

## Unit I

1

# Fundamentals of Object Oriented Programming

## 1.1 : Introduction to Procedural, Modular, Generic and Object Oriented Programming Techniques

**Q.1 Compare procedure oriented programming vs. object oriented programming.**

[SPPU : Dec.-19, May-19 Marks 4]

**Ans. :**

Sr. No.	Procedural Programming Language	Object Oriented Programming Language (OOP)
1.	The procedural programming executes series of procedures sequentially.	In object oriented programming approach there is a collection of objects.
2.	This is a top down programming approach.	This is a bottom up programming approach.
3.	The major focus is on procedures or functions.	The main focus is on objects.
4.	Data reusability is not possible.	Data reusability is one of the important feature of OOP.
5.	Data hiding is not possible.	Data hiding can be done by making it private.
6.	It is simple to implement.	It is complex to implement.
7.	For example : C, Fortran, COBOL	For example : C++, JAVA

**1.2 : Need of Object Oriented Programming**

**Q.2 Explain the need of object oriented programming.**

**Ans. :**

1. Emphasis is on data rather than procedures.
2. Programs can be divided into known objects.
3. Data needs to be hidden from the outside functions.
4. New data needs to be added frequently for maintaining the code.
5. Objects need to communicate with each other.
6. Some common properties are used by various functions.

**1.3 : Fundamentals of OOP**

**Q.3 Explain the features of object oriented programming.**

[SPPU : May-17, Marks 3, May-18, Marks 4, Dec.-19, Marks 6]

**Ans. :- 1. Object :**

- Object is an instance of a class.
- Objects are basic run-time entities in object oriented programming.
- In C++ the class variables are called objects. Using objects we can access the member variable and member function of a class.
- **Example**

Fruit f1;

For the class **Fruit** the object **f1** can be created.

**2. Classes :** • A class can be defined as an entity in which data and functions are put together. The concept of class is similar to the concept of structure in C.

- **Example**

```
class rectangle
{
    private :
        int len, br;
```

```

public :
    void get_data ( );
    void area( );
    void print_data ( );
};

```

### 3. Data Encapsulation :

- **Definition :** Encapsulation means binding of data and method together in a single entity called class.
- The data inside that class is accessible by the function in the same class. It is normally not accessible from the outside of the component.

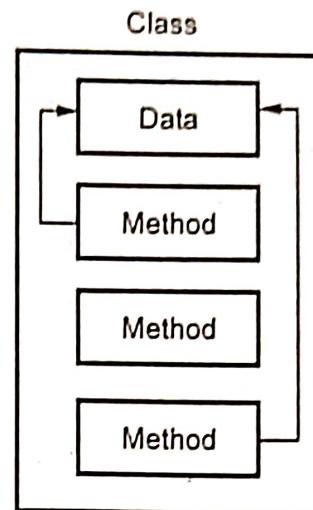
### 4. Inheritance :

- **Definition :** Inheritance is a property by which the new classes are created using the old classes. In other words the new classes can be developed using some of the properties of old classes.
- **Example :**
- Here the **Shape** is a base class from which the **Circle**, **Line** and **Rectangle** are the derived classes. These classes inherit the functionality **draw()** and **resize()**. Similarly the **Rectangle** is a base class for the derived class **Square**.

### 5. Polymorphism :

- **Definition :** Polymorphism is the ability to take more than one form and refers to an operation exhibiting different behavior in different instances (situations).

The behavior depends on the type of data used in the operation. It plays an important role in allowing objects with different internal structures to share the same external interface.



