ENTITY  yacy     SDATA  "[yacy     ]"--=строчная я, кириллица-->
<!ENTITY  YAcy     SDATA  "[YAcy     ]"--=прописная Я, кириллица-->
<!ENTITY  numero   SDATA  "[numero   ]"--=знак номер-->
```

Этот набор объектов состоит из символов кириллицы, которые не используются в русском языке.

<!-- (C) Международная организация по стандартизации, 1986

-->

<!-- Набор символьных объектов. Типичный вызов:

<!ENTITY % ISOcyr2 PUBLIC

"ISO 8879-1986//ENTITIES Non-Russian Cyrillic//EN">

%ISOcyr2;

-->

```
<!ENTITY   djcy      SDATA   "[djcy      ]"--=строчная dje, сербская-->
<!ENTITY   DJcy      SDATA   "[DJcy      ]"--=прописная DJE, сербская-->
<!ENTITY   gjcy      SDATA   "[gjcy      ]"--=строчная gje, македонская-->
<!ENTITY   GJcy      SDATA   "[GJcy      ]"--=прописная GJE македонская-->
<!ENTITY   jukcy     SDATA   "[jukcy     ]"--=строчная je, украинская-->
<!ENTITY   Jukcy     SDATA   "[Jukcy     ]"--=прописная JE, украинская-->
<!ENTITY   dscy      SDATA   "[dscy      ]"--=строчная dse, македонская-->
<!ENTITY   DScy      SDATA   "[DScy      ]"--=прописная DSE, македонская-->
<!ENTITY   iukcy     SDATA   "[iukcy     ]"--=строчная i, украинская-->
<!ENTITY   Iukcy     SDATA   "[Iukcy     ]"--=прописная I, украинская-->
<!ENTITY   yicy      SDATA   "[yicy      ]"--=строчная yi, украинская-->
<!ENTITY   YIcy      SDATA   "[YIcy      ]"--=прописная YI, украинская-->
<!ENTITY   jsercy    SDATA   "[jsercy    ]"--=строчная je, сербская-->
<!ENTITY   Jsercy    SDATA   "[Jsercy    ]"--=прописная JE, сербская-->
<!ENTITY   ljcy      SDATA   "[ljcy      ]"--=строчная lje, сербская-->
<!ENTITY   LJcy      SDATA   "[LJcy      ]"--=прописная LJE, сербская-->
<!ENTITY   njcy      SDATA   "[njcy      ]"--=строчная nje, сербская-->
<!ENTITY   NJcy      SDATA   "[Njcy      ]"--=прописная NJE, сербская-->
<!ENTITY   tshcy     SDATA   "[tshcy     ]"--=строчная tshe, сербская-->
<!ENTITY   TSHcy     SDATA   "[TSHcy     ]"--=прописная TSHE, сербская-->
<!ENTITY   kjcy      SDATA   "[kjcy      ]"--=строчная kje македонская-->
<!ENTITY   KJcy      SDATA   "[KJcy      ]"--=прописная KJE, македонская-->
<!ENTITY   ubrcy     SDATA   "[ubrcy     ]"--=строчная u, белорусская-->
<!ENTITY   Ubrcy     SDATA   "[Ubrcy     ]"--=прописная U, белорусская-->
<!ENTITY   dzcy      SDATA   "[dzcy      ]"--=строчная dze, сербская-->
<!ENTITY   DZcy      SDATA   "[DZcy      ]"--=прописная dze, сербская-->
```

D.4.3 Общее употребление

D.4.3.1 Цифровые и специальные графические символы

Этот набор включает, среди прочего, символы минимальных данных и символы разметки конкретного синтаксиса ссылок. Такие символы обычно могут непосредственно набираться на клавиатуре, но когда они назначены на роли разграничителей, может быть необходима ссылка на объект, чтобы ввести их как данные.

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
      удовлетворяющими SGML системами и приложениями, как
      определено в
      ИСО 8879, при условии, что это объявление будет включено во все
      копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOnum PUBLIC
        "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN">
      %ISOnum;
-->
<!ENTITY  half    SDATA  "[half    ]"--=дробь одна вторая-->
<!ENTITY  fracl2  SDATA  "[fracl2  ]"--=дробь одна вторая-->
<!ENTITY  fracl4  SDATA  "[fracl4  ]"--=дробь одна четвертая-->
<!ENTITY  frac34  SDATA  "[frac34  ]"--=дробь три четвертых-->
<!ENTITY  fracl8  SDATA  "[fracl8  ]"--=дробь одна восьмая-->
<!ENTITY  frac38  SDATA  "[frac38  ]"--=дробь три восьмых-->
<!ENTITY  frac58  SDATA  "[frac58  ]"--=дробь пять восьмых-->
<!ENTITY  frac78  SDATA  "[frac78  ]"--=дробь семь восьмых-->
<!ENTITY  supl    SDATA  "[supl    ]"--=верхний индекс один-->
<!ENTITY  sup2    SDATA  "[sup2    ]"--=верхний индекс два-->
<!ENTITY  sup3    SDATA  "[sup3    ]"--=верхний индекс три-->
<!ENTITY  plus    SDATA  "[plus    ]"--=знак плюс B:-->
<!ENTITY  plusmn  SDATA  "[plusmn  ]"--/pm B: =знак плюс или минус-->
<!ENTITY  lt      SDATA  "[lt      ]"--=знак меньше R:-->
<!ENTITY  equals  SDATA  "[equals  ]"--=знак равно R:-->
```

```
<!ENTITY   gt       SDATA  "[gt       ]"--=знак больше R:-->
<!ENTITY   divide   SDATA  "[divide   ]"--/div B: =знак деления-->
<!ENTITY   times    SDATA  "[times    ]"--/times B: =знак умножения-->
<!ENTITY   curren   SDATA  "[curren   ]"--=знак валюты-->
<!ENTITY   pound    SDATA  "[pound    ]"--=знак фунта-->
<!ENTITY   dollar   SDATA  "[dollar   ]"--=занк доллара-->
<!ENTITY   cent     SDATA  "[cent     ]"--=знак цента-->
<!ENTITY   yen      SDATA  "[yen      ]"--/yen =знак иены-->
<!ENTITY   num      SDATA  "[num      ]"--=знак номера-->
<!ENTITY   percnt   SDATA  "[percnt   ]"--=знак процента-->
<!ENTITY   amp      SDATA  "[amp      ]"--=амперсант-->
<!ENTITY   ast      SDATA  "[ast      ]"--/ast B: =звездочка-->
<!ENTITY   commat   SDATA  "[commat   ]"--=коммерческое at-->
<!ENTITY   lsqb     SDATA  "[lsqb     ]"--/lbrack 0: =левая квадратная скобка-->
<!ENTITY   bsol     SDATA  "[bsol     ]"--/backslash =обратная косая черта-->
<!ENTITY   rsqb     SDATA  "[rsqb     ]"--/rbrack C: =правая квадратная скобка-->
<!ENTITY   lcub     SDATA  "[lcub     ]"--/lbrace 0: =левая фигурная скобка-->
<!ENTITY   horbar   SDATA  "[horbar   ]"--=горизонтальная линия-->
<!ENTITY   verbar   SDATA  "[verbar   ]"--/vert =вертикальная линия-->
<!ENTITY   rcub     SDATA  "[rcub     ]"--/rbrace C: =правая фигурная скобка-->
<!ENTITY   micro    SDATA  "[micro    ]"--=знак микро-->
<!ENTITY   ohm      SDATA  "[ohm      ]"--=знак Ом-->
<!ENTITY   deg      SDATA  "[deg      ]"--=знак степени-->
<!ENTITY   ordm     SDATA  "[ordm     ]"--=порядковый указатель, мужской-->
<!ENTITY   ordf     SDATA  "[ordf     ]"--=порядковый указатель, женский -->
<!ENTITY   sect     SDATA  "[sect     ]"--=знак раздела-->
<!ENTITY   para     SDATA  "[para     ]"--=знак параграфа-->
<!ENTITY   middot   SDATA  "[middot   ]"--/centerdot B: =средняя точка-->
<!ENTITY   larr     SDATA  "[larr     ]"--/leftarrow /gets A: =левая стрелка-->
<!ENTITY   rarr     SDATA  "[rarr     ]"--/rightarrow /to A: =правая стрелка-->
<!ENTITY   uarr     SDATA  "[uarr     ]"--/uparrow A: ==верхняя стрелка-->
<!ENTITY   darr     SDATA  "[darr     ]"--/downarrow A: =нижняя стрелка-->
<!ENTITY   copy     SDATA  "[copy     ]"--=знак авторского права-->
<!ENTITY   reg      SDATA  "[reg      ]"--/circledR =охраняемый знак -->
<!ENTITY   trade    SDATA  "[trade    ]"--=знак торговой марки-->
<!ENTITY   brvbar   SDATA  "[brvbar   ]"--=разорванная (вертикальная) линия-->
<!ENTITY   not      SDATA  "[not      ]"--/neg /Inot =знак не-->
<!ENTITY   sung     SDATA  "[sung     ]"--=музыкальная нота (знак нотного текста)-->
<!ENTITY   excl     SDATA  "[excl     ]"--=восклицательный знак-->
<!ENTITY   iexcl    SDATA  "[iexcl    ]"--=перевернутый восклицательный знак-->
<!ENTITY   quot     SDATA  "[quot     ]"--=знак кавычек-->
<!ENTITY   apos     SDATA  "[apos     ]"--=апостроф-->
<!ENTITY   lpar     SDATA  "[lpar     ]"—0: =левая круглая скобка-->
<!ENTITY   rpar     SDATA  "[rpar     ]"—C: =правая круглая скобка-->
<!ENTITY   comma    SDATA  "[comma    ]"—P: =запятая-->
<!ENTITY   lowbar   SDATA  "[lowbar   ]"--=нижняя линия-->
<!ENTITY   hyphen   SDATA  "[hyphen   ]"--=дефис-->
<!ENTITY   period   SDATA  "[period   ]"--=точка-->
<!ENTITY   sol      SDATA  "[sol      ]"--=косая черта-->
<!ENTITY   colon    SDATA  "[colon    ]"--/colon P:-->
```

```
<!ENTITY  semi     SDATA  "[semi     ]"--=точка с запятой P:-->
<!ENTITY  quest    SDATA  "[quest    ]"--=вопросительный знак-->
<!ENTITY  iquest   SDATA  "[iquest   ]"--=перевернутый вопросительный знак-->
<!ENTITY  laquo    SDATA  "[laquo    ]"--=угловые кавычки, левые-->
<!ENTITY  raquo    SDATA  "[raquo    ]"--= угловые кавычки, правые-->
<!ENTITY  lsquo    SDATA  "[lsquo    ]"--=одиночные кавычки, левые-->
<!ENTITY  rsquo    SDATA  "[rsquo    ]"--= одиночные кавычки, правые-->
<!ENTITY  ldquo    SDATA  "[ldquo    ]"--=двойные кавычки, левые-->
<!ENTITY  rdquo    SDATA  "[rdquo    ]"--= двойные кавычки, правые-->
<!ENTITY  nbsp     SDATA  "[nbsp     ]"--=неразрывный пробел-->
<!ENTITY  shy      SDATA  "[shy      ]"--=мягкий перенос-->
```

D.4.3.2 Символы диакритических знаков

Эти объекты рассматриваются как представления независимых символов.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для
     использования с
     удовлетворяющими SGML системами и приложениями, как
     определено в
     ИСО 8879, при условии, что это объявление будет включено во все
     копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOdia PUBLIC
       "ISO 8879-1986//ENTITIES Diacritical Marks//EN">
     %ISOdia;
-->
<!ENTITY  acute    SDATA  "[acute    ]"--=акут-->
<!ENTITY  краткая  SDATA  "[краткая  ]"--=краткая-->
<!ENTITY  карон    SDATA  "[карон    ]"--=карон-->
<!ENTITY  cedil    SDATA  "[cedil    ]"--=седиль-->
<!ENTITY  circ     SDATA  "[circ     ]"--=циркумфлекс-->
<!ENTITY  dblac    SDATA  "[dblac    ]"--=двойной акут-->
<!ENTITY  die      SDATA  "[die      ]"--=трема-->
<!ENTITY  dot      SDATA  "[dot      ]"--=точка сверху-->
<!ENTITY  grave    SDATA  "[grave    ]"--=грав-->
<!ENTITY  macr     SDATA  "[macr     ]"--=удлиненная-->
<!ENTITY  ogon     SDATA  "[ogon     ]"--=огонек-->
```

