Padasalai's Telegram Groups!

(தலைப்பிற்கு கீழே உள்ள லிங்கை கிளிக் செய்து குழுவில் இணையவும்!)

- Padasalai's NEWS - Group
https://t.me/joinchat/NIfCqVRBNj9hhV4wu6_NqA

- Padasalai's Channel - Group
https://t.me/padasalaichannel

- Lesson Plan - Group
https://t.me/joinchat/NIfCqVWwo5iL-21gpzrXLw

- 12th Standard - Group
https://t.me/Padasalai_12th

- 11th Standard - Group
https://t.me/Padasalai_11th

- 10th Standard - Group
https://t.me/Padasalai_10th

- 9th Standard - Group
https://t.me/Padasalai_9th

- 6th to 8th Standard - Group
https://t.me/Padasalai_6to8

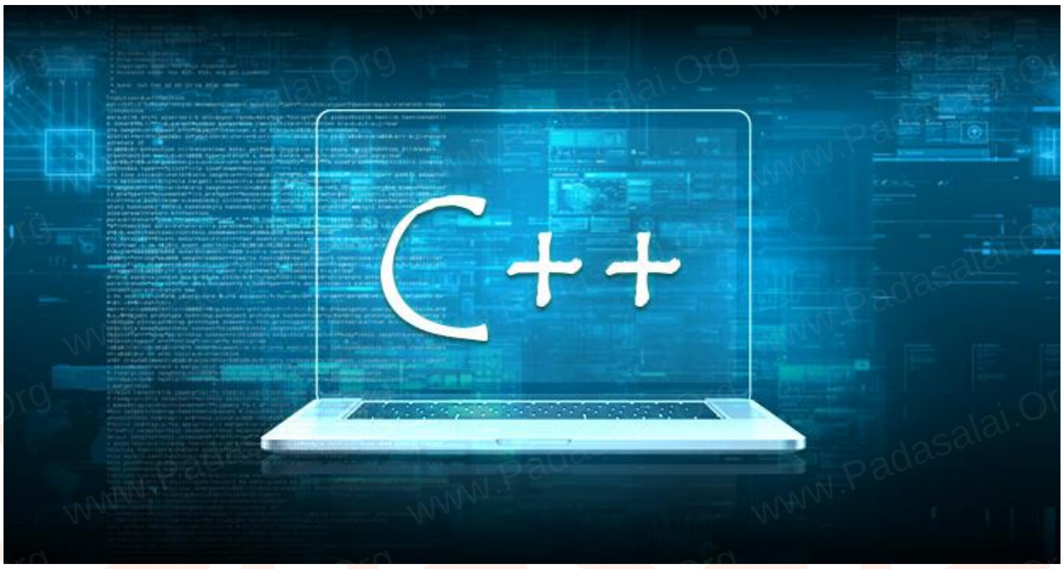
- 1st to 5th Standard - Group
https://t.me/Padasalai_1to5

- TET - Group
https://t.me/Padasalai_TET

- PGTRB - Group
https://t.me/Padasalai_PGTRB

- TNPSC - Group
https://t.me/Padasalai_TNPSC

Class : XI



Lesson -14

Class and Objects

Book Back Questions

Part -II

1. What are called members?
   - ✓ Class comprises of members. Members are classified as Data Members and Member functions.
   - ✓ Data members are the data variables that represent the features or properties of a class. Member functions are the functions that perform specific tasks in a class.
   - ✓ Member functions are called as methods, and data members are also called as attributes.
2. Differentiate structure and class though both are user defined data type.

SNO	STRUCTURE	CLASS
1	Members of a class are private by default	Members of a structure are public by default

2	struct Test
	{
	  int x; // x is public
	};
		class Test
		{
		  int x; // x is public
		};

3	A structure contains only member variables	A **class** can also contain functions [called methods] and member variables

3. What is the difference between the class and object in terms of oop?

Sno	Class	Object
1	The formation of a class doesn't allocate memory.	Creation of object consumes memory
2	A template or blueprint with which objects are created is known as Class.	An instance of a class is known as Object
3	Class is declared by using class keyword.	Object is invoked by new keyword

4. Why it is considered as a good practice to define a constructor though compiler can automatically generate a constructor?
   - ✓ When an object of the class is created a compiler can automatically generates a constructor if it is not defined.
   - ✓ It is considered that writing constructor for a class is a good practice because constructor takes over very important duty of initialization of an object being created and relieves us from this task
5. Write down the importance of destructor.
   - ✓ Destructor is used to de-allocate the memory of an object that was allocated by constructor.
6. What is class?

- ✓ **Class is a way to bind the data and its associated functions together**. Classes are needed to represent real world entities that not only have data type properties but also have associated operations.