**Fig. Q.3.1 Concept of encapsulation**

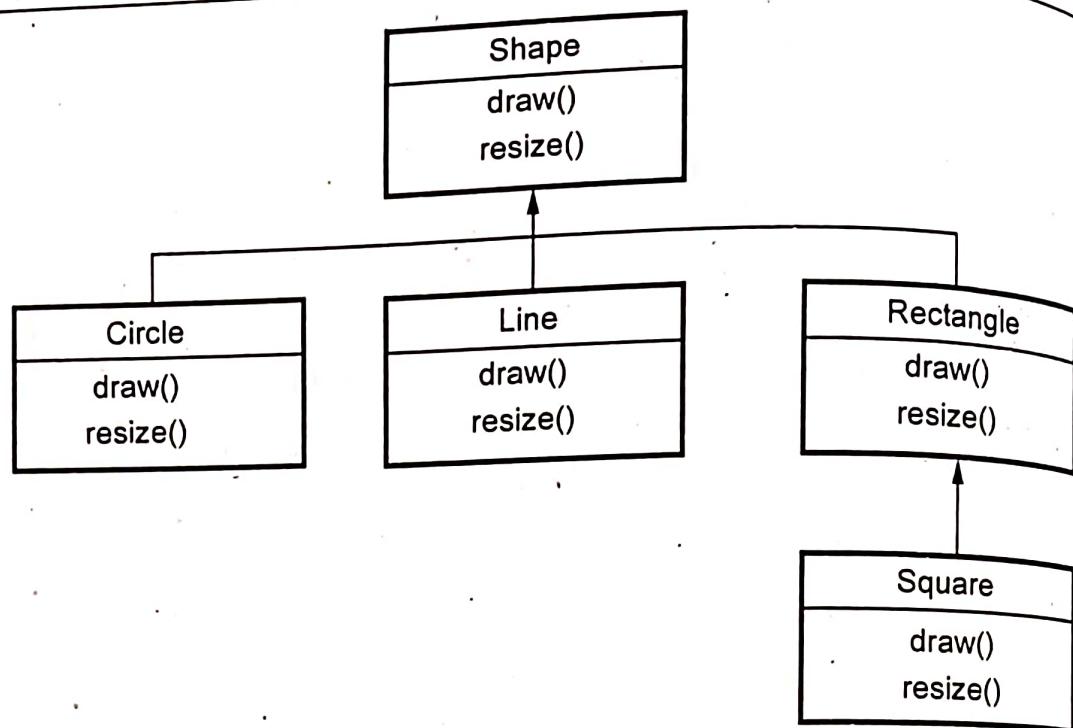


Fig. Q.3.2 Hierarchical structure of inheritance

**Q.4 What is the difference between data encapsulation and data abstraction ?**

**Ans. :**

Sr. No.	Data encapsulation	Data abstraction
1.	It is a process of binding data members of a class to the member functions of that class.	It is the process of eliminating unimportant details of a class. In this process only important properties are highlighted.
2.	Data encapsulation depends upon object data type.	Data abstraction is independent upon object data type.
3.	It is used in software implementation phase.	It is used in software design phase.
4.	Data encapsulation can be achieved by inheritance.	Data abstraction is represented by using abstract classes.

**1.4 : Benefits of OOP****Q.5 What are the benefits of OOP ?**

**Ans.** : Following are some advantages of object oriented programming -

1. Using **inheritance** the redundant code can be eliminated and the existing classes can be used.
2. The standard working **modules** can be created using object oriented programming. These modules can then communicate to each other to accomplish certain task.
3. Due to **data hiding** property, important data can be kept away from unauthorized access.
4. It is possible to create **multiple objects** for a given class.
5. For **upgrading the system** from small scale to large scale is possible due to object oriented feature.
6. Due to **data centered nature** of object oriented programming most of the details of the application model can be captured.
7. **Message passing technique** in object oriented programming allows the objects to communicate to the external systems.
8. **Partitioning the code** for simplicity, understanding and debugging is possible due to object oriented and modular approach.

**1.5 : C++ as Object Oriented Programming Language****Q.6 What is the difference between C and C++ ?**

**Ans. :**

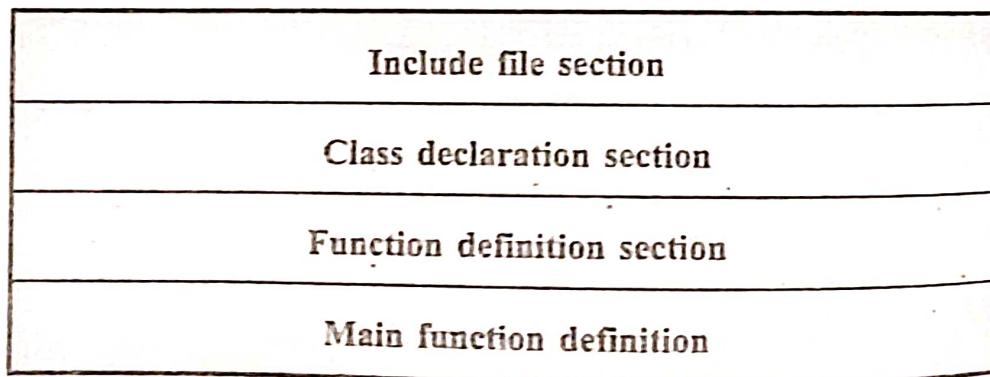
Sr. No.	C language	C++ language
1.	C is a procedure oriented language.	C++ is an object oriented programming language.
2.	C makes use of top down approach of problem solving.	C++ makes use of bottom up approach of problem solving.

3.	The input and output is done using <code>scanf</code> and <code>printf</code> statements.	The input and output is done using <code>cin</code> and <code>cout</code> statements.
4.	The I/O operations are supported by <code>stdio.h</code> header file.	The I/O operations are supported by <code>iostream.h</code> header file.
5.	C does not support inheritance, polymorphism, class and object concepts.	C++ supports inheritance, polymorphism, class and object concepts.
6.	The data type specifier or format specifier ( <code>%d</code> , <code>%f</code> , <code>%c</code> ) is required in <code>printf</code> and <code>scanf</code> functions.	The format specifier is not required in <code>cin</code> and <code>cout</code> functions.

