Лекция 3

# РАЗВЕТВЛЯЮЩИЕСЯ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ В VISUAL C++ 2010. УСЛОВНЫЕ ОПЕРАТОРЫ.

**Цель лекции:** изучение вопросов разработки разветвляющихся программ с использованием операторов условного и безусловного переходов, особенностей их конструкции и исполнения.

**Основные вопросы лекции**.

1. Разветвляющиеся вычислительные процессы.
2. Условный оператор **if…else**. Синтаксис и использование в программах.
3. Оператор вибору **switch**.
4. Безусловный оператор.

## 1. Разветвляющиеся вычислительные процессы

При решении практически любой задачи, в том числе и при помощи компьютера, приходится принимать те или иные решения. Поскольку вся информация в компьютере представлена в числовом виде, то сравнение числовых значений – это сущность механизма принятия решений в вычислительных процессах на языке С++.

Существует шесть базовых операторов для сравнения значений двух переменных:

| | |
|---|---|
| **< меньше** | **<= меньше или равно** |
| **> больше** | **>= больше или равно** |
| **== равно** | **!= не равно** |

Любой конкретный алгоритм может быть записан на языке программирования, использующем только три управляющий структуры: последовательное выполнение, ветвление и повторение.

Последовательность операторов выполняется в порядке их естественного расположения в программе, с возможным отклонением для вызова внешнего фрагмента (функции), но с обязательным возвратом в точку вызова.

## 2.Исследование условного оператора if…else

Любой алгоритм может быть записан на языке программирования с использованием только трех управляющих структур: последовательное выполнение, ветвление и повторение. Последовательное выполнение реализуется в линейных алгоритмах, исследованных в предыдущей лабораторной работе. В данной работе будут исследованы разветвленные алгоритмы, реализующие алгоритмы ветвления.

На рис. 1 приведен пример структуры ветвления. Операционный блок **Условие** состоит из выражения, содержащего логическое отношение, т. е. условие. Если условие выполняется, то реализуется **Действие 1** алгоритма, а в случае невыполнения - **Действие 2.**

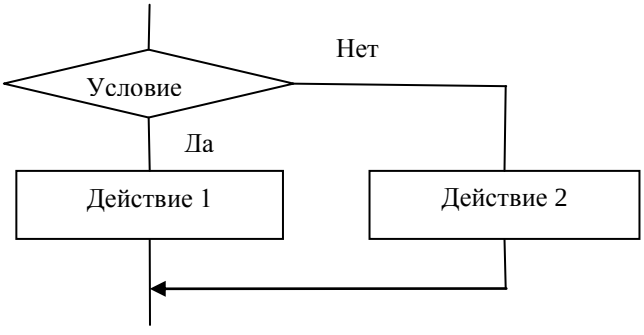


Рис. 1. Структура ветвления

Структура ветвления описывается в языке С++ с помощью **условного оператора**:

```
if (выражение)
  оператор_1;
else
  оператор_2;
```

где часть **else оператор_2;** может отсутствовать.

Вначале вычисляется **выражение** в скобках. Если оно истинно, то выполняется **оператор_1**. Если **выражение** ложно, то **оператор_1** пропускается и выполняется **оператор_2**. Вместо единичных операторов могут использоваться группы из нескольких операторов (сложные операторы). При этом они заключаются в фигурные скобки - **{ }.**

**Выражение** в скобках представляет собой условие, заданное с помощью операций отношений и логических операций.

**Сложные операторы.** К сложным операторам относят собственно сложные операторы и блоки. В обоих случаях это последовательность операторов, помещенная в фигурные скобки. Блок отличается от сложного оператора наличием **объявлений переменных** в теле блока. Например,

```
{                                          {
a=b+c;   // сложный оператор      и       int a,b,c,d;   // блок
d=a+b;                                     a=b+c;
}                                          d=a+b;
                                           }
```

Для лучшего понимания действия операторов в настоящей работе следует знать перевод следующих английских слов:

**if – если;**
**then – тогда;**
**else – иначе;**
**case – случай;**
**switch – переключатель;**
**break – прерывать;**
**default – не выполнять.**

**Пример 1.** Исследовать и выполнить программу для определения переменной **a** по следующему условию:

$$a = \begin{cases} b + c + d, & \text{если} \quad b > 10; \\ b - c - d, & \text{если} \quad b \le 10. \end{cases}$$

Блок-схема алгоритма решения задания 1 представлена на рис. 2.

