# B.Sc. Sixth Semester

# Computer Science(General)

# Paper: E601

## Part I: Object Oriented Programming using C++

## Topics: Constructor and Destructor, Inheritance

**Prepared by : Bhupesh Talukdar**

---

## Constructor

A constructor is a special member function of a class that initializes the objects of the class. It constructs a storage area for the data members of an object by allocating and initializing the memory for them.

For example,

```
class   Time
{
        int   hr,mn,sec;
        public:
                Time()
                {
                        hr=0;
                        mn=0;
                        sec=0;
                }

};
```

Here, *Time()* is a constructor.

A constructor has the following properties-

1. The name of the constructor must be same as its class name
2. A constructor does not have any return type, not even void. i.e. constructors never return any value.
3. A constructor is invoked automatically, whenever an object of its class is created.
4. A constructor is declared as public.
5. Constructors can be overloaded. i.e. a class can have more than one constructor.

**Types of constructor**

There are three types of constructors.

1. Default constructor
2. Parameterized Constructor
3. Copy Constructor

**Default Constructor**

A constructor that doesn't take any argument is called a default constructor.

For example-

```
class   Time
{
        int   hr,mn,sec;
public:
        Time()
        {
                hr=0;
                mn=0;
                sec=0;
        }

};
```

Here, *Time()* is a default constructor.

**Parameterized Constructor**

A constructor that takes values as arguments to initialize the data members of the class is called a parameterized constructor.

For example:

```
class   Time
{
        int   hr,mn,sec;
public:
        Time(int  a,int b,int c)
        {
                hr=a;
                mn=b;
                sec=c;
                }

        };
```

Here, *Time()* is a parameterized constructor. In case of parameterized constructor, the parameters are passed to the constructor during the declaration of the objects of the class.

For example,

```
Time        A(2,30,15);
```

**Copy Constructors**

A constructor that copies an existing object to a new object. i.e. a copy constructor takes an existing object as argument and copies the values of data members of the object into the corresponding data members of the new object.

For example,

```
        class   Time
                {
                        int   hr,mn,sec;
                public:
                        Time(int  a,int b,int c)
                        {
                                hr=a;
                                mn=b;
                                sec=c;
                        }
                        Time(Time      & T)
                        {
                                hr=T.hr;
                                mn=T.mn;
                                sec=T.sec;
                        }
                };
```

Here, *Time(Time & T)* is a copy constructor. Here *T* is an existing object.

The following program shows the use of the three types of constructors.

```
        #include<iostream.h>
        #include<conio.h>
        class   Time
        {
                int   hr,mn,sec;
            public:
                Time()
                {
                        hr=0;
                        mn=0;
                        sec=0;
                }
                Time(int  a,int b,int c)
                {
                        hr=a;
                        mn=b;
                        sec=c;
                }
                Time(Time      & T)
                {
                        hr=T.hr;
                        mn=T.mn;
                        sec=T.sec;
                }
                void display();
        };
        void Time::display()
        {
                cout<<hr<<"  Hour, "<<mn<<" Minute, "<<sec<<" Second"<<endl;
        }
        int main()
        {
                Time   T1;
                Time   T2(2,28,30);
                Time   T3(T2);
                T1.display();
                T2.display();
                T3.display();
                getch();
                return(0);
}
```

The output of the program will be as follows-

```
0 Hour, 0 Minute, 0 Second
2 Hour, 28 Minute, 30 Second
```

2 Hour, 28 Minute, 30 Second

## Destructor

A destructor is a special member function that destroys the objects of a class as soon as the scope of object ends. Like constructor the destructor is automatically called  when the work of the object is over.

The syntax for destructor is same as the constructor, but the difference here is that a **tilde** ~ symbol is prefixed to the name of the function..

For example,

```
class   Time
{
        public:
                Time()
                {
                        cout<<"Constructor is called here";
                }
                ~Time()
                {
                        cout<<"Destructor is called here";
                }
};
```

~Time() is the destructor here.

A destructor has the following properties-

1. The name of the destructor must be same as its class name
2. A destructor does not have any return type, not even void. i.e. destructor never return any value.
3. A destructor is automatically called  when the work of the object is over.
4. A destructor is declared as public.
5. A class can have only one destructor. i.e. a destructor cannot be overloaded

## Inheritance

Inheritance is a property of object oriented programming language through which one class can acquire the properties of an another class. It is a mechanism of deriving a new class from an existing class, where the existing class is called base class or super class or parent class and the new class is called derived class or sub class or child class. The derived class can inherit the protected and public data members and member functions of the base class. i.e. the private members of any class cannot be inherited.

Inheritance allows reusability of codes. i.e. it provides facility to the classes to reuse an existing code.