### 1.6 : C++ Programming Basics

**Q.7 Give the structure of C++.**

**Ans. :** There are four sections in the structure of C++ program -



**Fig. Q.7.1 Structure of C++ program**

The sample C++ program is as follows-

#### C++ Program

##### FirstProg.cpp

```
//This is my first C++ program
//This program simply prints the message Hello World
#include<iostream>
using namespace std;
```

```

int main()
{
    cout<<"Hello World\n";
}

```

### 1.7 : Data Types

**Q.8 What are primitive data types and user defined data types ?**

[SPPU : Dec.-18, Marks 3]

**Ans. :**

Type	Size	Range
char	1 byte	- 127 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	- 127 to 127
int	4 bytes	- 2147483648 to 2147483647
unsigned int	4 bytes	0 to 4294967295
signed int	4 bytes	- 2147483648 to 2147483647
short int	2 bytes	- 32768 to 32767
unsigned short int	2 bytes	0 to 65,535
signed short int	2 bytes	- 32768 to 32767
long int	4 bytes	- 2,147,483,647 to 2,147,483,647
signed long int	4 bytes	same as long int
unsigned long int	4 bytes	0 to 4,294,967,295
float	4 bytes	$\pm 3.4 \times 10^{-38}$
double	8 bytes	$\pm 1.7 \times 10^{-308}$
long double	8 bytes	$\pm 1.7 \times 10^{-308}$

**1.8 : Structures**

**Q.9 Explain the concept of structures with suitable example.**

**Ans. :** Definition : A structure is a group of items in which each item is defined by some identifier. These items are called members of the structure. Thus structure is a collection of various data items which can be of different data types.

The Syntax of declaring structure in C++ is as follows -

```
struct name {
    member 1;
    member 2;
    ...
    ...
    ...
    member n;
};
```

**Example :**

```
struct stud {
    int roll_no;
    char name[10];
    float marks;
};

struct stud stud1,stud2;
```

The stud 1 and stud 2 will look like this

roll_no	name [10]	marks
---------	-----------	-------

**Q.10 Write a C program to represent a complex number using structure and add two complex numbers.**

**Ans. :**

```
#include<iostream>
using namespace std;
typedef struct Complex
{
    float real;
    float img;
}C;
```



```
void main()
{
    C x, y, z;
    cout<<"\n Enter the real part of first complex number: ";
    cin>>x.real;
    cout<<"\n Enter the imaginary part of first complex number: ";
    cin>>x.img;
    cout<<"\n Enter the real part of second complex number: ";
    cin>>y.real;
    cout<<"\n Enter the imaginary part of second complex number: ";
    cin>>y.img;
    cout<<"\n\t The First complex number is
"<<x.real<<"+ "<<x.img<<"i";
    cout<<"\n\t The second complex number is
"<<y.real<<"+ "<<y.img << "i";
    z.real = x.real + y.real;
    z.img = x.img + y.img;
    cout<<"\n The Addition is: "<<z.real<<"+ "<<z.img << "i";
}
```

#### **Output**

Enter the real part of first complex number: 3

Enter the imaginary part of first complex number: 6

Enter the real part of second complex number: 4

Enter the imaginary part of second complex number: 8

The First complex number is 3+6i

The second complex number is 4+8i

The Addition is: 7+14i

**1.9 : Enumeration**

**Q.11 What is enumeration ? Explain it with example.**

**Ans. :** To create user defined type names the keyword enum is used.

The syntax of using enum is

**enum name{list of names} variables;**

**For example :**

enum day{Monday,Tuesday,Wednesday} d;

d=Tuesday;

By default the value of first name is 0, second name has value 1 and so on. But we can give some other specific value to the list element. For example

enum day{Monday=10,Tuesday=20,Wednesday=30};

Following is a simple C++ program that illustrates the use of enum

```
*****
Program for using enum
*****
#include<iostream>
using namespace std;
int main()
{
    enum mylist{small,middle=5,large} a,b;
    a=middle;
    b=large;
    cout<<"a= "<<a<" b= "<<b;
    return 0;
}
```

**Output**

a = 5 b = 6

## 1.10 : Control Structures

**Q.12 Explain the conditional statements used in C++.**

**Ans. :** There are two types of if statements - simple if statement and compound if statement. The **simple if statement** is a kind of if statement which is followed by single statement. The **compound if statement** is a kind of if statement for which a group of statements are followed. These group of statements are enclosed within the curly brackets.

If statement can also be accompanied by the **else** part.