7. Write the general form of class with example.

```
class  classname
{
          private:
                    member variable declarations;
                    member function declarations;
          public:
                    member variable declarations;
                    member function declarations;
          protected:
                    member variable declarations;
                    member function declarations;
};
```

8. Write the need for class.
   - ✓ **Class is a way to bind the data and its associated functions together**. Classes are needed to represent real world entities that not only have data type properties but also have associated operations. It is used to create user defined data type
9. What is data hiding?
   - ✓ **Data hiding** is one of the important features of Object Oriented Programming which allows preventing the functions of a program to access directly the internal representation of a class type.
   - ✓ The access restriction to the class members is specified by public, private, and protected sections within the class body.
10. Write syntax of defining non inline member function.

```
return_type class_name :: function_name (parameter list)
{
          function definition
}
```

11. What is the purpose of the class access specifier?
    - ✓ The access restriction to the class members is specified by public, private, and protected sections within the class body. The keywords public, private, and protected are called access specifiers. The default access specifier for members is private.
12. List the types of creating objects.
    - ✓ Local Object
    - ✓ Global Object
13. What is global object?

- ✓ An object is declared outside all the function bodies or by placing their names immediately after the closing brace of the class declaration then it is called as *Global objects*.
- ✓ These objects can be used by any function in the program

14. What is Local object?
    - ✓ An object is declared with in a function then it is called *local object*.It cannot be accessed from outside the function.
15. Write the general syntax of calling member function.

    The general syntax for calling the member function is:

    Object_name . function_name(actual parameter);
16. Write a short note on scope resolution operator.
    - ✓ If there are multiple variables with the same name defined in separate blocks then :: (scope resolution) operator will reveal the hidden file scope(global) variable.

    Example :

```
int a=100;

class A
{
int a;
public:
void fun()
{
a=20;
a+=::a; //using global variable value
cout<<a;
} };
```

17. What is container?
    - ✓ Whenever an object of a class is declared as a member of another class it is known as a container class. In the container-ship the object of one class is declared in another class.
18. What is the need of constructor?
    - ✓ The initialization of class type object at the time of declaration similar to a structure or an array is not possible because the class members have their associated access specifiers (private or protected or public).
    - ✓ Therefore Classes include special member functions called as *constructors*. The constructor function initializes the class object.
19. List the order of constructor invocation.
    - ✓ The constructors are executed in the order of the object declared. (If it is in same statement left to right)
    - ✓ For example

```
Test t1;
Test t2;
```

20. What is constructor?
    - ✓ When an instance of a class comes into scope, a special function called the *constructor* gets executed. The constructor function name has the same name as the class name.
    - ✓ The initialization of class type object at the time of declaration similar to a structure or an array is not possible because the class members have their associated access

specifiers (private or protected or public). Therefore Classes include special member functions called as *constructors*.

21. What is destructor?

✓ When a class object goes out of scope, a special function called the *destructor* gets executed. The destructor has the same name as the class tag but prefixed with a **~(tilde)**.Destructor function also return nothing and it does not associated with anydata type.

Part –III

1. Rewrite the following program after removing the syntax errors if any and underline the errors:

#include<iostream>
#include<stdio.h>
classmystud
{ intstudid =1001;
char name[20];
public
mystud( )
{ }
Void register()
{cin>>stdid;gets(name); }
	void display ( )
	{ cout<<studid<<":"<<name<<endl;}
	}
	int main( )
	{ mystud MS;
	register.MS( );
	MS.display( );
	}

Corrected Program :

```
#include<iostream>
#include<stdio.h>
using namespace std;
class mystud
{
int studid;
char name[20];
public:
mystud()
{
studid=0;
}
int register1()
{
        cin>>studid;
        gets(name);
}
void display()
{
cout<<studid<<":"<<name<<endl;
}
};
int main()
{
mystud MS;
MS.register1();
MS.display();
}
```

2. Write with example how will you dynamically initialize objects?
   - ✓ When the initial values are provided during runtime then it is called dynamic initialization.

     Example:

```
        #include<iostream>
        using namespace std;
        class X
        {
        float avg;
        public:
        X(float a)
        {
        avg=a;
        }
        void disp()
        {
        cout<<"\nAverage :- "<<avg;
        }
        };
        int main()
        {
        int avg;
        cout<<"\nEnter the Average";
        cin>>avg;
        X x(avg); // dynamic initialization
        x.disp();
        return 0;
}
```

3. What are advantages of declaring constructors and destructor under public accessibility?
   - ✓ A constructor can be defined either in private or public section of a class. But it is advisable to defined in public section of a class ,so that its object can be created in any function.

4. Given the following C++ code, answer the questions (i) & (ii).

```
class TestMeOut
{
public:
~TestMeOut() //Function 1
{cout<<"Leaving the examination hall"<<endl;}
TestMeOut() //Function 2
{cout<<"Appearing for examination"<<endl;}
void MyWork() //Function 3
{cout<<"Attempting Questions//<<endl;}
};
```

(i) In Object Oriented Programming, what is Function 1 referred as and when doesit get invoked / called ?

