Note Part-2

Part I: Object Oriented Programming using C++

Topics: Member functions, Friend functions, constant member functions.

Prepared
By
Bhupesh Talukdar

Member functions

A function declared inside a class is called a member function of the class. A member function can be defined in two ways-

1. Inside the class definition (called inline definition of function)
2. Out of the class definition

Inside the class definition (called inline definition of function)

The syntax for inside the class definition is –

```
class   class_name
        {
                ………
                Access_specifier:
                                return_type   function_name(Argument_list)
                                {
                                        Body of the function
                                }
        };
```

For example,

```
class complex
{
  int real,imag;
  public:
        void read()
        {
                cout<<"Enter the real and imaginary parts: ";
                cin>>real>>imag;
        }
};
```

Here, the member functions read() has been defined inside the class definition.

Out of the class definition

To define a member function of a class out of the class definition the scope resolution operator(::) is used.

The syntax for outside the class definition is –

```
class   class_name
        {
                ………
                Access_specifier:
                                return_type   function_name(Argument_list);


        };
Return_type        Class_name::Function_name(Argument_list)
{
                Body of the function
}
```

For example,

```
class complex
{
  int real,imag;
  public:
        void read();
};
void  complex::read()
{
        cout<<"Enter the real and imaginary parts: ";
        cin>>real>>imag;
}
```

Friend Function

A friend function is non-member function of a class that can access the private and protected members of the class. A function can be made friend of a class by declaring it with the keyword *friend* inside the class. A friend function is defined out of the class like normal function and the keyword friend is not used in the function definition .

The syntax for the declaration of friend function is-

```
        class   class_name
        {
                ………
                ……….
                friend   return_type   function_name(Argument_list);
        };
```

The syntax for the definition of friend function is-

```
        return_type   function_name(Argument_list);
        {
                Body of the function
        }
```

The following program shows the use of friend function to add two complex numbers.

```
#include<iostream.h>
#include<conio.h>
class complex
{
  int real,imag;
  public:
        void read()
        {
                cout<<"Enter the real and imaginary parts: ";
                cin>>real>>imag;
        }
        void display()
        {
                cout<<"Real part: "<<real<<endl;
```

```
                        cout<<"Imaginary part: "<<imag<<endl;
            }
            friend void add_complex(complex A,complex B);
};

void add_complex(complex A,complex B)
{
   complex C;
   C.real=A.real+B.real;
   C.imag=A.imag+B.imag;
   C.display();
}
int main()
{
  complex X,Y;
  X.read();
  Y.read();
  add_complex(X,Y);
  getch();
  return(0);
}
```

Use of friend function

Friend functions are used in two purposes-

1. In operator overloading
2. If a particular function is requires to be shared by two or more classes. In this case, the friend function is declared as friend function in all the classes.

Constant member function

A constant member function of class can only read and retrieve the data members of the calling object of the class. i.e. the data members cannot be modified through constant member function.
The keyword *const* is used to declare and define a constant member function.
The following program shows the use of constant member function-

```
#include<iostream.h>
#include<conio.h>
class student
{
            int rollno;
      public:
      void read();
      void display() const;
};

void student::read()
{
     cout<<"Roll No.: ";
     cin>>rollno;
}
void student::display()  const
{

     cout<<"Roll No.: "<<rollno;

}

int main()
 {
    student d;
   student d1;
   d.read();
   d.display();
   d1.read();
   d1.display();
   getch();
   return 0;
}
```