```
<!ENTITY  кольцо SDATA  "[кольцо  ]"--=кольцо-->
<!ENTITY  тильда SDATA  "[тильда  ]"--=тильда-->
<!ENTITY  uml    SDATA  "[uml    ]"--=знак умляут-->
```

D.4.3.3 Издательские символы

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для использования с
      удовлетворяющими SGML системами и приложениями, как определено в
      ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOpub PUBLIC
        "ISO 8879-1986//ENTITIES Publishing//EN">
      %ISOpub;
-->
<!ENTITY  emsp   SDATA  "[emsp   ]"--=длинный (эм) пробел-->
<!ENTITY  ensp   SDATA  "[ensp   ]"--=короткий (эн) пробел (1/2 длинного пробела)-->
<!ENTITY  emspl3 SDATA  "[emsp3  ]"--=l/3 длинного пробела-->
<!ENTITY  emspl4 SDATA  "[emsp4  ]"--=l/4 длинного пробела-->
<!ENTITY  numsp  SDATA  "[numsp  ]"--=цифровой пробел (ширина цифры)-->
<!ENTITY  puncsp SDATA  "[puncsp ]"--=пробел пунктуации (ширина запятой)-->
<!ENTITY  thinsp SDATA  "[thinsp ]"--=тонкий пробел (1/6 длинного пробела)-->
<!ENTITY  hairsp SDATA  "[hairsp ]"--=визирный пробел-->
<!ENTITY  mdash  SDATA  "[mdash  ]"--=длинное (эм) тире-->
<!ENTITY  ndash  SDATA  "[ndash  ]"--=короткое (эн) тире-->
<!ENTITY  dash   SDATA  "[dash   ]"--=дефис (истинно графический)-->
<!ENTITY  blank  SDATA  "[blank  ]"--=существенный пропуск-->
<!ENTITY  hellip SDATA  "[hellip ]"--=многоточие (горизонтальное)-->
<!ENTITY  nldr   SDATA  "[nldr   ]"--=сдвоенная точка шрифтовой линии (эн-пунктир )-->
<!ENTITY  fracl3 SDATA  "[fracl3 ]"--=дробь одна треть-->
<!ENTITY  frac23 SDATA  "[frac23 ]"--=дробь две трети-->
<!ENTITY  fracl5 SDATA  "[fracl5 ]"--=дробь одна пятая-->
<!ENTITY  frac25 SDATA  "[frac25 ]"--=дробь две пятых-->
<!ENTITY  frac35 SDATA  "[frac35 ]"--=дробь три пятых-->
<!ENTITY  frac45 SDATA  "[frac45 ]"--=дробь четыре пятых-->
<!ENTITY  fracl6 SDATA  "[frac16 ]"--=дробь одна шестая-->
<!ENTITY  frac56 SDATA  "[frac56 ]"--=дробь пять шестых-->
<!ENTITY  incare SDATA  "[incare ]"--=символ "на попечении"-->
```

```
<!ENTITY  block   SDATA  "[block    ]"--=полный блок-->
<!ENTITY  uhblk   SDATA  "[uhblk    ]"--=верхняя половина блока-->
<!ENTITY  lhblk   SDATA  "[lhblk    ]"--=нижняя половина блока-->
<!ENTITY  blkl4   SDATA  "[blkl4    ]"--=блок, заштрих. на 25%-->
<!ENTITY  blkl2   SDATA  "[blkl2    ]"--=блок, заштрих. на 50%-->
<!ENTITY  blk34   SDATA  "[blk34    ]"--=блок, заштрих. на 75%-->
<!ENTITY  marker  SDATA  "[marker   ]"--=гистограмма-->
<!ENTITY  cir     SDATA  "[cir      ]"--/circ B: =окружность, прозрачная-->
<!ENTITY  squ     SDATA  "[squ      ]"--=квдрат, прозрачный-->
<!ENTITY  rect    SDATA  "[rect     ]"--=прямоугольник, прозрачный-->
<!ENTITY  utri    SDATA  "[utri     ]"--/triangle =треугольник вверх, прозрачный-->
<!ENTITY  dtri    SDATA  "[dtri     ]"--/triangledown = треугольник вниз, прозрачный-->
<!ENTITY  star    SDATA  "[star     ]"--=звезда, прозрачная-->
<!ENTITY  bull    SDATA  "[bull     ]"--/bullet B: =круглый маркер, заштрих.-->
<!ENTITY  squf    SDATA  "[squf     ]"--/blacksquare =квадратный маркер, заштрих.-->
<!ENTITY  utrif   SDATA  "[utrif    ]"--/blacktriangle = треуг. вверх, заштрих.-->
<!ENTITY  dtrif   SDATA  "[dtrif    ]"--/blacktriangledown = треуг. вниз, заштрих. -->
<!ENTITY  ltrif   SDATA  "[ltrif    ]"--/blacktriangleleft R: = треуг. влево, заштрих. -->
<!ENTITY  rtrif   SDATA  "[rtrif    ]"--/blacktriangleright R: = треуг. вправо, заштрих. -->
<!ENTITY  clubs   SDATA  "[clubs    ]"--/clubsuit =карточный символ трефов-->
<!ENTITY  diams   SDATA  "[diams    ]"--/diamondsuit =карточный символ бубен-->
<!ENTITY  hearts  SDATA  "[hearts   ]"--/heartsuit =карточный символ червей-->
<!ENTITY  spades  SDATA  "[spades   ]"--/spadesuit =карточный символ пик-->
<!ENTITY  malt    SDATA  "[malt     ]"--/maltese =мальтийский крест-->
<!ENTITY  dagger  SDATA  "[dagger   ]"--/dagger B: =крестик-->
<!ENTITY  Dagger  SDATA  "[Dagger   ]"--/ddagger B: =двойной крестик-->
<!ENTITY  check   SDATA  "[check    ]"--/checkmark =черточка, знак отметки-->
<!ENTITY  cross   SDATA  "[ballot   ]"--=крест бюллетеня-->
<!ENTITY  sharp   SDATA  "[sharp    ]"--/sharp =музыкальный диез-->
<!ENTITY  flat    SDATA  "[flat     ]"--/flat =музыкальный бемоль-->
<!ENTITY  male    SDATA  "[male     ]"--=мужской символ-->
<!ENTITY  female  SDATA  "[female   ]"--=женский символ-->
<!ENTITY  phone   SDATA  "[phone    ]"--=символ телефона-->
<!ENTITY  telrec  SDATA  "[telrec   ]"--=символ телефонной записи -->
<!ENTITY  copysr  SDATA  "[copysr   ]"--=авторское право аудиозаписи-->
<!ENTITY  caret   SDATA  "[caret    ]"--=галочка (знак вставки)-->
<!ENTITY  lsquor  SDATA  "[lsquor   ]"--=поднимающ. одиночн. кавычки, левая (низкая)-->
<!ENTITY  ldquor  SDATA  "[ldquor   ]"--=поднимающ. двойные кавычки, левая (низкая)-->

<!ENTITY  fflig   SDATA  "[fflig    ]"--строчная лигатура ff-->
<!ENTITY  filig   SDATA  "[filig    ]"-- строчная лигатура fi-->
<!ENTITY  fjlig   SDATA  "[fjlig    ]"-- строчная лигатура fj-->
<!ENTITY  ffilig  SDATA  "[ffilig   ]"-- строчная лигатура ffi-->
<!ENTITY  ffllig  SDATA  "[ffllig   ]"-- строчная лигатура ffl-->
<!ENTITY  fllig   SDATA  "[fllig    ]"-- строчная лигатура fl-->

<!ENTITY  mldr    SDATA  "[mldr     ]"--эм пунктир-->
<!ENTITY  rdquor  SDATA  "[rdquor   ]"--поднимающ. двойные кавычки, правая (высокая)-->
<!ENTITY  rsquor  SDATA  "[rsquor   ]"--поднимающ. одиночн. кавычки, правая (высокая)-->
<!ENTITY  vellip  SDATA  "[vellip   ]"--вертикальное многоточие-->
```

```
<!ENTITY  hybull   SDATA  "[hybull    ]"--прямоугольник, заштрих. (дефисный маркер)-->
<!ENTITY  loz      SDATA  "[loz       ]"--/lozenge – ромб или знак итого-->
<!ENTITY  lozf     SDATA  "[lozf      ]"--/blacklozenge - ромб, заштрих.-->
<!ENTITY  ltri     SDATA  "[ltri      ]"--/triangleleft B: треугольник влево, прозрачный-->
<!ENTITY  rtri     SDATA  "[rtri      ]"--/triangleright B: треугольник вправо, прозрачный-->
<!ENTITY  starf    SDATA  "[starf     ]"--/bigstar - звезда, заштрих.-->

<!ENTITY  natur    SDATA  "[natur     ]"--/natural – музыка естественная-->
<!ENTITY  rx       SDATA  "[rx        ]"--лекарственный рецепт(Rx)-->
<!ENTITY  sext     SDATA  "[sext      ]"--секстиль(6-конечная звезда)-->

<!ENTITY  target   SDATA  "[target    ]"--зарегистрированный знак или мира-->
<!ENTITY  dicrop   SDATA  "[dicrop    ]"--нижняя левая ограничительная метка-- >
<!ENTITY  drcrop   SDATA  "[drcrop    ]"--нижняя правая ограничительная метка-->
<!ENTITY  ulcrop   SDATA  "[ulcrop    ]"-- верхняя левая ограничительная метка-->
<!ENTITY  urcrop   SDATA  "[urcrop    ]"--верхняя правая ограничительная метка-->
```

D.4.3.4 Символы рамок и заполнения

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для использования с
      удовлетворяющими SGML системами и приложениями, как определено в
      ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISObox PUBLIC
        "ISO 8879-1986//ENTITIES Box and Line Drawing//EN">
      %ISObox;
<!--  Все имена приведены в форме: boxl234, где:
      box = константа, которая идентифицирует объект рисования прямоугольник.
      1&2 = v, V, u, U, d, D, Ud или uD следующим образом:
      v   = вертикальная линия полной высоты.
      u   = верхняя половина вертикальной линии.
```

d = нижняя половина вертикальной линии.

3&4 = h, H, l, L, r, R, Lr или lR следующим образом:

h = горизонтальная линия полной длины.

l = левая половина горизонтальной линии.

r = правая половина горизонтальной линии.

Во всех случаях прописные буквы обозначают двойную или жирную линию.

-->

```
<!ENTITY  boxh    SDATA  "[boxh    ]"--горизонтальная линия-->
<!ENTITY  boxv    SDATA  "[boxv    ]"--вертикальная линия-->
<!ENTITY  boxur   SDATA  "[boxur   ]"--верхний правый квадрант-->
<!ENTITY  boxul   SDATA  "[boxul   ]"--верхний левый квадрант-->
<!ENTITY  boxdl   SDATA  "[boxdl   ]"--нижний левый квадрант-->
<!ENTITY  boxdr   SDATA  "[boxdr   ]"--нижний правый квадрант-->
<!ENTITY  boxvr   SDATA  "[boxvr   ]"--верхний и нижний правые квадранты-->
<!ENTITY  boxhu   SDATA  "[boxhu   ]"--верхние левый и правый квадранты-->
<!ENTITY  boxvl   SDATA  "[boxvl   ]"--верхний и нижний левые квадранты-->
<!ENTITY  boxhd   SDATA  "[boxhd   ]"--нижние левый и правый квадранты-->
<!ENTITY  boxvh   SDATA  "[boxvh   ]"--все четыре квадранта-->
<!ENTITY  boxvR   SDATA  "[boxvR   ]"--верхний и нижний правые квадранты-->
<!ENTITY  boxhU   SDATA  "[boxhU   ]"--верхние левый и правый квадранты-->
<!ENTITY  boxvL   SDATA  "[boxvL   ]"--верхний и нижний левые квадранты-->
<!ENTITY  boxhD   SDATA  "[boxhD   ]"--нижние левый и правый квадранты-->
<!ENTITY  boxvH   SDATA  "[boxvH   ]"-- все четыре квадранта-->
<!ENTITY  boxH    SDATA  "[boxH    ]"--горизонтальная линия-->
<!ENTITY  boxV    SDATA  "[boxV    ]"--вертикальная линия-->
<!ENTITY  boxUR   SDATA  "[boxUR   ]"--верхний правый квадрант-->
<!ENTITY  boxUL   SDATA  "[boxUL   ]"--верхний левый квадрант-->
<!ENTITY  boxDL   SDATA  "[boxDL   ]"--нижний левый квадрант-->
<!ENTITY  boxDR   SDATA  "[boxDR   ]"--нижний правый квадрант-->
<!ENTITY  boxVR   SDATA  "[boxVR   ]"--верхний и нижний правые квадранты-->
<!ENTITY  boxHU   SDATA  "[boxHU   ]"--верхние левый и правый квадранты-->
<!ENTITY  boxVL   SDATA  "[boxVL   ]"--верхний и нижний левые квадранты-->
<!ENTITY  boxHD   SDATA  "[boxHD   ]"--нижние левый и правый квадранты-->
<!ENTITY  boxVH   SDATA  "[boxVH   ]"--все четыре квадранта-->
<!ENTITY  boxVr   SDATA  "[boxVr   ]"--верхний и нижний правые квадранты-->
<!ENTITY  boxHu   SDATA  "[boxHu   ]"--верхние левый и правый квадранты-->
<!ENTITY  boxVl   SDATA  "[boxVl   ]"--верхний и нижний левые квадранты-->
<!ENTITY  boxHd   SDATA  "[boxHd   ]"--нижние левый и правый квадранты-->
<!ENTITY  boxVh   SDATA  "[boxVh   ]"--все четыре квадранта-->
<!ENTITY  boxuR   SDATA  "[boxuR   ]"--верхний правый квадрант-->
<!ENTITY  boxUl   SDATA  "[boxUl   ]"--верхний левый квадрант-->
<!ENTITY  boxdL   SDATA  "[boxdL   ]"--нижний левый квадрант-->
<!ENTITY  boxDr   SDATA  "[boxDr   ]"--нижний правый квадрант-->
```

```
<!ENTITY  boxUr    SDATA  "[boxUr    ]"--верхний правый квадрант-->
<!ENTITY  boxuL    SDATA  "[boxuL    ]"--верхний левый квадрант-->
<!ENTITY  boxDl    SDATA  "[boxDl    ]"--нижний левый квадрант-->
<!ENTITY  boxdR    SDATA  "[boxdR    ]"--нижний правый квадрант-->
```

D.4.4 Техническое употребление

Поскольку многие технические символы могут использоваться в разных контекстах, имена объектов этой категории обычно описывают графическое изображение визуально, нежели пытаясь передать связанную с ним семантическую концепцию.

Приведенные ниже сокращения используются с существенным постоянством :

Префиксы:

l=левый; r=правый; u=вверх; d=вниз; h=горизонтальный; v=вертикальный

b=назад, обратный

cu=изогнуто

g=больше чем; l=меньше чем;

n=отрицательный;

o=в окружности

s=малый, краткий;

sq=квадратный

thk=толстый;

x=расширенный, длинный, большой;

Корни:

ap=приблизительно;

arr=стрела; ha=гарпун

p=предшествует; sc=следует

sub=подмножество; sup=импликация

Суффиксы:

b=в квадрате;

f=заштрихован, черный, сплошной

e=одиночное равенство; E=двойное равенство;

hk=крюк;

s=наклон;

t=оперение;

w=волнистый;

2=два из

Прописная буква означает "сдвоенный" (или иногда "два из")

ПРИМЕЧАНИЕ. Визуальные изображения большинства объектов технического употребления идентифицируются по именам объектов в Association of American Publishers Electronic Manuscript Series: Markup of Mathematical Formulas, published by the Association of American Publishers, Inc., 2005 Massachusetts Avenue, N.W., Washington, DC 20036, U.S.A.

D.4.4.1 Общие объекты

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOtech PUBLIC
       "ISO 8879-1986//ENTITIES General Technical //EN">
     %ISOtech;
-->
```

```
<!ENTITY  aleph   SDATA  "[aleph   ]"--/aleph =алеф, иврит-->
<!ENTITY  and     SDATA  "[and     ]"--/wedge /land B: =логическое и-->
<!ENTITY  ang90   SDATA  "[ang90   ]"--=прямой угол (90 градусов)-->
<!ENTITY  angsph  SDATA  "[angsph  ]"--/sphericalangle =сферический угол-->
<!ENTITY  ap      SDATA  "[ap      ]"--/approx R: =приблизительность-->
<!ENTITY  becaus  SDATA  "[becaus  ]"--/because R: =потому что-->
<!ENTITY  bottom  SDATA  "[bottom  ]"--/bot B: =перпендикулярность-->
<!ENTITY  cap     SDATA  "[cap     ]"--/cap B: пересечение-->
<!ENTITY  cong    SDATA  "[cong    ]"--/cong R: =конгруэнтно с-->
<!ENTITY  conint  SDATA  "[conint  ]"--/oint L: =оператор интеграла по контуру-->
<!ENTITY  cup     SDATA  "[cup     ]"--/cup B: =объединение или логическое сложение-->
<!ENTITY  equiv   SDATA  "[equiv   ]"--/equiv R: =тождественность-->
<!ENTITY  exist   SDATA  "[exist   ]"--/exists = существует по крайней мере один-->
<!ENTITY  forall  SDATA  "[forall  ]"--/forall =для всех-->
<!ENTITY  fnof    SDATA  "[fnof    ]"--=функция (строчная f курсив)-->
<!ENTITY  ge      SDATA  "[ge      ]"--/geq /ge R: =больше чем или равно-->
<!ENTITY  iff     SDATA  "[iff     ]"--/iff =если и только если-->
<!ENTITY  infin   SDATA  "[infin   ]"--/infty =бесконечность-->
<!ENTITY  int     SDATA  "[int     ]"--/int L: =оператор интеграла-->
<!ENTITY  isin    SDATA  "[isin    ]"--/in R: =принадлежность множеству-->
<!ENTITY  lang    SDATA  "[lang    ]"--/langle 0: =левая угловая скобка-->
<!ENTITY  lArr    SDATA  "[lArr    ]"--/Leftarrow A: =следует из-->
<!ENTITY  le      SDATA  "[le      ]"--/leq /le R: =меньше чем или равно-->
<!ENTITY  minus   SDATA  "[minus   ]"--B: =знак минус-->
<!ENTITY  mnplus  SDATA  "[mnplus  ]"--/mp B: =знак минус или плюс-->
<!ENTITY  nabla   SDATA  "[nabla   ]"--/nabla =оператор Гамильтона-->
<!ENTITY  ne      SDATA  "[ne      ]"--/ne /neq R: =не равно-->
<!ENTITY  ni      SDATA  "[ni      ]"--/ni /owns R: =содержит-->
<!ENTITY  or      SDATA  "[or      ]"--/vee /lor B: =логическое или-->
<!ENTITY  par     SDATA  "[Par     ]"--/parallel R: =параллельность-->
<!ENTITY  part    SDATA  "[part    ]"--/partial =частный дифференциал-->
<!ENTITY  permil  SDATA  "[permil  ]"--=на тысячу-->
<!ENTITY  perp    SDATA  "[perp    ]"--/perp R: =перпендикулярность-->
<!ENTITY  prime   SDATA  "[prime   ]"--/prime =штрих или минута-->
<!ENTITY  Prime   SDATA  "[Prime   ]"--=двойной штрих или секунда-->
<!ENTITY  prop    SDATA  "[Prop    ]"--/propto R: =пропорциональность-->
<!ENTITY  radic   SDATA  "[radic   ]"--/surd =корень-->
<!ENTITY  rang    SDATA  "[rang    ]"--/rangle C: =правая угловая скобка-->
<!ENTITY  rArr    SDATA  "[rArr    ]"--/Rightarrow A: =влечет-->
<!ENTITY  sim     SDATA  [sim      ]"--/sim R: =подобие-->
<!ENTITY  sime    SDATA  [sime     ]"--/simeq R: =подобный, равенство-->
<!ENTITY  square  SDATA  "[square  ]"--/square B: =квадрат-->"
<!ENTITY  sub     SDATA  "[sub     ]"--/subset R: =подмножество или следует из-->
<!ENTITY  sube    SDATA  [sue      ]"--/subseteq R: = подмножество, равенство-->
<!ENTITY  sup     SDATA  "[sup     ]"--/supset R: =импликация или влечет-->
<!ENTITY  supe    SDATA  "[supe    ]"--/supseteq R: =импликация, равенство-->
<!ENTITY  there4  SDATA  "[there4  ]"--/therefore R: =следовательно-->
<!ENTITY  Verbar  SDATA  "[Verbar  ]"--/Vert =двойная вертикальная линия-->
<!ENTITY  angst   SDATA  "[angst   ]"--ангстрем=прописная A, кольцо-->
<!ENTITY  bernou  SDATA  "[bernou  ]"--функция Бернулли (прописная B скрипт)-->
```

```
<!ENTITY  compfn  SDATA  "[compfn   ]"--B: композиция (малая окружность)-->
<!ENTITY  Dot     SDATA  "[Dot      ]"--=трема или знак умляут-->
<!ENTITY  DotDot  SDATA  "[DotDot   ]"--четыре точки сверху-->
<!ENTITY  hamilt  SDATA  "[hamilt   ]"--Гамильтониан (прописная рукописная H)-->
<!ENTITY  lagran  SDATA  "[lagran   ]"--Лагранжиан (прописная рукописная L)-->
<!ENTITY  lowast  SDATA  "[lowast   ]"--низкая звездочка-->
<!ENTITY  notin   SDATA  "[notin"   ]"--N: не принадлежность множеству-->
<!ENTITY  order   SDATA  "[order    ]"--порядок (строчная рукописная o)-->
<!ENTITY  phmmat  SDATA  "[phmmat   ]"--физическая матрица M (прописная рукописная M)-->
<!ENTITY  tdot    SDATA  "[tdot     ]"--три точки сверху-->
<!ENTITY  tprime  SDATA  "[tprime   ]"--тройной штрих-->
<!ENTITY  wedgeq  SDATA  "[wedgeq   ]"--R: соответствует (равенство)-->
```

