Functions

Functions

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 as shown in the figure below. Two reasons to use functions: 1) to aid in the conceptual organization of a program. (However, object-oriented programming is more powerful). 2) to reduce program size.

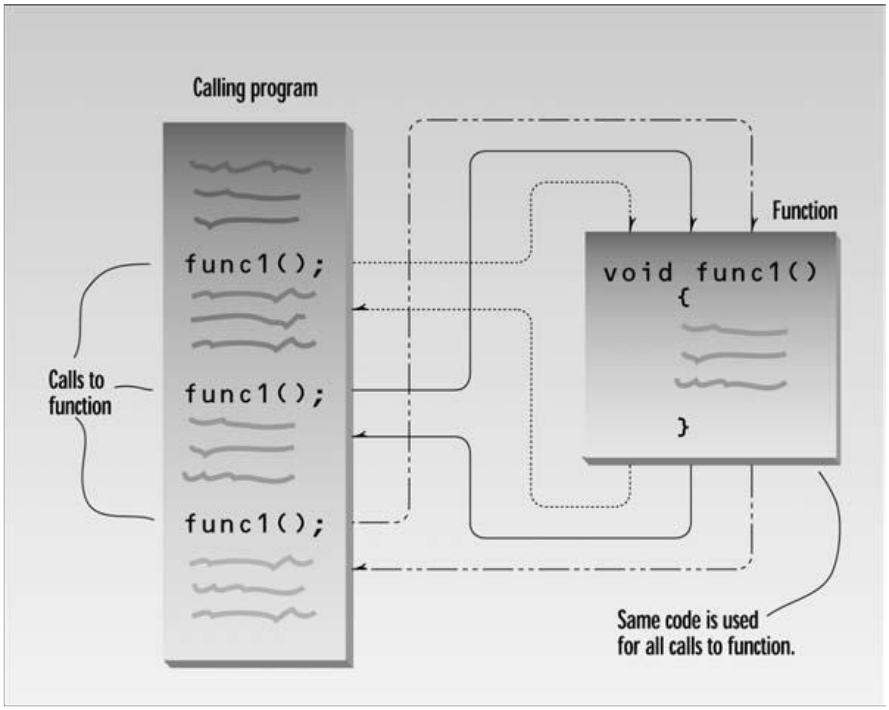


Figure 6.1 Flow of control to a function.

To add a function to the C++ program, we need to add three components to the program:

- function declaration (prototype)
- calls to the function
- function definition.

The structure of program will be as follows:

```
#include <filename.h>
function declaration;
void main ( )
{
  ...
  ...
  call of function;
  ...
  ...
  call of function;
  ...
  ...
}

function definition( )
{
   // function body
   ...
   ...
}
```

Function declaration (prototype)

The general form of the function declaration is:

return-type function-name(argument list) ;

Many arguments may be passed to a function, while **only one** argument may be returned from it.

Examples:

```
void  print();
long factorial(int);
float  average(int , int);
```

Calling Function

The function can be *called* (or *invoked*, or *executed*) many times in the program.

function-name(argument list) ;

Examples:

```
print();
fact = factorial(5);
cout<<average(90 , 70);
```

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.

Function Definition

The function definition contains the actual code for the function.

return-type function-name(argument list)

{

// function body

...

...

}

Examples:

```
void print()
{
  for(int i=0;i<80;i++)
    cout<<"*";
  cout<<endl;
```

```
}

float  average(int x, int y)
{
 return (x+y)/2.0;
}
```

Example: Write a C++ program that prints a line of 25 stars using the function **starline()**.

```
#include <iostream.h>
void starline(); //function declaration
(prototype)
void main()
{
  starline();          //call of function
  cout << "Student Name\tMark" << endl;
  starline();          //call of function
  cout<< "Ahmed\t\t80" << endl
      << "Ali\t\t90" << endl
      << "Hassan\t\t70" << endl
      << "Sara\t\t85" <<endl;
  starline();          //call of function
}

void starline() // function definition
{
  //function body
  for(int j=0; j<25; j++)
       cout << "*";
  cout << endl;
}
```

