Chapter 8

Functions in C++

8.1 Introduction

A function is a part of the program which is used to perform a task. Dividing a program into functions is one of the major principles of structured programming. The advantage of using functions is that it reduces the size of program by calling and using them at different places in the program.
In C++, many new features are added to functions to make them more reliable and flexible.

8.2 Function Prototype

The function prototype gives the information to the compiler about the function like the number and type of arguments and the return type. The function prototype is a declaration statement in the calling program and its syntax is as follows:
type function_name(arguments-list);
Example:

```
int sum(int a, int b);
```

In function declaration name of arguments are dummy variables and therefore they are optional. The following statement

```
int sum(int, int);
```

is a valid function declaration.

8.3 Call-by-reference

In call-by-value parameter passing method the actual parameters in calling program are copied to formal parameters in called function. The changes made by the called function on formal parameters are not reflect in calling program.
To make the changes in actual parameters in calling program, we use call-by-reference parameter passing method.
For example
Program 8.1: Call-by reference

```
#include<iostream>
using namespace std;
int  main()
{
    int count=0;
    void update(int &);
    cout<<"count="<<count<<"\n";
```

```
update(count);
cout<<"count="<<count;
return 0;
}
void update(int &x)
{
        x=x+1;
}
```

The output of the program 8.1 would be:

count=0
count=1

In the above program, the variable x in update function becomes the alias of the variable count in main function.

8.4 Return by reference

A function can also return a reference. For example

Program 8.2: Return by reference

```
#include<iostream>
using namespace std;
int main()
{
        int x=6,y=9;
        int & min(int &, int &);
        min(x,y)=-1;
        cout<<"x="<<x<<"\n";
        cout<<"y="<<y;
        return 0;
}

int & min(int &a, int &b)
{
         if(a<b)
         return a;
         else
         return b;
}
```

The output of the program 8.2 would be:

x=-1
y=9

In the above program, the return type of the function min is int &, the function

returns the reference to a or b. The function calling statement min(x,y); is a reference to either x or y depending on their values.

8.5 Function overloading

Same function name but different argument list can be used to perform different tasks, is known as function overloading. The correct function is to be called depends on the number and type of arguments but not on the return type of the function.

Program 8.3 Function overloading

```
#include<iostream>
using namespace std;
int sum(int, int);
int sum(int, int, int);
int main()
{
        cout<<"Sum of two numbers is "<<sum(5,10);
        cout<<"\n";
        cout<<"Sum of three numbers is "<<sum(10,20,30);
        return 0;
}
int sum(int x, int y)
{
        return(x+y);
}
int sum(int a, int b, int c)
{
        return(a+b+c);
}
```

The output of the program 8.3 would be:

```
Sum of two numbers is 15
Sum of three numbers is 60
```

In the above program, the function sum() is overloaded two times. When we pass two arguments to the function sum(), the function with two arguments is invoked and when we pass three arguments to the function sum(), the function with three arguments is invoked.

8.6 Inline function

When a function is called, control of execution is transferred from calling function to called function then again go back to the calling function. This is an overhead in program execution time. If the function body is small a

lot of time is spent in such overhead. The solution of this problem is inline function. An inline function is expanded in line when it invoked. The compiler replaces the function call statement with corresponding function body. To make a function inline , write inline keyword with the function declaration/definition statement.

For example

```
inline int sum(int x, int y)
{
return(x+y);
}
```

Important Points

- A function is a part of the program which is used to perform a task.
- The function prototype gives the information to the compiler about the function.
- A function can also return a reference.
- Same function name but different argument list can be used to perform different tasks, is called function overloading.
- An inline function is expanded in line when it invoked in which the compiler replaces the function call statement with corresponding function body.

Practice Questions

Objective type questions:

Q.1 Which is a valid function declaration

A. int fun(int a, int b); B. int fun(int, int);
C. Both A and B D. None of these

Q.2 In which parameter passing method the actual parameters are copied to formal parameters of function?

A. Call- by- reference B. Call-by-value
C. Call- by- address D. None of these

Q.3 In function overloading, the correct function is to be called is NOT depends on

A. Number of arguments B. Type of arguments
C. Return type of the function D. None of these

Q.4 Same function name but different argument list can be used to perform different tasks, is called

A. Function overloading B. Operator overloading
C. Class overloading D. None of these

Very Short Answer Type Questions

Q.1 What is function?

Q.2 What is function overloading?

Q.3 What is inline function?

Short Answer Type Questions

Q.1 What is function prototype?

Q.2 What is difference between call-by-value and call-by-reference?

Q.3 What are advantages of functions in structured programming?

Essay Type Questions

Q.1 Write a program to swap two values by using call-by-reference mechanism.

Q.2 Write a program to overload 'area()' function to compute the area of circle and the area of rectangle.

Answer Key

1. C 2. B 3. C 4. A