BASICS

C++ ==C++=

C+1=c+OOPS meaning

Stream: a sequence of characters/bytes which are having source /destination

why we need operations

- to control input output operations

Two types of streams

- istream(input)
- ostream(output) predefined objects and classes to control i/o operations

Base class:IO

- istream(class)-cin(object)
- iostream(class)-cout(object)
- combine to form iostream(predefined classes)

COUT:

- predefined object of ostream
- always refers console output i.e MONITOR
- no conversion required BECAUSE it has inbuilt conversion(implicit conversion)
- cout <<-insertion operator//put to

Bhojpuri MANIPULATORS:

- it is used to manipulate the output format
- **two types of manipulators:**
- manipulator operators: endl,ends;
- Manipulator functions:Setw(),serfill(),setprecision(),section flags
- setbase(16)<<number<<endl;//prints in hexadecimal

OOPS

- **C++ IS NOT A PURE OBJECT ORIENTED LANGUAGE BECAUSE WITHOUT OBJECTS AND CLASSES WE CAN EXECUTE C++ PROGRAMS**
- SO IT A MIXTURE OF FUNCTIONAL AND OBJECT ORIENTED programming LANGUAGE

why oops?

- global // variables int a
- where C language is a functional

programming language where global variables accessed by every variables

- which is data is not protected so in order to avoid such data access and data privacy is very important in world so
- OOPS IS introduced in C++
- where functions and data is combined in class.

OOPS REAL TIME EXAMPLE

- when u go to super market where many customers will buy items and every customer will have his bill containing his items
- and price so
- which is a object--> where every customer in market purchasing items is object
- each object has different price and different amount,items
- every object has its own data
- structure and class
- structure has only members and data is insecure
- class has data and functions and data

is secure has access specifiers private,public,protected

- structures by default public ,class by default public

CLASS

- access specifiers.
- member functions.
- data members.
- The default class is private where structures are public.
- private members are only accessed by member functions where data is hided it is also called data hiding
- binding data members and member functions together is called encapsulation.
- class is user defined data type
- class is blueprint(original) and object is a instances(xerox) of a class
- when class is created no memory allocation is done.
- When an object is created memory allocation is done.
- class is logical representation and where the object is physical

representation.

data members

- data members cannot be initialized in class like a=100;
- it can be initialized using member functions or constructors;
- when data is private, the data cannot be accessed by objects;

Objects:

- instance of class
- physical representation of class

Encapsulation :

- binding data members and member functional together

Inheritance:

- when class A is declared where it has some data types and member functions
- so class B also have some data types related to class A if we create class B with same data types of CLASS A
- it is wastage of memory in order to use memory efficiently c++ has feature

called Inheritance

Access specifiers:

1. Public: if anything declared as public then it will be available to all the classes
2. Private: if anything declared as private then it will be available to to its member function
3. Protected: if anything declared as protected then it will be available to its child class only.

- Multiple inheritance:A->C,B->C
- multilevel inheritance:A->B->C
- hierarchical inheritance :A->B,A->C Opposite to multiple inheritance
- Hybrid inheritance: it is a combination of more than one inheritance.

Abstract class with pure virtual functions:

- abstract class should have at least one virtual function.
- abstract class cannot have an object declaration.
- it is only used for inheriting properties of

the Abstract.

- **pure virtual functions** are declared as virtual and initialise rest 0 in the abstract class.Should be declared in the abstract class.

Templates: One function with different tasks.

```
#include<iostream>
using namespace std;
#include<string>
template <class t1,class t2>
float sum(t1 a,t2 b)
{
   return a>b?a:b;
}
int main()
{

 float c=sum(5,3);
 cout<<sum(5.5,2.5)<<endl;
 cout<<sum(8,5.5)<<endl;
 cout<<c;
}
```

Flles:

- When an object is created for the class when we read data from the number functions in the class so the data will be stored in the object.
- after execution of the program the data located in the stack memory or we can say RAM memory which is temporarily stored is also deleted.
- So in real time applications when the user interface enters The data we should be stored in a database or in secondary memory in order to use the data so we need to store.
- To avoid storing in temporary locations we have a files concept which stores in the secondary

storage we can say hard please so files are used to store the data entered by the user.