Following table illustrates various forms of if statement

Type of Statement	Syntax	Example
simple if	if(condition) statement	if(a<b) cout<<"a is smaller than b";
compound if	if(condition) { statement 1; ... ... }	if(a<b) { cout<<"a is smaller than b"; cout<<"b is larger than a"; }
if...else	if(condition) statement; else statement;	if(a<b) cout<<"a is smaller than b"; else cout<<"a is larger than b";

compound if...else

```

if(condition)
{
    statement 1;
    ...
}
else
{
    statement 1;
    ...
}

```

if(a&lt;b)

```

{
    cout<<"a is smaller
    than b";
    cout<<"b is larger than
    a";
}
else
{
    cout<<"a is larger than
    b";
    cout<<"b is smaller
    than a";
}

```

if...else if

```

if(condition)
{
    statement 1;
    ...
}

```

if(a&lt;b)

```

cout<<"a is smaller
than b";
else if(a<c)
    cout<<"a is smaller
    than b"
else
    cout<<"a is larger
    than b and c";

```

```

    }

else if(condition)

{

statement 1;

...

}

else

{

statement 1;

...

}

```

**Q.13 What is the use of switch-case statement ? Explain it with suitable example.**

**Ans. :**

From multiple cases if only one case is to be executed at a time then switch case statement is executed. Following table shows the use of switch case statements

Type of statement	Syntax	Example
switch ...case	switch(condition) { case caseno:statements break; .... default: statements	cout<<"\n Enter choice"; cin>>choice; switch(choice){ case 1: cout<<"You have selected 1"; break;

```

        }
        case 2: cout<<"You have selected 2";
        break;
        case 3: cout<<"You have selected 3";
        break;
        default: cout<<"good bye";
    }
}

```

### 1.11 : Arrays

**Q.14 What is arrays ? Explain the method of array initialization**

**Ans.** • Array is a collection of similar data type elements.

- Syntax of array is

Datatype Name[size]

- Example

int a[10];

This type of array is called **one dimensional array**.

**Method 1 :**

#### Syntax

Data type name[size]={value 0, value 1, value 2, ..., value n-1};

**Example**

int a[5]={10,20,30,40,50};

**Method 2 :**

```

cout<<"\n Enter the elements";
for(int i=0;i<n;i++)
    cin>>a[i];

```

- The arrays in which the elements are arranged in the form of rows and columns are called two dimensional arrays

- The initialization of two dimensional arrays is -

**Method 1 :**

```
int a[2][3]={  
    {10,20,30},  
    {40,50,60}  
};
```

**Method 2 :**

```
cout<<"\n Enter the elements";  
for(int i=0;i<n;i++)  
{  
    for(int j=0;j<m;j++)  
    {  
        cin>>a[i][j];  
    }  
}
```

**Q.15 Write a C++ program to store and retrieve the elements of an array.**

**Ans. :**

```
*****  
Program to store the elements in an array and then retrieve them  
*****  
  
#include<iostream.h>  
void main()  
{  
    int a[10],i,n;  
    cout<<"\n How many elements are there in the array?";  
    cin>>n;  
    cout<<"\n Enter the elements\n";  
    for(i=0;i<n;i++)  
        cin>>a[i];  
    cout<<"\n The elements of an array are ... \n";  
    for(i=0;i<n;i++)
```

```

cout<< " <<a[i];
}

}

```

**Output**

How many elements are there in the array?5

Enter the elements  
10 20 30 40 50

The elements of an array are ...  
10 20 30 40 50

### 1.12 : Strings

**Q.16 What is strings ? Explain the method of initialization of strings.**

**Ans. :** String is a collection of characters.

- **C/C++ representation :** In C / C++, the string is represented as array of characters. For example the string str can be represented as

char str[10];

- **Initialization of string :** Initializing of string can be done as follows -

char str[6]="INDIA"

Or

char str[]={'I','N','D','I','A'}

**Q.17 Write and explain the concept of string class used in C++.**

**Ans. :** The string class is used to define the strings. The syntax of using the string class is as follows -

string()

string(char \*str);

string(string &str);

**C++ Program**

```
#include<iostream>
#include<string>
using namespace std;
void main()
{
    string s1, s2;
    s1 = "Hello";
    cout << "\n The string : ";
    cout << s1;
    s2 = s1;
    cout << "\n The copied string is: ";
    cout << s2;
    s2 = "Friends";
    cout << "\n The new string is: " << s2;
    string s3;
    s3 = s1 + s2;
    cout << "\n The concatenated string is: ";
    cout << s3;
}
```

**Output**

The string : Hello  
The copied string is: Hello  
The new string is: Friends  
The concatenated string is:  
HelloFriends

**1.13 : Class and Object****Q.18 Define the term class.**

[SPPU : May-14, Marks 2]

- Ans. :**
- Each class is a collection of data and functions that manipulate the data.
  - Placing the data and functions together into a single entity is the central idea in object oriented programming.
  - The nouns in the system specification help the C++ programmer to determine the set of classes. The objects for these classes are created. These objects work together to implement the system.
  - Classes in C++ is the natural evolution of C notion of struct.

