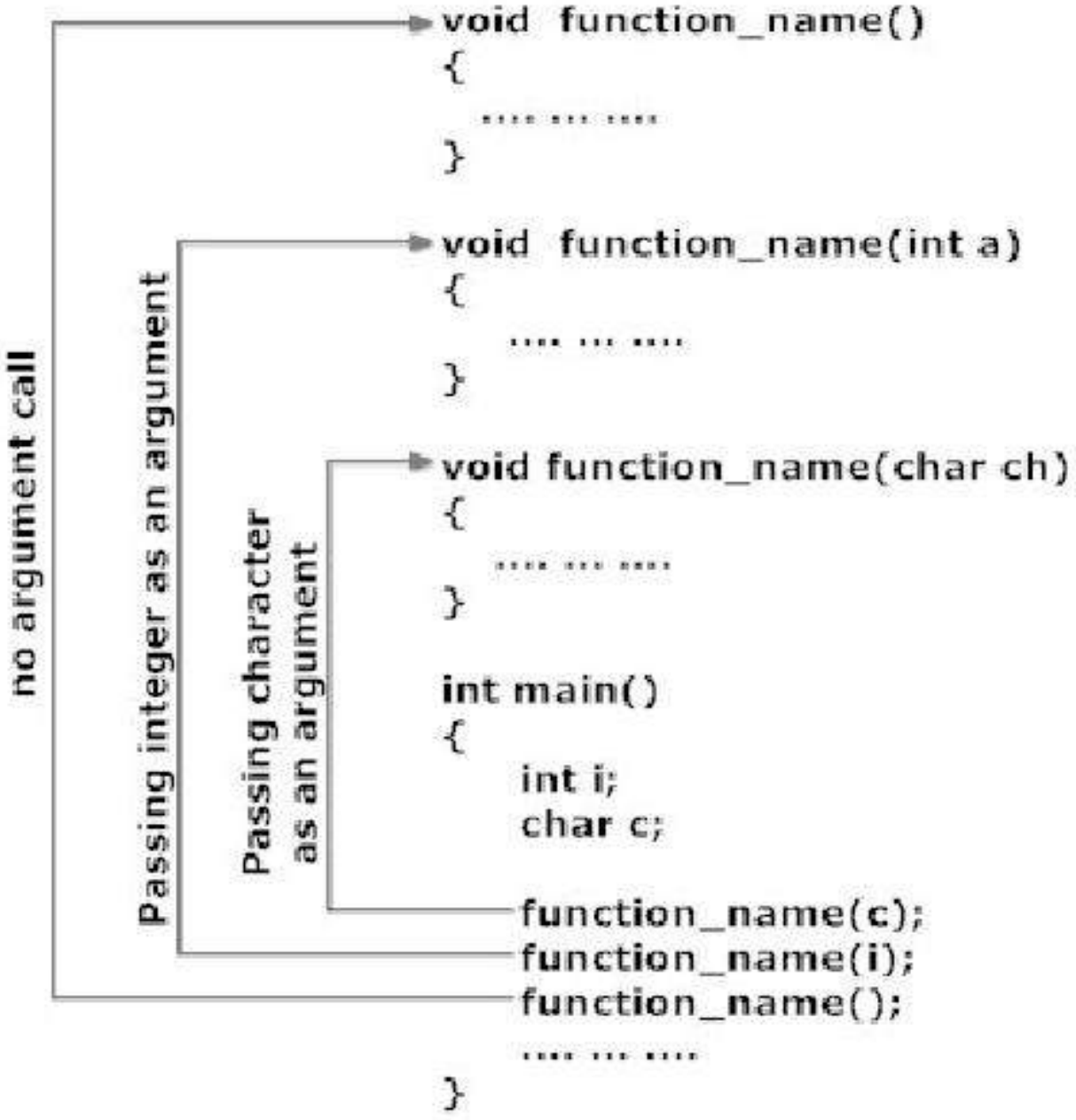
Chapter 8

FUNCTION OVERLOADING AND MEMBER FUNCTIONS

Objects:

- Ø Need for function overloading
- Ø Concept of polymorphism
- Ø Application of function overloading through simple examples
- Ø Concept of inline functions
- Ø Use friend functions



8.1 Introduction

In the previous chapter, we have learnt about classes and objects. The application of classes with data members and member functions is the principle feature of object oriented programming to develop and simplify programming methods. Another feature to help for easier programming is polymorphism. The definition of polymorphism has appeared in chapter 6. Let us recall the definition once again. Polymorphism refers to "one name having many forms, different behavior of an instance of an object depending on situations".

