

Лексика языка Си

Исходный файл с программой на языке Си представляет собой последовательность символов, разбитую на строки. Символы должны быть из алфавита языка Си.

Алфавит включает:

1. Латинские буквы: A-Z, a-z
2. Цифры: 0-9
3. Символ пробела
4. Управляющие символы: табуляции, перевод страницы, новая строка, возврат каретки
5. 29 графических символов, таких как + − , “ / ? и др.

Пробельные символы: символы из пунктов 3 и 4

Комментарий: /* это пример комментария */

Комментарии не могут вкладываться друг в друга.

Пробельные символы и комментарии разделяют *лексемы*. Последовательность из комментариев или пробельных символов равносильна одному пробелу.

Лексема — это минимальная смысловая единица языка. К лексемам относятся:

- операции;
- разделители;
- идентификаторы;
- ключевые слова;
- константы.

Операции и разделители

Простые операции: ! % ^ & * − + = ~ | . < > / ?

Составные операции присваивания: += -= *= /= %= <<= >>= &= ^= |=

Другие составные операции: --> ++ -- << >> <= >= == != && ||

Разделители: () [] { } , ; : ...

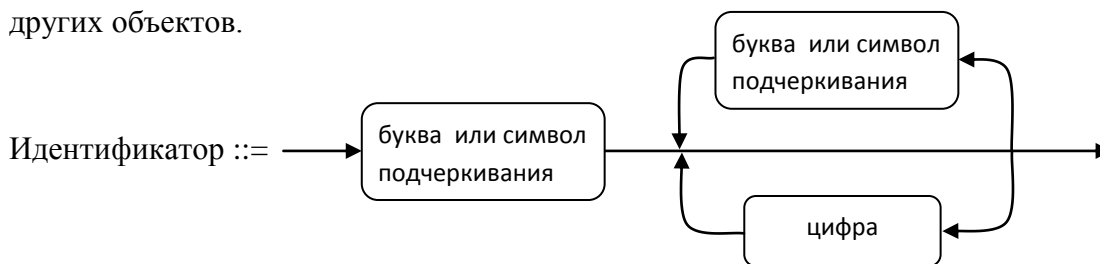
Ключевые слова

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Имена

Имя — это идентификатор, не совпадающий по написанию с ключевым словом.

Имена используются в программе для обозначения переменных, меток, функций, типов и других объектов.



буква ::= a | b | c | ... z | A | B | ... | Z

Примеры описаний переменных:

```
int i ;          /* переменная целого типа */
```

```
long l ;         /* переменная “длинного” целого типа */
```

```
char c;          /* символ */
```

```
double f ;       /* вещественного типа */
```

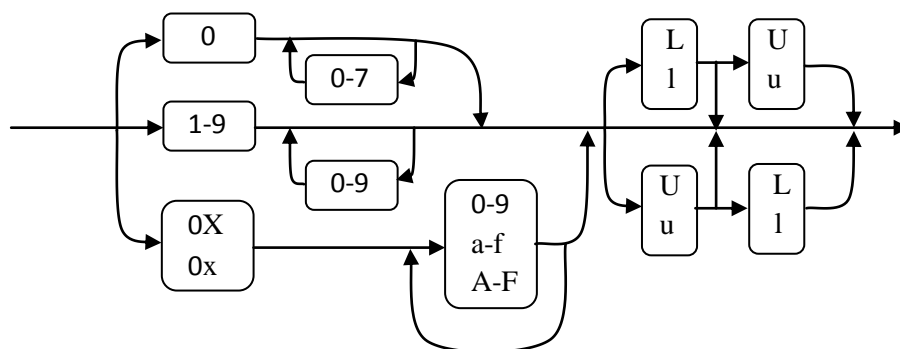
```
unsigned int u;  /* беззнаковое целое */
```

```
long double ld; /* длинное вещественное */
```

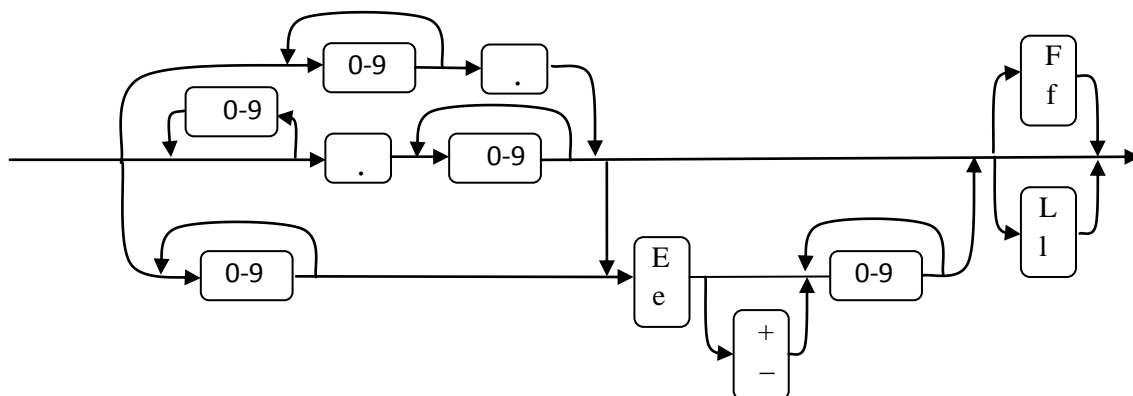
Константы

Служат для изображения значений некоторых типов в программе.