D.4.4.2 Греческие символы

Этот набор объектов определяет имена греческих символов для использования в качестве названий переменных в технических приложениях.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOgrk3 PUBLIC
       "ISO 8879-1986//ENTITIES Greek Symbols//EN">
     %ISOgrk3;
-->
<!ENTITY  alpha   SDATA  "[alpha    ]"--=строчная альфа, греческая-->
<!ENTITY  beta    SDATA  "[beta     ]"--=строчная бета, греческая-->
<!ENTITY  gamma   SDATA  "[gamma    ]"--=строчная гамма, греческая-->
<!ENTITY  Gamma   SDATA  "[Gamma    ]"--=прописная Гамма, греческая-->
<!ENTITY  gammad  SDATA  "[gammad   ]"--/digamma-->
<!ENTITY  delta   SDATA  "[delta    ]"--=строчная дельта, греческая-->
<!ENTITY  Delta   SDATA  "[Delta    ]"--=прописная Дельта, греческая-->
```

```
<!ENTITY  epsi     SDATA  "[epsi     ]"--=строчная эпсилон, греческая-->
<!ENTITY  epsiv    SDATA  "[epsiv    ]"--/varepsilon-->
<!ENTITY  epsis    SDATA  "[epsis    ]"--/straightepsilon-->
<!ENTITY  zeta     SDATA  "[zeta     ]"--=строчная зета, греческая-->
<!ENTITY  eta      SDATA  "[eta      ]"--=строчная эта, греческая-->
<!ENTITY  thetas   SDATA  "[thetas   ]"--/straight theta-->
<!ENTITY  Theta    SDATA  "[Theta    ]"--=прописная Тета, греческая-->
<!ENTITY  thetav   SDATA  "[thetav   ]"--/vartheta - изогнутая или прозрачная тета-->
<!ENTITY  iota     SDATA  "[iota     ]"--=строчная йота, греческая-->
<!ENTITY  kappa    SDATA  "[kappa    ]"--=строчная каппа, греческая-->
<!ENTITY  kappav   SDATA  "[kappav   ]"--/varkappa-->
<!ENTITY  lambda   SDATA  "[lambda   ]"--=строчная лямбда, греческая-->
<!ENTITY  Lambda   SDATA  "[Lambda   ]"--=прописная Лямбда, греческая-->
<!ENTITY  mu       SDATA  "[mu       ]"--=строчная мю, греческая-->
<!ENTITY  nu       SDATA  "[nu       ]"--=строчная ню, греческая-->
<!ENTITY  xi       SDATA  "[xi       ]"--=строчная кси, греческая-->
<!ENTITY  Xi       SDATA  "[Xi       ]"--=прописная Кси, греческая-->
<!ENTITY  Pi       SDATA  "[pi       ]"--=строчная пи, греческая-->
<!ENTITY  piv      SDATA  ″[piv      ]"--/varpi-->
<!ENTITY  Pi       SDATA  "[Pi       ]"--=прописная Пи, греческая-->
<!ENTITY  rho      SDATA  "[rho      ]"--=строчная ро, греческая-->
<!ENTITY  rhov     SDATA  "[rhov     ]"--/varrho-->
<!ENTITY  sigma    SDATA  "[sigma    ]"--=строчная сигма, греческая-->
<!ENTITY  Sigma    SDATA  "[Sigma    ]"--=прописная Сигма, греческая-->
<!ENTITY  sigmav   SDATA  "[sigmav   ]"--/varsigma-->
<!ENTITY  tau      SDATA  "[tau      ]"--=строчная тау, греческая-->
<!ENTITY  upsi     SDATA  "[upsi     ]"--=строчная юпсилон, греческая-->
<!ENTITY  Upsi     SDATA  "[Upsi     ]"--=прописная Юпсилон, греческая-->
<!ENTITY  phis     SDATA  "[phis     ]"--/straightphi - прямая пи-->
<!ENTITY  Phi      SDATA  "[Phi      ]"--=прописная Фи, греческая-->
<!ENTITY  phiv     SDATA  "[phiv     ]"--/varphi - изогнутая или открытая фи-->
<!ENTITY  chi      SDATA  "[chi      ]"--=строчная хи, греческая-->
<!ENTITY  psi      SDATA  "[psi      ]"--=строчная пси, греческая-->
<!ENTITY  Psi      SDATA  "[Psi      ]"--=прописная Пси, греческая-->
<!ENTITY  Omega    SDATA  "[Omega    ]"--=прописная Омега, греческая-->
<!ENTITY  omega    SDATA  "[omega    ]"--=строчная омега, греческая-->
```

D.4.4.3 Альтернативные греческие символы

Символы этого набора объектов могут использоваться вместе с предшествующими, когда необходим отдельный класс переменных. По соглашению они отображаются другим шрифтом или стилем (обычно утолщенным).

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
```

удовлетворяющими SGML системами и приложениями, как определено в
ИСО 8879, при условии, что это объявление будет включено во все копии.

-->

<!-- Набор символьных объектов. Типичный вызов:

<!ENTITY % ISOgrk4 PUBLIC

"ISO 8879-1986//ENTITIES Alternative Greek Symbols//EN">

%ISOgrk4;

-->

```
<!ENTITY  b.alpha     SDATA  "[b.alpha     ]"--=строчная альфа, греческая-->
<!ENTITY  b.beta      SDATA  "[b.beta      ]"--=строчная бета, греческая-->
<!ENTITY  b.gamma     SDATA  "[b.gamma     ]"--=строчная гамма, греческая-->
<!ENTITY  b.Gamma     SDATA  "[b.Gamma     ]"--=прописная Гамма, греческая-->
<!ENTITY  b.gammad    SDATA  "[b.gammad    ]"--/digamma-->
<!ENTITY  b.delta     SDATA  "[b.delta     ]"--=строчная дельта, греческая-->
<!ENTITY  b.Delta     SDATA  "[b.Delta     ]"--=прописная Дельта, греческая-->
<!ENTITY  b.epsi      SDATA  "[b.epsi      ]"--=строчная эпсилон, греческая-->
<!ENTITY  b. epsiv    SDATA  "[b.epsiv     ]"--/varepsilon-->
<!ENTITY  b. epsis    SDATA  "[b.epsis     ]"--/straightepsilon-->
<!ENTITY  b.zeta      SDATA  "[b.zeta      ]"--=строчная зета, греческая-->
<!ENTITY  b. eta      SDATA  "[b.eta       ]"--=строчная эта, греческая-->
<!ENTITY  b.thetas    SDATA  "[b.thetas    ]"--прямая тета-->
<!ENTITY  b.Theta     SDATA  "[b.Theta     ]"--=прописная Тета, греческая-->
<!ENTITY  b.thetav    SDATA  "[b.thetav    ]"--/vartheta - изогнутая или открытая тета-->
<!ENTITY  b.iota      SDATA  "[b.iota      ]"--=строчная йота, греческая-->
<!ENTITY  b.kappa     SDATA  "[b.kappa     ]"--=строчная каппа, греческая-->
<!ENTITY  b.kappav    SDATA  "[b.kappav    ]"--/varkappa-->
<!ENTITY  b.lambda    SDATA  "[b.lambda    ]"--=строчная лямбда, греческая-->
<!ENTITY  b.Lambda    SDATA  "[b.Lambda    ]"--=прописная Лямбда, греческая-->
<!ENTITY  b.mu        SDATA  "[b.mu        ]"--=строчная мю, греческая-->
<!ENTITY  b.nu        SDATA  "[b.nu        ]"--=строчная ню, греческая-->
<!ENTITY  b.xi        SDATA  "[b.xi        ]"--=строчная кси, греческая-->
<!ENTITY  b.Xi        SDATA  "[b.Xi        ]"--=прописная Кси, греческая-->
<!ENTITY  b.pi        SDATA  "[b.pi        ]"--=строчная пи, греческая-->
<!ENTITY  b.Pi        SDATA  "[b.Pi        ]"--=прописная Пи, греческая-->
<!ENTITY  b.piv       SDATA  "[b.piv       ]"--/varpi-->
<!ENTITY  b. rho      SDATA  "[b.rho       ]"--=строчная ро, греческая-->
<!ENTITY  b.rhov      SDATA  "[b.rhov      ]"--/varrho-->
<!ENTITY  b.sigma     SDATA  "[b.sigma     ]"--=строчная сигма, греческая-->
<!ENTITY  b.Sigma     SDATA  "[b.Sigma     ]"--=прописная Сигма, греческая-->
<!ENTITY  b.s igmav   SDATA  "[b.sigmav    ]"--/varsigma-->
<!ENTITY  b. tau      SDATA  "[b.tau       ]"--=строчная тау, греческая-->
```

```
<!ENTITY  b.ups i    SDATA  "[b.upsi      ]"--=строчная юпсилон, греческая-->
<!ENTITY  b.Upsi     SDATA  "[b.Upsi      ]"--=прописная Юпсилон, греческая-->
<!ENTITY  b.phis     SDATA  "[b.phis      ]"--/straightphi - прямая пи-->
<!ENTITY  b.Phi      SDATA  "[b.Phi       ]"--=прописная Фи, греческая-->
<!ENTITY  b.phiv     SDATA  "[b.phiv      ]"--/varphi - изогнутая или открытая фи-->
<!ENTITY  b.chi      SDATA  "[b.chi       ]"--=строчная хи, греческая-->
<!ENTITY  b.psi      SDATA  "[b.psi       ]"--=строчная пси, греческая-->
<!ENTITY  b.Psi      SDATA  "[b.Psi       ]"--=прописная Пси, греческая-->
<!ENTITY  b.omega    SDATA  "[b.omega     ]"--=строчная омега, греческая-->
<!ENTITY  b.Omega    SDATA  "[b.Omega     ]"--=прописная Омега, греческая-->
```

D.4.5 Дополнительные математические символы

D.4.5.1 Одиночные символы

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для использования с
      удовлетворяющими SGML системами и приложениями, как определено в
      ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOamso PUBLIC
        "ISO 8879-1986//ENTITIES Added Math Symbols: Ordinary//EN">
      %ISOamso;
-->
<!ENTITY  ang        SDATA  "[ang         ]"--/angle - угол-->
<!ENTITY  angmsd     SDATA  "[angmsd      ]"--/measuredangle - измеренный угол-->
<!ENTITY  beth       SDATA  "[both        ]"--/beth - бет, иврит-->
<!ENTITY  bprime     SDATA  "[bprime      ]"--/backprime - обратный штрих-->
<!ENTITY  comp       SDATA  "[comp        ]"--/complement - знак дополнения-->
<!ENTITY  daleth     SDATA  "[daleth      ]"--/daleth - далет, иврит-->
<!ENTITY  ell        SDATA  "[ell         ]"--/ell - строчный l курсив-->
<!ENTITY  empty      SDATA  "[empty       ]"--/emptyset /varnothing =строчная o, перечеркнутая-->
<!ENTITY  gimel      SDATA  "[gimel       ]"--/gimel - гимел, иврит-->
<!ENTITY  image      SDATA  "[image       ]"--/Im - мнимый-->
<!ENTITY  inodot     SDATA  "[inodot      ]"--/imath =строчная i, без точки-->
<!ENTITY  jnodot     SDATA  "[jnodot      ]"--/jmath - строчная j, без точки-->
<!ENTITY  nexist     SDATA  "[nexist      ]"--/nexists - не существует-->
<!ENTITY  oS         SDATA  "[oS          ]"--/circledS - прописная S в окружности-->
```