Ans: Its Destructor function and its executed automatically when instance of class goes out of scope.

(ii) In Object Oriented Programming, what is Function 2 referred as and when doesit get invoked / called ?

Ans: Its constructor function and its executed automatically when instance of class comes to scope.

5. Write the output of the following C++ program code :

```
#include<iostream>
using namespace std;
class Calci
{
char Grade;
int Bonus;
public:
Calci() {Grade='E'; Bonus=0;} //ascii value of A=65
void Down(int G)
{
Grade-=G;
}
void Up(int G)
{
Grade+=G;
Bonus++;
}
void Show()
{
cout<<Grade<<"#"<<Bonus<<endl;
}
};
```

```
int main()
{
Calci c;
c.Down(3);
c.Show();
c.Up(7);
c.Show();
c.Down(2);
c.Show();
return 0;
}
```

Output:

```
B#0
I#1
G#1
```

6. Write short note on class access specifier.
   - ✓ The access restriction to the class members is specified by public, private, and protected sections within the class body. The keywords public, private, and protected are called access specifiers. The default access specifier for members is private.

   Public:
   - ✓ A public member is accessible from anywhere outside the class but within a program. You can set and get the value of public data members even without using any member function.

   Protected:
   - ✓ A protected member is very similar to a private member but it provides one additional benefit that they can be accessed in child classes which are called derived classes (inherited classes).

   Private:
   - ✓ A private member cannot be accessed from outside the class. Only the class member functions can access private members. By default all the members of a class would be private.
7. Write a short note on non inline member function.
   - ✓ When Member function defined outside the class just like normal function definition (Function definitions you are familiar with) then it is be called as **outline member function or non-inline member function. Scope resolution operator (::) is used for this purpose.**

   Syntax:

   return_type class_name :: function_name (parameter list)
   {
   function definition
   }
8. Write a short note on memory allocation of objects.

- ✓ The member functions are created and placed in the memory space only when they are defined as a part of the class specification.
- ✓ Since all the objects belonging to that class use the same member function, **no separate space is allocated for member functions when the objects are created.**
- ✓ **Memory space required for the member variables are only allocated separately for each object** because the member variables will hold different data values for different objects

9. Write a short note on array objects.
   - An array which contains the class type of element is called array of objects.
   - It isdeclared and defined in the same way as any other type of array.

     Example:

     ```
     Class A
     {
     }a[5];
     ```

     Here **a[5]** is array objects of class A

10. Write a short note on nesting of member function.
    - We know that only public members of a class can be accessed by the object of that using dot operator.
    - A member function can call another member function directly using its name without using dot operator is called Nesting Member functions.

      Example:

      ```
      #include<iostream>
      using namespace std;
      class nest
      {
      public:
      int x=5,y=5;
      int sum( )
      {
      cout<<x+y; }
      int call( )
      {
      sum( ); }
      }a;
      void  main( )
      { a.call( ); }
      ```

11. Write a short note function returning objects.
    - Member function receive object as argument and it can also return an object.

    Example:

    ```
    #include<iostream>
    using namespace std;
    ```

```
class test
{
        public:
                int a;
                int set(int b)
                {
                        a=b;
                }
        test example(test c)
                {
                test d;
                d.a=c.a;
                return d;
                }
        int display()
                {
                cout<<a;
                }
};
int main()
{
test e,f;
e.set(10);
f.example(e);
e.display();
}
```

12. What is default constructor?

- A constructor that's accept no parameter is called default constructor.

For Example:

```
class test
{
public:
test( )
{
}
}t;
```

13. What is parameterized constructor?

- A constructor which can take arguments is called parameterized constructor .
- This type of constructor helps to create objects with different initial values.
- This is achieved by passing parameters to the function.
- Declaring a constructor with arguments hides the compiler generated constructor .After this we cannot invoke the compiler generated constructor.

For Example :

```
class test
{
public:
int A;
test(int x)
{
A=x;
}
}t(5);
```

14. What is copy constructer and how it's invoked?

- A constructor having a reference to an already existing object of its own class is called copy constructor .
- In other words Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type.
- It is usually of the form simple (simple&), where simple is the class name. The compiler provides a default Copy Constructor to all the classes.

Invoking of copy constructor:

- When an object is passed as a parameter to any of themember functions
    - Example void simple::putdata(simple x);
- When a member function returns an object
    - Example simple getdata() { }
- When an object is passed by reference to an instance of its own class
    - For example, simples1, s2(s1); // s2(s1) calls copy constructor

15. Write a short note on dynamic initialization of objects.

- When the initial values are provided during runtime then it is called dynamic initialization.

For Example:

```
#include<iostream>
using namespace std;
class test
{
        public:
        int set(int b)
        {
         cout<<b;
        }
        };
int main()
{
        int x;
        cout<<"\n Enter the value of x:";
        cin>>x;
        test e;
        e.set(x);
```

```
                    }
```

Part -IV

1. Explain nested class with example.

    - When one class becomes the member of another class then it is called Nested class and the relationship is called containership.