C++ implements polymorphism through function overloading and operator overloading. The function overloading allows the user to create new abstract data types. In this chapter we learn about the need for function overloading, definition and declaration of function overloading and some examples. The discussion is limited to design of a set of functions that perform essentially the same thing, but with a different argument list. The selection of overloaded function depends on matching arguments at the time of compilation. The study in this chapter is also extended to inline and friend functions.

8.2 Need for function overloading

Function overloading means two or more functions have same name, but differ in the number of arguments or data type of arguments. Therefore it is said that (function name) is overloaded. Function overloading therefore is the process of defining same function name to carry out similar types of activities with various data items. The advantages of function overloading are:

- When different functions are created for different operations, then user has to call respective function depending on the situation. Instead, for different situations if the same function is called with different arguments using function overloading, then the compiler automatically decides about the appropriate function by comparing the argument types used in the call to the function and calls the required function. Thus the code is executed faster.
- It is easier to understand the flow of information and debug.
- Code maintenance is easy.
- Easier interface between programs and real world objects.

8.3 Definition and declaration of overloaded functions

The main factor in function overloading is a function's argument list. C++ can distinguish overloaded functions by the number and type of arguments. If there are two functions having same name and different types of arguments or different number of arguments, then function overloading is invoked automatically by the compiler. Function Overloading is also known as Compile time polymorphism.

Example 8.1:
```
int sum(int a, int b);
float sum(float p,  float q);
```

The function sum() that takes two integer arguments is different from the function sum() that takes two float arguments. This is function overloading.

To overload a function, each overloaded function must be declared and defined separately.

Example 8.2:
```
int product(int p, int q, int r);
float product(float x,float y, float z);
int product(int p, int q, int r)
{
        cout<<"product="<<p*q*r<<endl;
}
float product(float x,float y,float z)
{
        cout<<product="<<x*x*y*y*z*z<<endl;
}
```

In this example the function product() is overloaded two times. First time with three integer values and integer as return value, second time with three float values and return type being a float. The compiler automatically chooses the right type of function depending on the number of arguments.

8.4 Restrictions on overloaded functions

- Ø Each function in a set of overloaded functions must have different argument list.
- Ø If typedef is used for naming functions, then the function is not considered as different type.

8.4.1 Calling overloaded functions

The following programming example shows how overloaded functions can be called.

Program 8.1: To compute volume of cone, cube and cylinder using overloaded functions.

```
#include<iostream.h>
#include<conio.h>
class funoverload
{
        public:
                int volume(int a)                    // Volume of Cube
                {
                        return a*a*a;
                }
```

```
        double volume(double r, double h)              // Volume of Cone
        {
                return (0.33*3.14*r*r*h);
        }
        double volume(double r, int h)         // Volume of Cylinder
        {
                return (3.14*r*r*h);
        }
        double volume(double l, double b, double h)    //Volume of Cuboid
        {
                return (l*b*h);
        }
};
int main()
{
        clrscr();
        funoverload f1;
        cout<<"Volume of the Cube: "<<f1.volume(10)<<endl;
        cout<<"Volume of the Cone: "<<f1.volume(2.0,3.0)<<endl;
        cout<<"Volume of the Cylinder: "<<f1.volume(2.0,3)endl;
        cout<<"Volume of the Cuboid: "<<f1.volume(5.0,6.0,7.0)<<endl;
        return 0;
        getch();
}
```

```
Volume of the Cube: 1000
Volume of the Cone: 12.4344
Volume of the Cylinder: 37.68
Volume of the Cuboid: 210
```

The above program finds volume of cube, cuboid, cone or cylinder depending on the number and type of arguments provided as input to the function called volume. Therefore though all functions have the same name volume, appropriate function is selected based on data type of argument list. The compiler selects the required function through function overloading.

8.5 Other functions in a class

The member functions of a class may be defined inside or outside a class. One such function defined inside a class is inline function.

8.5.1 inline functions

A function call generally involves the complex process of invoking a function, passing parameters to the function, allocating storage for local variables, thereby using extra time and memory space. It is possible to avoid these overheads of a function by using inline function. The inline function is a short function. Compiler replaces a function call with the body of the function.

The keyword inline is used to define inline functions. The inline function consists of function call along with function code and the process is known as expansion.

- Inline functions definition starts with keyword inline.
- The inline functions should be defined before all functions that call it.
- The compiler replaces the function call statement with the function code itself (expansion) and then compiles the entire code.
- They run little faster than normal functions as function calling overheads are saved.

Advantages of inline functions

- The inline member functions are compact function calls.
- Therefore the size of the object code is considerably reduced.
- The speed of execution of a program increases.
- Very efficient code can be generated.
- The readability of the program increases.