Example: Write a C++ program that adds two integer numbers using the function **add()**.

```
#include<iostream.h>
int add(int , int);        // function declaration
void main()
{
 int x , y ,z;
 cout<<"Enter two integer numbers: ";
 cin >> x >> y;
```

```
 z = add(x,y);                // call of function
 cout<<"Result= "<< z << endl;
}

int add(int a, int b)          // function definition
{
 return (a+b);
}
```

Passing Arguments to Functions

Passing by Value

Example: Write a C++ program that computes the square of an integer number using the function **sqr()**.

```
#include<iostream.h>
int sqr(int);
void main()
{
 int n;
 cout<<"Enter an integer number: ";
 cin>>n;
 cout<<"The square of "<< n <<" is "<<sqr(n)<<endl;
}

int sqr(int x)
{
 return (x*x);
}
```

Example: Write a C++ program that finds the maximum of two entered numbers using the function **max( )**.

```
#include<iostream.h>
int max(int , int);
void main()
{
 int x , y;
 cout<<"Enter two integer numbers: ";
 cin >> x >> y;
 cout<<"The maximum number is " << max(x,y) <<endl;
}

int max(int a , int b)
{
 if(a > b)
   return a;
 else
   return b;
}
```

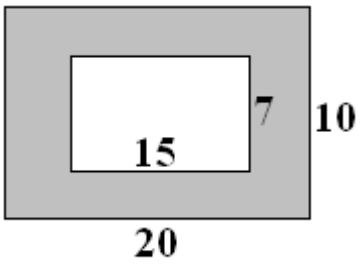
Example: Write a C++ program that computes the factorial of an integer number using the function **factorial()**.

```
#include<iostream.h>
long factorial(int);
void main()
{
 int x;
 cout<<"Enter an integer number: ";
 cin>>x;
 cout<<"The factorial is " << factorial(x)<<endl;
}

long factorial(int a)
{
 long fact = 1;
 for(int i = a ; i > 1 ; i--)
   fact *= i;
 return fact;
}
```

Example: Write a C++ program that computes the power of an integer number using the function **power()**.

```
#include<iostream.h>
long power(int , int);
void main()
{
 int num , pow;
 cout<<"Enter an integer number and power: ";
 cin >> num >> pow;
 cout<<"Result =  " << power(num,pow) << endl;
}

long power(int n , int p)
{
 long result = 1;
 for(int i=1 ; i<=p ; i++)
  result *= n;
 return result;
}
```

Example: Write a C++ program that computes the shaded area in the figure below using the function **area()**.




```
#include<iostream.h>
float area(float , float);
void main()
{
 float x1 = 10 , y1 = 20;
 float x2 = 7 , y2 = 15;
 cout<<"The shaded area is "
     << area(x1,y1) - area(x2,y2)
     << endl;
}

float area(float a , float b)
{
 return (a*b);
}
```

Example: Write a C++ program that tests an entered character if it is a small letter or not using the function **issmall( )**.

```
#include<iostream.h>
int issmall(char);
void main()
{
 char ch;
 cout<<"Enter an alphabetic character: ";
 cin >> ch;
 if(issmall(ch))
   cout<<"The character is a small letter."<<endl;
 else
   cout<<"The character is not a small letter."
       <<endl;
}

int issmall(char c)
{
 if (c >= 'a' && c <= 'z')
     return 1;
 else
     return 0;
}
```

Example: Write a C++ program that finds the max value in an integer array a[10] using the function **maxa()**.

```
#include<iostream.h>
int maxa(int b[10]);
void main()
{
 int a[10];
 cout<<"Enter ten integer values: ";
 for(int i=0; i<10 ; i++)
 cin >> a[i];
 cout<<"The max value is "<< maxa(a) << endl;
}

int maxa(int b[10])
{
 int m = b[0];
 for(int j=1 ; j<10; j++)
  if(b[j] > m)
    m = b[j];
 return m;
}
```