    Classes can be nested in two ways.

    - By defining a class within another class
    - By declaring an object of a class as a member to another class

    Defining a class with in another

    - When a class is declared with in another class, the inner class is called as Nested class (ie the inner class) and the outer class is known as Enclosing class.
    - Nested class can be defined in private as well as in the public section of the Enclosing class.

    Example:

```
          #include<iostream>
          using namespace std;
          class test
          {
               public:
               int a=10;
               int display()
               {
                    cout<<"\n"<<a;
                    s.print();
               }
     class exam
     {
     public:
     int print()
     {
     cout<<"\n hai ";
     }
     }s;
     };
     int main()
     {
     test e;
     e.display();
     }
```

2. Mention the differences between constructor and destructor

Sno	Constructor	Destructor
1	Constructor function is executed automatically when objects life time begins	Destructor function is executed automatically when objects life time ends
2	Constructor is overloaded	Destructor is not overloaded
3	Constructor have parameter	Destructor doesn't have

		parameter
4	Constructor is not inherited	Destructor also not inherited
5	Constructor is same name as class name	Destructor is same name as class name prefixed with ~ tidle character
6	In a class many constructor	In a class only one destructor

3. Define a class RESORT with the following description in C++ :

Private members:

Rno // Data member to store room number

Name //Data member to store user name

Charges //Data member to store per day charge

Days //Data member to store the number of days

Compute ( ) // A function to calculate total amount as Days * Charges and if the

//total amount exceeds 11000 then total amount is 1.02 * Days *Charges

Public member:

getinfo( ) // Function to Read the information like name , room no, charges and days

dispinfo ( ) // Function to display all entered details and total amount calculated

//using COMPUTE function

Program:

```
#include<iostream>
using namespace std;
class RESORT
{
        int Rno,Days;
        char name[50];
        float Charges,total;
        int compute()
        {
        total=Days*Charges;
        if(total>11000)
        {
                total=1.02*Days*Charges;
        }
        }
        public:
        int getinfo()
        {
        cout<<"\n Enter the rno,name,days,charges :";
```

```
          cin>>Rno>>name>>Days>>Charges;
          compute();
          }
          int dispinfo()
          {
          cout<<"\n Rno : "<<Rno;
          cout<<"\n Name :"<<name;
          cout<<"\n Days :"<<Days;
          cout<<"\n Charges :"<<Charges;
          cout<<"\n Total Wages:"<<total;
          }
};
          int main()
          {
                    RESORT R;
                    R.getinfo();
                    R.dispinfo();
          }
```

4. Write the output of the following

```
#include<iostream>
#include<stdio.h>
using namespace std;
class sub
{
int day, subno;
public :
sub(int,int); // prototype
void printsub()
{ cout<<" subject number : "<<subno;
cout<<" Days : " <<day;
}
};
sub::sub(int d=150,int sn=12)

{ cout<<endl<<"Constructing the object "<<endl;
day=d;
sub no=sn;
}
class stud
{
int rno;
float marks;
public:
stud( )
{ cout<<"Constructing the object of students "<<endl;
rno=0;
marks=0.0;
}
void getval()
{
cout<<"Enter the roll number and the marks secured ";
cin>>rno>>marks;
}
void printdet()
{ cout<<"Roll no : "<<rno<<"Marks : "<<marks<<endl;
}
};
class addmission {
sub obj;
stud objone;
float fees;
public :
add mission ( )
{ cout<< "Constructing the object of admission "<<endl;
fees=0.0;
}
void printdet( )
{ objone.printdet();
obj.printsub( );
cout<<"fees : "<<fees<<endl ;
}
};
int main()
{system("cls");
addmission adm;
cout<<endl<< "Back in main ( )";
return 0; }
```

Output:

```
Constructing the object
Constructing the object of students
Constructing the object of admission

Back in main ( )
```

5. Write the output of the following

```
#include<iostream>
#include<stdio.h>
using namespace std;
class P
{ public:
P ( )
{ cout<< "\nConstructor of class P "; }
~ P ( )
{ cout<< "\nDestructor of class P "; }
};
class Q
{ public:
Q( )
{ cout<<"\nConstructor of class Q "; }
~ Q( )
{ cout<< "\nDestructor of class Q "; }
};
class R
{ P obj1, obj2;
Q obj3;
public:
R ( )
{ cout<< "\nConstructor of class R ";}
~ R ( )
{ cout<< "\nDestructor of class R ";}
        };
        int main ( )
        {
        R R0;
        Q oq;
        P op;
        return 0;
        }
```

Output:

```
Constructor of class P
Constructor of class P
Constructor of class Q
Constructor of class R
Constructor of class Q
Constructor of class P
Destructor of class P
Destructor of class Q
Destructor of class R
Destructor of class Q
Destructor of class P
Destructor of class P
```