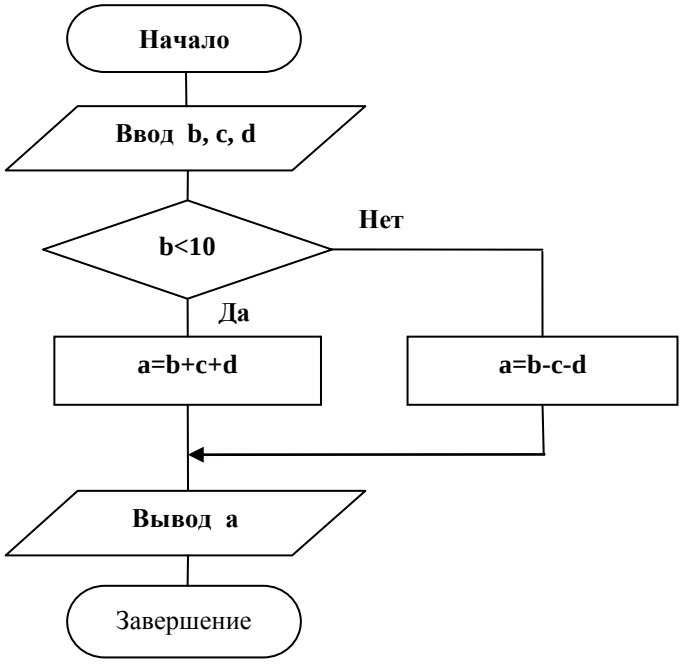


Рис. 2. Блок-схема алгоритма определения переменной **a** (простые операторы)

Программный код, реализующий алгоритм:

```
#include "stdafx.h"
#include <conio.h>                    //Файл conio.h обеспечивает задержку окна DOS на
                                      //экране дисплея
#include <iostream>                   //Директива include подключает файл ввода–вывода
using namespace std;                  //Подключает все имена из пространства имен  std

int _tmain(int argc, _TCHAR* argv[])  //Объявление главной функции _tmain
{                                     //Начало главной функции
int a, b;                             //Объявление переменных целого типа
int c=2, d=3;                         //Объявление переменных целого типа  и их
                                      //инициализация
        cout<<"Vvvedite b"<<endl;     //Вывод на экран надписи-приглашения  Vvedite b
        cin>>b;                       //Ввод значения переменной  b
        if (b<10)                     //Условный оператор if …else (1-я часть)
        a=b+c+d;                      //Вычисление переменной a при выполнении условия
        else                          //Условный оператор if …else (2-я часть)
        a=b-c-d;                      //Вычисление переменной a при невыполнении условия
        cout<<"a = "<<a;              //Вывод на экран значения переменной  a
getch();                              //Функция задержки окна DOS на экране
return 0;
}
```

**Пример 2.** Исследовать и выполнить программу для определения переменной **a** по условию **Примера 1**. Отличие в использовании сложных операторов в условном операторе **if…else**.

Блок-схема алгоритма решения данного задания представлена на рис. 3. Обратите внимание на отличия данной блок-схемы от схемы, приведенной на рис. 2. Объясните их.