```
<!ENTITY  planck  SDATA  "[planck     ]"--/hbar /hslash - постоянная Планка-->
<!ENTITY  real    SDATA  "[real      ]"--/Re - вещественный-->
<!ENTITY  sbsol   SDATA  "[sbsol     ]"--/sbs - короткая обратная косая черта-->
<!ENTITY  vprime  SDATA  "[vprime    ]"--/varprime - штрих, иной-->
<!ENTITY  weierp  SDATA  "[weierp    ]"--/wp - функция Вейерштрасса-->
```

D.4.5.2 Двоичные и большие операторы

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
      удовлетворяющими SGML системами и приложениями, как
      определено в
      ИСО 8879, при условии, что это объявление будет включено во все
      копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOamsb PUBLIC
        "ISO 8879-1986//ENTITIES Added Math Symbols: Binary
        Operators//EN">
      %ISOamsb;
-->
<!ENTITY  amalg   SDATA  "[amalg     ]"--/amalg B: amalgamation or coproduct-->
<!ENTITY  Barwed  SDATA  "[Barwed    ]"--/doublebarwedge B: логическое и, двойная черта-->
<!ENTITY  barwed  SDATA  "[barwed    ]"--/barwedge B: логическое и, черта сверху-->
<!ENTITY  Cap     SDATA  "[Cap       ]"--/Cap /doublecap B: двойное пересечение-->
<!ENTITY  Cup     SDATA  "[Cup       ]"--/Cup /doublecup B: двойное объединение-->
<!ENTITY  cuvee   SDATA  "[cuvee     ]"--/curlyvee B: изогнутое логическое или-->
<!ENTITY  cuwed   SDATA  "[cuwed     ]"--/curlywedge B: изогнутое логическое и-->
<!ENTITY  diam    SDATA  "[diam      ]"--/diamond B: прозрачный ромб-->
<!ENTITY  divonx  SDATA  "[divonx    ]"--/divideontimes B: деление на умножении-->
<!ENTITY  intcal  SDATA  "[intcal    ]"--/intercal B: intercal-->
<!ENTITY  lthree  SDATA  "[lthree    ]"--/leftthreetimes B:-->
<!ENTITY  ltimes  SDATA  "[ltimes    ]"--/Itimes B: знак умножения, закрыт слева-->
<!ENTITY  minusb  SDATA  "[minusb    ]"--/boxminus B: знак минус в квадрате-->
<!ENTITY  oast    SDATA  "[oast      ]"--/circledast B: звездочка в окружности-->
<!ENTITY  ocir    SDATA  "[ocir      ]"--/circledcirc B: прозрачная точка в окружности-->
<!ENTITY  odash   SDATA  "[odash     ]"--/circleddash B: дефис в окружности-->
<!ENTITY  odot    SDATA  "[odot      ]"--/odot B: центральная точка в окружности-->
<!ENTITY  ominus  SDATA  "[ominus    ]"--/ominus B: знак минус в окружности-->
```

```
<!ENTITY  opius   SDATA  "[opius    ]"--/opius B: знак плюс в окружности-->
<!ENTITY  osol    SDATA  "[osol     ]"--/oslash B: косая черта в окружности-->
<!ENTITY  otimes  SDATA  "[otimes   ]"--/otimes B: знак умножения в окружности-->
<!ENTITY  plusb   SDATA  "[plusb    ]"--/boxplus B: знак плюс в квадрате-->
<!ENTITY  plusdo  SDATA  "[plusdo   ]"--/dotplus B: знак плюс, точка сверху-->
<!ENTITY  rthree  SDATA  "[rthree   ]"--/rightthreetimes B:-->
<!ENTITY  rtimes  SDATA  "[rtimes   ]"--/rtimes B: знак умножения, закрыт справа-->
<!ENTITY  sdot    SDATA  "[sdot     ]"--/cdot B: малая центральная точка-->
<!ENTITY  sdotb   SDATA  "[sdotb    ]"--/dotsquare /boxdot B: малая точка в квадрате-->
<!ENTITY  setmn   SDATA  "[setmn    ]"--/setminus B: обратная косая черта-->
<!ENTITY  sqcap   SDATA  "[sqcap    ]"--/sqcap B: квадратное пересечение-->
<!ENTITY  sqcup   SDATA  "[sqcup    ]"--/sqcup B: квадратное объединение-->
<!ENTITY  ssetmn  SDATA  "[ssetmn   ]"--/smallsetminus B: малая обратная косая черта-->
<!ENTITY  sstarf  SDATA  "[sstarf   ]"--/star B: малая звезда, заштрихованная-->
<!ENTITY  timesb  SDATA  "[timesb   ]"--/boxtimes B: знак умножения в квадрате-->
<!ENTITY  top     SDATA  "[top      ]"--/top B: перевернутый перпендикуляр-->
<!ENTITY  uplus   SDATA  "[uplus    ]"--/uplus B: знак плюс в объединении-->
<!ENTITY  wreath  SDATA  "[wreath   ]"--/wr B: wreath product-->
<!ENTITY  xcirc   SDATA  "[xcirc    ]"--/bigcirc B: большая окружность-->
<!ENTITY  xdtri   SDATA  "[xdtri    ]"--/bigtriangledown B: большой треуг. вниз, прозр.-->
<!ENTITY  xutri   SDATA  "[xutri    ]"--/bigtriangleup B: большой треуг. вверх, прозр.-->
<!ENTITY  coprod  SDATA  "[coprod   ]"--/coprod L: coproduct operator-->
<!ENTITY  prod    SDATA  "[prod     ]"--/prod L: оператор произведения-->
<!ENTITY  sum     SDATA  "[sum      ]"--/sum L: оператор суммирования-->
```

D.4.5.3 Отношения

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
      удовлетворяющими SGML системами и приложениями, как
      определено в
      ИСО 8879, при условии, что это объявление будет включено во все
      копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOamsr PUBLIC
        "ISO 8879-1986//ENTITIES Added Math Symbols: Relations//EN">
      %ISOamsr;
-->
```

```
<!ENTITY  ape     SDATA  "[ape     ]"--/approxeq R: приближенно, равенство-->
<!ENTITY  asymp   SDATA  "[asymp   ]"--/asymp R: асимптотически равен-->
<!ENTITY  bcong   SDATA  "[bcong   ]"--/backcong R: обратная конгруэнтность-->
<!ENTITY  bepsi   SDATA  "[bepsi   ]"--/backepsilon R: так что-->
<!ENTITY  bowtie  SDATA  "[bowtie  ]"--/bowtie R:-->
<!ENTITY  bsim    SDATA  "[bsim    ]"--/backsim R: обратное подобие-->
<!ENTITY  bsime   SDATA  "[bsime   ]"--/backsimeq R: обратное подобие, равенсто-->
<!ENTITY  bump    SDATA  "[bump    ]"--/Bumpeq R: выпуклый знак равенства-->
<!ENTITY  bumpe   SDATA  "[bumpe   ]"--/bumpeq R: выпуклый знак равенства, равенство-->
<!ENTITY  cire    SDATA  "[cire    ]"--/circeq R: окружность, равенство-->
<!ENTITY  colone  SDATA  "[colone  ]"--/coloneq R: двоеточие, равенство-->
<!ENTITY  cuepr   SDATA  "[cuepr   ]"--/curlyeqprec R: изогнутое равенство, предшествует-->
<!ENTITY  cuesc   SDATA  "[cuesc   ]"--/curlyeqsucc R: изогнутое равенство, следует-->
<!ENTITY  cupre   SDATA  "[cupre   ]"--/curlypreceq R: изогнутое предшествует, равенство-->
<!ENTITY  dashv   SDATA  "[dashv   ]"--/dashv R: тире, вертикальное-->
<!ENTITY  ecir    SDATA  "[reci    ]"--/eqcirc R: окружность на знаке равенства-->
<!ENTITY  ecolon  SDATA  "[ecolon  ]"--/eqcolon R: равенство, двоеточие-->
<!ENTITY  eDot    SDATA  "[eDot    ]"--/doteqdot /Doteq R: равенство, симметр. точки-->
<!ENTITY  esdot   SDATA  "[esdot   ]"--/doteq R: равенство, одна точка сверху-->
<!ENTITY  efDot   SDATA  "[efDot   ]"--/fallingdotseq R: равенство, падающие точки-->
<!ENTITY  egs     SDATA  "[egs     ]"--/eqslantgtr R: равно-или-больше, с наклоном-->
<!ENTITY  els     SDATA  "[els     ]"--/eqslantless R: равно-или-меньше, с наклоном-->
<!ENTITY  erDot   SDATA  "[erDot   ]"--/risingdotseq R: равенство, поднимающиеся точки-->
<!ENTITY  fork    SDATA  "[fork    ]"--/pitchfork R: вилка, направленная вниз-->
<!ENTITY  frown   SDATA  "[frown   ]"--/frown R: дуга, направленная вниз-->
<!ENTITY  gap     SDATA  "[gap     ]"--/gtrapprox R: больше, приближенно-->
<!ENTITY  gsdot   SDATA  "[gsdot   ]"--/gtrdot R: больше чем, одна точка-->
<!ENTITY  gE      SDATA  "[gE      ]"--/geqq R: больше, двойное равенство-->
<!ENTITY  gel     SDATA  "[gel     ]"--/gtreqless R: больше, равенство, меньше-->
<!ENTITY  gEl     SDATA  "[gEl     ]"--/gtreqqless R: больше, двойное равенство, меньше-->
<!ENTITY  ges     SDATA  "[ges     ]"--/geqslant R: больше-или-меньше, с наклоном-->
<!ENTITY  Gg      SDATA  "[Gg      ]"--/ggg /Gg /gggtr R: тройное больше чем-->
<!ENTITY  gl      SDATA  "[gl      ]"--/gtrless R: больше, меньше-->
<!ENTITY  gsim    SDATA  "[gsim    ]"--/gtrsim R: больше, подобие-->
<!ENTITY  Gt      SDATA  "[Gt      ]"--/gg R: двойной знак больше чем-->
<!ENTITY  lap     SDATA  "[lap     ]"--/lessapprox R: меньше, приближенно-->
<!ENTITY  ldot    SDATA  "[ldot    ]"--/lessdot R: меньше чем, с точкой-->
<!ENTITY  lE      SDATA  "[lE      ]"--/leqq R: меньше, двойное равенство-->
<!ENTITY  lEg     SDATA  "[lEg     ]"--/lesseqqgtr R: меньше, двойное равенство, больше-->
<!ENTITY  leg     SDATA  "[leg     ]"--/lesseqgtr R: меньше, равенство, больше-->
<!ENTITY  les     SDATA  "[les     ]"--/leqslant R: меньше-чем-или-равно, с наклоном-->
<!ENTITY  lg      SDATA  "[lg      ]"--/lessgtr R: меньше, больше-->
<!ENTITY  Ll      SDATA  "[Ll      ]"--/Ll /lll /llless R: тройное меньше чем-->
<!ENTITY  lsim    SDATA  "[lsim    ]"--/lesssim R: меньше, подобие-->
<!ENTITY  Lt      SDATA  "[Lt      ]"--/ll R: двойной знак меньше чем-->
<!ENTITY  ltrie   SDATA  "[ltrie   ]"--/trianglelefteq R: левый треугольник, равенство-->
<!ENTITY  mid     SDATA  "[mid     ]"--/mid R:-->
<!ENTITY  models  SDATA  "[models  ]"--/models R:-->
<!ENTITY  pr      SDATA  "[pr      ]"--/prec R: предшествует-->
<!ENTITY  prap    SDATA  "[prap    ]"--/precapprox R: предшествует, приближенно-->
```

```
<!ENTITY  pre      SDATA  "[pre      ]"--/preceq R: предшествует, равенство-->
<!ENTITY  prsim    SDATA  "[prsim    ]"--/precsim R: предшествует, подобие-->
<!ENTITY  rtrie    SDATA  "[rtrie    ]"--/trianglerighteq R: правый треугольник, равенство-->
<!ENTITY  samalg   SDATA  "[samalg   ]" --/smallamalg R: small amalg-->
<!ENTITY  sc       SDATA  "[sc       ]"--/succ R: следует-->
<!ENTITY  scap     SDATA  "[scap     ]"--/succapprox R: следует, приближенно-->
<!ENTITY  sccue    SDATA  "[sccue    ]"--/succcurlyeq R: следует, изогнутое равенство-->
<!ENTITY  see      SDATA  "[see      ]"--/succeq R: следует, равенство-->
<!ENTITY  scsim    SDATA  "[scsim    ]"--/succsim R: следует, подобие-->
<!ENTITY  sfrown   SDATA  "[sfrown   ]"--/smallfrown R: малая дуга, направленная вниз-->
<!ENTITY  smid     SDATA  "[smid     ]"--/shortmid R:-->
<!ENTITY  smile    SDATA  "[smile    ]"--/smile R: дуга, направленная вверх-->
<!ENTITY  spar     SDATA  "[spar     ]"--/shortparallel R: короткие параллели-->
<!ENTITY  sqsub    SDATA  "[sqsub    ]"--/sqsubset R: квадратное подмножество-->
<!ENTITY  sqsube   SDATA  "[sqsube   ]" --/sqsubseteq R: квадрат. подмножество, равенство-->
<!ENTITY  sqsup    SDATA  "[sqsup    ]"--/sqsupset R: квадратная импликация-->
<!ENTITY  sqsupe   SDATA  "[sqsupe   ]"--/sqsupseteq R: квадратная импликация, равенство-->
<!ENTITY  ssmile   SDATA  "[ssmile   ]"--/smallsmile R: малая дуга, направленная вверх-->
<!ENTITY  Sub      SDATA  "[Sub      ]"--/Subset R: двойное подмножество-->
<!ENTITY  subE     SDATA  "[subE     ]"--/subseteqq R: подмножество, двойное равенство-->
<!ENTITY  Sup      SDATA  "[Sup      ]"--/Supset R: двойная импликация-->
<!ENTITY  supE     SDATA  "[supE     ]"--/supseteqq R: импликация, двойное равенство-->
<!ENTITY  thkap    SDATA  "[thkap    ]"--/thickapprox R: толстое приближенно-->
<!ENTITY  thks im  SDATA  "[thksim   ]"--/thicksim R: толстое подобие-->
<!ENTITY  trie     SDATA  "[trie     ]"--/triangleq R: треугольник, равенство-->
<!ENTITY  twixt    SDATA  "[twixt    ]"--/between R: между-->
<!ENTITY  vdash    SDATA  "[vdash    ]"--/vdash R: вертикаль, тире-->
<!ENTITY  Vdash    SDATA  "[Vdash    ]"--/Vdash R: двойная вертикаль, тире-->
<!ENTITY  vDash    SDATA  "[vDash    ]"--/vDash R: вертикаль, двойное тире-->
<!ENTITY  veebar   SDATA  "[veebar   ]"--/veebar R: логическое или, линия снизу-->
<!ENTITY  vltri    SDATA  "[vltri    ]"--/vartriangleleft R: левый треуг., прозрачный, иной-->
<!ENTITY  vprop    SDATA  "[vprop    ]"--/varpropto R: пропорциональность, иная-->
<!ENTITY  vrtri    SDATA  "[vrtri    ]"--/vartriangleright R: прав. треуг., прозрачный, иной-->
<!ENTITY  Vvdash   SDATA  "[Vvdash   ]"--/Vvdash R: тройная вертикаль, тире-->
```

D.4.5.4 Отрицательные отношения

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для использования с
      удовлетворяющими SGML системами и приложениями, как определено в
      ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
```

```
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOamsn PUBLIC
       "ISO 8879-1986//ENTITIES Added Math Symbols: Negated
       Relations//EN">
     %ISOamsn;
-->

<!ENTITY  gnap    SDATA  "[gnap    ]"--/gnapprox N: больше, не приближенно-->
<!ENTITY  gne     SDATA  "[gne     ]"--/gneq N: больше, не равенство-->
<!ENTITY  gnE     SDATA  "[gnE     ]"--/gneqq N: больше, не двойное равенство-~>
<!ENTITY  gnsim   SDATA  "[gnsim   ]"--/gnsim N: больше, не подобие-->
<!ENTITY  gvnE    SDATA  "[gvnE    ]"--/gvertneqq N: больше, верт., не дв. рав.-->
<!ENTITY  lnap    SDATA  "[lnap    ]"--/Inapprox N: меньше, не приближенно-->
<!ENTITY  lnE     SDATA  "[lnE     ]"--/Ineqq N: меньше, не двойное равенство-->
<!ENTITY  lne     SDATA  "[lne     ]"--/Ineq N: меньше, не равенство-->
<!ENTITY  lnsim   SDATA  "[lnsim   ]"--/Insim N: меньше, не подобие-->
<!ENTITY  lvnE    SDATA  "[lvnE    ]"--/Ivertneqq N: меньше, верт., не дв. рав.-->
<!ENTITY  nap     SDATA  "[nap     ]"--/napprox N: не прилиженно-->
<!ENTITY  ncong   SDATA  "[ncong   ]"--/ncong N: не конгруэнтно с-->
<!ENTITY  nequiv  SDATA  "[nequiv  ]"--/nequiv N: не тождественно-->
<!ENTITY  ngE     SDATA  "["ngE    ]"--/ngeqq N: не больше, двойное равенство-->
<!ENTITY  nge     SDATA  "[nge     ]"--/ngeq N: не больше-чем-или-равно-->
<!ENTITY  nges    SDATA  "[nges    ]"--/ngeqslant N: не бол.-или-равно, с наклон.-->
<!ENTITY  ngt     SDATA  "["ngt    ]"--/ngtr N: не больше-чем-->
<!ENTITY  nle     SDATA  "[nle     ]"--/nieq N: не меньше-чем-или-равно-->
<!ENTITY  nlE     SDATA  "[nlE     ]"--/nieqq N: не меньше, двойное равенство-->
<!ENTITY  nles    SDATA  "[nles    ]"--/nieqslant N: не мен.-или-равно, с наклон.-->
<!ENTITY  nlt     SDATA  "[nlt     ]"--/niess N: не меньше-чем-->
<!ENTITY  nltri   SDATA  "[nltri   ]"--/ntriangleleft N: не левый треугольник-->
<!ENTITY  nltrie  SDATA  "[nltrie  ]"--/ntrianglelefteq N: не лев. треуг., равенство-->
<!ENTITY  nmid    SDATA  "[nmid    ]"--/nmid-->
<!ENTITY  npar    SDATA  "[npar    ]"--/nparallel N: не параллельно-->
<!ENTITY  npr     SDATA  "[npr     ]"--/nprec N: не предшествует-->
<!ENTITY  npre    SDATA  "[npre    ]"--/npreceq N: не предшествует, равенство-->
<!ENTITY  nrtri   SDATA  "[nrtri   ]"--/ntriangleright N: не правый треугольник-->
<!ENTITY  nrtrie  SDATA  "[nrtrie  ]"--/ntrianglerighteq N: не прав. треуг., рав.-->
<!ENTITY  nsc     SDATA  "[nsc     ]"--/nsucc N: не следует-->
<!ENTITY  nsce    SDATA  "[nsce    ]"--/nsucceq N: не следует, равенство-->
<!ENTITY  nsim    SDATA  "[nsim    ]"--/nsim N: не подобие-->
<!ENTITY  nsime   SDATA  "[nsime   ]"--/nsimeq N: не подобие, равенство-->
<!ENTITY  nsmid   SDATA  "[nsmid   ]"--/nshortmid-->
<!ENTITY  nspar   SDATA  "[nspar   ]"--/nshortparallel N: не короткие параллели-->
<!ENTITY  nsub    SDATA  "[nsub    ]"--/nsubset N: не подмножество-->
<!ENTITY  nsube   SDATA  "[nsube   ]"--/nsubseteq N: не подмножество, равенство-->
<!ENTITY  nsubE   SDATA  "[nsubE   ]"--/nsubseteqq N: не подмнож., дв. равенство-->
<!ENTITY  nsup    SDATA  "[nsup    ]"--/nsupset N: не импликация-->
<!ENTITY  nsupE   SDATA  "[nsupE   ]"--/nsupseteqq N: не имплик., дв. равенство-->
```

```
<!ENTITY  nsupe   SDATA  "[nsupe    ]"--/nsupseteq N: не импликация, равенство-->
<!ENTITY  nvdash  SDATA  "[nvdash   ]"--/nvdash N: не вертикаль, тире-->
<!ENTITY  nvDash  SDATA  "[nvDash   ]"--/nvDash N: не вертикаль, двойное тире-->
<!ENTITY  nVDash  SDATA  "(nVDash   ]"--/nVDash N: не дв. вертикаль, дв. тире-->
<!ENTITY  nVdash  SDATA  "[nVdash   ]"--/nVdash N: не дв. вертикаль, тире-->
<!ENTITY  prnap   SDATA  "[prnap    ]"--/precnapprox N: предшеств., не приближ.-->
<!ENTITY  prnE    SDATA  "[prnE     ]"--/precneqq N: предшеств., не дв. равенство-->
<!ENTITY  prnsim  SDATA  "[prnsim   ]"--/precnsim N: предшеств., не подобие-->
<!ENTITY  scnap   SDATA  "[scnap    ]"--/succnapprox N: следует, не приближ.-->
<!ENTITY  scnE    SDATA  "[scnE     ]"--/succneqq N: следует, не дв. равенство-->
<!ENTITY  scnsim  SDATA  "[scnsim   ]"--/succnsim N: следует, не подобие-->
<!ENTITY  subne   SDATA  "[subne    ]"--/subsetneq N: подмножество, не равенство-->
<!ENTITY  subnE   SDATA  "[subnE    ]"--/subsetneqq N: подмнож., не дв. равенство-->
<!ENTITY  supne   SDATA  "[supne    ]"--/supsetneq N: импликация, не равенство-->
<!ENTITY  supnE   SDATA  "[supnE    ]"--/supsetneqq N: имплик., не дв. равенство-->
<!ENTITY  vsubnE  SDATA  "[vsubnE   ]"--/subsetneqq N: подмнож., не дв. рав., иное-->
<!ENTITY  vsubne  SDATA  "[vsubne   ]"--/subsetneq N: подмнож., не равенство, иное-->
<!ENTITY  vsupne  SDATA  "[vsupne   ]"--/supsetneq N: имплик., не равенство, иное-->
<!ENTITY  vsupnE  SDATA  "[vsupnE   ]"--/supsetneqq N: имплик., не дв. рав., иное-->
```