6. Explain about defining methods of class.

- Member functions are the functions that perform specific tasks in a class. Member functions are called as methods.
- The member functions of a class can be defined in two ways.
  - ✓ Inside the class definition
  - ✓ Outside the class definition

- **Inside the class definition**
  - When a member function is defined inside a class, it behaves like inline functions. These are called Inline member functions.
  - Example:

```
class Example
{
public:
int sum( )
{

}
};
```

- **Outside the class definition**
  - ✓ When Member function defined outside the class just like normal function definition (Function definitions you are familiar with) then it is be called as **outline member function or non-inline member function. Scope resolution operator (::) is used for this purpose.**

Syntax:

```
return_type class_name :: function_name (parameter list)
{
function definition
}
```

Example:

```
class Example
{
public:
int sum( );
};
int  Example::sum( )
{
}
```

7. Explain about creating objects with example.
   - ✓ A class specification just defines the properties of a class. To make use of a class specified, the variables of that class type have to be declared.
   - ✓ The class variables are called *object*. Objects are also called as *instance* of class.

- ✓ Objects can be created in two methods,
    - Global object
    - Local object
- ✓ **Global Object :**
    - If an object is declared outside all the function bodies or by placing their names immediately after the closing brace of the class declaration then it is called as *Global object*. These objects can be used by any function in the program
- ✓ **Local Object**
    - If an object is declared with in a function then it is called *local object*.It cannot be accessed from outside the function.

Example:

```
#include<iostream>
using namespace std;
class P
{
        int a;
        public:
                int display()
                {
                        cout<<"\n enter the value:";
                        cin>>a;
                        cout<<"\n Entered value is "<<a;
                }
};
P b; → Global object
int main()
{
        P c;→local object
        b.display();
        c.display();
}
```

8. Explain about how the objects are passed to function arguments.
    - Objects can also be passed as arguments to a member function just like any other data type of C++.Objects can also be passed in both ways
        (1) Pass By Value
        (2) Pass By Reference

Pass By Value

- When an object is passed by value the function creates its own copy of the object and works on it. Therefore any changes made to the object inside the function do not affect the original object.

Example:

```
#include<iostream>
#include<stdio.h>
using namespace std;
```

```
class P
{
    int a;
    public:
            int assign(int x)
            {
            a=x;
            }
            int printing(P a1, P a2)
            {
            a1.a=20;
            a2.a=30;
            cout<<"\n Changed Value "<<a1.a;
            cout<<"\n Changed Value"<<a2.a;
            }
};
int main()
{
    P c,b,d;
    b.assign(10);
    c.assign(10);
    d.printing(b,c);
}
```

- **Pass By Reference**
    - When an object is passed by reference , its memory address is passed to the function so the called function works directly on the original object used in the function call. So any changes made to the object inside the function definition are reflected in original object.

Example:

```
#include<iostream>
#include<stdio.h>
using namespace std;
class P
{
        int a;
        public:
                int assign(int x)
                {
                a=x;
                }
                int printing(P &a1, P &a2)
                {
                a1.a=20;
                a2.a=30;
                cout<<"\n Changed Value "<<a1.a;
                cout<<"\n Changed Value"<<a2.a;
                }
};
int main()
{
        P c,b,d;
        b.assign(10);
        c.assign(10);
        d.printing(b,c);
}
```

9. Explain about invocation of constructors.
    - There are two ways to create an object using parameterized constructor
        - Implicit call
        - Explicit call
    - **Implicit call**
        - In this method ,the parameterized constructor is invoked automatically whenever an object is created.
    - **Explicit call**
        - In this method ,the name of the constructor is explicitly given to invoke the parameterized constructor so that the object can be created and initialized .
        - Explicit call method is the most suitable method as it creates a temporary object, the chance of data loss will not arise. A temporary object lives in memory as long as it is being used in an expression. After this it gets destroyed.

Example:

```
        #include<iostream>
        using namespace std;
        class simple
        {
private:
        int a;
public:
        simple(int m)
        {
        a= m ;
        cout<< "\n Constructor of class-simple invoked for implicit and explicit call"<<endl;
        }
        void putdata()
        {
        cout<<"\nThe integers are... "<<a<<endl;
        }
};
```

```
int main()
{
simple s1(10); //implicit call
simple s2=simple(30); //explicit call
cout<<"\n\t\tObject 1\n";
s1.putdata();
s2.putdata();
return 0;
}
```

10. Explain about characteristic of constructor and destructors.

Characteristics of constructor:

- The name of the constructor must be same as that of the class
- No return type can be specified for constructor
- A constructor can have parameter list
- The constructor function can be overloaded
- They cannot be inherited but a derived class can call the base class constructor
- The compiler generates a constructor, in the absence of a user defined constructor.
- Compiler generated constructor is public member function
- The constructor is executed automatically when the object is created
- A constructor can be used explicitly to create new object of its class type