**For example :**

```
class rectangle
{
    private:
    int len,br;
public:
    void get_data();
    void area();
    void print_data();
};
```

**Q.19 Give the difference between structure and class.**

**Ans. :**

Sr. No.	Structure	Class
1.	By default the members of structure are <b>public</b> .	By default the members of class are <b>private</b> .
2.	The structure <b>can not be inherited</b> .	The class <b>can be inherited</b> .
3.	The structures <b>do not require constructors</b> .	The classes <b>require constructors</b> for initializing the objects.
4.	A structure contains <b>only data members</b> .	A class contains the <b>data as well as the function members</b> .

**Q.20 What is class and object ? Differentiate between class and object.**

[SPPU : Dec.-17, Marks 6]

**Ans. : Class :** Refer Q.18.

**Object :** The object is an instance of a class. Hence object can be created using the class name. The object interacts with the help of procedures or the methods defined within the class.

**Difference between Class and Object**

Sr. No.	Class	Object
1.	For a single class there can be any number of objects. For example - If we define the class as River then Ganga, Yamuna, Narmada can be the objects of the class River.	There are many objects that can be created from one class. These objects make use of the methods and attributes defined by the belonging class.
2.	The scope of the class is persistent throughout the program.	The objects can be created and destroyed as per the requirements.
3.	The class can not be initialized with some property values.	We can assign some property values to the objects.
4.	A class has unique name.	Various objects having different names can be created for the same class.

**1.14 : Class and Data Abstraction****Q.21 How does class play an important role in data abstraction ?****Explain****Ans. :**

- (1) Data abstraction is one of the characteristics of C++ by which simplified representation of the C++ program is presented to the user.
- (2) The implementation details are hidden.
- (3) Class is used to achieve data abstraction in C++.

**(4) Example**

```
class Student
{
```

**Private:**

```
    int roll;
    char name[10];
```

**public:**

```
    void input();
    void display();
```

};

In the main function, we can access the functionalities using the object.  
For instance -

```
Student obj;
obj input();
obj.display();
```

From above code the implementation details are hidden

### 1.15 : Class Scope and Accessing Class Members

**Q.22 Explain the method of accessing class members used in C++ with the help of suitable example.**

**Ans. :**

- (1) The class has two types of members - i) Data members and ii) member functions.
- (2) Typically the data members are declared as private and member functions are declared as public. Hence the data members of class are accessed used by member functions of a class.
- (3) For accessing the class members outside the class, the object of that class is created and used.
- (4) The member functions can be defined inside a class or outside a class. For defining the member functions outside the class the scope resolution operator is used.

**(5) Example :**

```
#include <iostream>
```

```
using namespace std;
```

```
class Test
```

```
{
```

```
private:
```

```
    int a,b,c;
```

Data members are declared as private.

```
public:
```

```
    int Addition(int a, int b)
```

```
{
```

```
        c = a+b;
```

```
        return c;
```

Member functions are declared as public.  
Note that: Only within the function the data members can be manipulated.

```

        }

    void Display( );
};

void Test::Display( ) //Using Scope resolution operator
{
    //display function is defined outside the class
    cout << "The sum is:" << c << "\n";
}

int main()
{
    Test obj;
    obj.Addition(10,20);
    obj.Display();
    return 0;
}

```

Only member functions  
 are accessible outside the  
 class because they are  
**public.**

### 1.16 : Access Specifiers

#### Q.23 Define the term - Member Access Control

 [SPPU : May 2014, 2 Marks]

**Ans. :** Member Access Control defines the manner by which the data can be accessible. The keywords public, private and protected are used to control access to member.

#### Q.24 Explain the terms private, public and protected.

**Ans. :** • By default the access specifier is of private type. When the data and functions are declared as private then only members of same class can access them. This achieves the **data hiding** property.

- The **public** access specifier allows the function declared outside the class to access the data and functions declared under it.
- The **protected** access specifier allows the functions and data declared under it be accessible by the belonging class and the immediate derived class. Outside the belonging class and derived class these members are not accessible.

### 1.17 : Separating Interface from Implementation

**Q.25 What are interface and implementation files ? Explain the advantages of separating interface from implementation.**

**Ans. :** The interface file is a file that contains the declaration of the class. The implementation files are the files that define the actual functionalities and invoke those functionalities.

Thus the C++ program can be split into

- **Header files** - contains class definitions and function prototypes
- **Source-code files** - contains member function definitions

The advantage of this arrangement is that the modification of the program becomes easy.

### 1.18 : Functions

**Q.26 What is function prototype ?**

**Ans. :** A function prototype is declaration of function without specifying the function body. That is, the function prototype specifies name of function, argument type and return type.