D.4.5.5 Стрелочные отношения

```
<!--  (C) Международная организация по стандартизации, 1986
      Предоставляется разрешение копировать в любой форме для
      использования с
      удовлетворяющими SGML системами и приложениями, как
      определено в
      ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!--  Набор символьных объектов. Типичный вызов:
      <!ENTITY % ISOamsa PUBLIC
        "ISO 8879-1986//ENTITIES Added Math Symbols: Arrow
        Relations//EN">
      %ISOamsa;
-->
<!ENTITY  cularr   SDATA  "[cularr   ]"--/curvearrowleft A: левая изогнутая стрелка-->
<!ENTITY  curarr   SDATA  "[curarr   ]"--/curvearrowright A: правая изогнутая стрелка-->
<!ENTITY  dArr     SDATA  "[dArr     ]"--/Downarrow A: двойная стрелка вниз -->
<!ENTITY  darr2    SDATA  "[darr2    ]"--/downdownarrows A: две стрелки вниз-->
<!ENTITY  dharl    SDATA  "[dharl    ]"--/downleftharpoon A: вниз гарпун слева-->
<!ENTITY  dharr    SDATA  "[dharr    ]"--/downrightharpoon A: вниз гарпун сверху-->
<!ENTITY  lAarr    SDATA  "[lAarr    ]"--/Lleftarrow A: левая тройная стрелка-->
```

```
<!ENTITY  Larr    SDATA  "[Larr    ]"--/twoheadleft arrow A:-->
<!ENTITY  larr2   SDATA  "[larr2   ]"--/leftleftarrows A: две левых стрелки-->
<!ENTITY  larrhk  SDATA  "[larrhk  ]"--/hookleftarrow A: левая стрелка с крюком-->
<!ENTITY  larrlp  SDATA  "[larrlp  ]"--/looparrowleft A: левая стрелка с петлей-->
<!ENTITY  larrtl  SDATA  "[larrtl  ]"--/leftarrowtail A: левая стрелка с оперением-->
<!ENTITY  lhard   SDATA  "[lhard   ]"--/leftharpoondown A: левый гарпун снизу-->
<!ENTITY  lharu   SDATA  "[lharu   ]"--/leftharpoonup A: левый гарпун сверху-->
<!ENTITY  hArr    SDATA  "[hArr    ]"--/Leftrightarrow A: двойная стрелка влево и вправо-->
<!ENTITY  harr    SDATA  "[harr    ]"--/leftrightarrow A: стрелка влево и вправо-->
<!ENTITY  lrarr2  SDATA  "[lrarr2  ]"--/leftrightarrows A: стрелка влево над стрелк. вправо-->
<!ENTITY  rlarr2  SDATA  "[rlarr2  ]"--/rightleftarrows A: стрелка вправо над стрелк. влево-->
<!ENTITY  harrw   SDATA  "[harrw   ]"--/leftrightsquigarrow A: волн. стрелка влево и вправо-->
<!ENTITY  rlhar2  SDATA  "[rlhar2  ]"--/rightleftharpoons A: правый гарпун над левым-->
<!ENTITY  lrhar2  SDATA  "[lrhar2  ]"--/leftrightharpoons A: левый гарпун над правым-->
<!ENTITY  lsh     SDATA  "[lsh     ]"--/Lsh A:-->
<!ENTITY  map     SDATA  "[map     ]"--/mapsto A:-->
<!ENTITY  mumap   SDATA  "[mumap   ]"--/multimap A:-->
<!ENTITY  nearr   SDATA  "[nearr   ]"--/nearrow A: стрелка в направлении вправо вверх-->
<!ENTITY  nlArr   SDATA  "[nlArr   ]"--/nLeftarrow A: не следует из-->
<!ENTITY  nlarr   SDATA  "[nlarr   ]"--/nieftarrow A: не левая стрелка-->
<!ENTITY  nhArr   SDATA  "[nhArr   ]"--/nLeftrightarrow A: не левая и правая дв. стрелка-->
<!ENTITY  nharr   SDATA  "[nharr   ]"--/nieftrightarrow A: не левая и правая стрелка-->
<!ENTITY  nrarr   SDATA  "[nrarr   ]"--/nrightarrow A: не правая стрелка-->
<!ENTITY  nrArr   SDATA  "[nrArr   ]"--/nRightarrow A: не влечет-->
<!ENTITY  nwarr   SDATA  "[nwarr   ]"--/nwarrow A: стрелка в направлении влево вверх-->
<!ENTITY  olarr   SDATA  "[olarr   ]"--/circlearrowleft A: левая стрелка в окружности-->
<!ENTITY  orarr   SDATA  "[orarr   ]"--/circlearrowright A: правая стрелка в окружности-->
<!ENTITY  rAarr   SDATA  "[rAarr   ]"--/Rrightarrow A: правая тройная стрелак-->
<!ENTITY  Rarr    SDATA  "[Rarr    ]"--/twoheadrightarrow A:-->
<!ENTITY  rarr2   SDATA  "[rarr2   ]"--/rightrightarrows A: две правых стрелки-->
<!ENTITY  rarrhk  SDATA  "[rarrhk  ]"--/hookrightarrow A: правая стрелка с крючком-->
<!ENTITY  rarrlp  SDATA  "[rarrlp  ]"--/looparrowright A: правая стрелка с петлей-->
<!ENTITY  rarrtl  SDATA  "[rarrtl  ]"--/rightarrowtail A: правая стрелка с оперением-->
<!ENTITY  rarrw   SDATA  "[rarrw   ]"--/squigarrowright A: волнистая правая стрелка-->
<!ENTITY  rhard   SDATA  "[rhard   ]"--/rightharpoondown A: правый гарпун снизу-->
<!ENTITY  rharu   SDATA  "[rharu   ]"--/rightharpoonup A: правый гарпун сверху-->
<!ENTITY  rsh     SDATA  "[rsh     ]"--/Rsh A:-->
<!ENTITY  drarr   SDATA  "[drarr   ]"--/searrow A: стрелка в направлении вправо вниз -->
<!ENTITY  dlarr   SDATA  "[dlarr   ]"--/swarrow A: стрелка в направлении влево вниз -->
<!ENTITY  uArr    SDATA  "[uArr    ]"--/Uparrow A: двойная стрелка вверх-->
<!ENTITY  uarr2   SDATA  "[uarr2   ]"--/upuparrows A: две стрелки вверх-->
<!ENTITY  vArr    SDATA  "[vArr    ]"--/Updownarrow A: двойная стрелка вверх и вниз-->
<!ENTITY  varr    SDATA  "[varr    ]"--/updownarrow A: стрелка вверх и вниз-->
<!ENTITY  uharl   SDATA  "[uharl   ]"--/upleftharpoon A: вверх гарпун слева
<!ENTITY  uharr   SDATA  "[uharr   ]"--/uprightharpoon A: вверх гарпун справа
<!ENTITY  xlArr   SDATA  "[xlArr   ]"--/Longleftarrow A: длинная левая двойная стрелка-->
<!ENTITY  xhArr   SDATA  "[xhArr   ]"--/Longleftrightarrow A: длин. лев. и прав. дв. стрелка-->
<!ENTITY  xharr   SDATA  "[xharr   ]"--/longleftrightarrow A: длин. лев. и прав. стрелка-->
<!ENTITY  xrArr   SDATA  "[xrArr   ]"--/Longrightarrow A: длинная правая двойная стрелка-->
```

D.4.5.6 Открывающие и закрывающие разграничители

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Набор символьных объектов. Типичный вызов:
     <!ENTITY % ISOamsc PUBLIC
       "ISO 8879-1986//ENTITIES Added Math Symbols: Delimiters//EN">
     %ISOamsc;
-->
<!ENTITY rceil  SDATA "[rceil  ]"--/rceil C: потолок справа-->
<!ENTITY rfloor SDATA "[rfloor ]"--/rfloor C: пол справа-->
<!ENTITY rpargt SDATA "[rpargt ]"--/rightparengtr C: правая скобка, больше-->
<!ENTITY urcorn SDATA "[urcorn ]"--/urcorner C: верхний правый угол-->
<!ENTITY drcorn SDATA "[drcorn ]"--/lrcorner C: нижний правый угол-->
<!ENTITY lceil  SDATA "[lceil  ]"--/lceil 0: потолок слева-->
<!ENTITY lfloor SDATA "[lfloor ]"--/lfloor 0: пол слева-->
<!ENTITY lpargt SDATA "[lpargt ]"--/leftparengtr 0: левая скобка, больше-->
<!ENTITY ulcorn SDATA "[ulcorn ]"--/ulcorner 0: верхний левый угол-->
<!ENTITY dlcorn SDATA "[dlcorn ]"--/llcorner 0: нижний левый угол-->
```

ПРИЛОЖЕНИЕ Е

ПРИМЕРЫ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Е.1 Определение типа документа

Следующий пример приведен как иллюстрация практического определения типа документа. Он прежде всего предназначен для иллюстрации правильного использования объявлений разметки, но он также в целом отражает надлежащую практику проектирования.

```
<!-- (C) Международная организация по стандартизации, 1986
     Предоставляется разрешение копировать в любой форме для использования с
     удовлетворяющими SGML системами и приложениями, как определено в
     ИСО 8879, при условии, что это объявление будет включено во все копии.
-->
<!-- Определение типа документа. Типичный вызов:
     <!DOCTYPE general PUBLIC "ISO 8879-1986//DTD General Document//EN" [
     <!ENTITY % ISOnum PUBLIC
     "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN">
     <!ENTITY % ISOpub PUBLIC
     "ISO 8879-1986//ENTITIES Publishing//EN">
     %ISOnum; %ISOpub;
     (Здесь могут быть определены объекты параметров и дополнительные элементы.)
```

]>

-->

<!ENTITY % doctype "general" -- Родовой идентификатор типа документа-->

<!--Это определение типа документа для документа "general".

Оно содержит необходимые элементы для использования во многих приложениях, и

Организовано таким образом, что в подмножество объявления типа документа

могут быть добавлены другие элементы.-->

<!--Условные соглашения наименования объектов-->

<!--

Префикс = где используется:

p. = в параграфах (также во фразах, если суффикс ph)

s. = в разделах (т.е. среди параграфов)

ps. = в параграфах и разделах

i. = где допускается включением

m. = модель содержания или объявленное содержание

a. = определение атрибута

NONE= специальное использование, определенное в моделях

Suffix = допустимое содержание

.ph = элементы, чьим содержанием является %m.ph

.d = элементы, чье содержание имеет то же определение

NONE= элементы с уникальными определениями

-->

<!--Маркеры элементов-->

<!ENTITY % p.em.ph "hpl|hp2|hp3|hp0|cit" – Выделенные фразы -->

<!ENTITY % p.rf.ph "hdref|figref" -- Фразы ссылок -->

<!ENTITY % p.rf.d "fnref|liref" -- Ссылки (пустые) -->

<!ENTITY % p.zz.ph "(%p.em.ph;)|(%p.rf.ph;)|(%p.rf.d;)" –

```
                        Все фразы -->
<!ENTITY % ps.ul.d      "ol | sl | ul | nl" -- Списки элементов - позиций-->
<!ENTITY % ps.list      "%ps.ul.d; | dl | gl" -- Все списки -->
<!ENTITY % ps.elem      "xmp |lq | lines | tbl | address | artwork" -- Другие
                        элементы -->
<!ENTITY % ps.zz        "(%ps.elem;) | (%ps.list;)" -- Подэлементы
                        параграфов/разделов-->
<!ENTITY % s.p.d        "p | note" -- Простые параграфы -->
<!ENTITY % s.top        "topl | top2 | top3 | top4" -- Темы -->
<!ENTITY % s.zz         "(%s.p.d;) | (%ps.zz;) | (%s.top;)" -- Подэлементы
                        разделов-->
<!ENTITY % i.float      "fig | fn" -- Плавающие элементы-->
<!ENTITY % fm.d         "abstract | preface" -- Вступление -->
<!ENTITY % bm.d         "glossary | bibliog" -- Завершение -->

                         <!-- Группы модели -->
<!ENTITY % m.ph         "(#PCDATA | (%p.zz.ph;))*" -- Модель фразы -->
<!ENTITY % m.p          "(#PCDATA | (%p.zz.ph;)|(%ps.zz;))*" -- Модель
                        параграфа -->
<!ENTITY % m.pseq       "(P, ((%s.p.d;) | (%ps.zz;))*)" --
                        Последовательность параграфов -->
<!ENTITY % m.top        "(th?, p, (%s.zz;)*)" -- Модель темы-->

                        <!--Структура документа-->
<!--МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT %doctype; - -  (frontm?, body, appendix?, backm?)
                         +(ix|%i.float;)>
<!ELEMENT Frontm    - 0  (titlep, (%fm.d;|hl)*, toc?, figlist?)>
<!ELEMENT Body      - 0  (h0+|hl+)>
<!ELEMENT Appendix  - 0  (hl+)>
```