Characteristics of Destructor:

- The destructor has the same name as that of the class prefixed by the tilde character '~'.
- The destructor cannot have arguments
- It has no return type
- Destructors cannot be overloaded i.e., there can be only one destructor in a class
- In the absence of user defined destructor, it is generated by the compiler
- The destructor is executed automatically when the control reaches the end of class scope to destroy the object
- They cannot be inherited

All the Best

HSC I FIRST YEAR

11

COMPUTER SCIENCE

VOLUME –II

LESSON 15

Prepared By : M.Dhanapal., MCA.,B.Ed., 9790573672 Literacy Mission MHSS,Tirupur

LESSON 15

POLYMORPHISM

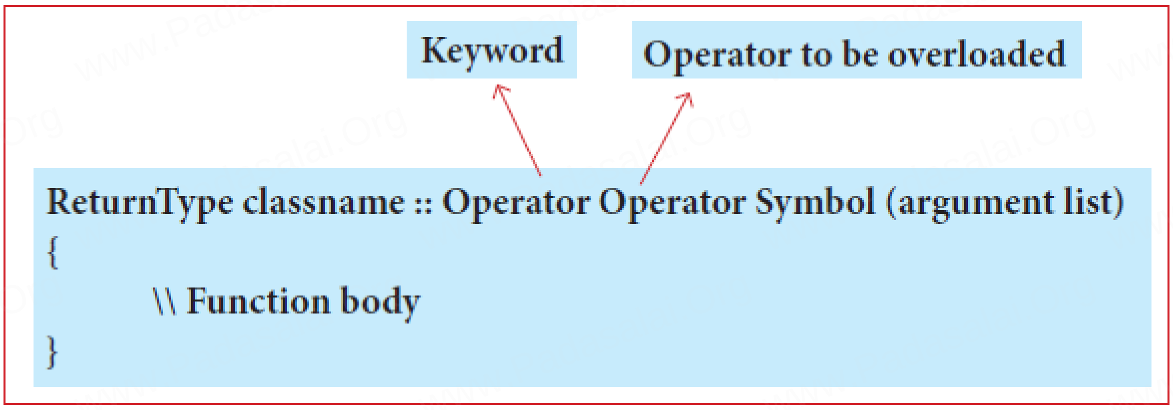
Part –II

1. What is function overloading?
   - ✓ The ability of the function to process the message or data in more than one form is called as function overloading.
   - ✓ In other words function overloading means two or more functions in the same scope share the same name but their parameters are different. In this situation, the functions that share the same name are said to be overloaded and the process is called function overloading.
2. List the operator that cannot be overloaded.
   - ✓ Operator that are not overloaded are follows
     - scope operator **::**
     - sizeof
     - member selector **.**
     - member pointer selector *****
     - ternary operator **?:**
3. class add{ int x; public: add(int) }; Write an outline definition for the constructor.

```
add ::add(int y)
{
        x=y;
 }
```

4. Does the return type of a function help in overloading a function?
   - ✓ No, The return type of overloaded functions are not considered for overloading same data type.
5. What is the use of overloading a function?
   - ✓ Function overloading is not only implementing polymorphism but also reduces the number of comparisons in a program and makes the program to execute faster.
   - ✓ It also helps the programmer by reducing the number of function names to be remembered.
6. What is overloaded resolution?
   - ✓ The process of selecting the most appropriate overloaded function or operator is called **overload resolution.**

7. What is overloaded function?
   - ✓ The term overloading means name having two or more distinct meaning. Then function have more than one distinct meaning is called overloaded function.
8. What is function signature?
   - ✓ The functions that share the same name are said to be overloaded and the process is called function overloading. The number and types of a function's parameters are called the **function's signature.**
9. Write the syntax of the operator overloading function.



```
ReturnType classname :: Operator Operator Symbol (argument list)
{
        \\ Function body
}
```

Part –III

1. What are the rules for function overloading?
   - ✓ The overloaded function must differ in the number of its arguments or data types.
   - ✓ The return type of overloaded functions is not considered for overloading same data type.
   - ✓ The default arguments of overloaded functions are not considered as part of the parameter list in function overloading.
2. How does a compiler decide as to which function should be invoked when there are many functions? Give an example.
   - ✓ When you call an overloaded function, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function with the parameter types specified in the definitions.
   - ✓ The process of selecting the most appropriate overloaded function or operator is called **overload resolution.**

   Example:

```
float area(int radius);                 → area(5);
float area(int length, int breadth);    → area(5,6);
```

