Chapter-8

FUNCTION OVERLOADING AND MEMBER FUNCTION

➢ Introduction:

- User defined function is a function defined by the user to solve his/her problem. Such a function can be called from anywhere and any number of times in the program.
- C++ implements polymorphism through function overloading and operator overloading.
- Function overloading allows the user to create new abstract data type.

➢ Function Overloading:

- Function Overloading means two or more functions have same name, but differ in the number of arguments or data types of arguments.
- Function overloading is the process of defining same function name to carry out similar types of activities with various data items.

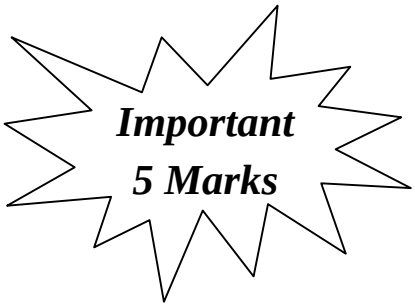
➢ Definition and Declaration of overloaded functions:

- The main factor in function overloading is a functions argument list.
- 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:

```
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:

```
int product ( int p, int q, int r);
float product ( float x, float y);
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 * y <<endl;
}
```



➢ Calling Overloaded Functions:

- The following program shows how overloaded functions can be called.

Program: To compute area of rectangle, circle, triangle using overloaded functions.

```
#include<iostream.h>
#include<conio.h>
class funoverloaded
{
        public:
                int area ( int l, int b)                    // area of rectangle
                {
                        return (l * b);
                }
                float area ( float r)                       // area of circle
                {
                        return (3.14 * r * r);
                }
                float area (float b, float h)
                {
                        return (0.5 * b * a);               // area of triangle
                }
};
void main( )
{
        funoverloaded f;
        clrscr( );
        cout<<" Area of Rectangle:"<<f.area(4,6)<<endl;
        cout<<"Area of Circle:"<<f.area(10)<<end;
        cout<<"Area of Triangle:"<<f.area(3.0, 7.0)<<endl;
        getch();
}
```

OUTPUT:

Area of Rectangle: 24

Area of Circle: 314.2857

Area of Triangle: 10.5

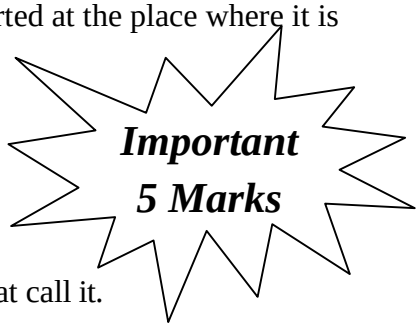
➢ Need for function overloading:

- The advantage of function overloading are:
    - 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.

➢ Restrictions on Overloaded Functions:

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

➢ Inline Function:



- An ***Inline*** function is a special type of function whose body is inserted at the place where it is called, instead of transferring the control to the function.
- The keyword ***inline*** is used to define inline function.
- Rules:
  - Inline function definition starts with keyword ***inline***.
  - The inline function should be defined before all function that call it.
  - The compiler replaces the function call statement with the function code itself (expansion) and then compiles the entire code.
- The general format for the inline function declaration is given below:

```
inline Returntype Fun_Name ( [Argument] )
{
        …….. ;
        return expression ;
}
```

- **Program to find the cube of a number using inline function:**

```
#include<iostream.h>
#include<conio.h>
inline int cube ( int a )
{
    return a * a * a;
}
void main( )
{
    int n ;
    clrscr( );
        cout<<"Enter the input number"<<endl;
    cin>>n;
    cout<<"Cube of " <<n<<" = "<<cunbe(n);
    getch();
}
```

OUTPUT:

Enter the input number

4

Cube of 4 = 64

✓ Advantage of inline function:

- The size of the object code is considerably reduced.
- The speed of execution of a program increases.
- Very efficient code can be generated.
- There is no burden on the system for function calling.
- It also saves the overhead of return call from a function.
- The readability of the program increases.

✓ **Disadvantage of inline function:**

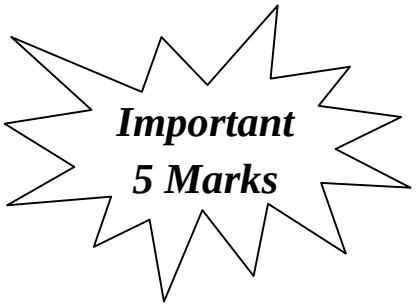
- May increase the size of the executable file
- More memory is needed.
- If used in header file, it will make your header file size large and may also make it unreadable.

✓ **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.

➢ Friend Function:

- A friend function is a non-member function of a class has the access permission to the private member of the 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.
- The general format for the friend function is given below:




```
class class_name
{
        public:
                friend return_type function_name ( [arguments] );
}
```

✓ **The friend functions have the following properties:**

- Friend function is not a member function of the class, has full access permission to private and protected members of the class.
- It can be declared either in public or private part of a class.
- A friend function cannot be called using the object of that class. It can be invoked like any normal function.
- The function is declared with keyword friend. But while defining friend function it does not use either keyword friend or : : operator.
- They are normal external functions that are given special access privileges.
- It cannot access the data member variables directly and has to use an object name.membername.
- Use of friend function is rare, since it violates the rule of encapsulation and data hiding.

✓ **Program to check a number is even or odd using a friend function.**

```
#include<iostream.h>
#include<conio.h>
class number
{
    private:
            int a;
    public:
            void readdata( )
            {
                    cout<<"Enter the Number"<<endl;
                    cin>>a;
            }
            friend int even(number);
};
int even(number n)
{
    if(n.a % 2 = = 0)              //friend function can access the private data a of the object n
            return 1;
    else
            return 0;
}
void main( )
{
    number num1;
    clrscr( );
        num1.readadata( );
        if( even(num1) )                        //friend function call
                cout<<"Number is Even";
    else
            cout<<'Number is Odd";
    getch();
}
```

OUTPUT:

```
Enter the Number
10
Number is Even
Enter the Number
11
Number is Odd
```

CHAPTER 8 – Function Overloading and Member Function BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	-	-	01 Question	01 Question
-	-		Question No 32	05 Marks

Important Questions

5 Marks Question:

1. What is function overloading? Explain the need for function overloading.
2. Discuss overloaded functions with syntax and example.
3. What is inline function? Write a simple program for it.
4. Mention the advantage and disadvantage of inline function.
5. Explain friend function and their characteristics.
6. Program to check whether a number is prime or not using inline function:

```
#include<iostream.h>
#include<conio.h>
inline int prime ( int n )
{
        for(int i=2; i<n/2; i++)
                if ( n % i = = 0)
                        return 0;
                return 1;
}
void main( )
{
        int num ;
        clrscr( );
        cout<<"Enter the input number"<<endl;
        cin>>num;
        if ( prime(num))        //inline function call
                cout<<"Number is Prime " ;
        else
                cout<<"Number is not a Prime " ;
        getch();
}
```

OUTPUT:

Enter the input number

5

Number is Prime