**Q.27 Explain call by value and call by reference with suitable examples**

**Ans. :** 1. **Call By Value** : In this method, actual values are passed as parameters to the function. For example

```
void sum(int x,int y)/*definition*/
{
    int c;
    c=x+y;
    cout<<"\n The Addition is: "<<c;
}
int main()
{
    int a,b;
    cout<<"\n Enter The two numbers: ";
```

```

    cin >> a;
    cin >> b;
    sum(a,b); /* call */
    return 0;
}

```

**2. Call By Reference :** In this method, the parameters are passed by reference. Pointers are used for passing this type of parameters. For example

```

void sum(int *x,int *y)//function definition
{
    int c;
    c = *x + *y;
    cout << "The addition of two numbers is: " << c;
}

int main()
{
    int a,b;
    void sum(int *,int *); //function declaration
    cout << "\n Enter The Two Numbers: ";
    cin >> a;
    cin >> b;
    sum(&a,&b); //call to the function
    return 0;
}

```

**Q.28 Consider the following declaration :**

```

class TRAIN
{
    int trainno;
    char dest[20];
    float distance;
    public:
    void get(); //To read an object from the keyboard
    void put(); //To write an object into a file
    void show(); //To display the file contents on the monitor
};

```

Complete the member functions definitions [SPPU : May-17, Marks 8]

Ans. :

```
#include <iostream>
#include <fstream>
using namespace std;
class TRAIN
{
protected:
    int trainno;
    char dest[20];
    float distance;

public:
    void get()
    {
        cout << "\n Enter trainno: ";
        cin >> trainno;
        cout << "\n Enter destination: ";
        cin >> dest;
        cout << "\n Enter distance: ";
        cin >> distance;
    }
    void show()
    {
        ifstream in_obj;
        in_obj.open("test.dat", ios::binary);
        in_obj.read((char*)this, sizeof(TRAIN));

        cout << "\n Train No : " << trainno;
        cout << "\n Destination : " << dest;
        cout << "\n Distance :" << distance;
    }
    void put()      // Member function for write file
    {
```

```

ofstream out_obj;
out_obj.open("test.dat", ios::app | ios::binary);
out_obj.write((char*)this, sizeof(TRAIN)); //writes current record
to file
}
};

int main()
{
    TRAIN obj;
    cout << "\nEnter the Record:";
    obj.get();
    cout << "\tWriting the Record....";
    obj.put();
    cout << "\nThe Record is:";
    obj.show();
    return 0;
}

```

### 1.19 : Accessing Function and Utility Function

**Q.29 Explain the concept of access and utility function.**

**Ans. :**

- **Access Function**

- It is a function that can read or display data.
- Another use of access function is to check the truth or false conditions. Such functions are also called as predicate functions.

- **Utility Function**

- It is a helper or supporting function that can be used by access function to perform some common activities.
- It is a private function because it is not intended to use outside the class.

**double division(double a, double b)//Utility Function**

```

{
    c = a / b;
}

```

```

    return c;
}
Void Get_data (double x, double y) // Access function
{
    a = x;
    b = y;
}

```

### 1.20 : Constructors

**Q.30 What is constructors ?**

**Ans.** : Constructors are the functions created for initializing the objects or data members of a class. The name of constructor is same as that of class name. The constructors must be declared in public mode.

**Q.31 Enlist the characteristics of constructors.**

**Ans.** : Following are some rules that must be followed while making use of constructors -

1. Name of the constructor must be same as the name of the class for which it is being used.
2. The constructor must be declared in the public mode.
3. The constructor gets invoked automatically when an object gets created.
4. The constructor should not have any return type. Even a void type should not be written for the constructor.
5. The constructor can not be used as a member of union or structure.
6. The constructors can have default arguments.

**Q.32 Write a C++ program for parameterized constructor.**

**Ans. : C++ Program**

```

#include<iostream>
using namespace std;
class image
{
    private:

```



```

        int height,width;
public:
    image(int x,int y) //constructor
    {
        height=x;
        width=y;
    }
    int area()
    {
        return (height*width);
    }
};

int main()
{
    image obj1(5,3);
    cout<<"The area is :"<<obj1.area()<<endl;
    return 0;
}

```

**Output**

The area is :15

**Q.33** Write a class called "arithmetic" having two integer and one character data members. It performs the operation on its integer members indicated by character member(+,-,\*,/) For example \* indicates multiplication on data members as  $d1*d2$ . Write a class with all necessary constructors and methods to perform the operation and print the operation performed in format Ans= d1 op d2. Test your class using main()