3. What is operator overloading? Give some example of operators which can be overloaded.
   - ✓ The term operator overloading is giving additional functionality to an operator.
   - ✓ Overloaded operator are +,++,-,--,*,-=,+=,*=, etc.,
4. Discuss the benefit of constructor overloading?
   - ✓ A class can have more than one constructor with different signature.
   - ✓ Constructor overloading provides flexibility of creating multiple type of objects for a class.
5. class sale (int cost, discount ;public: sale(sale &); Write a non inline definition for constructor specified;

```
sales::sales(sales &a)
{
        cost=a.cost;
        discount=a.discount;
}
```

Part-IV

1. What are the rules for operator overloading?
   - ✓ Precedence and Associativity of an operator cannot be changed.
   - ✓ No new operators can be created, only existing operators can be overloaded.
   - ✓ Cannot redefine the meaning of an operator's procedure. You cannot change how integers are added. Only additional functions can be to an operator
   - ✓ Overloaded operators cannot have default arguments.
   - ✓ When binary operators are overloaded, the left hand object must be an object of the relevant class.
   - ✓ Operator that are not overloaded are follows
     - scope operator **::**
     - sizeof
     - member selector **.**
     - member pointer selector *****
     - ternary operator **?:**
2. Answer the question (i) to (v) after going through the following class.

```
class Book
{
int BookCode ; char Bookname[20];float fees;
public:
```

```
Book( ) //Function 1
{
fees=1000;
BookCode=1;
strcpy (Bookname,"C++");
}
void display(float C) //Function 2
{
cout<<BookCode<<":"<<Bookname<<":"<<fees<<endl;
}
~Book( ) //Function 3
{
cout<<"End of Book Object"<<endl;
}
Book (int SC,char S[ ],float F) ; //Function 4
};
```

(i) In the above program, what are Function 1 and Function 4 combined together referred as?

Answer:

constructors. function 1 refers default constructor and function 4 refers parameterized constructor

(ii) Which concept is illustrated by Function3? When is this function called/ invoked?

Answer: Destructor. When an instance of scope goes out special function destructor gets executed.

(iii) What is the use of Function3?

Answer: Destructor function will de-allocate the memory of an object

(iv) Write the statements in main to invoke function1 and function2

```
int  main( )
{
Book b,c1(2,"C++",185.6);
b.display(150.6);
}
```

(v) Write the definition for Function4.

```
Book(int SC,char S[ ],float F)
{
fees=F;
BookCode=SC;
Bookname=S;
}
```

3. Write the output of the following program. Refer the question Book back progam (3) in part -IV

```
Seminar starts now
Welcome to Seminar
Recap of Previous Seminar Content
Lectures in the seminar on
Vote of thanks
Vote of thanks
Vote of thanks
```

4. Debug the program (Refer the program in page no 260,Q/No : 4)

Sno	Error code	Resaon	Correct Code
1	charstr[20];	Data type and variable are separated by whitespace	char str[20];
2	void accept_string	Parenthesis missing in the function prototype	void accept_string( )
3	String operator *(String x)	Overloaded operator is +	String operator + (String x)
4	String s;	No need to declare the object. Because it is un-used in the program	
5	strcpy(s.str,str);	No need	
6	strcat(str,str);	Right hand side argument is mentioned wrongly	strcat(str,x.str);
7	Header file missing	While using strcat function we must use string.h header file	#include<string.h>
8	goto s;	No need	
9	}	Class not terminated	};
10	cout>>	cout object use << operator	cout<<

11	Str3=str1+str2; cout<<"\n\n Concatenated string is";	Swap the two coding	cout<<"\n\n Concatenated string is"; Str3=str1+str2;
12	str3.display_string();	No need	

5. Refer the page no :261 and Q/A:5

i.Mention the objects which will have the scope till the end of the program.

Ans: ob,ob1

ii. Name the object which gets destroyed in between the program

Ans: ob

iii. Name the operator which is over loaded and write the statement that invokes it.

Ans: == and ob==ob1;

iv. Write out the prototype of the overloaded member function

Ans: void comp::operator ==(comp ob)

v. What types of operands are used for the overloaded operator?

Ans: user defined operand and type of operator is binary-relational operator (==)

vi.Which constructor will get executed? Write the output of the program

Ans: Constructor : default constructor is executed in this program

Output :

Enter First String:computer

Enter Second String: Science

Strings are not Equal

6. Explain about function overloading with example.

Function Overloading:

- ✓ The ability of the function to process the message or data in more than one form is called as function overloading.
- ✓ In other words function overloading means two or more functions in the same scope share the same name but their parameters are different. In this situation, the functions that share the same name are said to be overloaded and the process is called function overloading.

Function Signature :

- ✓ The functions that share the same name are said to be overloaded and the process is called function overloading. The number and types of a function's parameters are called the **function's signature.**

Overloaded resolution:

- ✓ The process of selecting the most appropriate overloaded function or operator is called **overload resolution.**

Rules :

- ✓ The overloaded function must differ in the number of its arguments or data types.
- ✓ The return type of overloaded functions is not considered for overloading same data type.
- ✓ The default arguments of overloaded functions are not considered as part of the parameter list in function overloading.

