Для объявления переменной в языке С необходимо написать:

тип идентификатор;

Например:

```
int a;
double d;
```

Можно объявить сразу несколько переменных, перечислив их через запятую:

```
int first, second;
```

Переменной можно дать начальное значение при объявлении:

```
int a = 0;
```

Стандартом языка С определены следующие типы:

Тип	Типичный размер в битах	Минимально допустимый диапазон значений
char	8	от -128 до 127
unsigned char	8	от 0 до 255
signed char	8	от -128 до 127
int	16 или 32	от -32768 до 32767
unsigned int	16 или 32	от 0 до 65535
signed int	16 или 32	то же, что и int
short int	16	от -32768 до 32767
unsigned short int	16	от 0 до 65535
signed short int	16	то же, что и short int
long int	32	от -2 147 483 648 до 2 147 483 647
long long int	64	от -2^{63} до $2^{63} - 1$, добавлен стандартом С99
signed long int	32	то же, что и long int
unsigned long int	32	от 0 до 4 294 967 295
unsigned long long int	64	от 0 до $2^{64} - 1$, добавлен в С99
float	32	точность не менее 6 значащих десятичных цифр
double	64	точность не менее 10 значащих десятичных цифр
long double	80	точность не менее 10 значащих десятичных цифр

К ряду типов могут применяться модификаторы:

char может быть **unsigned** (беззнаковый) или **signed** (знаковый, по умолчанию).

int может быть **unsigned** (беззнаковый) или **signed** (знаковый, по умолчанию).
int также может быть **short** (16 бит) или **long** (32 бита).

Тип **double** может быть **long** (80 бит).

Например:

unsigned char ch;

unsigned long l;

Операторы языка С имеют следующие приоритеты:

Наивысший
( ) [ ] -> .
! ~ ++ -- - (type) * & sizeof
* / %
+ -
<< >>
< <= > >=
== !=
&
^
|
&&
||
?:
= += -= *= /= и т.д.
,
Наинизший

Условный оператор if имеет следующий синтаксис:

```
if (условие)
  оператор;
else
  оператор;
```

Ветка else исполняется в случае невыполнения условия и может быть опущена:

```
if (условие)
  оператор;
```

В случае, если необходимо выполнить несколько операторов, то они берутся в фигурные скобки:

```
if (условие) {
  операторы;
}
else {
  операторы;
}
```

Условие может состоять из нескольких подусловий. Между ними применяются специальные операторы || (или), && (и), ! (отрицание). В языке С истинным считается любое ненулевое значение. Для сравнения значений применяется оператор ==. Для повышения приоритета при проверке часть условий может быть взята в скобки.

Одновременно a = 5 и b = 6:
if (a == 5 && b == 6)

Или a < 10, или же a > 100:
if (a < 10 || a > 100)

Отрицание предыдущего условия:
if (!(a < 10 || a > 100))

Проверка на то, что a не равно нулю:
if (a)

Оператор **switch** позволяет последовательно сравнить целочисленную переменную с целочисленными константами и выполнить определенные действия в случае, если она совпадает с одной из констант:

```
switch (a) {
   case 1:
      /*операторы для случая a равно 1*/
      break;
 case 2:
      /*операторы для случая a равно 2*/
      break;
 case 10:
      /*операторы для случая a равно 10*/
      break;
  default:
      /*операторы для случая a не равно ни одной из констант.
         эта ветка может быть опущена*/
      break;
}
```

После каждой из веток обычно присутствует оператор **break**, завершающий работу **switch**. В противном случае исполнение будет продолжаться пока не встретится такой оператор (**break**) или же не закончится весь **switch**.

Оператор цикла **while** позволяет выполнять указанные операторы до тех пор, пока верно условие:

```
while (условие)
   оператор;


while (условие) {
  операторы
}
```

Например:

```
int val = 10, summa = 0;
while (val > 0) {
  summa = summa + val;
  val--;
}
```

Условие формируется по тем же правилам, что и в операторе **if**.

Оператор **do-while** сначала делает команды и только затем проверяет условие. Поэтому блок команд всегда выполнится хотя бы один раз:

```
do {
    операторы;
}
while(условие);
```

Цикл **for** имеет следующий вид:

```
for (инициализация; условие; приращение)  {
   операторы;
}
```

Перед началом выполнения цикла исполняется секция инициализации, которая может состоять из одной или нескольких команд, разделенных запятыми. Перед каждым витком цикла проверяется условие и, если оно верно, начинают исполняться операторы. После каждого витка исполняется секция приращения, которая может изменять управляющие переменные. Она также состоит из одного или нескольких операторов, разделенных запятыми. Любая из секций (или даже все секции) может быть опущена, однако всегда должно быть две точки с запятой.

Пример:

```
int i, summa;
for (i = 0, summa = 0; i < 10; i++)
  summa = summa + i;
```

В любом из циклов может быть использован оператор **break**, который прекращает исполнение цикла и переходит к следующему за циклом оператору.

В любом из циклов может быть использован оператор **continue**, который переходит к следующему витку цикла. Для циклов **while** и **do-while** это проверка условия. Для цикла **for** это сначала исполнение секции приращения и затем проверка условия.