Disadvantage

As the body of inline function is substituted in place of a function call, the size of the executable file increases and more memory is needed.

Program 8.2: Finding the square of a number using inline functions.

```
#include <iostream.h>
inline int square (int a)
{
   return(a*a);
}
int  main( )
{
        int x, y;

        x=square(5);
        cout<<"Square of 5 = "<<x<<endl;
        y=square(10);
        cout<<"Square of 10 = "<<y<<endl;
        return 0;
}
```

```
Square of 5 = 25
Square of 10 = 100
```

In the above example square() is an inline function that finds the square of a number.

Note: The inline function may not work some times for one of the following reasons:

- The inline function definition is too long or too complicated.
- The inline function is recursive.
- The inline function has looping constructs
- The inline function has a switch or goto.

8.5.2 friend functions

We have seen that private and protected members of a class cannot be accessed from outside the class in which they are declared. In other words non-member function does not have access to the private data members of a class. But there could be a situation where two classes must share a common function. C++ allows the common function to be shared between the two classes by making the common function as a friend to both the classes, thereby allowing the function to have access to the private data of both of these classes.

A friend function is a non-member function that is a friend of a class. The friend function is declared within a class with the prefix **friend**. But it should be defined outside the class like a normal function without the prefix friend. It can access public data members like non-member functions.

Syntax:

```
class class_name
{
        public:
        friend  void  function1(void);
        friend  returntype_specifer  function_name(arguments);
};
```

Example 8.3:

```
class  base
{
                int val1, val2;
        public:
                void getdata()
                {
                        cout<<“Enter two values:”;
                        cin>>val1>>val2;
                }
};
```

```
friend float mean(base ob);
float mean(base ob)
{
        return float(ob.val1+ob.val2)/2;
}
```

In the above example mean() is declared as friend function that computes mean value of two numbers that are input using getdata() function.

- A friend function although not a member function, has full access right to the private and protected members of the class.
- A friend function cannot be called using the object of that class. It can be invoked like any normal function.
- They are normal external functions that are given special access privileges.
- It cannot access the member variables directly and has to use an object name.membername (Here, is a membership operator).
- The function is declared with keyword friend. But while defining friend function it does not use either keyword friend or :: operator.

Program 8.3 To indicate the use of friend function.

```
#include <iostream.h>
class myclass
{
        private:
                int a,b;
        public:
                void set_val(int i, int j);
                friend int add(myclass obj);
};
void myclass::set_val(int i,int j)
{
    a = i;
    b = j;
}
int add(myclass obj)
{
        return (obj.a+obj.b);
}
```

```
int  main()
{
        myclass object;
        object.set_val(34, 56);
        cout << "Sum of 34 and 56 is "<<add(object)<<endl;
        return 0;
}
```

```
Sum of 34 and 56 is 90
```

In the above program segment the function add() is a friend of class myclass. Since it is not a member of any class, it cannot be called like an object. Since it is a non-member function, add() should be declared inside a class under public or private or protected access specifier.

Points to remember

- A function name that has several definitions with respect to the number of arguments and type of arguments is known as function overloading.
- Function overloading implements polymorphism
- Inline functions are member functions defined inside a class.
- The function code is written along with function definition inside a class
- Friend function is a non member function of a class that has access to both private and protected access members

Review questions

One mark questions

1. What is meant by function overloading?
2. When is function overloading needed?
3. Write an advantage of function overloading.
4. Name one condition for overloading of functions.
5. What is an inline function?
6. Write one advantage of inline function.
7. The inline function is always converted to a code block. True/false
8. What is a friend function?
9. Write an example for friend function declaration.
10. Write one condition for using a friend function.
11. Can the keyword friend be used while declaring a friend function?
12. Overloading a function means using different names to perform the same operations. True/false

Two marks questions:

1. What is meant by overloading implements polymorphism?
2. Write example for definition and declaration of function overloading
3. What are the restrictions on overloaded functions?
4. What are inline functions? Give an example.
5. Write the advantages of inline functions.
6. Create an inline function to check if a number is prime or not.
7. When are friend functions used? Write syntax for declaration of friend function.
8. Write any two characteristics of friend function.

Three mark questions:

1. Write any three reasons for function overloading.
2. What are the advantages of inline functions?
3. Write the advantages and disadvantages of inline functions.
4. When is it possible that an inline function may not work?
5. Write any three characteristics of friend function.

Five mark questions:

1. Explain the need for function overloading.
2. Discuss overloaded functions with syntax and example.
3. Explain inline functions with syntax and example.
4. Explain friend functions and their characteristics.

****