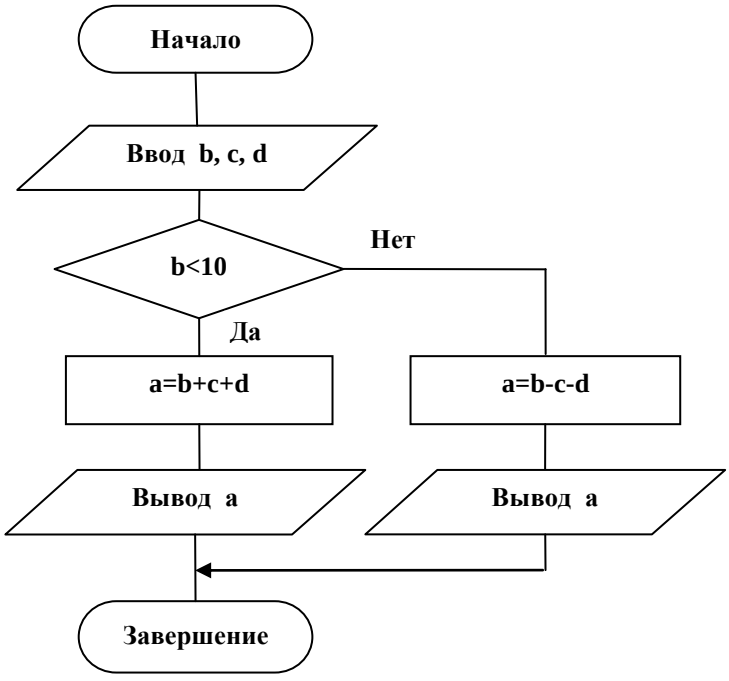


Рис. 3. Блок-схема алгоритма определения переменной **a** (сложные операторы)

Введите и выполните программный код, реализующий алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
#include "stdafx.h"
```

```
#include <conio.h>                  //Файл conio.h обеспечивает задержку окна DOS на
                                    //экране дисплея
#include <iostream>                 //Директива include подключает файл ввода–вывода
using namespace std;                //Подключает все имена из пространства имен  std

int _tmain(int argc, _TCHAR* argv[])  //Объявление главной функции _tmain
{                                   //Начало главной функции
int a, b;                           //Объявление переменных целого типа
int c=2, d=3;                       //Объявление переменных целого типа  и их
                                    //инициализация
        cout<<"Vvvedite b"<<endl;   //Вывод на экран надписи-приглашения  Vvedite b
        cin>>b;                     //Ввод значения переменной  b
        if (b<10)                   //Условный оператор if…else (1-я часть)
                {                   //Начало сложного оператора
                a=b+c+d;            //Вычисление переменной a при выполнении условия
                cout<<"a=b+c+d= "<<a;  //Вывод на экран значения переменной  a
                }                   //Конец сложного оператора
        else                        //Условный оператор if …else (2-я часть)
                {                   //Начало сложного оператора
                a=b-c-d;            //Вычисление переменной a при невыполнении условия
                cout<<"a=b-c-d= "<<a;  //Вывод на экран значения переменной  a
                }                   //Конец сложного оператора
getch();                            //Функция задержки окна DOS на экране
return 0;
}
```

Введите, отладьте и выполните данный программный код самостоятельно. Зафиксируйте результаты для b=8 и b=12. Объясните результаты.

### 3. Использование условного оператора if…else для 3-х интервалов значений переменой

Выше исследовались случаи применения условного оператора **if…else** для 2-х интервалов значений переменой, т. е. рассматривались простые условия (логические выражения) типа **x<a**. При этом число **a** делит числовую ось **x** на два интервала (рис. 4).

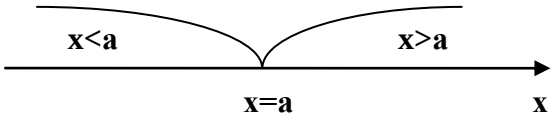


Рис. 4. Графическая интерпретация логического выражения для двух интервалов значений **x**

Язык C++ и среда Visual C++2010 позволяют в случае необходимости применять в условном операторе **if…else** и более сложные логические выражения с использованием различных логических операций.

Рассмотрим случай, когда  в условии оператора **if…else** указан некоторый интервал значений, с которым сравнивается переменная. Например, переменная **x** должна находиться в интервале значений от **a** до **b** (**a<b**) (рис. 5). В этом случае условие запишется следующим образом:

**if (x>=a && x<=b) ,**

где **&&** - операция логического **И**.

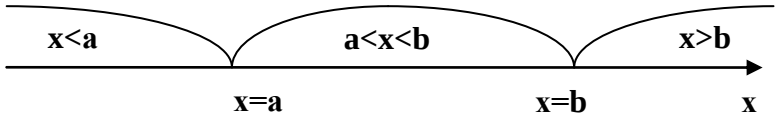


Рис. 5. Графическая интерпретация логического выражения для трех интервалов значений **x**

### 4. Исследование вложенного условного оператора if…else

Во многих случаях использование простого (одинарного) оператора **if…else** не обеспечивает решение поставленной задачи. Очевидно, что данный оператор, например, не позволяет вычислительному процессу принять решение в случае, когда для какой-то переменной существует четыре и более интервала ее значений. В этом случае применяют вложенный условный оператор **if…else**.

На рис. 6 приведен пример структуры вложенного условного оператора **if…else** (в качестве базовой использована структура, изображенная на рис. 2.1).

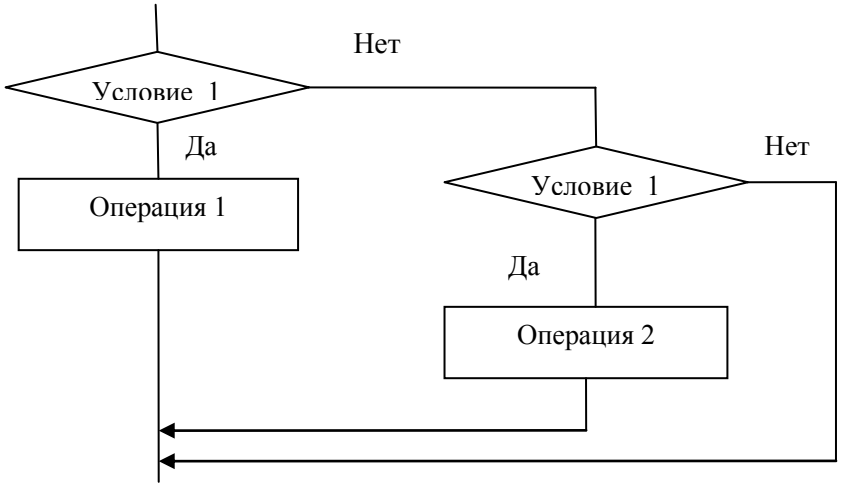


Рис.6. Структура вложенного условного оператора **if…else**

### 3. Оператор выбора switch

Оператор **switch** (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Он обеспечивает выбор из нескольких вариантов на основании некоторого фиксированного параметра. Формат оператора следующий:

