Практическое занятие 14

Тема: «Функции»

Теоретическая часть

Системный подход к программированию основывается на том, что поставленная перед разработчиком задача, как правило, разбивается на несколько меньших, которые, в свою очередь, делятся еще на несколько менее сложных задач, и так до тех пор, пока самые мелкие задачи не будут решены с помощью стандартных процедур. Таким образом осуществляется так называемая функциональная декомпозиция.

Ключевым элементом данной модели для решения конкретной задачи в C++ выступает функция. Функцию можно представить как подпрограмму или некую процедуру, несущую законченную смысловую нагрузку.

Параметры и аргументы функций

Каждая функция, которую предполагается использовать в программе, должна быть в ней объявлена. Обычно объявления функций размещают в заголовочных файлах, которые затем подключаются к исходному тексту программы с помощью директивы **#include**. Объявление функции описывает ее прототип (иногда говорят, сигнатура). **Прототип** функции объявляется следующим образом:

возвращаемый_тип FunctionName(перечень объявляемых параметров);

Здесь возвращаемый_тип - возвращаемый функцией тип данных (например, int, double, char, void и т.д.). Если возвращаемый тип данных не указан, то по умолчанию возвращаемым функцией типом будет считаться int.

Список объявляемых параметров задает тип и имя каждого параметра функции, разделенные запятыми. Допускается опускать имя параметра. Список объявляемых параметров функции может быть пустым. Примеры прототипов функций:

```
int swap(int, int) ;
double max (double par1, double par2);
void func() ;
```

Прототип функции может быть пропущен, если определение функции следует до первого ее вызова из любой другой функции. Последний вариант считается плохим стилем программирования, так как бывают случаи перекрестных вызовов, а любые вносимые в программу изменения будут требовать анализа всего ее кода.

Определение функции состоит из ее заголовка и тела, которое заключено в фигурные скобки и несет смысловую нагрузку. Если функция возвращает значение, отличное от типа void, в теле функции обязательно должен присутствовать оператор return с параметром того же типа, что и возвращаемое значение. В случае, если возвращаемое значение не будет использоваться в дальнейшем в программе (void), оператор return следует без параметра или вообще может быть опущен, тогда возврат из функции осуществляется по достижении закрывающейся скобки.

Для того чтобы функция выполнила определенные действия, она должна быть вызвана в программе. При обращении к функции она выполняет поставленную задачу, а по окончании работы возвращает в качестве результата некоторое значение. Вызов функции представляет собой указание идентификатора функции (ее имени), за которым в круглых скобках следует список аргументов, разделенных запятыми:

имя_функции (аргумент_1,
аргумент_2,
аргумент_3,
….
аргумент_N)

Каждый аргумент функции представляет собой переменную, выражение или константу, передаваемые в тело функции для дальнейшего использования в вычислительном процессе. Список аргументов функции может быть пустым.

Функция может вызывать другие функции (одну или несколько), а те, в свою очередь, производить вызов третьих и т.д. Также функция может вызывать сама себя. Это явление в программированин называется рекурсией.

Любая программа на C++ обязательно включает в себя главную функцию main(). Именно с этой функции начинается выполнение программы.

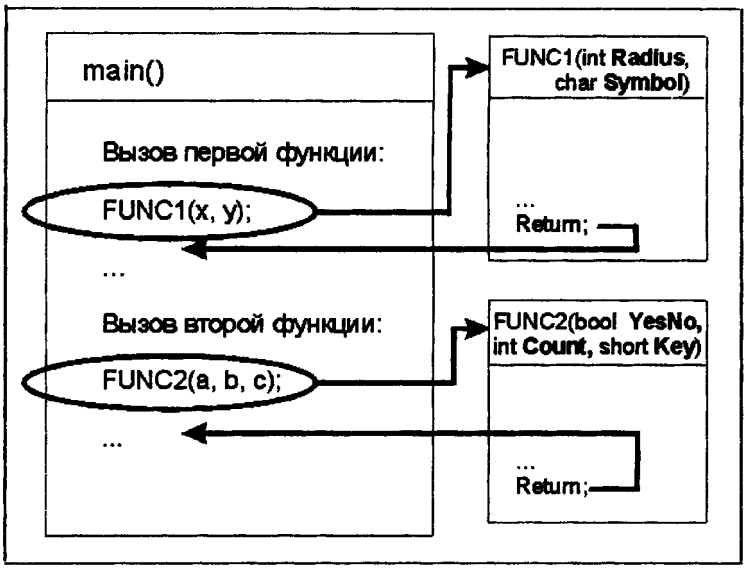

Рис.1. Схематический порядок вызова функций.

Рассмотрим простой пример вычисления квадрата числа с использованием функции.

```
#include <cstdlib>
#include <iostream>

using namespace std;
// Прототип функции:
long MySquare(int) ;

int main(int argc, char *argv[])
{
    int Variable = 5;
    cout << MySquare(Variable);
    cin.get();
    return EXIT_SUCCESS;
}

// определение функции:
long MySquare(int x)
{
    return x*x;
}
```

Перед вызовом main() объявляется прототип используемой функции, в приведенной программе это MySquare(int). Эта функция имеет один целочисленный аргумент. Сама функция задана уже после закрытия main(). В результате работы программы будет выведено 25.

В C++ допускается при вызове функций опускать некоторые ее параметры. Достигается это указанием в прототипе функции значений аргументов по умолчанию. Ниже приведен пример, в котором функция itShow() при вызове может иметь различный вид в зависимости от ситуации.

```
#include <cstdlib>
#include <iostream>

using namespace std;
// В прототипе функции заданы значения 2 аргументов
void itShow(int i, bool Flag=true, char symbol='A');

int main(int argc, char *argv[])
{
// функция вызывается трижды с различным набором аргументов
    cout << "First call" << endl;
    itShow(100);
    cout << endl << "Second call" << endl;
        itShow(200,false);
        cout <<  endl << "Third call" << endl;
            itShow(300,false,'B');
```

```
    cin.get();
    return EXIT_SUCCESS;
}
```

// определение функции:

```
void itShow(int i, bool Flag, char symbol)
{
    cout << "i=" << i << endl;
    cout << "Flag=" << Flag << endl;
    cout << "symbol=" << symbol << endl;
}
```

Практическая часть

1. Изучите примеры объявления и использования функций, приведенные в теоретической части. Реализуйте приведенные примеры. Изучите работу примеров внося в них свои изменения.

2. Напишите программу, в которой заданы следующая функция:

$$y(x)=\begin{cases} x^2-x-1/2, x\leqslant -1/2 \\ -x^2-x+2, -1/2<x\leqslant 1 \\ -x^2+x+1/2, x>1 \end{cases}$$

   Вычислите 10 значений функции на разных участках интервала от -2 до 2.

3. Подготовьте отчет. В отчете представьте код программ и скриншоты их работы.