**Ans. :**

```

#include<iostream>
using namespace std;
class Arithmetic
{
    int d1,d2;
    char op;
public:

```

```
Arithmetic(int x,char c,int y)
{
    d1=x;
    d2=y;
    op=c;
}
int operation()
{
    int c;
    switch(op)
    {
        case '+':c=d1+d2;
                    break;
        case '-':c=d1-d2;
                    break;
        case '*':c=d1*d2;
                    break;
        case '/':c=d1/d2;
                    break;
    }
    return c;
};
int main()
{
    Arithmetic obj1(10,'+',20);
    Arithmetic obj2(20,'-',10);
    Arithmetic obj3(10,'/',5);
    Arithmetic obj4(10,'*',20);
    cout<<"\n Addition of 10+20= "<<obj1.operation();
    cout<<"\n Subtraction of 20-10= "<<obj2.operation();
```

```

cout << "\n Division of 10/5= " << obj3.operation();
cout << "\n Multiplication of 10*20= " << obj4.operation();
return 0;
}

```

**Output**

Addition of  $10+20= 30$   
Subtraction of  $20-10= 10$   
Division of  $10/5= 2$   
Multiplication of  $10*20= 200$

**Q.34 Write short note on - Copy Constructor:**

[Dec 2014, Marks 3, Dec 2015, Marks 2 ]

**Ans. :** *Definition* : Copy constructor is a special type of constructor in which new object is created as a copy of existing object.

- In other words in copy constructor one object is initialized by the other object. The general form of copy constructor is -

```

classname (classname &object)
{
    //body of the constructor
}

```

While invoking the copy constructor we will use following syntax

```
classname new_object_name(old_object_name);
```

Thus the copy constructor takes a reference to an object of same class as an argument.

**C++ Program**

```

class test
{
    int x;
public:
    test(); //default constructor
    test(int val) //parameterized constructor
    {
        x = val;
    }
}

```

```

    }
    test(test &obj) //copy constructor
    {
        x = obj.x;//entered the value in obj.x
    }
    void show()
    { cout << x; }
};

int main()
{
    test Old(100);
    //call for copy constructor
    test New(Old); //<---- object 'Old' is passed as argument to object
                    'New'
    cout << "\n The original value is: ";
    Old.show();
    cout << "\n The New copied value is: ";
    New.show();
    cout << endl;
    return 0;
}

```

The output of above program will be 100 for both original and copied value.

### 1.21 : Destructor

**Q.35 What is destructor ? Explain it with suitable example.**

**Ans. :** The destructor is called when the object is destroyed. The object can be destroyed automatically when the scope of the objects end (i.e. when the function ends) or explicitly by using operator delete.

The destructor must have the same name as the class but preceded with a tilde sign (~) and it must also return no value.

**C++ Program**

```
#include<iostream>
using namespace std;
class image
{
private:
    int *height,*width;
public:
    image(int,int);//constructor
    ~image();//destructor
    int area();//regular function
};
image::image(int x,int y)
{
    height=new int;//assigns memory dynamically using 'new'
    width=new int;
    *height=x;
    *width=y;
}
image::~image()
{
    delete height;//destroys the memory using 'delete'
    delete width;
}
int image::area()
{
    return (*height* *width);
}
int main()
{
    image obj1(10,20);
    cout<<"The area is :"<<obj1.area()<<endl;
    return 0;
}
```

**Output**

The area is : 200

### 1.22 : Object and Memory Requirements

**Q.36 Explain how memory is allocated for the objects in C++?**

- Ans. :
- The memory space objects is allocated when they are declared.
  - The members functions are created and placed in the memory only when they are defined as a part of a class specification.
  - As the objects are using the same member functions of the belonging class, no separate memory space is created for these function when objects are created.
  - Only for the member variables the separate memory space is created for each object.

Fig. Q.36.1 illustrates this concept

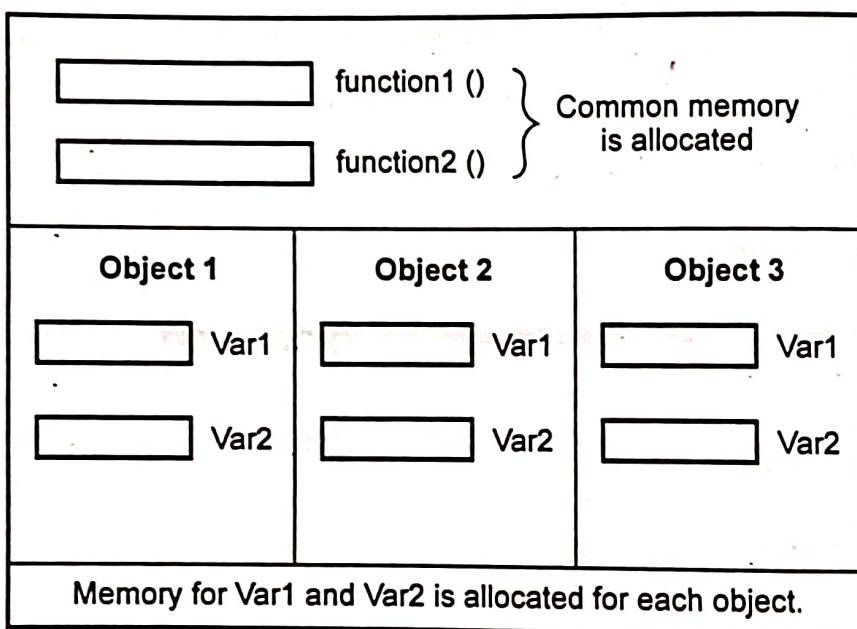


Fig. Q.36.1 Memory allocation for objects

### 1.23 : Static Members

**Q.37 Define the term-static data member**

[May 2014, Dec 2015, Marks 2]