```
switch (выражение)
{
  case константа_1: оператор_1;
  case константа_2: оператор_2;
  case константа_3: оператор_3;
  ...
     case константа_n: оператор_n;
     [default: оператор;]
}
```

Блок-сема алгоритма, реализующего работу оператора выбора **switch**, приведена на рис. 2.4.

Выполнение оператора **switch** начинается с вычисления выражения в скобках (оно должно быть целочисленным). Его значение последовательно сравнивается с константами, которые записаны следом за каждым оператором **case**. Затем управление передается тому оператору, который помечен константным выражением (меткой), значение которого совпало с вычисленным выражением в условии. После этого последовательно выполняются все остальные ветви, если выход из переключателя явно не указан.

Все константные выражения должны иметь разные значения, но быть одного и того же целочисленного типа. Несколько меток могут следовать подряд. Если совпадения не произошло,

выполняются операторы, расположенные после слова **default** (а при его отсутствии управление передается следующему за **switch** оператору).

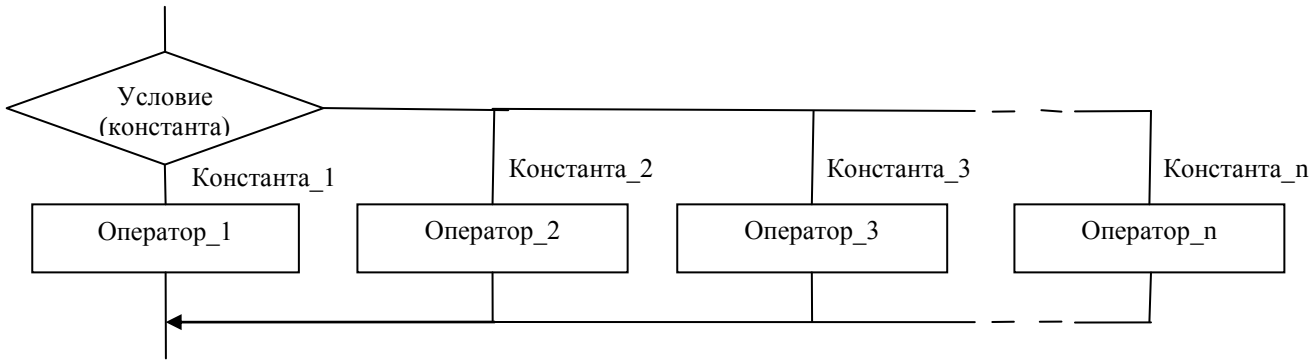


Рис. 2.4. Блок-схема алгоритма работы оператора **switch**

**Пример 3.** Проанализируем работу оператора **switch** на примере программы, реализующей простейший калькулятор на 4 действия. Здесь консольно вводятся две переменные **a** и **b** и знак операции, которую необходимо совершить с этими переменными. Оператор **switch** сравнивает введенный знак операции со знаком, содержащемся в четырех метках **case**, и производит необходимую арифметическую операцию. Результат выводится на экран. Если знак операции не совпадает с содержащимися в метках, то на экран выводится сообщение о неизвестной операции.