```
<!ELEMENT Backm      - 0  ((%bm.d;|hl)*, index?)>
<!ELEMENT (toc | figlist | index )      --      Оглавление,      список
          иллюстраций, --
                - 0   EMPTY      --    и    индекс    генерируются
                      содержанием-->

                <!--Элементы титульного листа-->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT  titlep  - 0  (title & docnum? & date? & abstract? &
                        (author | address | %s.zz;)*)>
<!ELEMENT  (docnum|date|author)
                - 0  (#PCDATA) -- Номер документа и т.д. -->
<!ELEMENT  title   - 0  (tline+) -- Заглавие документа-->
<!ELEMENT  tline   0 0  %m.ph; -- Строка заглавия-->

                <!-- Озаглавленные разделы -->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT h0       - 0    (h0t, (%s.zz;)*, hl+) -- Часть -->
<!ELEMENT (hl | %bm.d; | %fm.d;)
                - 0    (h1t, (%s.zz;)*, h2*) -- Глава -->
<!ELEMENT h2       - 0    (h2t, (%s.zz;)*, h3*) -- Раздел -->
<!ELEMENT h3       - 0    (h3t, (%s.zz;)*, h4*) -- Подраздел -->
<!ELEMENT h4       - 0    (h4t, (%s.zz;)*) -- Подподраздел -->
<!ELEMENT (h0t | hlt | h2t | h3t | h4t)
                0 0    %m.ph; -- Названия озаглавленных разделов-->

                <!-- Темы (подразделы с названиями) -->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT top1     - 0  %m.top; -(top1) -- Тема 1 -->
<!ELEMENT top2     - 0  %m.top; -(top2) -- Тема 2 -->
```

```
<!ELEMENT top3    - 0  %m.top; -(top3) -- Тема 3 -->
<!ELEMENT top4    - 0  %m.top; -(top4) -- Тема 4 -->
<!ELEMENT  th     - 0  %m.ph; -- Заголовок темы -->

                <!-- Элементы в разделах или параграфах -->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT Address     - -  (aline+)>
<!ELEMENT aline       0 0  %m.ph; -- Строка адреса -->
<!ELEMENT Artwork     - 0  EMPTY>
<!ELEMENT dl          - -  ((dthd+, ddhd)?, (dt+, dd)*)>
<!ELEMENT  dt         - 0  %m.ph; -- Термин определения-->
<!ELEMENT (dthd       | - 0  (#PCDATA) -- Заголовки для dt и dd -->
           ddhd)
<!ELEMENT  dd         - 0  %m.pseq; -- Описание определения -->
<!ELEMENT gl          - -  (gt, (gd|gdg)*) -- Список глоссария -->
<!ELEMENT  gt         - 0  (#PCDATA) -- Термин глоссария -->
<!ELEMENT  gdg        - 0  (gd+) -- Группа определения глоссария -->
<!ELEMENT  gd         - 0  %m.pseq; -- Определение глоссария -->
<!ELEMENT (%ps.ul.d;) - -  (li*) -- Списки элементов -->
<!ELEMENT  1i         - 0  %m.pseq; -- Элемент списка -->
<!ELEMENT lines       0 0  %m.pseq; -- Элементы строк -->
<!ELEMENT (lq | xmp)  - -  %m.pseq; -(%i.float;) -- Длинная кавычка-->
<!ELEMENT (%s.p.d;)   - 0  %m.p; -- Параграфы-->

                <!-- Таблица -->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT Tb1     - -   (hr*, fr*, r+)>
<!ELEMENT  hr     - 0   (h+) -- Верхняя строка -->
<!ELEMENT  fr     - 0   (f+) -- Нижняя строка-->
<!ELEMENT  r      0 0   (c+) -- Строка (корпус таблицы) -->
```

```
<!ELEMENT  c       0 0   %m.pseq; -- Ячейка в строке корпуса-->
<!ELEMENT  (f|h)   0 0   (#PCDATA) -- Ячейка в fr или hr -->

                         <!-- Фразы -->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT (%p.em.ph;) - -   %m.ph; -- Выделенные фразы -->
<!ELEMENT q             - -   %m.ph; -- Цитата -->
<!ELEMENT (%p.rf.ph);   - -   %m.ph; -- Фразы ссылок -->
<!ELEMENT (%p.rf.d;)    - 0   EMPTY-- Сгенерированные ссылки-->

                         <!-- Включаемые подэлементы-->
<! --МИНИМАЛЬНОЕ СОДЕРЖАНИЕ ЭЛЕМЕНТОВ (ИСКЛЮЧЕНИЯ)-->
<!ELEMENT  fig        - -   (figbody, (figcap, figdesc?)?) -(%i.float;)>
<!ELEMENT  figbody    0 0   %m.pseq; -- Корпус рисунка -->
<!ELEMENT  figcap     - 0   %m.ph; -- Название рисунка -->
<!ELEMENT  figdesc    - 0   %m.pseq; -- Описание рисунка -->
<!ELEMENT  fn         - -   %m.pseq; -(%i.float;) -- Примечание -->
<!ELEMENT  ix         - 0   (//PCDATA) -- Элемент индекса -->

                         <!-- Списки определений атрибутов -->
<!-- Поскольку это определение типа документа предназначено для
базовых документов SGML, в которых функции связи не
поддерживаются, было необходимо включить в определения атриубты
связи.
-->
<!—ЭЛЕМЕНТЫ ИМЯ ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ-->
<!ATTLIST   %doctype;  security CDATA  #IMPLIED
                       status   CDATA  ""
                       version  CDATA  #IMPLIED>
<!ATTLIST   title      stifle   CDATA  #IMPLIED>
```

```
<!ATTLIST   (h0 | hl | h2 | %bm.d; |
            %fm.d;)
                    id          ID          #IMPLIED
                    stitle      CDATA       #IMPLIED>
<!ATTLIST   (h3|h4) id          ID          #IMPLIED>
<!ATTLIST   Artwor  sizex       NMTOKE      Textsize
            k                   N
          -- По умолчанию ширина текущего текста в колонке. --
                    sizey       NMTOKEN     #REQUIRED
          -- (Размеры указываются в единицах, поддерживаемых
             приложением, в котором появляется это объявление;
             для sizex может использоваться ключевое слово "textsize"
             для обозначения "ширины, на которую был установлен
          предыдущий текст").
          -->
<!ATTLIST   gl      compact     (compact)   #IMPLIED
                    termhi      NUMBER      2>
<!ATTLIST   dl      compact     (compact)   #IMPLIED
                    headhi      NUMBER      2
                    termhi      NUMBER      2
                    tsize       NUMBERS     9
          -- Число элементов dt на один элемент dd должно равняться
             количеству номеров, указанных для tsize (здесь 1).
             Число элементов dthd должно быть таким же.
          -->
<!ATTLIST   gd          source      CDATA       #IMPLIED>
<!ATTLIST   (%ps.ul.d;) compact     (compact)   #IMPLIED>
<!ATTLIST   li          id          ID          #IMPLIED>
<!ATTLIST   xmp         depth       NUTOKE      #IMPLIED
                                    N
```

```
                        keep        NMTOKE  all
                                    N
                        lines       (flow | lines) lines>
<!ATTLIST   Tbl         cols        NUMBE  #REQUIRED
                                    RS
          -- Число элементов c на элемент r должно быть равно
             количеству номеров, указанных для cols
             (аналогично, число h на hr и f на fr).
          -->
<!ATTLIST   c           heading   (h)           #IMPLIED
          -- Если указано h, ячейка является названием строки.
          -->
<!ATTLIST   (%p.rf.ph;) refid     IDREF         #CONREF
                        page      (yes | no)    yes>
<!ATTLIST   fnref       refid     IDREF         #REQUIRED>
<!ATTLIST   liref       refid     IDREF         #REQUIRED
                        page      (yes | no)    yes>
<!ATTLIST   fig         id        ID            #IMPLIED
                        frame     (box  |  rule  | none
                                  none)
                        place     (top  |  fixed  | top
                                  bottom)
                        width     (column       | page
                                  page)
                        align     (left | center | center
                                  right)
                        lines     (flow | lines)  lines>
<!ATTLIST   ix          id        ID          #IMPLIED
                        print     CDATA       #IMPLIED
                        see       CDATA       #IMPLIED
```

```
                    seeid     IDREF     #IMPLIED>
<!ATTLIST  fn       id        ID        #IMPLIED>

                <!-- Объекты для кратких ссылок-->
<!ENTITY  ptag      STARTTA   "p  -- Начальный тег параграфа -->
                    G         "
<!ENTITY  qtag      STARTTA   "q  -- Начальный тег цитаты -->
                    G         "
<!ENTITY  qetag     ENDTAG    "q  -- Конечный тег цитаты -->
                              "
<!ENTITY  endtag    ENDTAG    "" -- Пустой конечный тег для
                                  любого элемента-->
<!SHORTREF docmap                  -- Отображение для общего
                                   использования --
                    "&3#RS;&# ptag -- Пустая строка - это <p> --
                    RE;"
                    ’"’       qtag -- " -это <q> -->
<!USEMAP   docmap %doctype;>
<!SHORTREF qmap                    -- Отображение для цитат --
                    ’"’       qetag -- " - это </q> -->
<!USEMAP   qmap
           q>
<!SHORTREF ixmap                   -- Отображение для записей
                                   индекса--
                    "&#RE;"   endtag -- Конец записи это </> -->
<!USEMAP   ixmap ix>
```

Расчет доступного объема для этого определения типа документа приведен ниже.

Объявлены 23 объекта с 732 символами текста.

Объявлены 78 типов элементов с 696 маркерами моделей и 8 группами исключений с 13 именами

Объявлены 39 атрибутов с 23 членами группы и 80 символами текста значения по умолчанию.

Определены 0 ID и 0 ссылок на ID.

0 нотаций содержания данных с 0 текстовых символов.

Объявлены 3 отображения кратких ссылок.

Требуется 8644 пунктов доступного объема (24% из допустимых 35000).

E.2 Метафайл машинной графики

Изображения, сгенерированные программой машинной графики, могут быть включены в документ SGML с помощью методов, проиллюстрированных на рис. 12 и 13.

Иллюстрируемые нотации - это кодировки метафайла машинной графики (CGM). Поскольку в таком файле могут встречаться разграничители SGML, он должен быть встроен с помощью ссылки на содержание, чтобы он не просматривался синтаксическим анализатором SGML. Он может быть безопасно внедрен в объект SGML, только если он был исследован и найден свободным от разграничителей, которые должны распознаваться в заменяемых символьных данных, или если он предварительно обработан с целью конвертирования таких разграничителей в ссылки на объект.

Обратите внимание, что элемент "graphic" не содержит атрибутов для позиционирования или ссылки на него. Элемент аналогичен форматированному тексту или незаполненному пространству, оставленному для вклеивания (подобно элементу "artwork" в определении типа документа E.1), в котором он обычно форматируется в том же месте

логического документа, где он введен. Чтобы переместить его в другое место, или заключить в рамку, или присвоить ему ID, по которому на него можно будет ссылаться, его необходимо включить в корпус элемента рисунка.

Если объявление элемента для "graphic" сохраняется в файле "graphic.etd", следующее объявление позволит графическим метафайлам появляться в рисунках в пределах документа "general":

<!DOCTYPE general PUBLIC "ISO 8879-1986//DTD General Document//EN" [

<!ENTITY % ps.elem "graphic | xmp | lq | lines | tbl | address | artwork">

<!ENTITY % graphic SYSTEM "graphic.etd">

%graphic;

] >

E.3 Аппаратно - независимое расширение кода

ПРИМЕЧАНИЕ. Этот раздел предназначен только для того, чтобы проиллюстрировать подходы к использованию ИСО 2022 метода расширения кода графического алфавита ИСО 2022 с SGML; не все комбинации методов рассматриваются, не для всех предлагается законченный проект. Предполагается, что читатель знаком с ИСО 2022, ИСО 4873 и ИСО 6937.

Документы SGML могут содержать множественные графические символьные алфавиты, используя методы расширения кода ИСО 2022. Кроме того, эти методы могут быть дополнены ссылками на объект SGML и краткими ссылками, чтобы достичь такой степени свободы от устройства и кодовых зависимостей, которая не возможна с одним расширением кода.

Основной принцип, на котором основано расширение кода - возможность битовой комбинация представлять больше чем один символ: какой именно - зависит от управляющих символов и escape-последовательностей, которые ему предшествуют. Поэтому, при использовании расширения кода с SGML, необходимо разрешить присутствие управляющих символов расширения кода в объектах SGML таким способом, который будет предотвращать смешивание разграничителей с ними или с расширенными символами.

E.3.1 Средства расширения кода

Те средства, которые требуют наименьшего объема усилий пользователя и синтаксического анализа, чтобы избежать смешивания с разграничителями, когда не используется подавление разметки, это средства, в которых все графические символы разметки находятся в наборе G0, который занимает левую (или единственную) сторону кодовой таблицы, а код:

a) 8-разрядный, с набором G1, всегда занимает правую сторону. Символьный алфавит G1 может быть при необходимости изменен с соответствующим обозначением escape-последовательности. Смещения с блокировкой не используются.

b) 8-разрядный, с наборами G1, G2 и G3, перемещенный смещениями с блокировкой в правую сторону кодовой таблицы. Если необходимо более трех дополнительных наборов, символьные алфавиты G1, G2 и G3 могут быть изменены с соответствующим обозначением escape-последовательности.

c) 7-разрядный или 8-разрядный, с символами из наборов G2 и G3, перемещенный одиночными смещениями в левую (или единственную) сторону кодовой таблицы. Как и выше, символьные алфавиты G2 и G3 могут быть изменены, как необходимо. В 8-разрядных кодах это средство может быть объединено с любым из двух других.

На рис. 14 проиллюстрирован параметр идентификации символа функции, который должен заменить такой параметр многокодового базового конкретного синтаксиса при использовании вышеупомянутых средств расширения кода.

```
<!NOTATION  cgmchar      PUBLIC
            "ISO 8632/2//NOTATION Character encoding//EN" -->
<!NOTATION  cgmclear  PUBLIC
             "ISO 8632/4//NOTATION Clear text encoding//EN" -->
<!ELEMENT   graphic   - 0 RCDATA>
<!ATTLIST   graphic
--          ИМЯ     ЗНАЧЕНИЕ                    ПО УМОЛЧАНИЮ--
            file        ENTITY                  #CONREF
            -- Внешний объект, содержащий метафайл.
               Если не определено, метафайл является синтаксическим
               содержанием элемента.
            --
            coding    NOTATION (cgmclear | cgmchar)    cgmclear
            -- Нотация содержания данных, когда меатфайл не является
               внешним. Игнорируется, если поскольку
               нотация определяется в объявлении объекта.
            --
            picnum    NUMBER                    1
            -- Порядковый номер рисунка, если метафайл
               содержит больше одного рисунка.
            --
            x0          CDATA --нижний левый угол--     #IMPLIED
            xl          CDATA                           #IMPLIED
            y0          CDATA --верхний правый угол--   #IMPLIED
            yl          CDATA                           #IMPLIED
            -- Координаты места просмотра рисунка в
               координатах виртуального устройства (вещественное число
               в формате -n.nEn). По умолчанию величины,
               ссылающиеся на расширение VDC.
            --
```

Рис. 12. Атрибуты графического метафайла (1 из 2): кодирование и просмотр

E.3.1.1 Предотвращение ложного распознавания разграничителей

Когда расширение кода делается с единственным набором G1 в

соответствии с ИСО 4873, возможность ложного распознавания мала или отсутствует. Другие методы расширения кода допускают большую возможность, которой можно избежать, если разработчик и/или пользователь соблюдают следующие правила:

a) Вся разметка, включая краткие ссылки и ссылки на объект, должна быть сделана в наборе G0.

b) Если объект смещается в дополнительный набор, он должен вернуться в набор G0 (если только цель объекта не заключалась в смещении).

c) Escape-последовательности, отличные от функций смещения (например, последовательности указателя и оповещателя) должны быть введены в объявления комментария.

Должна быть назначена строка разграничителя ***com***, которая не появляется ни в одной из не смещающих escape-последовательностей. Например, с набором разграничителей ссылок все публичные escape-последовательности длиной до пяти символов и частные длиной до трех символов, могут быть введены в объявления комментария без возможности ложного распознавания разграничителя.