Example: Write a C++ program that reads, sorts and prints an integer array a[10] using three functions **setarray()**, **sortarray()**, and **putarray()**.

```
#include<iostream.h>
void setarray(int b[10]);
void sortarray(int b[10]);
void putarray(int b[10]);
void main()
{
 int a[10];
 setarray(a);
 sortarray(a);
 cout<<"Sorted array is ";
 putarray(a);
}

void setarray(int b[10])
{
  cout<<"Enter ten integer array: ";

  for(int i=0; i<10; i++)
```

```
    cin >> b[i];
}
void sortarray(int b[10])
{
 for(int i=0; i<9; i++)
   for(int j=i+1; j<10; j++)
     if(b[i] > b[j])
      {
       int temp = b[i];
       b[i] = b[j];
       b[j] = temp;
      }
}

void putarray(int b[10])
{
 for(int j=0; j<10; j++)
  cout<<b[j] <<" ";
 cout<<endl;
}
```

Passing by Reference

- When we need the function returns more than one value, we can pass arguments by reference to the function.
- A reference provides an alias—another name—for a variable (It's actually the memory address of the variable that is passed).
- An important advantage of passing by reference is that the function can access the actual variables in the calling program. This provides a mechanism for passing more than one value from the function back to the calling program.

Example: simple program to understand the reference.

```
#include<iostream.h>
void main()
{
 int y = 10;
 int &w = y;
 cout<< y << endl;
 w++;
 cout<< y << endl;
}
```

Example: The same program of **sqr()** function but the argument is passed by reference.

```
#include<iostream.h>
void sqr(int&);
void main()
{
 int n;
 cout<<"Enter an integer number: ";
 cin>>n;
 sqr(n);
 cout<<"The square is "<< n << endl;
}

void sqr(int& x)
{
 x = x * x;
}
```

Example: Write a C++ program that exchanges two integer numbers using the function **swap()**.

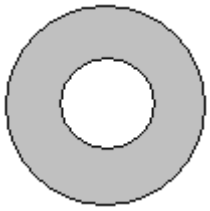
```
#include<iostream.h>
void swap(int& , int&);
void main()
{
 int x , y;
 cout<<"Enter x: ";
 cin >> x;
 cout<<"Enter y: ";
 cin >> y;
 swap(x , y);
 cout<<"x = "<< x <<"\ny = "<< y << endl;
}
void swap(int& a , int& b)
{
 int temp = a;
 a = b;
 b = temp;
}
```

Example: Write a C++ program that reads a real number, and then separates this number into an integer and fractional part using the function **intfrac()**.

```
#include<iostream.h>
void intfrac(float, float& , float&);
void main()
{
 float number , intpart , fracpart;
 cout<<"Enter a real number: ";
 cin >> number;
 intfrac(number, intpart , fracpart);
 cout<<"The integer part is "<< intpart << endl
     <<"The fraction part is "<< fracpart << endl;
}

void intfrac(float n, float& intp , float& fracp)
{
 intp = float(long(n));
 fracp = n - intp;
}
```

Homework:

1. Write a C++ program that determines whether an integer number is positive, negative or zero using the function **test()**.
2. Write a C++ program that computes the shaded area in the figure below using the function **carea( )**. The radius values are entered by the user.



3. Write a C++ program that converts a positive integer number into binary using the function **binary()**.
4. An integer number is said to be a prime if it is divisible only by 1 and itself. Write a C++ program that inputs an integer number and determines whether the number is a prime or not using the function **isprime()**.
5. Write a C++ program to test if an entered character is a numeric digit or not using the function **isndigit( )**.
6. Write a C++ program that computes the number of decimal digits in an entered positive integer number using the function **nodec()**.
7. Write a C++ program that converts a small letter into capital letter using the function **tocapital( )**.
8. Write a C++ program that inputs a time in seconds and then converts the time from seconds into hours, minutes and seconds using the function **converttime( )**. The program should then print the time in a formal format (e.g. 12:59:59) using the function **displaytime()**.