```
#include "stdafx.h"
#include <conio.h>                    //Файл conio.h обеспечивает задержку окна DOS на
                                      //экране дисплея
#include <iostream>                   //include подключает файл ввода–вывода iostream
using namespace std;                  //Подключает все имена из пространства имен  std

 int _tmain(int argc, _TCHAR* argv[]) //Объявление главной функции _tmain
{                                     //Начало главной функции
 double a, b, res;                    //Объявление вещественных переменных
 char operation;                      //Объявление символьной  переменной
 cout << "Vvedite a : "<<endl;        //Вывод на экран надписи-приглашения  “Vvedite a :”
        cin >> a;                     //Ввод значения переменной  a
 cout << "Vvedite operation : "<<endl; //Вывод на экран надписи-приглашения  “Vvedite
                                      //operation :”
        cin >> operation;             //Ввод знака операции
 cout << "Vvedite b : "<<endl;        //Вывод на экран надписи-приглашения  “Vvedite b :”
        cin >> b;                     //Ввод значения переменной  b
        switch (operation)            //Условие (с чем будет сравнение) в операторе  switch
        {                             //Начало оператора  switch
        case '+': res = a + b; break; //Метка ”+” и вычисл-е переменной res для этой метки
        case '-': res = a - b; break; //Метка ”-” и вычисл-е переменной res для этой метки
        case '*': res = a * b; break; //Метка ”*” и вычисл-е переменной res для этой метки
        case '/': res = a / b; break; //Метка ”/” и вычисл-е переменной res для этой метки
        default : cout <<"Unknown operation"<<endl;  //Вывод сообщения о неизвестной
                                                     //операции
        }                             //Конец сложного оператора
 cout << "Result : " << res;          //Вывод на экран результата вычислений
        getch();                      //Функция задержки окна DOS на экране
 return 0;
}                                     //Конец главной функции
```

## 4. Безусловный оператор goto

Переход к любому оператору программы без условия (директивный переход) в среде Visual C++2010 осуществляется с помощью оператора безусловного перехода **go to**. В отличие от оператора **if…else** этот оператор осуществляет переход в нужное место программы без выполнения каких-либо условий. Оператор имеет следующий вид:

**goto Метка;**

Меткой обозначают какой-либо оператор, на который должен быть осуществлен переходи з места установки оператора **goto**. В качестве метки выступает идентификатор с расположенным за ним символом двоеточия:

**Метка: оператор;**

Как только выполнение программы достигает оператора **goto**, управление передается оператору, помеченному меткой А.

**Пример 4.** Проанализируем работу оператора **goto** на примере программы из задания 2. Расположим оператор **goto** с меткой **A** перед условным оператором, а метку **A** – перед концом программы. Теперь условный оператор не выполнится, т. к. компьютер выполнит оператор **goto A** и перейдет сразу к выводу надписи **"Perehod po metke ";**