Ans. : The static data members are basically the data members that are declared with keyword static. The static members have the same

properties as that of global variables. For example we can declare the static integer variable as follows

```
static int count;
```

**Q.38 Give a C++ code to illustrate the use of static data member.**

**Ans. :** Static data member is created to make the variable act like a global variable. For example

**C++ program using static keyword**

```
#include<iostream>
using namespace std;
void fun()
{
    static int cnt=0;
    cout << "\ncount = " << cnt;
    cnt++;
}
int main()
{
    for (int i = 0; i < 5; i++)
        fun();
    return 0;
}
```

**Output**

```
count = 0
count = 1
count = 2
count = 3
count = 4
```

**C++ Program without using static**

```
#include<iostream>
using namespace std;
void fun()
{
    int cnt=0;
    cout << "\ncount = " << cnt;
    cnt++;
}
int main()
{
    for (int i = 0; i < 5; i++)
        fun();
    return 0;
}
```

**Output**

```
count = 0
```

**Q.39 What is static member function ? Give an example for the same.** [SPPU : May-14, Marks 4, Dec.-19, Marks 3]

**Ans. :** (1) The static member functions are the member functions declared with the keyword **static**. This function is independent of any particular object of the class.

(2) A static member function can only access static data member, other static member functions and any other functions from outside the class.

### (3) Example Program

```

class Test
{
private:
    static int number;
public:
    static int Increment_count(); //static member function
};

int Test::Increment_count()
{
    return number++;
}

int Test::number = 0; // definition of static data
int main()
{
    Test a,b,c;
    //will increment the count 3 times
    cout << " a = " << a.Increment_count() << endl; //output:0

    cout << " b = " << b.Increment_count() << endl; //output:1
    cout << " c = " << c.Increment_count() << endl; //output:2
    return 0;
}

```

**1.24 : Inline Function****Q.40 Define the term - Inline function.**

[SPPU : May-14, Marks 2, Dec.-19, Marks 4]

**Ans. : Definition :** The inline function is a function whose code is copied in place of each function call. The inline specifies that the compiler should insert the complete body of function in every context where the function is used.

**Programming example :**

```
#include<iostream.h>
inline largest(int x,int y,int z)

{
    if(x>y&&x>z)
    {
        cout<<" First number is greatest and it is: "<<x;
    }
    else if(y>x&&y>z)
    {
        cout<<" Second number is greatest and it is: "<<y;
    }
    else
    {
        cout<<" Third number is greatest and it is: "<<z;
    }
}
```

**Q.41 Define class number which has inline function mult ( ) and cube ( ) for calculating the multiplication of 2 double numbers given and cube of the integer number given.**

[SPPU : Dec.-16, Marks 4]

**Ans. :**

```
#include <iostream>
using namespace std;
    inline double Mult(double x, double y)
```

```

    {
        return (x*y);
    }
    inline int cube(int n)
    {
        return n*n*n;
    }
}
int main()
{
    number obj;
    cout << "Mult (20,10): " << obj.Mult(20, 10) << endl;
    cout << "Cube (10): " << obj(cube(10)) << endl;
    return 0;
}

```

**Output**

Mult (20,10): 200  
 Cube (10): 1000

**1.25 : Friend Function**

**Q.42 What is friend function ? Explain with suitable example.**

[SPPU : Dec.-16, Marks 2, May-19, Marks 4]

**Ans. :** The friend function is a function that is not a member function of the class but it can access the private and protected members of the class.

The friend function is given by a keyword *friend*.

These are special functions which are declared anywhere in the class but have given special permission to access the private members of the class.

### C++ Program

```

#include<iostream>
using namespace std;
class test
{
    int data;
    friend int fun(int x); //declaration of friend function
}

```

```

public:
    test()//constructor
{
    data = 5;
}
int fun(int x)
{
    test obj;
    //accessing private data by friend function
    return obj.data + x;
}
int main()
{
    cout << "Result is = "<< fun(4)<< endl;
    return 0;
}

```

**Output**

**Result is = 9**

**Q.43 Explain merits and demerits of friend function.**

[SPPU : May-14, Marks 2]

**Ans. : Merits**

1. We can able to access the other class members in our class if, we use friend keyword.
2. We can access the members without inheriting the class.

**Demerits**

1. It can not be invoked by using object of particular class.
2. It can not be used to achieve runtime polymorphism.

**END... ↗**