## How to define a derived class

The syntax for defining a derived class is as follows-

```
class    Derived_Class_Name :  Access_specifier      Base_Class_Name

{

         Access_Specifier :

                              Data members

                              Member functions

         ……………

         …………..

};
```

For example,

```
class    shape

{

         protected :

                  float    length, height;

         public :

                  void     read()

                  {

                           cout<<"Enter length and height: "<<endl;

                           cin>>length>>height;

                  }

};
```

```
class	triangle : public	shape
{
		float	ar;
	public:
		void	find_area()
		{
			ar=(length*height)/2;
			cout<<"The area of the triangle is: "<<ar;
		}
};
```

Here, *shape* is the base class and *triangle* is the derived class.

**Modes of derivation of a class**

A class can be derived from a base class by using the following three modes-

1. **Private** : When a class is derived through *private* access specifier, all the protected and public members of the base class become private in the derived class.
2. **Public :** When a class is derived through *public* access specifier, then the protected and public members of the base class remain same in the derived class. i.e. if a member of the base class is protected then it becomes protected in derived class and if it is public member of the base class then it becomes public in derived class.
3. **Protected :** When a class is derived through *protected* access specifier, all the protected and public members of the base class become protected in the derived class.

Example of class derivation using private mode

```
class	shape
{
	protected :
		float	length, height;
	public :
		void	read()
```

```
                {
                        cout<<"Enter length and height: "<<endl;
                        cin>>length>>height;
                }
};

class   triangle : private      shape
{
                float   ar;
        public:
                void    find_area()
                {
                        read();
                        ar=(length*height)/2;
                        cout<<"The area of the triangle is: "<<ar;
                }

};
```

Example of class derivation using protected mode

```
class   shape
{
        protected :
                float   length, height;
        public :
                void    read()
                {
                        cout<<"Enter length and height: "<<endl;
                        cin>>length>>height;
                }
};

class   triangle : protected    shape
{
                float   ar;
        public:
                void    find_area()
                {
                        read();
                        ar=(length*height)/2;
                        cout<<"The area of the triangle is: "<<ar;
                }
};
```

Example of class derivation using public mode

```
class   shape
{
        protected :
                float   length, height;
        public :
                void    read()
                {
                    cout<<"Enter length and height: "<<endl;

                    cin>>length>>height;

          }

};


class     triangle : public         shape

{

          float     ar;

     public:

          void      find_area()

          {

                    ar=(length*height)/2;

                    cout<<"The area of the triangle is: "<<ar;

          }

};
```

The following program shows the use of inheritance-

```
     #include<iostream.h>
     #include<conio.h>
     class     student
     {
          protected :
               int       rollno;
               char      name[50];
          public :
               void      read_student()
               {
                    cout<<"Enter Roll No. : ";
                    cin>>rollno;
                    cout<<"Enter Name. : ";
                    cin>>name;
```

```
                }
        };

        class   science_student : public        student
        {
                int     mark_csc,mark_ph,mark_ch,mark_maths,total_marks;
            public:
                void    read_marks();
                void    ctotal();
                void    show();
        };

        void    science_student::read_marks()
        {
            cout<<"Enter the marks of Comuter Science,Physics,Chemistry and Mathematics:
        "<<endl;
            cin>>mark_csc>>mark_ph>>mark_ch>>mark_maths;
        }

        void    science_student:: ctotal()
        {
            total_marks=mark_csc+mark_ph+mark_ch+mark_maths;
        }

        void    science_student:: show()
        {
            cout<<"Roll No. : "<<rollno<<endl;
            cout<<"Name : "<<name<<endl;
            cout<<"Computer Science : "<<mark_csc<<endl;
            cout<<"Physics : "<<mark_ph<<endl;
            cout<<"Chemistry : "<<mark_ch<<endl;
            cout<<"Mathematics"<<mark_maths<<endl;
            cout<<"Total Marks : "<<total_marks<<endl;
        }

        class   arts_student : public   student
        {
                int     mark_eng,mark_edu,mark_his,mark_eco,total_marks;
            public:
                void    read_marks();
                void    ctotal();
                void    show();
        };

        void    arts_student::read_marks()
        {
                cout<<"Enter the marks of English,Education,History,Economics: "<<endl;
                cin>>mark_eng>>mark_edu>>mark_his>>mark_eco;
        }

        void    arts_student:: ctotal()
        {
                total_marks=mark_eng+mark_edu+mark_his+mark_eco;
        }

        void    arts_student:: show()
        {
                cout<<"Roll No. : "<<rollno<<endl;
                cout<<"Name : "<<name<<endl;
                cout<<"English : "<<mark_eng<<endl;
                cout<<"Education : "<<mark_edu<<endl;
                cout<<"History : "<<mark_his<<endl;
                cout<<"Economics"<<mark_eco<<endl;
                cout<<"Total Marks : "<<total_marks<<endl;
        }
        int main()
        {
                clrscr();
                science_student     S;
                arts_student    A;
                S.read_student();
                S.read_marks();
                S.ctotal();
                A.read_student();
                A.read_marks();
                A.ctotal();
                S.show();
                A.show();
                getch();
                return(0);
}
```

Here, *student* is the base class and *science_student* and *atrs_student* are the derived classes.