```
   sizex          NMTOKEN                 #IMPLIED
   sizey          NUTOKEN                 #IMPLIED
-- Могут быть определены либо sizex (размер x), либо sizey (размер y),
   либо оба (в единицах измерения, поддерживаемых DTD, в котором
   появляется это объявление; для sizex может быть использовано
   ключевое слово "textsize" для обозначения "ширины, на которую был
   установлен предыдущий текст"). Если определен один из
   параметров, атрибуты scaling (масштабирование), alignx
   (выравнивание x) и aligny (выравнивание y) игнорируются, т.к.
   scaling будет обязательно иметь значение uniform
   (пропорциональное).
   --
   scaling        (uniform | nonunif)     uniform
   alignx         (left | center | right) center
   aligny         (top | middle | bottom) middle
-- Если определены оба sizex и sizey, то если scaling имеет значение
   nonuniform (непропорционально), alignx и aligny игнорируются, т.к.
   фактические x и y будут равны указанным sizex и sizey. В противном
   случае фактические x и y будут такими, что один будет равен
   указанному для него размеру, а другой будет меньше своего
   указанного размера. Атрибут alignx или aligny будет применяться к
   тому, который меньше своего указанного размера.
--
--*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*--
-- В последующей таблице сведены соотношения между атрибутами
   размера, масштаба и выравнивания:
   SizeX          SizeY  Scaling   AlignX        AlignY
   (yes)          (no)
   (no)           (yes)
   (yes)          (yes)  nonunif
   (yes)          (yes)  uniform   если x<SizeX  если y<SizeY --
--*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*--
   orient         (0 | 90 | 180 | 270)          0
-- Угол поворота в градусах. --
 >
```

Рис. 13. Атрибуты графического метафайла (2 из 2): размер и поворот

Не смещающие escape-последовательности могут также вводиться в смешанном содержании, но только если они не содержат строки разграничителей, которые могут быть распознаны, или для предотвращения ложного распознавания определены краткие ссылки, как объяснено ниже.

d) Строки, которые появляются в функциях смещения, не должны быть назначены на роли разграничителей, распознаваемые в режиме распознавания CON, если только краткие ссылки не определены, чтобы предотвратить ложное распознавание.

Например, в наборе разграничителей ссылок, правая фигурная скобка, тильда и вертикальная линия являются разграничителями краткой ссылки, но они также присутствуют в последовательностях смещения LS1R, LS2R и LS3R, необходимых, когда используются наборы G2 и G3. Поскольку они являются разграничителями краткой ссылки, они могут быть легко выключены, просто не отображая их на объекты. Но если они необходимы, они могут сохраняться доступными, путем определения последовательностей LS2R и LS3R как кратких ссылок, которые никогда не отображаются на объекты.

