

Специфицирована только первая фаза сканера: разбиение исходного текста на лексемы. Преобразование лексем в токены не рассматривается.

Описание выполнено с применением РБНФ.

Использованы следующие расширения РБНФ:

1. При описании классов элементарных терминальных символов используется метасимвол `".."` для обозначения очевидных альтернатив. Очевидность этих альтернатив следует из упорядоченности элементарных терминальных символов.
2. Для обозначения отдельного элементарного символа по его коду используется нотация, принятая в языке Оберон. Например: `20X` — означает символ пробела.
3. Терминальные символы могут быть заключены как в двойные, так и в одинарные кавычки.
4. Метасимволы «левая кавычка» (`"` и `'`) и «правая кавычка» (`"` и `'`) не различаются.

Лексемы

```
lexeme = (ident | number | string | "(" | ")" | ":@" | "**") {blank | comment}.
```

```
ident = letter {letter | digit}.           // идентификатор
number = digit {digit}.                   // число
string = ''' {char} ''' | '"' {char} '".   // строковая константа
```

Разделители лексем

```
blank = 09X | 0DX | 20X.                 // «белый» символ
comment = "(" {sym} ")".                 // комментарий
```

Классы элементарных терминальных символов

```
letter = "A" | .. | "Z".                 // буква
digit = "0" | .. | "9".                  // цифра
char = 20X | .. | 0FEX.                  // печатный символ
sym = 0X | .. | 0FFX.                    // любой элементарный символ

ignor = 0X | .. | 08X | 0BX | 0CX | 0EX | .. | 1FX | 0FFX.
                                           // игнорируемые элементарные символы
```

Класс лексемы	Нетерминал	First()	Follow()
Идентификатор	ident	{ <code>"A".."Z"</code> }	{ <code>"0".."9"</code> , <code>"A".."Z"</code> }
Число	number	{ <code>"0".."9"</code> }	{ <code>"0".."9"</code> }
Строковая константа 1	string	{ <code>'''</code> }	{ <code>20X</code> , <code>21X</code> , <code>23X</code> .. <code>0FEX</code> }
Строковая константа 2	string	{ <code>'"'</code> }	{ <code>20X</code> .. <code>26X</code> , <code>28X</code> .. <code>0FEX</code> }
Левая скобка	—	{ <code>"("</code> }	{}
Правая скобка	—	{ <code>)"</code> }	{}
Знак присваивания	—	{ <code>:@"</code> }	{ <code>=</code> }
Звёздочка	—	{ <code>**"</code> }	{}
—	comment	{ <code>"("</code> }	{ <code>0X</code> .. <code>0FFX</code> }

Комментарии не являются лексемами. В таблице последняя строка приведена только для того, чтобы подчеркнуть, что сканер обязан распознавать также и комментарии, а не только лексемы. Если бы комментарии были лексемами, то «левая скобка» и «комментарий» относились бы к одному классу лексем. В начале разбора сканер так и считает.

На вход сканера подаётся поток элементарных символов, обработанный драйвером исходного текста. Драйвер исходного текста выполняет следующую обработку:

- Все концы строк, записанные в нотации Unix и Windows, преобразует к символу `0DX` (конец строки в нотации Macintosh).
- Удаляет из исходной последовательности все игнорируемые символы.