C++

Assignment No - 13

Q1] What is mean by reference in C++? Explain with an example?

→ Reference is new added data type in C++. The concept of reference is applicable in C++ & Java.

Reference is considered as derived data type.

It just considered as another name to that variable.

It just another name so that their is no separate memory allocation for variable.

To create a reference we use & (reference operator).

If the & operator use after assignment operator (RHS) then it considered as address of operator.

= & (address of operator).

If the & operator used before assignment operator then is used as reference operator.

(Reference operator) & =

- We can also create reference which refer to any primitive data type.
- We can also create a reference which refer to pointer.
- The name of reference as well as name of original variable refer to same memory location. That's why address and value of both is same.
- If we change value of variable using it original name the value of reference variable changes automatically.

when we create reference variable its entry get inside the symbol table.
Symbol table contain one column name as another name it contain reference name of variable.
e.g.

```
#include <stdio.h>
Using namespace std;
int main()
{ int no = 11;
   int &noref = no;
   int rahul = 99;
   int &pappu = rahul;
   int *ptr = &no;
   int *(&ptrref) = ptr;

   cout <<" no :: " <<no <<"\n";
   cout <<" noref ::" << noref <<"\n";
   cout <<" &noref :: " << &noref <<"\n";
   cout <<" *ptr :: " << ptr <<"\n";
   cout <<" ptrref :: " << *ptrref <<"\n";
   no ++;
   cout <<" no :: " << no <<"\n";
   cout <<" noref :: " <<noref <<"\n";
return 0;
}
```

Name	Address	Size	Value	Data Type	another name
no	100	4	11	int	noref
rahul	200	4	99	int	pappu
ptr	300	8	100	pointer	ptrref

Q2] What is mean by constructor & destructor? When are they get called.

→ Constructor-

A constructor is a special member function of a class that initializes objects of that class. It has the same name as the class and does not have a return type.

- It used to allocate resources or set initial values for object data members.
- Automatically called when an object of class is created.

Types of Constructor

1) Default Constructor - Take no parameters.
2) Parameterized Constructor - Takes arguments to initialize object member.
3) Copy Constructor - Create a new object as a copy of an existing object.

Destructor-

A destructor is special member function of a class used to destroy objects. It has the same name as the class prefixed with a tilde (~) and does not have a return type or parameter.

- Used to release resources like memory, file handles or network connections allocated by the constructor or other class methods.
- It automatically called when an object goes out of scope or is explicitly deleted.

Q.3] Write down the rules which must be followed while writting constructor and destructor.

→ 1] Name of constructor should be same as class name.
2] All the constructors and destructor should be under public specifier.
3] Their should not any return type for constructor and destructor.
4] If we create an obj by passing some parameters then parameterized constructor get call
5] If we create an object without passing any parameter then default constructer get call.
6] By passing another object then copy constructor will call.
7] No need of explicite call to constructor & Destructor.
8] Constructor is used to initialized the characteristics as well as it is used to allocate resources.
9] Destructor is not used to allocate memory for object.
10] Destructor is used to deallocate the resources which where allocated inside the constructor.
11] Destructor get automatically call before deallocating memory of an object.
12] The & operator used inside copy constructor is considered as reference operator.
13] If that operator is not ~~written~~ written then it may leads to recursive call.

Q4] What are different types of constructors? Explain with example with a sample program.

→ 1) Default Constructor -

A constructor that does not take any arguments or has default arguments.

- It initializes an object with default values.

2) Parameterized Constructor

A constructor that takes arguments.

It allows initializing objects with specific values at the time of creation.

3) Copy Constructor -

Constructor that creates a new object as a copy of an existing object.

It used when an object is passed by ~~me~~ value, returned from a function, or explicitly copied.

e.g.

```
#include <iostream>
Using namespace Std;
Class d Demo
{ public:
    int i;
    int j;
  Demo ()  // default Constructor
   { cout <<" inside default constructor \n";}
  Demo (int A, int B) // parameterized Constructor
    { i = A;
      j = B;
      cout <<" inside parameterized constructor \n";}
  Demo (Demo &ref)   // copy Constructor
    {
```

```
    i = ref.i;
    j = ref.j;
    cout<<" inside copy constructor \n";}
};
int main()
{ Demo obj1;
  Demo obj2 (11,22);
  Demo obj3 (obj2);
return 0;
}
```

Q5) What is mean by pointer? what is the prototype of this pointer. explain with example.

→ Pointer is considered as derived data type in c, c++.

Pointer is such a special ~~function~~ Variable which stores address of anything.

The power of pointer is to fetch the data whose address is stores in it.

we can create a pointer which points to any primitive data type.

Pointer is a variable which holds address, each address is of type unsigned long due to which size of every pointer is 8byte.

This pointer.

this pointer is a special pointer in c++ that points to the current object of a class.

It is automatically passed as a hidden implicite parameter to all nonstatic member functions of a class.

