Лекция 8
ФУНКЦИИ И ИХ ПРИМЕНЕНИЕ В СРЕДЕ VISUAL C++

Цель лекции. Изучить назначение и особенности разработки и использования функций в среде Visual C++ 2010

Основные вопросы лекции:

1. Понятие, назначение и объявление функций в среде Visual C++.
2. Описание (программный код) функций в Visual C++.
3. Вызов функций в Visual C++.
4. Передача аргументов функции в Visual C++.
5. Области действия и видимости переменных в программах в среде Visual C++.
6. Функции и массивы в Visual C++.

1. Понятие, назначение и объявление функций в среде Visual C++ 2010

Основную часть программного кода в C++ составляют функции. **Функция** – это самостоятельная единица программы для решения конкретной задачи. Функции позволяют разбивать программу на отдельные автономные блоки. Любая программа содержит, по крайней мере, одну функцию – главную, например, **tmain ( )**.

Для создания правильного кода компилятору необходимо сообщить в начале программы имя функции, тип возвращаемого результата, а также количество и типы аргументов. Для этой цели в C++ используется так называемый **прототип функции**. Прототип функции задается следующим образом:

ТипРезультата ИмяФункции (ТипПараметра1 [ИмяПараметра1], …);

Использование прототипа функции является **объявлением функции**. Чаще всего прототип функции совпадает с заглавием функции. В отличие от заглавия функции прототип заканчивается (!) **точкой с запятой.**

Имена формальных параметров функции при ее объявлении не играют роли. Поэтому прототип функции может выглядеть следующим образом:

```
int function(int a, float b, float c);
```

или

```
int function(int, float, float);
```

Два этих объявления функции **function** равносильны.

2. Описание (программный код) функций в Visual C++ 2010

Основная форма описания или **программный код** функции имеет следующий вид:

```
Тип ИмяФункции (ТипПараметра1 ИмяПараметра1, …)
{
Тело функции
}
```

Описание функции состоит из заглавия функции и тела функции. Все исследованные нами выше программы имели по умолчанию такое описание главной функции:

```
int _tmain(int argc, _TCHAR* argv[])
{
Тело функции
}
```

В заглавии **Тип** перед именем функции определяет тип значения, которое возвращает функция. Если тип не указан, то по умолчанию предусматривается, что функция возвращает целое значение (тип **int**).

Список параметров состоит из перечня типов и имен параметров, разделенных запятыми. Функция может не иметь параметров, но круглые скобки необходимы всегда.

В списке параметров для каждого параметра должен быть указан тип. Например,

function (int x, int в, float z) - правильный список параметров;

function (int x, в, float z) - неправильный список параметров.

В теле функции обязательно должен присутствовать оператор **return** (возвратить) с параметром того же типа, что и возвращаемое значение.

Оператор **return** имеет два варианта использования.

1. Вызывает немедленный выход из функции и возвращение в программу, которая ее вызвала.
2. Используется для возвращения значения функции.

Если возвращаемое значение не используется в дальнейшем в программе, то оператор **return** следует без параметра или **вообще может быть опущен**. В этом случае возвращение в программу осуществляется после достижения фигурной закрывающейся скобки **}.**

В случае, когда оператора **return** в теле функции нет или за ним нет значения, то значение, возвращаемое функцией, неизвестно (не определено). Если функция должна возвращать значение, но не делает этого, компилятор выдает предупреждение. Все функции, которые возвращают значение, могут использоваться в выражениях языка C++.

Функция может вызывать другие функции (одну или несколько). А те, в свою очередь, проводить вызов третьих и т.д. Кроме того, функция может вызывать саму себя. Это явление в программировании называется **рекурсией.**

Любая программа в среде Visual C++ 2010 обязательно включает главную функцию **tmain()**. С этой функции начинается выполнение программы.

3. Вызов функций в Visual C++ 2010

Для того чтобы функция выполняла определенные действия в программе, она должна быть вызвана. Функция выполняется только при обращении к ней. По окончании работы функция возвращает в основную программу в качестве результата значение некоторой переменной и т. п.

Вызов функции осуществляется путем указания в программе ее имени (идентификатора), за которым в круглых скобках следует список аргументов, разделенных запятыми.

ИмяФункции(аргумент 1, аргумент 2, … аргумент N) .

