OBJECT ORIENTED PROGRAMMING

Classes and Objects

Zuhaib A. Shaikh,
Asst. Prof., CSE Deptt.,QUEST
Web: zuhaib-shaikh.neocities.org

Introduction

- The class and object approach lead the program towards object oriented programming
- Enforces modularity
- Class is description that consists members as a single unit
- Members of class can be:
  - Data members (attributes), which carry data specified with-in class
  - Member functions (methods), which operate data members of same class as described
- Class itself is just description and doesn't exist physically (i.e. doesn't allocate memory)
- Variable or instance of class is called an object
- Object is physical existence of a class and allocate memory
- Many objects of same class can be created according to requirements
- Relationship b/w class and its object is similar to structure and its variable
- Data encapsulation and data hiding is provided with help of classes & objects

Classes and Objects

- The class is usually defined globally in program
- While its objects can be created locally or globally.
- Syntax

```
class class_name
{
access_specifier:
        members:
 access_specifier:
        members:
}Objects(optional);
```

```
class COMPLEX
{
private:
        int real, img;
 public:
        void display(void)
        {
                cout<<real<<"+"<<img;
        }
};
```

- The access specifiers in the class provides data hiding
- Allows programmers to restrict access to members of class

Classes and Objects

- Access Specifiers:
  - The private specifier allows the members to be only accessed with-in the class in which those are declared/defined
  - While, public allows members to be used within and outside of the class
    - The public members can be accessed outside with help of the object
- Public access specifier is used by default
- The objects of a class can be declared like variables
- Syntax

  Class_name Object_name; **COMPLEX c1;**
- Public members of an object can be accessed using member **.** Operator
- Syntax

  Object_name .Member_name; **c1.display();**
- Each object can refers to its members only

Constructor

- It is a special member function of the class
- Automatically called when object of the class is created
- Conditions:
  - Constructor should always be defined as public
  - Name of the constructor should be same as the name of class
- It is used for variety of tasks including assigning initial values to the data members or helps in creating temporary objects of the class
- E.g.

```
class COMPLEX
{
        private:
                int real, img;
         public:
                void COMPLEX(void)
                {
                        real = 0;
                        img = 0;
                }
};
```

Destructor

- It is a special member function of the class
- Automatically called when object of the class is destroyed (deallocated from memory)
- Conditions:
  - Destructor should always be defined as public
  - Name of the destructor should be same as the name of class starting with tilde ~ operator
  - Destructor can't return any value not takes any parameters
- E.g.

```
class COMPLEX
{
        private:
                int real, img;
         public:
                ~COMPLEX()
                {
                        cout<<"Object deallocated from memory"<<endl;
                        cout<<"Object destroyed";
                }
};
```

Friend Function

- A special function, allowed to access all members of the class with help of its object
- Not considered as class's method
- Declared inside class with friend keyword
- Defined and called like simple function
  - e.g.

```
class COMPLEX
{
        private:
                int real, img;
                char sign;
        public:
                friend void display(COMPLEX);
};
void display(COMPLEX C)
{
        cout<<C.real<<C.sign<<C.img<<endl;
}
```

Friend Class

- A class can be created as friend
- Friend class is allowed to access all members of a class(in which it was declared)
- Declared inside class with friend keyword
- Defined and called like simple class
  - e.g.

```
class NUMBER
{
        private:
                int val;
        public:
                friend class dispnum;
};
class dispnum
{
        public:
        void display (void)
        {
                NUMBER N;
                cout<<N.val<<endl;
        }
};
```

Static Members

- A method and/or attribute can be declared as static inside a class and then globally declared outside of class with scope resolution operator
- Single static member is used for all objects of the class
- Static attribute shares same memory space among all objects
- While, static method can be called without creating its object
  - e.g.

```
class COUNTER
{
        private:
                static int val;
        public:
                COUNTER()
                {
                        val++;
                }
                static display ()
                {
                        cout<<val<<endl;
                }
};

int COUNTER::val = 0;

COUNTER C1, C2;
COUNTER::display();
COUNTER C3;
COUNTER::display();
```

Questions