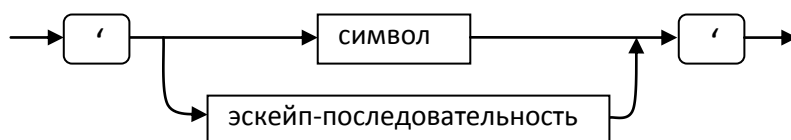
целая константа ::=



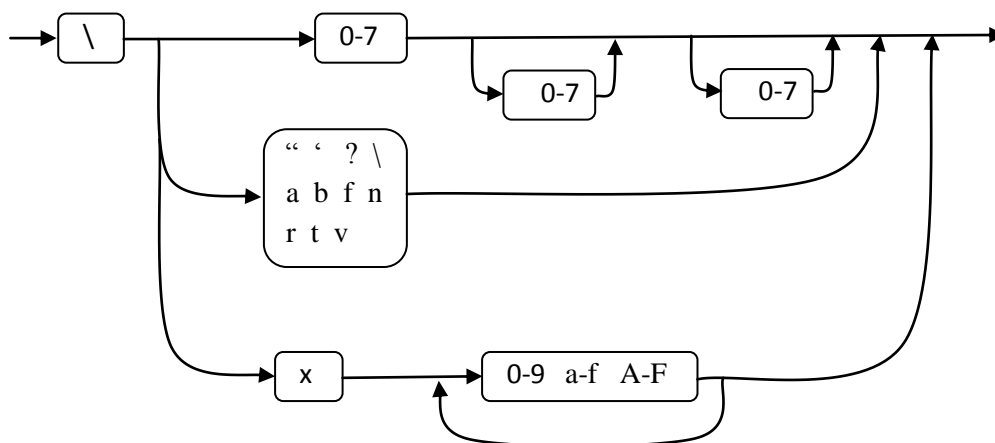
вещественная константа ::=



символьная константа ::=



эскейп-последовательность ::=



Символ – из набора символов на данной машине.

Примеры: `'\x30'` означает `'0'`; `'\007'` означает `'\b'`.

Значение некоторых эскейп-последовательностей

<code>'\n'</code>	новая строка	<code>'\a'</code>	сигнал звонок
<code>'\r'</code>	возврат каретки	<code>'\b'</code>	возврат на шаг
<code>'\t'</code>	табуляция	<code>'\f'</code>	перевод страницы
<code>'\v'</code>	вертикальная табуляция		

Тип константы определяется по ее виду. Символьная константа имеет тип `int`. Ее значением является код символа.

Целая константа без суффикса имеет тип `int` (или тип `long`, если она столь велика, что «не умещается» в тип `int`). Целая константа с суффиксом `u` или `U` имеет тип `unsigned`, с суффиксом `l` или `L` — тип `long`, с обоими суффиксами — тип `unsigned long`.

Вещественная константа без суффикса имеет тип `double`, с суффиксом `f` или `F` — тип `float`, с суффиксом `l` или `L` — тип `long double`.

Примеры констант

56 — целая константа (`int`), 6.7 — вещественная константа (`double`), '0' означает целое число, равное 48, если в реализации используется ASCII кодировка, '\0' означает целое число ноль, это так называемая null-литера. Все литерные константы имеют целый тип `int`.

Строковые константы

Примеры:

"это строка\n"

"" /* пустая строка */

Внутреннее представление в виде массива:

э	т	о		с	т	р	о	к	а	\n	\0
---	---	---	--	---	---	---	---	---	---	----	----

\0

Строки

`char str1[] = "можно изменять эту строку"; /* str1 — массив из 26 символов */`

`const char str2[] = "изменять нельзя"; /* const — квалификатор типа */`

Еще о комментариях

В стандарте C99 введен еще один вид комментария как в C++ : две косые черты начинают комментарий и он оканчивается концом строки:

`// это комментарий в стиле C++`

Рекомендуется не смешивать комментарии разных стилей. Будем использовать комментарии в стиле C89.

Для игнорирования фрагмента кода лучше использовать не комментарий, а директивы препроцессора

```
#if 0
...
#endif
```

этим фрагмент игнорируется компилятором