1. **IFSTREAM**:Input file stream
2. **OSTREAM**: output file stream

This are available in
#include<fstream.h>

POLYMORPHISM(MANY FORMS)

- using one object in different forms reusability

Two types

1)overloading

- 1)function overloading
- 2)operator overloading

- **2)binding**
- 1)static binding/early/compile time
- 2)dynamic/late/run time binding

Overloading:

Overloading is having the same

name and same features with same function or maybe same object is also known as overloading and it is also known as Polymorphism because a single object of single function is turned into many forms so it is also known as polymorphism.

```
#include<iostream>
#include<string.h>
using namespace std;
class stu
{
 string name;
 string course;

float fee;
public:
   stu(string name,string course)
//Parameterized Constructor
   {
    this->name=name;
    this->course=course;
```

```
      fee=0;
    }
    stu(string name,string course,float
fee) //Parameterized Constructor
    {
     this->name=name;
     this->course=course; //Invoke One
Constructor with in one Constructor
    this->fee=fee;
    }
        void showstu()
        {
        cout<<"Name= "<<name<<endl;
        cout<<"Course= "<<course<<endl;
        cout<<"fees= "<<fee<<endl;
        }

};
int main()
{
```

```
stu s1("rama","C++");
stu s2("Varun","C++",5000);
s1.showstu();
s2.showstu();
}
```

Constructors:

- Constructor are used to define or initialise the data.
- Constructor is always defined in public only.
- Constructor is automatically called when the object is created

DESTRUCTORS:

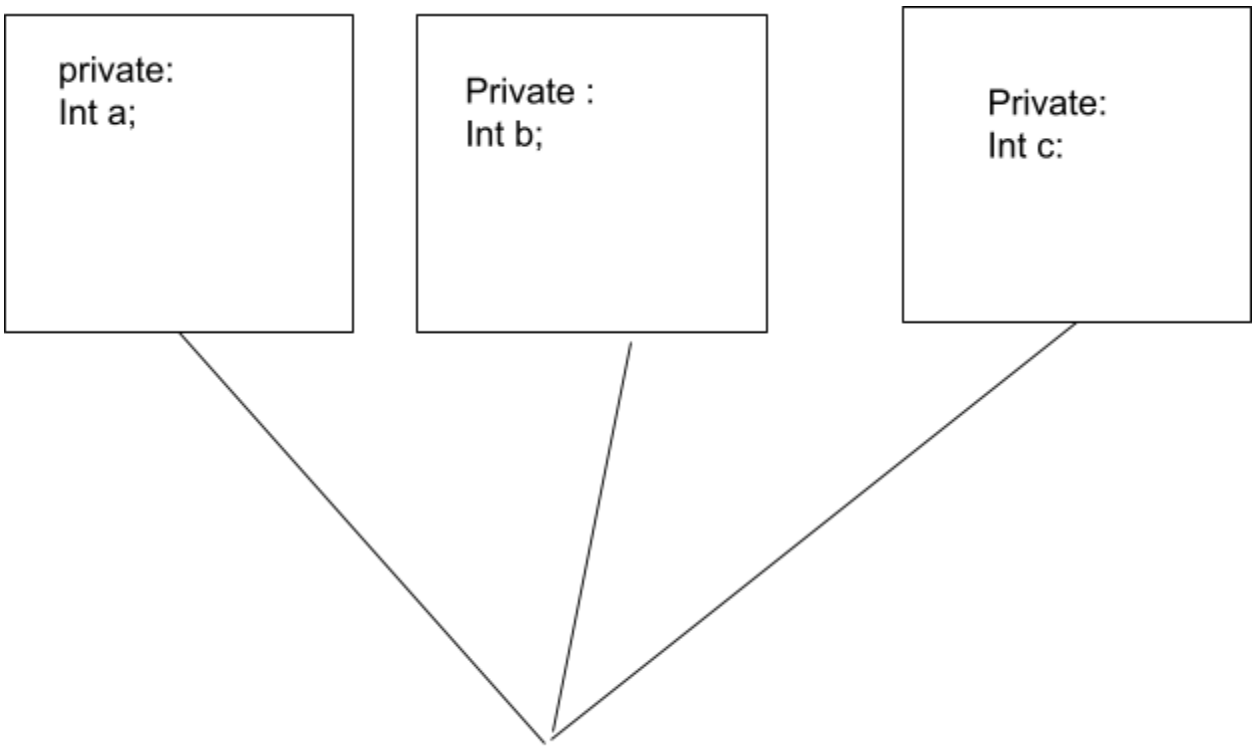
Destructor is used to release memory that created by the constructor.

Represented as
~class name
{

}

Friend functions:

Function is not Member function of a class but it able to access all the data members of the class.



Accessing from outside it not possible because it is private in order to access :friend function is used

Friend function can access private data members outside the class

Friend function is is against the Data Hiding concept because it reveals the data outside the class show so in order to maintain their data security or data privacy better avoid using concept in your

real time application.

```
#include<iostream>
#include<string.h>
using namespace std;
class A
{
   int a;
   public:
     void geta()
     {
         cout<<"eneter a";
         cin>>a;
     }
     friend void print(A,B);
};
class B
{
   int b;
```

```
   public:

   void getb()
   {
        cout<<"enter b";
         cin>>b;
   }

   friend void print(A,B);
};

void print(A m,B n)
{
 if(m.a>n.b)
 {
     cout<<m.a<<" big"
 }
}
int main()
{
```

```
  A m;
  B n;
  m.geta();
  n.getb();
  print(m,n);

}
```

Friend class:

Class is declared the private members of the class only accessible inside the class only for we can say it is Data Hiding in order to access the data members of the class, outside of the class we have to use of declare friend class inside its.

so when we declare it as a friend class so we can access the data members of Private data member inside the class into the another class .