Prototype- classname* const this,

eg.

```
#include<iostream>
using namespace std;
class Demo
{ public :
    int i;
    int j;
    int k; //Demo (Demo *this, int a=10, int b=20 int c=30)
  Demo (int a=10, int b=20, int c= 30)
  { cout<<" Value of this is "<< this << endl;
    this->i = a;
    this->j = b;
    this->k = c;
  }
  //void fun (Demo *this, int No1, int No2)
  void fun (int No1, int No2)
  { cout<<" inside fun :"<<endl;
    cout<<" Value of i :"<< this->i << endl;
    cout<<" value of j :"<< this->j << endl;
    cout<<" value of k :"<< this->k << endl;
  }
  //void display (Demo *this)
  void display ()
  { cout<<" inside display :"<<endl;
    cout<<" value of this pointer is :"<<this<<endl;
  }
};
int main()
{ Demo obj1;
```

```
// Demo obj3(5);
Demo obj2(40,50,60);
obj1.Fun(5,6);
obj2.fun(8,9);

cout<<"Address of obj1:"<<&obj1<<endl;
obj1.display();

cout<<"Address of obj2:"<<&obj2<<endl;
obj2.display();
return 0;
}
```

Q6] What is inheritance? explain types of inheritance with program.

→ Inheritance is considered as one of the object oriented paradigm of C++ & java.

- In simple term inheritance is a reusability which acquires characteristics and behaviour of another class.

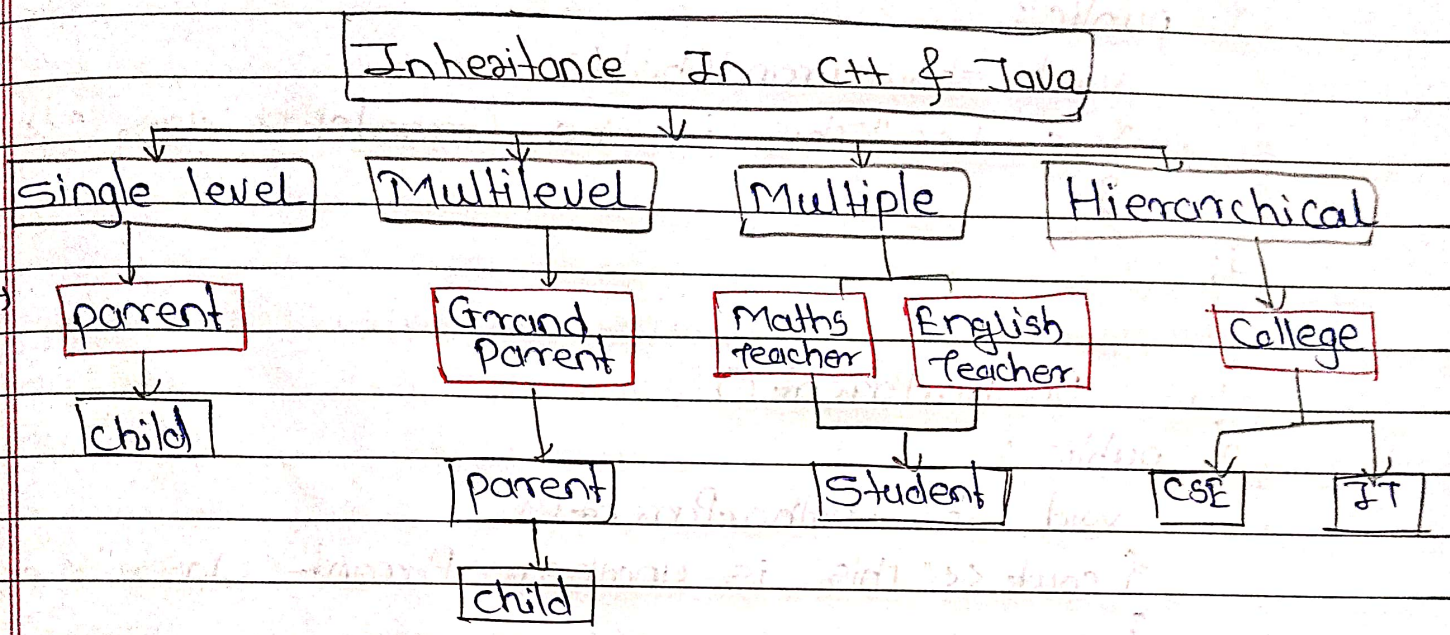
- According to structure layout there are types of inheritance

1] Single level inheritance
2] Multi level inheritance
3] Multiple inheritance.

1] Single Inheritance-
A subclass inherits from a single parent class.