Каждый аргумент функции является **переменной, выражением или константой**. Они передаются в тело функции для последующего использования в вычислительном процессе. Список аргументов может быть пустым.

На рис. 1 схематично показано, как взаимодействует основная программа и функции в **Visual C++ 2010.**

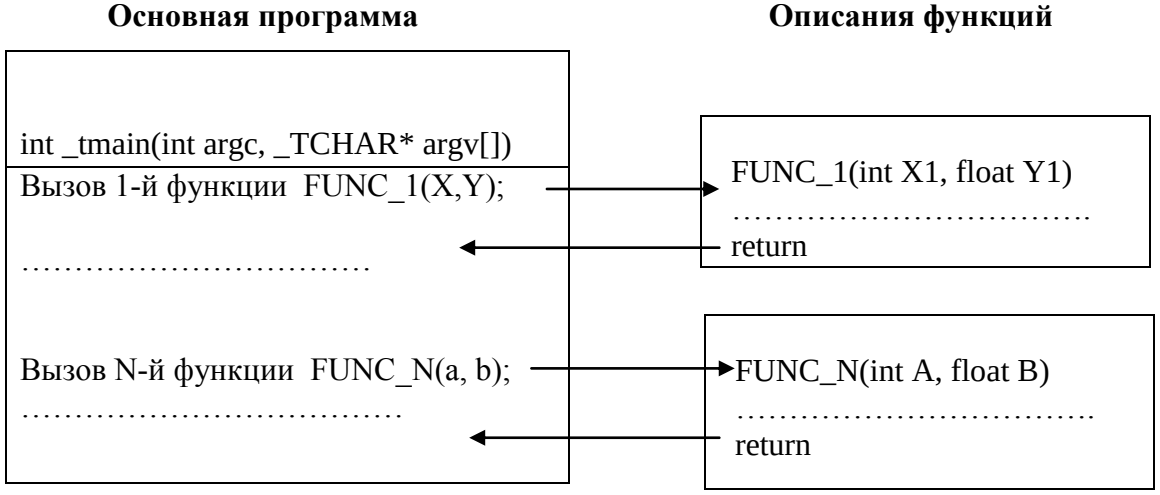


Рис. 1. Порядок вызова функций

Аргументы, которые указаны в заглавии функции, носят название **формальных**, например, в **FUNC(int X1, float Y1)** формальные параметры – **X1** и **Y1.**

Аргументы, которые указаны в имени функции при ее вызове, называются фактическими. Например, при вызове **FUNC(X,Y)** фактические параметры – **X** и Y. Фактические параметры принимают конкретные значения, передающиеся формальным параметрам.

В языке C++ есть особенность – все аргументы функции передаются по значению.

Например, при трансляции функции **float func(float x, float в)** в стеке выделяется место для ее формальных параметров. В это выделенное место заносятся значения фактических параметров, то есть значения параметров при вызове функции. Далее функция использует эти параметры (см. рис. 2) **.**

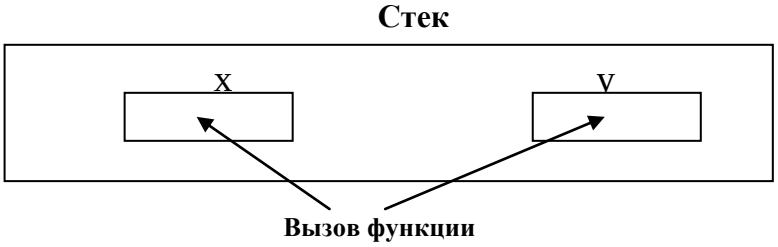


Рис. 2. Вызов функции **func (a,b)**

4. Передача аргументов функции в Visual C++ 2010

Существует два способа передачи аргументов функции в C++: по значению и по ссылке.