Class A
{

```
Int a;
fun()
{
cin>>a;
}
  Friend class B;
}
Class B
{

Fun1
{
  cout<<a;
}
}
```

friend class allows the concept of reusability of the code.

Static data type: A static int variable remains in memory while the program is running. A normal or auto variable is destroyed when a function call where the variable was declared is over.

- Static data members initialized zero automatically
- Static data members are class members

- So it can be initialised outside the class with Scope resolution

Static data members Are created in data segment against Public Data Hidin concept

STATIC MEMBER FUNCTIONS:

- Static member functions can only access static data members .
- static Member function can be accessed using class name or class object

Operator overloading:

If we want to add 2+3 then we get 5

We want to add our own objects then it will give u error because + operator is predefined

If we want to operate on user defined data type than its called need operator overloading

```
#include<string.h>
#include<iostream>
using namespace std;
class A
{
```

```
int real;
int img;
public:
void get()
{
     cout<<"enter value";
     cin>>real;
     cin>>img;
  }
   A operator +(A n)
   {
        A c;
        c.real=real+n.real;
        c.img=img+n.img;
        return c;
   }
   A operator *(A n)
   {
        A c;
        c.real=real*n.real;
        c.img=img*n.img;
        return c;
   }
   A operator -(A n)
```

```
    {
        A c;
        c.real=real-n.real;
        c.img=img-n.img;
        return c;
    }
    void operator ==(A n)
    {
        if(real==n.real)
        {
            cout<<"real are equal";
        }
        else if(img==n.img)
        {
            cout<<"img are equal"<<endl;
        }
    }
    void print()
    {
        cout<<real<<"+"<<img<<"i"<<endl;
    }
};

int main()
```

```
{
   A m,n;
   m.get();
   n.get();
    m==n;
  A q=m+n;
  A l=m*n;
  A s=m-n;

  q.print();
  l.print();
  s.print();
}
```

Problem with pointers

- Pointers contain addresses of other variables and stores in it .
- so So when we are using real time applications The data needs to be secure by using pointers we may change the address of the variable at any time at anywhere
- so in order to solve this problem so in order to keep the applications more secure c++ has introduced another feature reference data type which is also called as

reference variables.

- it is also known as alias as we can say another name to the variable which may have data in it

Reference data type represented as

data type &variable =variable eg int &a=b;

we cannot change the declaration or we cannot add other reference variable to the a variable this makes a more secure than pointer because it cannot change it in reference variables and pointers are changle;

THIS POINTER:

In C programming in order to know the address of the variable you have use *'&'* variable in C plus plus programming we use this pointer to Know of the address of the Current object.

```
Class A
{
   private:
   int a;
   public:
     void print()
     {
        cout<<" "<<this<<endl;
```

```
    }
};
int main()
{
  A a;
  A b;
  a.print();
  b.print();
}
```

OUTPUT:

0x7ffd39849080--->ADDRESS OF A OBJECT
0x7ffd39849084--->ADDRESS OF B OBJEct

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

```
    int a;
    public:
       void set()
       {
            a=20;
       }
       void print(int a)
       {
          a=a;
          cout<<a<<endl;
          cout<<this->a;
       }
};
int main()
{
 A a;
 a.set();
 a.print(100);
}
```

OUTPUT:
100->Local value
20->object value

Now this can be solved using this in order give preferenece

```
void print(int a)
      {
         this->a=a;
         cout<<a;
      }
```

CONSTRUCTORS:

- constructors are used to initialise the Private data present in the class data members which are also known as Private Data in order to ensure data security

- so constructors are used to initialise

the Private data present in the class member

- so if we want to access the data in the private mode outside the class or initialise the data members or data outside the class it is not possible because in order to ensure the data security or data privacy
- Class A{
  Private:
  Int a=10;
  Int b=20; //

- so this throws error because u cannot initialize the data Class private Mode Initialise the data present in the private mode and In order to secure data for access from outside you input from the outside so special topic Called Constructors
-
    - Constructors are called automatically when an object of the class is created.

- So in order to initialise the data present in the private so this needs another function to initialise the data present in the private mode so constructors are automatically insurance data without causing any function
- so this makes it easy to initialise data so constructors are very helpful in i to initialise the data present in the private mode so constructors are automatically insurance data without calling any function so this makes it easy to initialise data so constructors are very helpful in initialising the data.

```
Class test
{
 test()
{
///
};
Int main()
{
```

}

- **Two types of Constructors:**

  - Compile time constructor:If the constructor is not Defined by the user the compiler automatically defines the constructor inside the class in the compile time which is also known as intake instructor so it does not contain any God
  - user defined constructor: it is the user defined function after it is called when the object is created so it is also known as user defined constructor.

```
Test()
{
 //User-defined
}
```