Example:

```
#include<iostream>
#include<string.h>
using namespace std;
int sum(int x,int y,int z)
{
      return x+y+z;
}
int sum(int x1,int y1)
{
      return x1+y1;
}
int main()
{
cout<<sum(4,5,6)<<endl;
cout<<sum(5,6);
return 0;
}
```

Output:

```
15
11
```

7. Explain about operator overloading with example.
   - ✓ The term operator overloading, refers to giving additional functionality to the normal C++ operators like +,++,-,—,+=,-=,*.<,>.
   - ✓ It is also a type of polymorphism in which an operator is overloaded to give user defined meaning to it.
- ✓ Operator that are not overloaded are follows
  - scope operator **::**
  - sizeof
  - member selector **.**
  - member pointer selector ***** **and** ternary operator **?:**

Operator Overloading Syntax

```
returnType classname :: Operator Operator Symbol (argument list)
{
\\ Function body
}
```

Restrictions on Operator Overloading

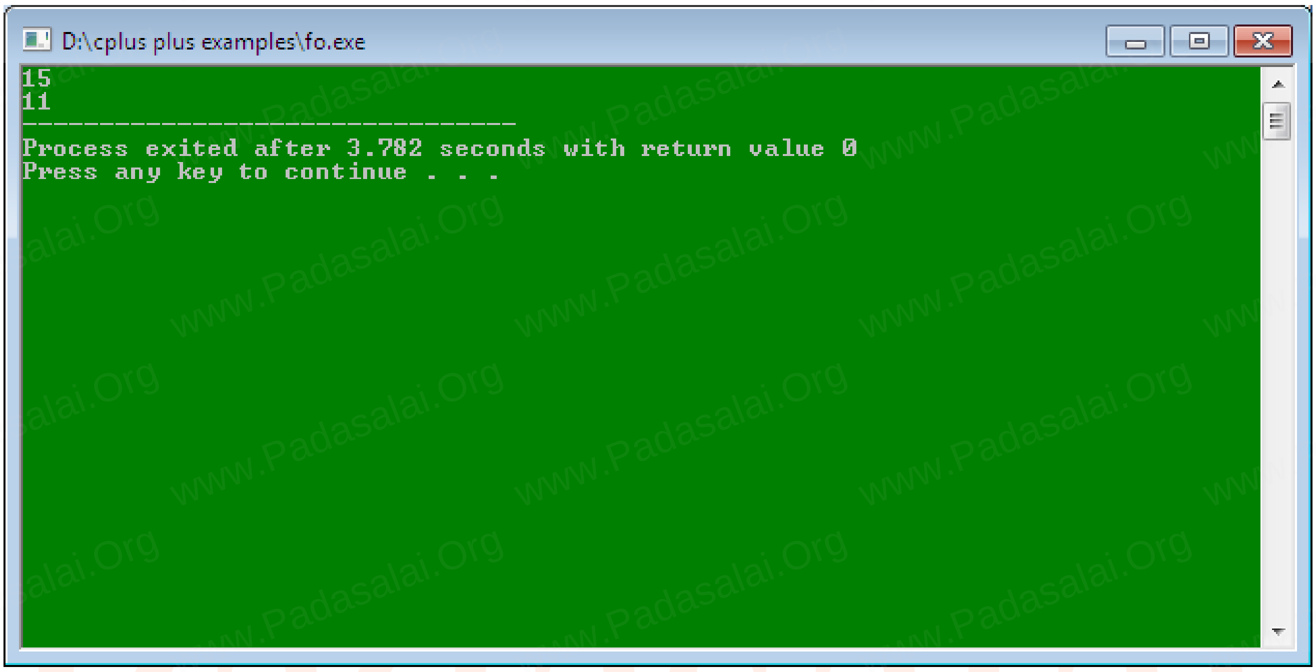
- ✓ Following are some restrictions to be kept in mind while implementing operator overloading.
  - Precedence and Associativity of an operator cannot be changed.
  - No new operators can be created, only existing operators can be overloaded.
  - Cannot redefine the meaning of an operator's procedure. You cannot change how integers are added.Only additional functions can be to an operator
  - Overloaded operators cannot have default arguments.
  - When binary operators are overloaded, the left hand object must be an object of the relevant class

Example:

```
#include<iostream>
#include<string.h>
using namespace std;
class example
{
        public:
                int x;
        int getdata()
        {
                cout<<"\n Enter the x value : ";
                cin>>x;
        }
        example operator -();
}e;
example example::operator-()
        {
                x=-x;
                cout<<"\n value of X  after overloading is = "<<x;
        }
int main()
{
```

```
e.getdata();
-e;
return 0;
}
```

Output:



D:\cplus plus examples\fo.exe

```
15
11
--------------------------------
Process exited after 3.782 seconds with return value 0
Press any key to continue . . .
```