```
FUNCTION  RE                    13
          RS                    10
          SPACE                 32
          TAB       SEPCHAR      9
-- Функции для расширения кода графического алфавита--
-- Разметка распознается, пока они используются. --
          ESC       FUNCHAR      27 --Escape --
          LSO       FUNCHAR      15 --Смещение с блокировкой ноль (набор
                                     G0) --
                                     --LS1R, LS2R и LS3R
                                     являются ESC последовательностями--
          SS2       FUNCHAR     142 --Одиночное смещение два (набор G2) --
          SS3       FUNCHAR     143 --Одиночное смещение три (набор G3) --
```

Рис. 14. Символы функций для аппаратно - независимых многокодовых конкретных синтаксисов

К параметру *разграничителей краткой ссылки объявления* SGML будут добавлены следующие параметры:

SHORTREF SGMLREF

-- Добавлены SHORTREF для предотвращения ложного распознавания--

-- односимвольных SHORTREF, когда им предшествует ESC--

"&#ESC;|" -- LS3R contains SHORTREF "|" --

"&#ESC;}" -- LS2R contains SHORTREF "}" --

"&#ESC;~" -- LS2R contains SHORTREF "~" --

Поскольку более длинная строка разграничителя всегда имеет преимущество при распознавании перед более короткой строкой, содержащийся в ней (даже если более длинная является краткой ссылкой, которая ни на что не отображается), вертикальная линия и правая фигурная скобка не будут ошибочно распознаваться как разграничители, когда они появляются в пределах последовательностей смещения.

е) Если используются одиночные смещения, каждая последовательность, составленная из SS2 или SS3, сопровождаемых битовой комбинацией первого символа строки разграничителя, распознаваемой в режиме CON, должна быть определена как краткая ссылка, чтобы предотвратить ложное распознавание разграничителя, описанным выше способом.

В наборе разграничителей ссылок к затрагиваемым символам относятся амперсант (&), знак " меньше чем " (<), право квадратная скобка (]) и косая черта (/), плюс первый символ разграничителей краткой ссылки, которые начинаются с графического символа.

E.3.1.2 Устранение аппаратных и кодовых зависимостей

Документ SGML, который использует средства расширения кода ИСО 2022, может быть передан другим системам или устройствам, которые используют те же самые средства. В таких случаях могут использоваться конкретные синтаксисы, описанные в D.3.1, а методы, описанные в этом разделе не обязательны. Однако, не все системы или

устройства, которым может быть адресован документ SGML, поддерживают все методы расширения кода ИСО 2022. Например, многие форматеры и фотонаборные машины будут требовать специализированной смены шрифта и команд позиционирования, чтобы выполнить то, чего расширение кода достигает с помощью escape-последовательностей и смещений.

Обычным методом SGML для достижения независимости вывода является использование ссылки на объект для любого символа, который не включен ни в набор G0, ни в *набор символов трансляции - ссылок*. Тогда как имя объекта остается неизменным, определение меняется в зависимости от системы вывода, как объяснено в D.4. Для этой цели могут использоваться публичные наборы объектов, определенные в этом пункте; среди прочего, они содержат имена объектов для существующих и предлагаемых символьных алфавитов, определенных ИСО 6937.

Когда расширение кода используется для создания документа, бремя ввода ссылок на объект может быть значительно сокращено, поскольку каждый дополнительный символ набора может быть определен как краткая ссылка и отображен на соответствующий объект. Однако, это не возможно, когда символы появляются в контексте, в котором краткие ссылки не распознаются, таком как элемент *символьных данных* или отмеченный раздел, или где распознавание разметки подавлено символами функций.

ПРИЛОЖЕНИЕ F

СООБРАЖЕНИЯ ПО РЕАЛИЗАЦИИ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Система SGML имеет два облика — одним она обращена к размеченному документу, а другим к прикладным программам, которые обрабатывают размеченный документ.

Требования и разрешенные изменения для интерфейса документа определены в основном содержании этого международного стандарта. Интерфейс с прикладными программами не стандартизирован, но это приложение идентифицирует некоторые полезные функции поддержки разметки для систем SGML, которые обеспечивают разработанным пользователями программам доступ к синтаксическому анализатору SGML и менеджеру объектов.

Обсуждение ведется в общих терминах и не охватывает все аспекты реализации. Оно предназначено исключительно для ознакомления с некоторыми проблемами реализации, и ни в каком смысле не может рассматриваться как спецификация.

F.1 Модель синтаксического анализа SGML

При обсуждении функций в качестве основы необходимо иметь в виду модель процесса синтаксического анализа SGML.

ПРИМЕЧАНИЕ. Этот раздел не должен рассматриваться как требование к конкретной архитектуре синтаксического анализатора SGML или методам реализации. Любые термины, которые могут нести иной смысл (такие как, например, процедура, стек, служебная

программа), используются только как средства выражения функциональных возможностей.

Кроме того, описанная здесь модель - просто та модель, которую разработчики SGML нашли полезной в ходе обсуждений проекта: это не единственная возможная модель, может быть не лучшая, конечно не полная, и может даже быть внутренне противоречивая. Тем не менее, она полезна для понимания того, как связан синтаксический анализатор SGML с другими компонентами системы, и предлагается только для этой цели.

F.1.1 Физический ввод

Синтаксический анализатор SGML не читает физический входной поток. Все управление вводом осуществляется менеджером объектов. Синтаксический анализатор вызывается системой, ему передаются данные для анализа.

F.1.1.1 Объекты

Когда синтаксический анализатор распознает ссылку на объект, он сообщает об этом менеджеру объекта, который обрабатывает новый объект как физический источник ввода до конца объекта, о котором менеджер объектов уведомляет синтаксический анализатор (т.е. генерирует ***Ee***). После этого менеджер объектов получит последующие входные данные из первоначального объекта.

Синтаксический анализатор полагает границы объекта существенными при распознании многосимвольных разграничителей и контекстных разграничителей, и при обеспечении соблюдения правил языка ссылками на объект в разграниченном тексте. В противном случае он видит только непрерывную строку символов безотносительно объектов, в которых они появляются.

F.1.1.2 Границы записей

Если документ содержит границы записи, они обрабатываются как определено в этот международном стандарте. Распознавание строк разграничителей, которые содержат границы записи, происходит в ходе начального синтаксического анализа ввода. Однако, определение того, что данные ***RE*** должны игнорироваться (как требуется в п.7.6.1), не может быть сделано до того, как не будут разрешены все ссылки и не будет нормализована минимизированная разметка.

F.1.2 Режимы распознавания

Существенное свойством SGML является различение режимов распознавания разграничителей, как описано в 9.6.1. После пролога документ первоначально просматривается в режиме CON, при этом текстовые символы обрабатываются как определено в 7.6, пока одно из следующих событий не вызывает смену режима:

— Описательная разметка (начальный тег или конечный тег):

Синтаксический анализ продолжается в режиме TAG.

Идентифицируется ID, а атрибуты и их значения подтверждаются и сохраняются со значениями по умолчанию, предоставленными для неопределенных атрибутов. Управление передается процедуре, сопоставленной с GI, вместе с указателем на имена и значения атрибута .

В зависимости от приложения, процедура может нуждаться в сохранении содержания всего элемента перед его обработкой, либо она может просто изменять состояние приложения и затем возвращать управление. Когда процедура элемента сохраняет содержание, процедуры для вложенных в него элементов должны также обработать их содержание и возвращать его сохраненному тексту.

ПРИМЕЧАНИЕ. Сохранение текста без анализа и последующий

его анализ, как если бы он появился в другом месте документа, является нарушением данного международного стандарта. Напротив, текст должен анализироваться как только он обнаружен, и внутренняя форма анализируемого текста (включая информацию GI, атрибутах и т.д.) является тем, что должно сохраняться для последующего использования.

После того как система возвращает управление, анализ возобновляется в режиме CON после ***tagc*** (или его эквивалента).

— Команда обработки:

Команда анализируется как CDATA (в режиме PI) и результирующая строка символов после удаления разграничителей передается прикладной программе. Программа выполняет команду и возвращает управление. Синтаксический анализ возобновляется в режиме CON после ***pic***.

— Объявление разметки:

Начало и имя идентифицированы; синтаксический анализ переходит в режим MD, чтобы идентифицировать индивидуальные параметры и конец объявления. Затем вызывается семантическая подпрограмма индивидуального объявления, чтобы выполнить объявление, и управление возвращается.

— Ссылки:

Ссылка анализируется в режиме REF, чтобы определить имя (или номер символа) и найти конец ссылки.

Если это ссылка на символ, синтаксический анализатор заменяет ее правильным символом в текущем объекте. Если это ссылка на не относящийся к SGML символ или на символ функции как данные, символ отмечается, чтобы не перепутать его с нормальным символом.

Если это ссылка на объект, имя объекта передается менеджеру объектов, который модифицирует указатель на входные буферы и возвращает управление. Для кратких ссылок, не отображенных на объект, ссылка обрабатываются, как описано в 9.4.6.

— Данные: символы данных могут передаваться процедуре как только они обнаружены, либо они могут буферизироваться до появления какой-либо разметки. В последнем случае расположение и длина данных передаются в процедуру, а синтаксический анализ продолжается в разметке.

Переданная строка данных может содержать помеченные символы из ссылок на символы, которые система должна уметь обрабатывать.

F.1.3 Минимизация разметки

Минимизация разметки позволяет пользователю опускать полностью или частично начальный или конечный теги элемента .

Для минимизации SHORTTAG система должна следить за текущим расположением в структуре элемента, обычно храня запись GI открытых элементов. Также необходимо интерпретировать определения атрибута, но не обязательно для понимания модели содержания, если только не предусматриваются службы проверки данных.

Однако, для функций OMITTAG и DATATAG знание моделей содержания необходимо.

ПРИМЕЧАНИЕ. Разрешение прикладным программам, написанным пользователями, обеспечивать поддержку минимизации разметки, является нарушением этого международного стандарта, поскольку это создает возможность различного анализа документа различными приложениями.

F.1.4 Трансляция

SGML предполагает, что набор символов документа тот же, что и набор символов системы; т.е., что синтаксический анализатор SGML не должен быть осведомленным о какой-либо трансляции. Любая требуемая трансляция осуществляется системой до обработки, или в ходе обработки таким способом, который прозрачен для синтаксического анализатора SGML.

F.1.5 Аналогия командного языка

Описательные теги SGML являются антитезой команд языка процедур, и разработчикам типа документа, в частности, не следует думать о них, как о подобных. Тем не менее, могут быть выведены аналогии между атрибутами элемента и параметрами команды, которые могут способствовать пониманию разработчика.

Термин SGML	*Термин программирования*
список определений атрибутов	список формальных параметров
определение атрибута	формальный параметр
список спецификаций атрибутов	список фактических параметров
спецификация атрибута	фактический параметр

Обратите внимание, что в отличие от многих списков формальных параметров, порядок определений отдельных атрибутов не предписывает порядок, в котором должны появляться спецификации атрибутов.

F.2 Инициализация

Пользователь должен быть способен указать системе в начале работы определенную информацию, некоторая часть которой, например, типы активного документа и типы связи, должна быть доступна синтаксическому анализатору SGML. Способ передачи информации зависит от системы.

F.2.1 Начальное отображение процедуры

Пользователь должен определить отображения между типами элементов, с одной стороны, и процедурами, которые будут для них выполняться, с другой стороны.

Это могло быть сделано в форме имени отдельной процедуры (например, процедура для элемента документа), которая создает отображения для других элементов. Сам синтаксический анализатор не нуждается в такой информации, но если бы она была доступна, синтаксический анализатор мог бы возвращать соответствующее имя процедуры всякий раз, когда он возвращает описательную информацию тега.

F.2.2 Спецификация процесса связи

При форматировании тип неформатированного документа часто называется "логическая структура", а тип отформатированного документа называется "структура верстки". Документ SGML, который должен быть отформатирован, будет всегда содержать логическую структуру (обычно тип базового документа). Если должен использоваться явный процесс связи, он будет также содержать определение типа документа для структуры верстки ("родовая структура верстки"), и возможно полностью отформатированный экземпляр документа.

Для приложения форматирования при вызове обработки пользователь должен определить тип связи, который будет генерировать экземпляр желательной структуры верстки из логической структуры, а также любые параметры, необходимые приложению. Эта информация не сохраняется как часть разметки документа, поскольку она изменяется от одного выполнения приложения к другому. Однако, разметка содержит объявления типов связи, которые определяют, которые элементы структуры верстки создаются из элементов логической структуры.

F.2.3 Параллельные экземпляры документа

Полностью отформатированный экземпляр обычно не используется

для приложения форматирования, поскольку он является результатом работы такого приложения. Он с большей вероятностью будет использоваться в процессах индексации или каталогизации, где номера строк и страниц в структуре верстки используются вместе с информацией из логической структуры. Здесь, связь между двумя структурами на высшем уровне и между подэлементами является постоянной характеристикой документа и выражается сопоставлением тегов от двух типов документов. Во время инициализации пользователь должен определить два типа документа, чьи экземпляры будут обрабатываться.

F.3 Динамическое отображение процедур

Система должна поддержать отображение процедур как динамический процесс, доступный процедурам, так как в некоторых приложениях процедура для типа элемента может определять обработку его подэлементов. Например, в приложении форматирования процедура для элемента типа "корпус" может определять, что элементы типа "заголовок" должны быть в шрифте Roman, тогда как процедура для родового идентификатора "приложение" может определять, что встречающиеся в нем элементы "заголовок" должны быть в курсиве.

Другими словами, можно считать, что элемент заголовка квалифицируется типом элемента, в чьих границах он появляется. Напрашивается модель фамилии и имени: заголовок совместно использует нечто общее с всеми другими заголовками (то же для имени), но это модифицируется тем фактом, что он является частью корпуса или приложения (две различных фамилии).

F.4 Обработка ошибок

Этот международный стандарт требует, чтобы сообщение об ошибке

разметки включало ее природу и местонахождение. Разработчики свободны в выборе наилучших средств для выполнения этого требования, с учетом их уникальных системных сред и интерфейсов пользователя.

Следует иметь в виду следующие общие соображения:

a) Расположение в структуре объекта может быть выражено как имя объекта, конец записи и позиция символа внутри него. Расположение может также включать:

i) список открытых объектов;

ii) расположение в структуре объекта, где на каждый открытый объект имеется ссылка;

iii) для каждого открытого объекта, к которому обращается краткая ссылка, имя отображения и строка краткой ссылки.

b) Расположение в структуре элемента может быть столь же полезно, как и в структуре объекта. Оно может быть выражено как список текущих открытых элементов.

c) При сообщении о характере ошибки, следует учесть минимизацию разметки или другие функции, которые могут быть использованы, и которые могут повлиять на восприятие ошибки пользователем.

ПРИЛОЖЕНИЕ G

КЛАССИФИКАЦИЯ И СЕРТИФИКАЦИЯ СООТВЕТСТВИЯ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Этот международный стандарт предлагает ряд параметров соответствия системам SGML. Ожидается, что агентства, которые удостоверяют соответствие, пожелают определить значимое подмножество этих параметров для тестирования. Им также понадобится простой способ классификации сертифицированных систем, который точно идентифицирует функции и изменения синтаксиса, которые поддерживает система .

Это приложение описывает схему классификации, которая выполняет обе эти задачи, и рассматривает некоторые из значений для сертификационных агентств.

G.1 Код классификации

Схема классификации приведена на рис. 15. Схема присваивает системам SGML код классификации соответствия, который состоит из трех подчиненных кодов: функция, синтаксис и проверка данных. По этой причине, код классификации соответствия может называться "FSV".

Например, FSV "0768RS064" состоит из кода функции "0768", кода синтаксиса "RS" кода проверки данных "064". Это код классификации для системы, которая может проверять правильность базовых документов SGML без опций функций или проверки данных.

Согласно рисунку, имя классификации "основной, подтверждающий правильность". Если бы код проверки данных был "000", суффикс проверки данных изменился бы, и имя классификации было бы

"основной, не подтверждающий правильность".

Поскольку функции, конкретный синтаксис и опции проверки данных соответствующей системы могут измениться независимо друг от друга, невозможно определить единственную иерархию классификации соответствия. Вместо этого имеются две иерархии, одна, основанная на коде функции, а другая на коде проверки данных, как показано в рисунке.

G.1.1 Код функции

Код функции определяется суммированием коэффициентов, назначенных в следующем списке каждой поддерживаемой функции:

Функция	***Вес***
FORMAL	1
CONCUR	2
EXPLICIT	4
IMPLICIT	8
SIMPLE	16
SUBDOC	32
RANK	64
DATATAG	128
OMITTAG	256
SHORTTAG	512

При необходимости код дополняется старшими нулями, чтобы сделать его той же длины, что и максимальный возможный код.

Номера функций являются взвешенными (они представляют собой степени двух), чтобы код функции уникально идентифицировал поддерживаемые функции. Например, код функции 0384 означает, что поддерживаются только функции DATATAG и OMITTAG.

ПРИМЕЧАНИЕ. Описанный метод работает, когда все функции определены независимо, и каждая из них может поддерживаться или не

поддерживаться; т.е. имеются два возможных значения для каждой функции, 0 (не поддерживается) и 1 (поддерживается). В общем случае существуют N возможных значений, V, от V=0 до V = N-1. Для каждой функции имеется коэффициент F, который является произведением N предыдущей функции и G (G = 1 для первой функции). Вес для данного значения, W (V), является произведением V и G. (Суммируя: F=N*G и W (V) =V*G, где звездочка означает умножение.)

ИМЯ КЛАССА СООТВЕТСТВИЯ	КОД ФУНКЦИИ: F	КОД СИНТАК-СИСА:S	КОД ПОДТВЕРЖД. ПРАВИЛЬН.:V	СУФФИКС ПОДТВЕРЖД. ПРАВИЛЬН.
Минимальный	0000	CS	000	не
Минимальный с опциями	0001-0767	CS	000	подтвержд.
Базовый	0768	RS	000	
Базовый с опциями	0769-1023	RS	000	
Полный	1024	RS	000	
Многокодовый минимальный	0000	MC	000	
Многокодовый минимальный с опциями	0001-0767	MC	000	
Многокодовый базовый	0768	MB	000	
Многокодовый базовый с опциями	0769-1023	MB	000	
Многокодовый полный	1024	MB	000	
Минимальный	0000	CS	064	подтвержд.
Минимальный с опциями	0001-0767	CS	064	
Базовый	0768	RS	064	
Базовый с опциями	0769-1023	RS	064	
Полный	1024	RS	064	
Многокодовый минимальный	0000	MC	064	
Многокодовый минимальный с опциями	0001-0767	MC	064	
Многокодовый базовый	0768	MB	064	
Многокодовый базовый с опциями	0769-1023	MB	064	
Многокодовый полный	1024	MB	064	
Минимальный	0000	CS	065 - 127	подтвержд.
Минимальный с опциями	0001-0767	CS	065 - 127	с опциями
Базовый	0768	RS	065 - 127	
Базовый с опциями	0769-1023	RS	065 - 127	
Полный	1024	RS	065 - 127	
Многокодовый минимальный	0000	MC	065 - 127	
Многокодовый минимальный с опциями	0001-0767	MC	065 - 127	
Многокодовый базовый	0768	MB	065 - 127	
Многокодовый базовый с опциями	0769-1023	MB	065 - 127	
Многокодовый полный	1024	MB	065 - 127	

Рис. 15. Классификация соответствия FSV

G.1.2 Код подтверждения правильности

Код подтверждения правильности определяется путем суммирования коэффициентов, назначенных в следующем списке каждой поддерживаемой функции подтверждения:

Функция	***Вес***
CAPACITY	1
EXCLUDE	2
SGML	4
MODEL	8
FORMAL	16
NONSGML	32
GENERAL	64

При необходимости код дополняется старшими нулями, чтобы сделать его той же длины, что и максимальный возможный код.

Система, которая делает только общую проверку правильности, будет иметь код подтверждения 064. Поскольку этот международный стандарт требует общей проверки правильности в качестве предварительного условия для любой опции проверки правильности, единственно возможными кодами подтверждения могут быть 000, и 064-127.

ПРИМЕЧАНИЕ. При модификации этой схемы GENERAL должен иметь наибольший вес, и никакой код подтверждения не может быть меньше веса GENERAL, если только он не 0.

G.1.3 Код синтаксиса

Коды для конкретного синтаксиса приведены в следующем списке:

Код	***Конкретный синтаксис***
CS	Конкретный синтаксис ядра
RS	Конкретный синтаксис ссылок
MC	Многокодовый конкретный синтаксис ядра
MB	Многокодовый базовый конкретный синтаксис

Перечисленные синтаксисы определены в этом документе.

Схема классификации предполагает, что проверка соответствия будет проводиться для одиночного конкретного синтаксиса, используемого как в прологе, так и в элементе документа.

G.2 Соображения по сертификации

Важно заметить, что схема классификации соответствия не ограничивает ни агентства по сертификации, ни разработчиков. Соответствие определено в 15 этот международного стандарта и формально выражается в объявлении системы, для которого код классификации соответствия является только неофициальным частичным резюме.

Разработчик может предлагать так много функций и вариантов конкретного синтаксиса, как он считает необходимым для его пользователей. Агентство сертификации может предлагать тестирование для столь узкой или широкой их группы, как оно пожелает. В результате, данная сертификация может включать все функции, которые заявляет тестируемая система, но это не подразумевает, что система не смогла пройти тест на соответствие этому международному стандарту. Это просто означает, что в отношении некоторых функций системы агентство сертификации не высказывает мнения относительно их соответствия стандарту.

Агентства сертификации должны быть свободны в изменении предложенной схемы классификации для выполнения своих требований, или в полном ее игнорировании. Однако, агентство, независимо от схемы классификации, которой оно придерживается, должно признавать, что оно не может объявлять систему соответствующей или несоответствующий на основании каких-либо критериев, кроме тех, которые определены в этом международном стандарте. В частности, оно

не может считать систему несоответствующей исключительно потому, что сочетание ее функций не соответствует схеме классификации агентства. Агентство должно проводить сертификацию (или не проводить в соответствии со схемой тестирования) относительно тех функций, которые оно желает тестировать, и объявить, что оно не имеет мнения относительно других.

ПРИЛОЖЕНИЕ H

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МОДЕЛИ СОДЕРЖАНИЯ SGML

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Нотация *группы модели* SGML была преднамеренно разработана таким образом, чтобы напоминать нотацию регулярных выражений теории автоматов, поскольку теория автоматов обеспечивает теоретическую основу для некоторых аспектов понятия соответствия *модели содержания.* Это приложение определяет отношения между нотацией *группы модели* и регулярными выражениями и обсуждает некоторые ситуации, в которых группы модели ведут себя подобно автоматам, и некоторым ситуации, в которых их поведение отлично.

H.1 Нотация группы модели

Нотация *группы модели* SGML соответствует нотации регулярного выражения теории автоматов или приводятся к ней следующим образом:

a) группы and приводятся к группе or перестановок групп seq; например:

(a & b)

эквивалентно регулярному выражению (или группе модели SGML):

((a, b) | (b, a))

b) маркер с индикатором появления opt сводится к такому же маркеру в группе or с нулевым маркером; например:

(a?)

эквивалентно регулярному выражению:

(a |)

которое, кстати, не является действительной группой модели SGML).

H.2 Применение теории автоматов

Проверка соответствия *модели содержания* по существу эквивалентна проблеме признания (принятие или отклонение) регулярных выражений, и поэтому регулярные выражения обеспечивают полезную теоретическую основу для некоторых аспектов контекстной проверки SGML.

Можно показать (с помощью теоремы Kleene), что регулярное выражение эквивалентно детерминированному конечному автомату (DFA). Поэтому, теоретически синтаксический анализатор может обработать любую *группу модели*, сводя ее к регулярному выражению и создавая DFA с путями переходов между состояниями, соответствующими маркерам *группы модели*. На практике, однако, возникают некоторые трудности.

Одна проблема возникает при приведении групп ***and***. Так как число требуемых перестановок групп ***seq*** является факториальной функцией, даже маленькая группа ***and*** может вести к недопустимо большому числу соответствующих групп ***seq***. Например, группа ***and*** с шестью членами, потребует 6! или 720 групп ***seq***.

Другая проблема заключается в конструкции DFA. Один метод заключается в том, чтобы сначала создать недетерминированный конечный автомат (NFA) непосредственно из регулярного выражения, а затем получить детерминированный эквивалент. Хотя этот и другие алгоритмы для построения DFA не полиномиальны и едва ли тяжелы для удобочитаемых для человека моделей, предусмотренных SGML, они могут быть слишком ресурсоемки для некоторых реализаций.

Данный международный стандарт избегает этих проблем, устраняя необходимость построения DFA. Это делается путем запрета моделей, которые являются неоднозначными или требуют "просмотра правого контекста"; то есть *группа модели* ограничена таким образом, чтобы в любом данном контексте, элемент (или строка символов) в документе мог соответствовать одному и только одному примитивному маркеру содержания (см. 11.2.4.3). В действительности, допустимый набор регулярных выражений сводится к тем, чьи соответствующие NFA могут быть детерминировано прослежены, потому что либо:

a) от данного узла может исходить только одна последовательность дуг, которая содержит только единственную помеченную дугу, чья метка - метка соответствующего содержания; или

b) если больше чем одна такая последовательность дуг могут исходить из узла, только одна из них может входить в него, и этой последовательности дается приоритет правилами SGML.

В результате, контекстная проверка может быть проведена упрощенными алгоритмами, которые используют только NFA.

Ограничение, которое может быть в принципе оправдано, поскольку люди также должны выполнять контекстную проверку при создании документа, и они с большей вероятностью будут делать это успешно, если регулярные выражения остаются простыми. Ограничение также может быть оправдано с практической точки зрения, потому что, несмотря на ограничение, всегда может быть получена желательная структура документа путем ввода промежуточных элементов.

H.3 Расхождение с теорией автоматов

Не следует делать каких-либо предположений относительно общей применимости теории автоматов к моделям содержания. Например, следует отметить, что модели содержания, которые приводятся к

эквивалентным регулярным выражениям, не обязательно эквивалентны для других целей. В частности, определение того, может ли начальный тег технически быть опущенным, зависит от точной формы выражения модели содержания. Минимизация разрешалась бы для "b" в

(a?, b)

тогда как она бы не разрешалась в

((a, b) | b)

даже несмотря на то, что обе модели приводятся к эквивалентным регулярным выражениям.

ПРИЛОЖЕНИЕ I

НЕСООТВЕТСТВУЮЩИЕ ОТКЛОНЕНИЯ

(Настоящее приложение не является неотъемлемой частью данного международного стандарта)

Исторически, многие преимущества обобщенной разметки были получены в результате использования языка процедур обработки текста, как если бы это был описательный язык. Этот подход обязательно менее эффективен, чем использование Стандартного Обобщенного Языка Разметки, который был разработан специально с целью описания документов.

В этом приложении рассматриваются некоторые общие отклонения от данного международного стандарта, которые возникают, когда процедурные языки используются для описательной разметки.

I.1 Родовые идентификаторы фиксированной длины

В языке с GI фиксированной длины не поддерживаются атрибуты и не имеется ***tagc***. Содержание начинается с первого символа после начального тега GI .

В следующем примере используются 2-символьные GI:

<раЭто параграф с

<sqкраткой цитатой< /sqв нем.</ра

Некоторые недостатки GI фиксированной длины:

— документ - источник может быть труден для чтения человеком (хотя машины не имеют каких-либо неприятностей);

— некоторые "естественные" мнемоники тега будут непригодны в сочетаниях из-за их длины (например, "p" для параграфа и "h1" для заголовка первого уровня не могут использоваться вместе);

— типы документа, которые имеют атрибуты, не могут быть обработаны.

Документ, созданный с GI фиксированной длины, может быть преобразован в документ SGML (при условии, что не существуют никакие другие различия) путем вставки разграничителя ***tagc***.

I.2 Одиночный разграничитель

В языке с единственным разграничителем один символ используется для ролей разграничителя ***stago***, ***etago***, ***ero, mdo*** и ***pio***.

Этот подход накладывает некоторые ограничения:

— GI, имена объектов, имена объявлений, имена команд обработки и имена макрокоманд все должны отличаться от друг друга, так как нет никакого другого способа различить их;

— конечные теги могут вводиться, только когда контекст не создает возможность смешения их с начальными тегами, поскольку они выглядят одинаково.

Некоторые недостатки использования единственного разграничителя:

— структура документа больше не прозрачна для читателя, поскольку описательные теги не могут легко различаться от команд обработки, объявлений разметки или ссылок на объект;

— команды обработки, которые применяются только к единственному приложению и системе, не могут легко обнаруживаться для изменения, если документ используется в иных приложении или системе;

— разработчик типа документа должен знать имя каждой команды обработки и макрокоманды, и не может использовать те же имена для GI. Точно так же программист текста не может создавать новую макрокоманду без того, чтобы убедиться, что не

существует GI с тем же именем;

— обмен документами строго ограничен из-за трудности в предотвращении конфликта GI документа с именами макрокоманд получающей системы.

Очевидное решение этих проблем состоит в том, чтобы использовать соглашения об именах для различения типов имен. Например, имена объявлений могут начинаться с восклицательного знака, имена команд обработки с вопросительного знака и т.д.

<!ENTITY abc SYSTEM>

<?NEWPAGE>

Конкретный синтаксис ссылок SGML использует в качестве разграничителей многосимвольные строки, таким образом создавая визуальный эффект единственного разграничителя с соглашениями по наименованию, но без каких-либо недостатков.