**Functions**

Function: Function is a sub program that acts on data and often returns a value. A function groups a number of program statements into a unit and gives it a name. This unit can then be invoked from other parts of the program.

Types of Functions

There are two types of functions in C++ programming:

1. Built-in Functions: Built-in Functions are the functions which are declared in the C++ header files such as sqrt(), strlen(), pow(), etc.

2. User-defined functions: User-defined functions are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.

 FIGURE

Flow of control to a function.

**Simple Functions**

Our first example demonstrates a simple function whose purpose is to print a line of 45 asterisks.

The example program generates a table, and lines of asterisks are used to make the table

more readable. Here’s the listing for TABLE:

// table.cpp

// demonstrates simple function

#include <iostream>

using namespace std;

void starline(); //function declaration

// (prototype)

int main()

{

starline(); //call to function

cout << “Data type Range” << endl;

starline(); //call to function

cout << “char -128 to 127” << endl

<< “short -32,768 to 32,767” << endl

<< “int System dependent” << endl

<< “long -2,147,483,648 to 2,147,483,647” << endl;

starline(); //call to function

return 0;

}

//--------------------------------------------------------------

// starline()

// function definition

void starline() //function declarator

{

for(int j=0; j<45; j++) //function body

cout << ‘\*’;

cout << endl;

}

The output from the program looks like this:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Data type Range

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

char -128 to 127

short -32,768 to 32,767

int System dependent

long -2,147,483,648 to 2,147,483,647

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The program consists of two functions: main() and starline(). You’ve already seen many

programs that use main() alone. What other components are necessary to add a function to the

program? There are three: the function declaration, the calls to the function, and the function

definition.

The Function Declaration

Just as you can’t use a variable without first telling the compiler what it is, you also can’t use a

function without telling the compiler about it. There are two ways to do this. The approach we

show here is to declare the function before it is called. (The other approach is to define it

before it’s called)

In the TABLE program, the function starline() is declared in the line

void starline();

The declaration tells the compiler that at some later point we plan to present a function called

starline. The keyword void specifies that the function has no return value, and the empty

parentheses indicate that it takes no arguments.

Calling the Function

The function is called (or invoked, or executed) three times from main(). Each of the three

calls looks like this:

starline();

This is all we need to call the function: the function name, followed by parentheses. The syntax

of the call is very similar to that of the declaration, except that the return type is not used.

The call is terminated by a semicolon. Executing the call statement causes the function to execute;

that is, control is transferred to the function, the statements in the function definition

are executed, and then control returns to the statement following the function call.

The Function Definition

Finally, we come to the function itself, which is referred to as the function definition. The definition

contains the actual code for the function. Here’s the definition for starline():

void starline() //declarator

{

for(int j=0; j<45; j++) //function body

cout << ‘\*’;

cout << endl;

}

The definition consists of a line called the declarator, followed by the function body. The

function body is composed of the statements that make up the function, delimited by braces.

The declarator must agree with the declaration: It must use the same function name, have the

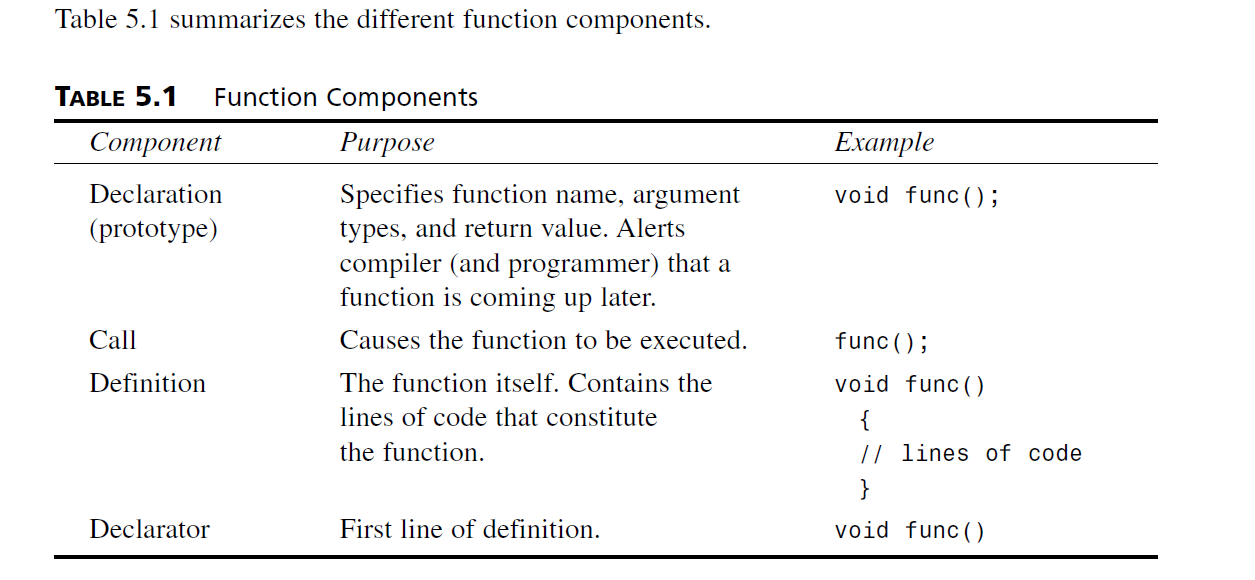
same argument types in the same order (if there are arguments), and have the same return type.

Notice that the declarator is not terminated by a semicolon.

Figure 5.2 shows the syntax of the function declaration, function call, and function definition.

 FIGURE

Function syntax.



**Advantage of functions**

There are many advantages of functions.

1) Code Reusability

By creating functions in C++, you can call it many times. So, we don't need to write the same code again and again.

2) Code optimization

It makes the code optimized so no need to write much code. Suppose, you have to check 3 numbers (531, 883 and 781) whether it is prime number or not. Without using function, you need to write the prime number logic 3 times. So, there is repetition of code. But if you use functions, you need to write the logic only once and you can reuse it several times.

3) Another reason to use functions is to reduce program size.