2] Multilevel Inheritance-
A subclass inherits from parent class, and then another class inherits from the subclass

3) Multiple Inheritance :-
A class inherits from more than one parent class.



e.g.

```
#include<iostream>
using namespace std;
//Base class for single Inheritance
class Parent {
public:
    void showParent()
    { cout << " This is the parent class " << endl;
    }
};
//Derived class for single Inheritance
class Child : public Parent
{ public :
    void show child()
    { cout<<" this is the Child class "<<endl;
    }
};
```

```
//Intermediate class for multilevel Inheritance
class GrandChild : public Child
{ public:
    void showGrandChild()
    { cout<< "This is the Grandchild class "<<endl
    }
};
//Another base class for multiple Inheritance
class AnotherParent()
{ public:
    void showAnotherParent()
    { cout <<" This is another Parent class "<<endl;
    }
};
//Derived class for Multiple Inheritance.
class MultipleDerived : public GrandChild ,
    public AnotherParent
 { public:
    void showMultipleDerived()
    { cout<<" This is MultipleDerived class "<<endl;
    }
};
int main()
{ MultipleDerived obj;
   // Accessing Parent class Methods
    obj. showParent();
   // Accessing child class methods
    obj. showChild();
   //Accessing Grandchild class method
    obj. showGrandParent();
    obj. showMultipleDerived(); // Multiple class method
return 0;
}
```

Q7] What is mean by polymorphism? explain its types with program.

→ Polymorphism is a key concept in object Oriented Programming that allows methods or functions to behave differently based on the object that calls them.

The word polymorphism means many forms.

It allows a single interface to represent different data types or methods.

Types of Polymorphism

1] Compile Time 2] Run Time

Compile Time Polymorphism (static)

Concept of Compile Time Polymorphism is Considered as Early Binding

Because desicions taken by compiler is ~~on~~ in the early phase.

To achieve compile time polymorphism in C++ & Java we have to use the term called overloading

There are two types of overloading

1] Function Overloading

2] Operator Overloading.

Run Time Polymorphism (Dynamic)

This is achieved through method overriding using inheritance and virtual functions.

The function to be called is determined at runtime.

eg.

```
#include <iostream>
using namespace std;
// Compile Time (Method overloading)
class CompileTime
```

```
{ public:
    void display()
    { cout << "display with no parameters" << endl;
    }
    void display(int num)
    { cout << "Display with an integer:" << num << endl;
    }
    void display(String str, int num)
    { cout << "Display with a string and an int"
           << str << ", " << num << endl;
    }
};
// Run time Polymorphism (Base class)
class Animal
{ public:
    virtual void sound() // virtual fun for overloading
    { cout << " Animals make sound " << endl;
    }
};
class Dog : public Animal
  { public
    void sound() override
    { //override base class fun
      cout << " Dog barks " << endl;
    }
};
class Cat : public Animal
{ public
   void sound() override
   { // override base class fun
     cout << " Cat meows " << endl;
}; }
```

```
int main()
{ // Demonstrating Compile time
    Compiletime obj;
    obj.display();
    obj.display(42);
    obj.display("Hello", 100);

    cout<<"------"<<endl;
    // Demonstrating Run time
    Animal * animal; // Base class pointer

    Dog dog;
    Cat cat;

    animal = &dog; // ptr point to dog obj
    animal -> sound(); calls dog's sound()

    animal = &cat; //ptr point to cat obj
    animal -> sound(); // calls Cat's sound()

return 0;
}
```

Q8] What is the use of & operator in case of copy constructor.

→ The & operator in a copy constructor is used to pass the argument by reference. This is essential for creating an efficient and accurate copy of an object in C++.

- Why use & in copy constructor.

1) Avoid Infinite Recursion.

2] Modify original object
3] Const Safety.

Syntax-

```
class className {
public :
    className (const className & obj);
};
```

Q 9] Why the name of constructor and destructor is same as class Name.

→ The name of constructor and destructor is same as the class name to provide a direct and clear association between the class and its special member functions.
This design choice simplifies the object oriented programming model in C++.
- A constructor is automatically called when an object is created. Matching its name to the class makes it intuitive and directly links it to the class's initialization process.
- Destructor is automatically called when an object goes out of scope or is deleted. Having the destructor's name match the class name, simplifies understanding that it is responsible for cleanup related to the class.

Q 10] What is mean by Overloading? Explain Function Overloading.

→ Concept of compile time Polymorphism is considered as early binding. Because desicions taken by compiler is in early ple phase.

- To achieve compile time polymorphism in C++ & java we have to use the term called overloading.

Overloading -

Overloading in programming refers to the ability to define multiple functions or operators with the same name but with different functionality. It allows methods or operators to behave differently based on the parameters or operand types provided.

Overloading is a form of compile time polymorphism

Types of overloading are:

1) Function Overloading
2) Operator overloading.

Function overloading -

- It takes place on in single class.

Function overloading allows multiple functions to have the same name but with different parameters (number, type or order of parameter)

- Functions must differ in the number or type of parameters.

- the return type alone cannot distinguish overloaded functions.