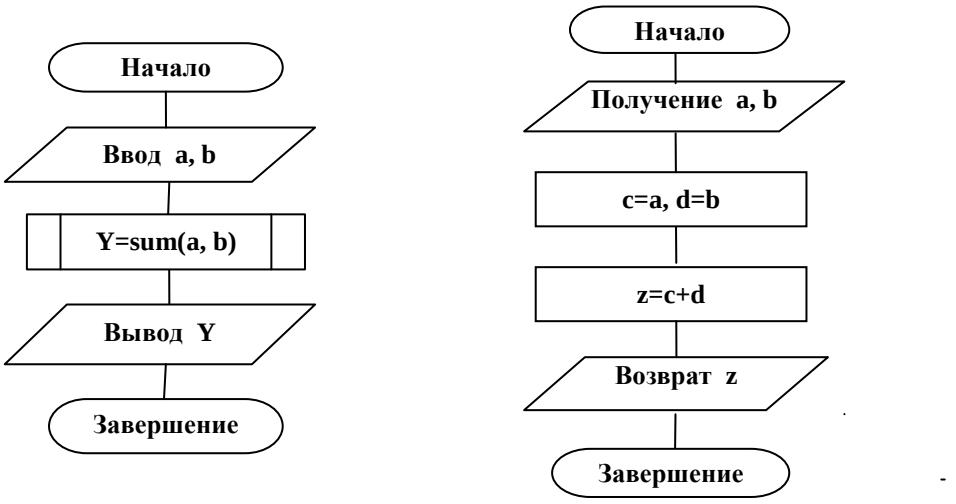
Когда происходит передача переменной-аргумента по значению, в функции создается локальная переменная с именем аргумента, в которую записывается его значение. Внутри функции может измениться значение этой переменной, но не самого аргумента.

Тип каждого фактического параметра (константы или переменной) в инструкции вызова функции должен совпадать с типом соответствующего формального параметра, указанного в объявлении функции.

Если параметр функции используется для возвращения результата, то в объявлении функции этот параметр должен быть ссылкой, а в инструкции вызова функции как фактический параметр должен быть указан адрес переменной (передача аргументов по ссылке).

Пример 1. Разработать программу, вычисляющую **Y** по формуле **Y=a+b**, в которой расчет **Y** оформлен в виде функции.

Блок-схема алгоритма решения данной задачи приведена на рис. 3.



а) б)

Рис. 3. Блок-схема алгоритма вычисления **Y** по формуле **Y=a+b** с использованием функции:
а) основная программа; б) функция **sum(double c, double d)**

Программа, использующая функцию для вычисления **Y** по формуле **Y=a+b**, будет иметь следующий вид:

```
double sum(double c, double d);          //Объявление (прототип) функции вычисления Y
int _tmain(int argc, _TCHAR* argv[])
{
  double Y, a, b;
  cout<<"Vvedite a,  b:"<<endl;
  cin>>a>>b;                             //Ввод чисел a и b
  Y=sum(a, b);                           //Вызов функции, вычисляющей сумму двух чисел
  cout<<endl<<"Y = "<<Y<<endl;           //Вывод значения Y
  getch();
  return 0;
}

double sum(double c, double d)           //Описание функции вычисления z=c+d
{
  double z;
  z=c+d;                                 //Вычисление  z=c+d
  return z;                              //Возврат значения  z  в основную программу
}
```

Результат выполнения программы следующий:

```
Vvedite a,  b:
5.4 4.3
Y=9.7
```

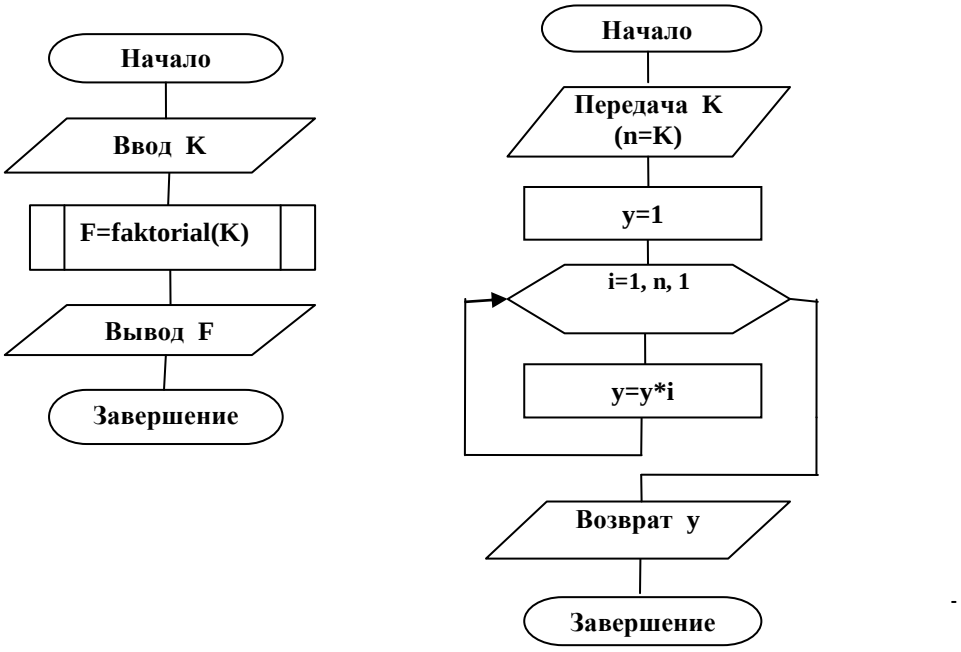
Пример 2. Разработать программу, использующую функцию вычисления факториала числа:

