C++- Functions

Introduction

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is *main()*, and all the most trivial programs can define additional functions.

A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

Defining a Function

The general form of a C++ function definition is as follows:

```
return_type function_name( parameter list ) {
   body of the function
}
```

A C++ function definition consists of a function header and a function body. Here are all the parts of a function:

- **Return Type** − A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.
- **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function signature.
- **Parameters** − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- **Function Body** − The function body contains a collection of statements that define what the function does.

Function Declarations

A function declaration tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts:

```
return_type function_name( parameter list );
```

Calling a Function

While creating a C++ function, you give a definition of what the function has to do. To use a function, you will have to call or invoke that function.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value.

Function Arguments

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the formal parameters of the function.

Default Values for Parameters

When you define a function, you can specify a default value for each of the last parameters. This value will be used if the corresponding argument is left blank when calling to the function.

This is done by using the assignment operator and assigning values for the arguments in the function definition. If a value for that parameter is not passed when the function is called, the default given value is used, but if a value is specified, this default value is ignored, and the passed value is used instead.

Passing parameters by reference

By default, we pass parameters by value, we make a copy of the variable and the changes to the variable within the function do not occur outside the function. For example:

```
#include <iostream>
using namespace std;
// pass-by-value
void increment(int a) {
 a = a + 1;
 cout << "a in increment " << a << endl;
}
int main() {
 int q = 3;
 increment(q); // does nothing
 cout << "q in main " << q << endl;
 return 0;
}
```

The output is:

```
a in increment 4
q in main 3
```

The original variable is not modified. If you want to modify the original variable as opposed to making a copy, you must pass the variable by reference (*int &a* instead of *int a*).

```
#include <iostream>
using namespace std;
//pass by-reference
void increment(int &a) {
 a = a + 1;
 cout << "a in increment " << a << endl;
}
int main() {
 int q = 3;
 increment(q); // works
 cout << "q in main " << q << endl;
 return 0;
}
```

The output is:

```
a in increment 4
q in main 4
```

Exercises

1. Write a program that will ask the user to input three integer values from the keyboard inside the main function. Then it will print the smallest and largest of those numbers.
2. Write a program in C++ which swap the values of two variables not using third variable.