

Условные операторы в Си

Рассмотрим подробнее **структуру алгоритма** «развилка».

Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется **ветвью алгоритма**.

Признаком разветвляющегося алгоритма является наличие операций проверки условия. Чаще всего для проверки условия используется условный оператор **if**.

Условный оператор if

Условный оператор **if** может использоваться в форме **полной** или **неполной** развилки.

Неполная развилка	Полная развилка
<pre>1 if (Условие) 2 { 3 БлокОпераций1; 4 }</pre>	<pre>1 if (Условие) 2 { 3 БлокОпераций1; 4 } 5 else 6 { 7 БлокОпераций2; 8 }</pre>
<pre>graph TD Entry(()) --> Cond{Условие?} Cond -- да --> Block[Блок операций] Cond -- нет --> Join(()) Block --> Join Join --> Exit(())</pre>	<pre>graph TD Entry(()) --> Cond{Условие?} Cond -- да --> Block1[Блок операций 1] Cond -- нет --> Block2[Блок операций 2] Block1 --> Join(()) Block2 --> Join Join --> Exit(())</pre>

В случае неполной развилки если **Условие** истинно, то **БлокОпераций1** выполняется, если **Условие** ложно, то **БлокОпераций1** не выполняется.

В случае полной развилки если **Условие** истинно, то выполняется **БлокОпераций1**, иначе выполняется **БлокОпераций2**.

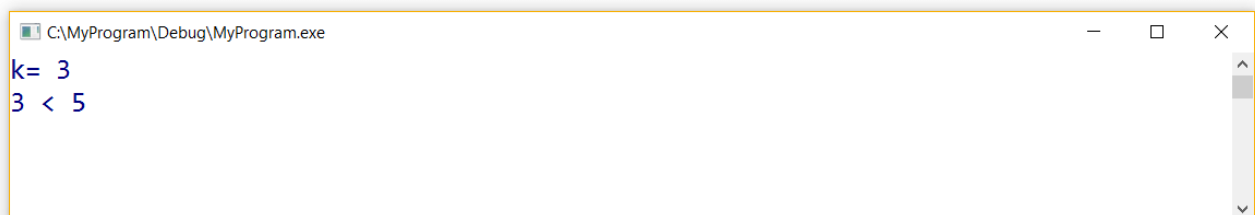
БлокОпераций может состоять из одной операции. В этом случае наличие фигурных скобок, ограничивающих блок, необязательно.

Основными операциями, проверяемыми внутри условного блока, являются **операции отношения**.

Пример на C:

```
1  #define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
2  #include <stdio.h>
3  int main()
4  {
5      int k;           // объявляем целую переменную k
6      printf("k= ");   // выводим сообщение
7      scanf("%d", &k); // вводим переменную k
8      if (k >= 5)       // если k>5
9          printf("%d >= 5", k); // выводим "ЗНАЧЕНИЕ >= 5"
10     else              // иначе
11         printf("%d < 5", k);  // выводим "ЗНАЧЕНИЕ < 5"
12     getchar(); getchar();
13     return 0;
14 }
```

Результат выполнения



```
C:\MyProgram\Debug\MyProgram.exe
k= 3
3 < 5
```

Оператор **if** может быть вложенным.