$$F = n!\ .$$

Для составления программы данное выражение запишем следующим образом:

$$F = n! = 1*2*3*4* \ldots *n = \prod_{j=1}^{n} j;$$

Блок-схема алгоритма решения данной задачи приведена на рис. 4.



а) б)

Рис. 4. Блок-схема алгоритма вычисления факториала числа с использованием функции:
а) основная программа; б) функция **faktorial(int n)**

Программа для функции, вычисляющей факториал числа **n**, будет иметь следующий вид:

```
int faktorial(int  n);                          //Объявление (прототип) функции вычисления факториала
int _tmain(int argc, _TCHAR* argv[])
{
  int K;
  cout<<"Vvedite K:"<<endl;
  cin>>K;                                       //Ввод числа K
  F=faktorial(K);                               //Вызов функции, вычисляющей факториал числа
  cout<<endl<<"Faktorial K= "<<K<<endl;         //Вывод значения факториала числа K
  getch();
  return 0;
}

int faktorial(int  n)                           //Описание функции вычисления факториала числа
{
  int j, y;
  y=1;
          for(j=1; j<=n; j++)                   //Цикл для вычисления  факториала числа
          y=y*j;
  return y;                                     //Возврат значения  y  в основную программу
}
```

Результат выполнения программы следующий:

```
Vvedite K:
5
Faktorial K= 120
```

Пример 3. Разработать программу для вычисления числа соединений **R** из **N** элементов по **M** по формуле

$$R = C_N^M = \frac{N!}{M!(N-M)!};$$

где расчет **R** производится с использованием функции.

Воспользуемся созданной в **Примере 2** функцией **factorial(int n)** для вычисления факториала числа и будем обращаться к этой функции непосредственно в формуле для расчета **R**:

```
int faktorial(int  n);                          //Объявление (прототип)  функции вычисления факториала
int _tmain(int argc, _TCHAR* argv[])
{
  int N,M,R;
```

```
  cout<<"Vvedite N i M (N>M):"<<endl;
  cin>>N>>M;                                  //Ввод значений  N и M
  R=faktorial(N)/(faktorial(M)*faktorial(N-M));  //Формула для R (с трехкратным вызовом функции faktorial)
  cout<<endl<<"R= "<<R<<endl;                 //Вывод значения R
  getch();
  return 0;
}

int faktorial(int n)                          //Описание функции вычисления факториала числа
{
  int j, y;
  y=1;
  for(j=1; j<=n; j++)
          y=y*j;
  return y;                                   //Возвращение в основную программу значения  y
}
```

Результат выполнения программы следующий:

Vvedite N и M:
8 4
R= 70

5. Области действия и видимости переменных в программах в среде Visual C++ 2010

Область действия переменной – это часть или части программного кода, в которых данные переменные **определены** (доступны для действий с ними в данном месте программы).

С точки зрения области действия переменных различают три типа переменных:

- локальные;
- глобальные;
-формальные.

Локальные переменные – это переменные, объявленные в середине блока, в частности, внутри описания функции. Локальная переменная доступна в середине блока, в котором она объявлена. Блок открывается и закрывается фигурными скобками. Область действия локальной переменной – данный блок или функция.

Формальные переменные (параметры) – это переменные, объявленные при описании функции как ее аргументы. Формальные параметры используются в теле функции, как локальные переменные. Область действия формальных параметров – тело функции.

