Оператор ветвления

if (условие) оператор_1; else оператор_2;

В операторе if слово else может отсутствовать.

В операторе ветвления после ключевых слов if и else должны следовать операторы. Если хотя бы один из них является оператором if, его называют вложенным. Согласно принятому в языке Си соглашению слово else всегда относится к ближайшему предшествующему ему if.

Оператор варианта

```
switch (выражение)
{
   case константа_1:  операторы_1; break;
   case константа_2:  операторы_2; break;
   ........           ........
   default:           операторы_default;
}
```

Оператор варианта

вычисляется значение целого выражения и сравнивается со всеми константами. При совпадении выполнится соответствующий вариант операторов. Вариант default реализуется, если ни один другой не подошел (слово default может и отсутствовать). Если default отсутствует, а все результаты сравнения отрицательны, то ни один вариант не выполняется.

Оператор варианта

Для прекращения проверок после успешного выбора варианта используется оператор break, обеспечивающий немедленный выход из переключателя switch.

Допускаются вложенные конструкции switch.

Циклы с условием

while (выражение) тело_цикла

do {тело_цикла} while (выражение);

Выражение может принимать ненулевое (истинное) или нулевое (ложное) значение. Если оно истинно, то выполняется тело цикла и выражение вычисляется снова. Если выражение ложно, то цикл while заканчивается.

Оператор безусловного перехода

goto метка;

Метка - это любой идентификатор, после которого поставлено двоеточие. Ее не надо объявлять.

Цикл for

for (выражение1; выражение2; выражение3)
тело_цикла

Выражение1 присваивает начальное значение управляющей переменной, выражение3 изменяет его на каждом шаге, Если выражение 2 истинно, цикл продолжается.

Пример:

```
for (i = 1; i < 10; i++)
{   ...
}
```

Цикл for

for (выражение1; выражение2; выражение3)
тело_цикла

Выражение1 присваивает начальное значение управляющей переменной, выражение3 изменяет его на каждом шаге, Если выражение 2 истинно, цикл продолжается.

Пример:

```
for (i = 1; i < 10; i++)
{   ...
}
```

Цикл for

Любое из трех выражений в цикле for может отсутствовать, однако точка с запятой должна оставаться. Таким образом,

for (; ;) {...}

это бесконечный цикл

```
for (ch = 'a'; ch != 'p';) scanf ("%c", &ch);
/* Цикл будет выполняться до тех пор, пока с клавиатуры не будет введен символ 'p' */
```

Ввод-вывод

scanf("управляющая строка", аргумент1, аргумент2,...);

Аргументы scanf( ) должны быть указателями на соответствующие значения. Для этого перед именем переменной записывается символ &.

```
scanf("%d", &a);
scanf("%c", &b);
scanf("%d%c%f",&a, &b, &t);
```

Символы управляющей строки scanf( ) (указываются после символа %):

c - на входе ожидается появление одиночного символа;

d или i - ожидается десятичное целое число и аргумент является указателем на переменную типа int;

D или l - ожидается десятичное целое число и аргумент является указателем на переменную типа long;

e или E - ожидается вещественное число с плавающей точкой;

f - ожидается вещественное число с плавающей точкой;

g или G - ожидается вещественное число с плавающей точкой;

s - ожидается строка символов;

u / U ожидается беззнаковое целое число и аргумент является указателем на unsigned int/unsigned long;

Ввод-вывод

getchar( ) считывает один символ с клавиатуры

Оператор вида:

x = getchar( );

присваивает переменной x очередной вводимый символ. Переменная x должна иметь символьный или целый тип.

putchar(x) выдает значение переменной x в стандартный выходной поток (на экран).

Ввод-вывод

Функция *getchar( )* после вввода символа ожидает нажатия <Enter>.

Функции *getch( )* и *getche( )* вводят символ сразу же после нажатия соответствующей клавиши . Отличие между ними в том, что *getche( )* отображает вводимый символ на экране дисплея, а *getch( )* - нет.

Функции описаны в файле conio.h (консольный ввод/вывод).

Знак операции	Назначение операции
!	Логическое отрицание
~	Поразрядное отрицание
-	Изменение знака
++	Увеличение на единицу
--	Уменьшение на единицу
&	Взятие адреса
*	Обращение по адресу
(тип)	Преобразование типа (т.е. (float) a)
*	Умножение
/	Деление
%	Определение остатка от деления
+	Сложение
-	Вычитание

<<	Сдвиг влево
>>	Сдвиг вправо
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
= =	Равно
!=	Не равно
&	Поразрядное логическое "И"
^	Поразрядное исключающее "ИЛИ"
|	Поразрядное логическое "ИЛИ"
&&	Логическое "И"
||	Логическое "ИЛИ"
=	Присваивание
+=, - =, *=, /=	Составные операции присваивания (например, a *= b (т.е. a = a * b) и т.д.)

Операция присваивания (=).

Выражение вида *x = y;*
присваивает переменной x значение переменной y.

Операцию "=" можно использовать многократно в одном выражении:

```
x = y = z = 100;
```

Операция присваивания (=).

Выражение с операцией присваивания, заключенное в круглые скобки, возвращает значение, равное присваиваемому. Например, выражение (a=7+2) имеет значение 9. После этого можно записать другое выражение, например: ((a=7+2)<10), которое в данном случае будет всегда давать истинное значение

Операция присваивания (=).

Следующая конструкция:

```
((ch = getch( )) == 'i')
```

позволяет вводить значение переменной ch и давать истинный результат только тогда, когда введенным значением является символ 'i'.