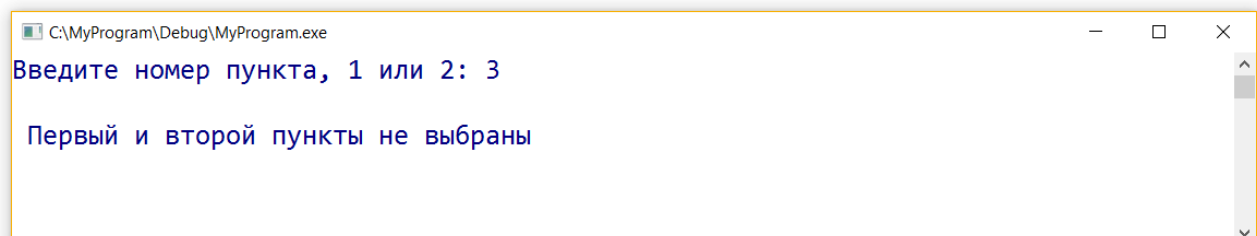
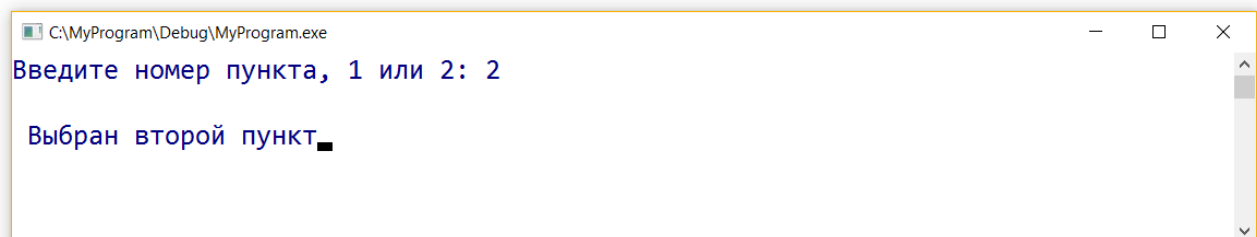
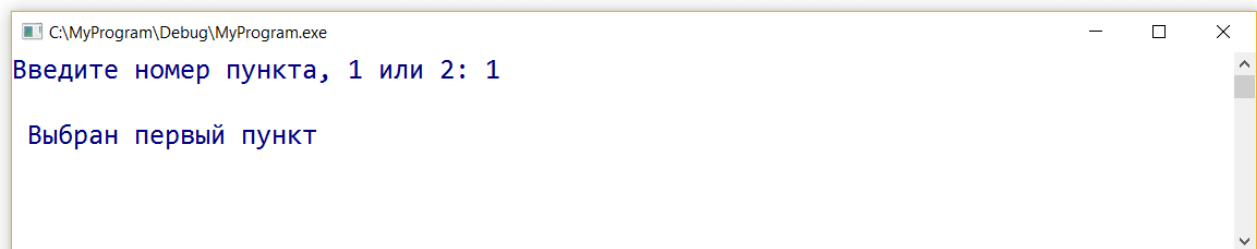
Пример на C:

```

4   int main() {
5       int key; // объявляем целую переменную key
6       system("chcp 1251"); // переходим в консоли на русский язык
7       system("cls"); // очищаем окно консоли
8       printf("Введите номер пункта, 1 или 2: ");
9       scanf("%d", &key); // вводим значение переменной key
10      if (key == 1) // если key = 1
11          printf("\n Выбран первый пункт"); // выводим сообщение
12      else if (key == 2) // иначе если key = 2
13          printf("\n Выбран второй пункт"); // выводим сообщение
14      else // иначе
15          printf("\n Первый и второй пункты не выбраны"); // выводим сообщ
16      getchar(); getchar();
17      return 0;
18  }

```

Результат выполнения



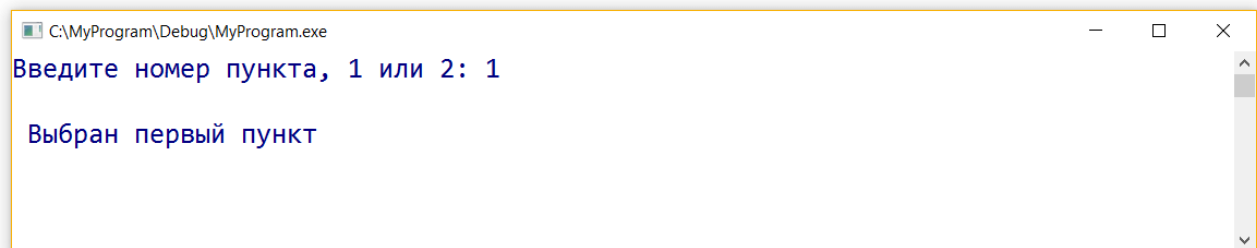
При использовании вложенной формы оператора `if` опция `else` связывается с последним оператором `if`. Если требуется связать опцию `else` с предыдущим оператором `if`, внутренний условный оператор заключается в фигурные скобки:

```

1  #define _CRT_SECURE_NO_WARNINGS // для возможности использования sc
2  #include <stdio.h>
3  #include <stdlib.h> // для использования функции system
4  int main() {
5      int key; // объявляем целую переменную key
6      system("chcp 1251"); // переходим в консоли на русский язык
7      system("cls"); // очищаем окно консоли
8      printf("Введите номер пункта, 1 или 2: ");
9      scanf("%d", &key); // вводим значение переменной key
10     if (key != 1) { // если key не равен 1
11         if (key == 2) // если key равен 2
12             printf("\n Выбран второй пункт"); // вывод сообщения
13     } // если key - не 1 и не 2, то ничего не выводится
14     else // иначе, если key равен 1
15         printf("\n Выбран первый пункт"); // вывод сообщения
16     getchar(); getchar();
17     return 0;
18 }