Глобальные переменные – это переменные, объявленные в основной программе вне какой-либо функции. Они могут быть использованы в любом месте программы. Область действия глобальной переменной – вся программа.

6. Функции и массивы в Visual C++ 2010

Если в качестве аргумента функции используется массив, то необходимо указать адрес начала массива и его размер.

Заглавие функции, обрабатывающей массив, необходимо записать следующим образом:

float function (float A[n]);

или

float function (float A[ ], int n);

В этом случае вызов функции **function** из основной программы запишется следующим образом:

function (B, k);

Пример 4. Задан массив **A** из **N** произвольных чисел. Сформировать новый массив **B**, каждый элемент которого равняется частному от деления соответствующего элемента массива **A** на его максимальный элемент. На экран вывести начальный массив **A**, его максимальный элемент и массив **B**. Поиск максимального элемента массива **A** оформить функцией.

Запрограммируем функцию нахождения максимального элемента массива и его номера и используем ее в разрабатываемой программе следующим образом:

```
double Max(double B[],int n);                 //Объявление (прототип)  функции поиска макс. элемента
int _tmain(int argc, _TCHAR* argv[])
{
  const int N=50;                             //Максимальный размер исходного массива
  int i, k;                                   //Объявление параметра  цикла и реального размера массива
  double A[N], M;                             //Объявление массива A и переменной  M
  cout<<"Vvedite razmer massiva:"<<endl;
  cin>>k;                                     //Ввод реального размера массива
  cout<<" Vvedite massiv:"<<endl;
  for(i=0; i<k; i++)
          cin>>A[i];                          //Ввод исходного массива A
```

```
  cout<<endl;
  M=Max(A, k);                          //Обращение к функции поиска  Max()
  cout<<" Ishodnui massiv "<<endl;
  for(i=0; i<k; i++)                    //Цикл для печати  исходного массива A
          cout<<A[i]<<' ' ;
  cout<<endl<<"Max= "<<M<<endl;         //Печать максимального  элемента исходного  массива
  cout<<" Novui massiv "<<endl;
  for(i=0; i<k; i++)                    //Цикл для расчета и печати элементов  нового  массива
          cout<<A[i]/M<<' ';
  cout<<endl;
  getch();
  return 0;
}
double Max(double B[],int n)            //Описание функции поиска максимального элемента массива
{
  int j;
  double C=B[0];                        //Присваивание переменной C начального знач-я (1-го элемента)
  for(j=0; j<n; j++)                    //Цикл для перебора элементов массива и сравнения их с C
          if (B[j]>C) C=B[j];
  return C;                             //Возвращение в основную программу значения  C
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite razmer massiva :
7
 Vvedite massiv:
1 6 0 3 7 9 5

 Ishodnui massiv
1 6 0 3 7 9 5
Max= 9
 Novui massiv
0.111111 0.666667 0 0.333333 0.777778 1 0.555556
```

Пример 5. Задан массив **A** из **N** произвольных чисел. Разработать программу для нахождения максимального элемента этого массива и его номера, которые вывести на экран. Поиск максимального элемента массива и его номера оформить функцией, а сами переменные использовать как глобальные, т. е. “видимые” и в основной программе, и в описаниях функций.

Запрограммируем функцию нахождения максимального элемента массива и его номера и используем ее в разрабатываемой программе следующим образом:

```
double Max(double B[], int n);          //Объявление (прототип) функции Max()
double C;                               //Объявление глобальной переменной C
int k;                                  //Объявление глобальной  переменной k
int _tmain(int argc, _TCHAR* argv[])
{
  const int N=50;                       //Объявление максимального размера исходного массива
  int i, m;                             //Объявление параметра цикла и реального размера массива
  double A[N];                          //Объявление исходного массива  A[N]
  cout<<"Vvedite razmer massiva:"<<endl;
  cin>>m;                               //Ввод реального размера исходного массива
  cout<<endl<<"Vvedite massiv :"<<endl;
  for(i=0; i<m; i++) cin>>A[i];         //Ввод элементов исходного массива A[i]
  cout<<endl;
  Max(A, m);                            //Обращение (вызов) к функции Max()
  cout<<"Max= "<<C<<endl;               //Печать макс. элемента массива A (глобальная переменная)
  cout<<" Nomer max "<<k<<endl;         //Печать номера макс. элемента (глобальная переменная)
  getch();
  return 0;
}

double Max(double B[], int n)           //Заголовок описания (кода) функции  Max
{
  int j;                                //Объявление параметра цикла
  C=B[0];                               //Глобальной переменной C присваиваем первый элемент
          for(j=0; j<n;j++)             //Цикл для поиска максимального элемента
```

```
            if(B[j]>C)
            {
            C=B[j];                    //Определяем максимальный элемент
            k=j;                       //Номер макс. элемента заносим в глобальную переменную k
            }
  return 0;                            //Функция не возвращает никаких значений
}
```