```
#include "stdafx.h"
#include <conio.h>                  //Файл conio.h обеспечивает задержку окна DOS на
                                    //экране дисплея
#include <iostream>                 //Дир. include подключает файл ввода–вывода iostream
using namespace std;                //Подключает все имена из пространства имен  std

int _tmain(int argc, _TCHAR* argv[])  //Объявление главной функции _tmain
{                                   //Начало главной функции
        int a, b;                   //Объявление переменных целого типа
        int c=2, d=3;               //Объявление переменных целого типа  и их
                                    //инициализация
                cout<<"Vvvedite b"<<endl;//Вывод на экран надписи-приглашения  Vvedite b
                cin>>b;             //Ввод значения переменной  b
                goto A;             //оператор goto
                if (b<10)
        {
                a=b+c+d;
                cout<<"a=b+c+d= "<<a;
        }
                else
        {
                a=b-c-d;
                cout<<"a=b-c-d= "<<a;
        }
        A: cout<<"Perehod po metke ";       //Метка A и оператор вывода
        getch();                            //Функция задержки окна DOS на экране
        return 0;
}
```

### Вопросы для самоконтроля.

1. Як записати в операторі **if** перевірку умови на рівність змінної нулю?
   1) **if (x)**
   2) **if (!x)**
   3) **if (x=0)**
2. Записаний оператор **if(a>b) x=a else x=b.** Скільки помилок зроблено при запису оператора?

3. Поточний оператор виконується деяке число раз доти поки не буде задоволена умова його завершення. Це:
   1) послідовна структура
   2) структура вибору
   3) структура повторення

4. Є два числа **a** і **b.** Якщо **a>b,** то поміняти їх місцями. Який з операторів записаний правильно?
   1) **if (a>b) { c=a; a=b; b=c };**
   2) **if (a>b) { c=a; b=c; a=b };**
   3) **if (a>b) { b=c; a=b; c=a };**

5. В операторі **if** перевіряється умова **if (( x>=a) &&(x<=b) ).**Що означає перевірка цієї умови?
   1) **x** належить відрізку **a,b**
   2) **x** не належить відрізку **a,b**
   3) Так записувати умову не можна

6. Вкажіть правильний запис оператора умовного переходу:
   1) **if(x>1) then y=x; else y:=0;**
   2) **if(x>1)y=x else y=0;**
   3) **if(x>1)y=x; else y=0;**

7. У загальному вигляді оператор умовного переходу має вигляд **if (умова) S1; else S2;** . **S1** і **S2** це:
   1) будь-які оператори мови С
   2) оператори присвоювання
   3) спеціальні оператори мови С

8. Записаний оператор: **if (умова) S;** Якщо умова не виконується, то який оператор буде виконаний?
   1) оператор **S**
   2) наступний оператор програми
   3) відбудеться перехід на кінець програми

9. У загальному вигляді оператор умовного переходу має вигляд: **if (умова) S1; else S2; .**
Якщо умова не виконується, то який оператор буде виконаний?
   1) оператор **S1**
   2) оператор **S2**
   3) наступний оператор програми

10. Вкажіть правильний запис оператора безумовного переходу:
   1) **goto A1;**
   2) **goto A1,A2,A3**;
   3) **goto 1;**

11. Записаний оператор: **if (X>0) { X = X-1; Y= 0; } else Y= Y+1 .** Скільки помилок у приведеному операторі?

12. Вкажіть на можливість такого запису оператора **if(1< X < 3) S1; else S2;**
   1) можливо
   2) можливо в окремих випадках
   3) не можливо

13. Яке значення матиме змінна **Z** після виконання операторів?
   **X=1; Y= 1; Z =0; if(X>0) if(Y>0) Z=1; else Z=2;**

14. В операторі **if** необхідно записати умову, що "X не належить відрізку [ A,B ]". Вкажіть на правильний запис цієї умови.
   1) **if(A > X > B)**
   2) **if((X>A)&&(X>B))**
   3) **if((X<A)and(X>B))**