```

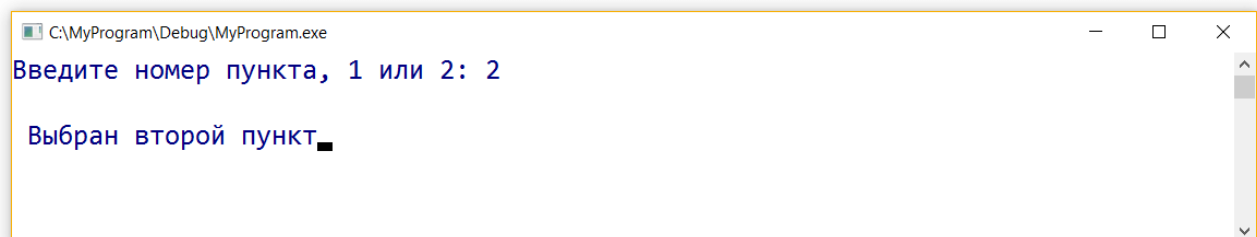
Результат выполнения



C:\MyProgram\Debug\MyProgram.exe

Введите номер пункта, 1 или 2: 1

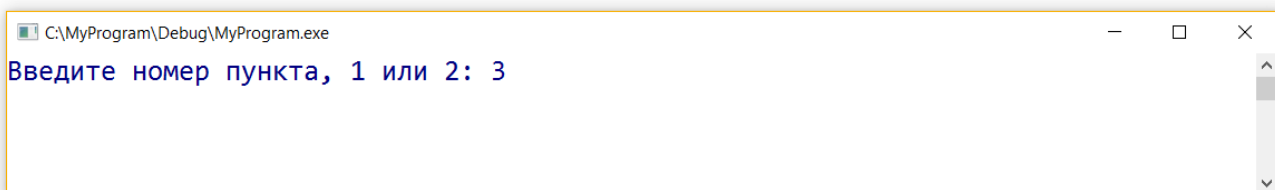
Выбран первый пункт



C:\MyProgram\Debug\MyProgram.exe

Введите номер пункта, 1 или 2: 2

Выбран второй пункт



C:\MyProgram\Debug\MyProgram.exe

Введите номер пункта, 1 или 2: 3

Логические операции в условных операторах

Условный оператор может проверять

- одновременное выполнение всех условий (операция И - &&)
- выполнение хотя бы одного из условий (операция ИЛИ - ||)
- выполнение только одного из условий (операция исключающее ИЛИ - ^)

Пример на Си: Найти максимум из 3 чисел

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  int main()
4  {
5      int a, b, c;
6      printf("a=");
7      scanf("%d", &a);
8      printf("b=");
9      scanf("%d", &b);
10     printf("c=");
11     scanf("%d", &c);
12     if ((a >= b) && (a >= c))
13         printf("Max = %d", a);
14     else if ((b >= a) && (b >= c))
15         printf("Max = %d", b);
16     else
17         printf("Max = %d", c);
18     getchar();
19     getchar();
20     return 0;
21 }
```

Пример на C++: Найти максимум из 3 чисел

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a, b, c;
6      cout << "a=";
7      cin >> a;
8      cout << "b=";
9      cin >> b;
10     cout << "c=";
11     cin >> c;
12     if ((a >= b) && (a >= c))
```

```

13     cout << "Max = " << a;
14     else if ((b >= a) && (b >= c))
15         cout << "Max = " << b;
16     else
17         cout << "Max = " << c;
18     cin.get();
19     cin.get();
20     return 0;
21 }

```

Тернарные операции

Тернарная условная операция имеет 3 аргумента и возвращает свой второй или третий операнд в зависимости от значения логического выражения, заданного первым операндом. Синтаксис тернарной операции в языке Си

```
Условие ? Выражение1 : Выражение2;
```

Если выполняется **Условие**, то тернарная операция возвращает **Выражение1**, в противном случае - **Выражение2**.

Тернарные операции, как и операции условия, могут быть вложенными. Для разделения вложенных операций используются круглые скобки.

Приведенный выше пример с использованием тернарных операций можно представить в виде

```

1  #define _CRT_SECURE_NO_WARNINGS // для возможности использования ss
2  #include <stdio.h>
3  #include <stdlib.h> // для использования функции system
4  int main() {
5      int key; // объявляем целую переменную key
6      system("chcp 1251"); // переходим в консоли на русский язык
7      system("cls"); // очищаем окно консоли
8      printf("Введите номер пункта, 1 или 2: ");
9      scanf("%d", &key); // вводим значение переменной key
10     key == 1 ? printf("\n Выбран первый пункт") :
11         (key == 2 ? printf("\n Выбран второй пункт") :
12             printf("\n Первый и второй пункты не выбраны"));
13     getchar(); getchar();
14     return 0;
15 }

```

Оператор ветвления `switch` (оператор множественного выбора)

Оператор `if` позволяет осуществить выбор только между двумя вариантами. Для того, чтобы производить выбор одного из нескольких вариантов необходимо использовать вложенный оператор `if`. С этой же целью можно использовать оператор ветвления `switch`.

Общая форма записи

```
switch (ЦелоеВыражение)
{
    case Константа1: БлокОпераций1;
        break;
    case Константа2: БлокОпераций2;
        break;
    . . .
    case Константаn: БлокОперацийn;
        break;
    default: БлокОперацийПоУмолчанию;
        break;
}
```

Оператор ветвления `switch` выполняется следующим образом:

- ❑ вычисляется **ЦелоеВыражение** в скобках оператора `switch`;
- ❑ полученное значение сравнивается с метками (**Константами**) в опциях `case`, сравнение производится до тех пор, пока не будет найдена метка, соответствующая вычисленному значению целочисленного выражения;
- ❑ выполняется **БлокОпераций** соответствующей метки `case`;
- ❑ если соответствующая метка не найдена, то выполнится **БлокОперацийПоУмолчанию**, описанный в опции `default`.