После выполнения программы экран будет иметь следующий вид:

```
Uvedite razmer massiva :
8

Uvedite massiv :
11 23 15 7 8 12 4 9

Max= 23
 Nomer max 1
```

Вопросы для самоконтроля

1. При обращении к функции реальные и формальные параметры должны совпадать…
   1) по количеству и месту
   2) по количеству, типу и месту
   3) по количеству и типу
2. Обращение к функции осуществляется…
   1) по заглавию функции
   2) по имени функции
   3) с помощью специального оператора
3. Функции могут быть…
   1) только с параметрами
   2) только без параметров
   3) с параметрами и без параметров
4. В программе используются две независимые функции **F1** и **F2**. В каком порядке они должны быть описаны в разделе прототипов?
   1) Сначала **F1**, а затем **F2**
   2) Сначала **F2**, а затем **F1**
   3) В любом порядке
5. В массиве **A** необходимо определить максимальный элемент и его номер. Возможно ли этот блок оформить функцией?
   1) Возможно
   2) Возможно в отдельных случаях
   3) Невозможно
6. В программе описано две функции **F1** и **F2**. Возможно ли в них использовать одинаковые переменные?
   1) Возможно
   2) Не возможно
   3) Возможно в отдельных случаях
7. Отдельный блок программы целесообразно оформить функцией, если в нем есть…
   1) один результат
   2) два и более результата
   3) Все равно сколько результатов
8. В программе описана функция **int f(int x)** . Возможно ли так обратится к этой функции: **r = f(n)/(f(m)*f(n-m));**?
   1) Возможно
   2) Невозможно
   3) Возможно в отдельных случаях
9. Укажите правильный прототип функции **FF**, использующей в качестве аргумента целый массив **W[n]**.
   1) **double FF(double B[], int n);**
   2) **double FF(int W[], int n);**
   3) **FF(double B[], int n);**
10. Укажите правильное обращение к функции **FF**, использующей в качестве аргумента целый массив **W[n]**.
    1) **func (int W[], n);**
    2) **FF(W, n);**
    3) **func (int W[][]);**
11. Когда функция не возвращает никакого значения, можно ли ее использовать в выражениях языка С++?
    1) можно 2) нельзя 3) можно, когда выражение принимает определенное значение
12. Параметры функции могут быть переменные, которые…
    1) объявлены в функции
    2) объявлены вне функции
    3) все равно где

13. Укажите правильную запись заглавия функции:
    1) **func(int x, в, float z)**
    2) **func(x, int в, float z)**
    3) **func(int x, int в, float z)**
14. Оператор **return**…
    1) вызывает выход из функции
    2) используется для возвращения значения функции
    3) и первое, и второе
15. В теле функции оператор **return**…
    1) может отсутствовать
    2) должен быть обязательно
    3) все зависит от назначения функции
16. Когда в теле функции отсутствует оператор **return**, выход из функции осуществляется…
    1) когда выполнится последний оператор функции
    2) в данном случае возникнет зависание программы
    3) оператор **return** должен быть в теле функции обязательно
17. Если функция не имеет аргументов, то в прототипе такой функции необходимо…
    1) ничего не писать в скобках функции
    2) в скобках записать служебное слово int
    3) в скобках записать служебное слово void
18. Локальная переменная – это переменная, которая объявлена…
    1) в теле данной функции
    2) вне данной функции
    3) в качестве параметра в заглавии функции
19. В заглавии функции записываются…
    1) формальные параметры
    2) фактические параметры
    3) глобальные переменные
20. При обращении к функции реальные параметры должны совпадать с формальными…
    1) по месту
    2) по месту и типу
    3) по месту, типу и количеству