1. NOTES OF OOPS
2. We were using the procedural programming by which we write the list of instruction in a single block

Suitable for small program

Eg

Void main(){} // thats all

1. Modular Programming dividing a program into functions and each function has a clear purpose.

Eg

Void function();

Void main()

{

function(); //

}

Problem : in modular programming data remains alive within modules, so we need some data to be global .

Problem :: if we use more global variable, if atm has function like withdrawal() , withdrawal(), getn.PIN() mini stmt() global variable : blc, pin, name,   
The issue is that we have global data into global variables and four functions are using all the data, so for the programmer it is harder to track where the value is getting from.  
  
SO, in a large project , we have a very large number of functions and it becomes very difficult to track. Difficult to conceptualize. Difficult to modify.  
   
The root cause is the function and data are separate from each other. So we need to bind them together   
So, with OOP we bind our function and data together maling the CLASS  
  
Class is the collections of variable and functions:  
  
Class  
  
Name

Pin

Balance

Function  
withdrawal()

chk\_blc()

gen\_PIN()

MINI\_STMT()

The class is the blueprint of the object with the help of class , we can create the object   
  
It relates with real world like name pin balance and their function are capable of using their own variable . like ram.name

ENCAPSULATION IS ENCAPSULATING ALL THINGS IN THE CLASS

Private means we can only use inside the class only

Private and public are class specifiers.   
   
 For member variable we write private, for member function we write public :

ACCESS SPECIFIERS:

Class house

{

private :

Int length, breadth; // member variable

Public:

Void setData(int x, int y)

Void area()

{

{Cout << length\*breadth;}

}

Private :  
Int x;

Protected:

Int y

Public :

Int z

IF WE USE INHERITANCE, WE CAN USE THE FEATURE OF ANOTHER CLASS…

INheriting Class 1

Private :

Can’t access

Protected :

y=10;

Public :

z=20

Other : (LIKE FROM MAIN FUNCTION sita.name)

Private :

Can’t access

Protected:

Can’t access

Public :

z=20;

Charactersistsc os OOPs;

Class:

* Class is a blueprint and object is instance of class
* Class is user-defined data type which holds its own data members and member functions.
* Helps in code reusability

Encapsulation:

* Encapsulation is wrapping up variables and methods in class.
* It help in data hiding

Polymorphism:

* Function overloading is example
* The property in which a single is capable of performing multiple task and can exist in multiple forms is called polymorphism
* Helps in reducing the complexity and length of code bcz we are not writing the functionalities aggain n again

Inheritance:

* Inherit the properties of a parent class to child class  
  Like:-

We can create the Animal Class and   
Class Animal{string name, int legs, int age};

Inherit to the Dog, Cat,Cow.

Abstraction:

* Hiding complicated things from the user

OOP’s approach

Class- combine data and function together

Class is building print

Class is user defined data type which holds its own member and member functions

Object is an instance of class, when created memory is allocated to member variables of member function

Access Sepcifiers

* Data hiding
* Private  
  With same class
* Protected  
  Within same class , inheriting class
* Public   
  Anywhere

Charactersistscs

1. Class Object
2. Encapsulation
3. Polymorphism(the same thing is capable of showing many forms helps in code reusability)
4. Inheritance
5. Abstraction - Data hiding simply

Constructors

Boss (i have got critical assignment hat needs to be done with great precision)  
  
- i give instructions to secretary

* Secratary - Employee

Second way

Calling employee and giving him instructions face to face  
- No intermediate

IN case 1:  
- there is chance of misconception

* It may convey improper info to employee
* It is time consuming ( conveying same info twice)

IN case 2:

* No erros as above

Lets take a example

#include <iostream>

Class A

{

private:

Int age;

public:

//—----------------------------------------------

Void setData(int x =0)

{age =x;}

//—-----------------------------------------------

Int getData()

{return age;}

//—-----------------------------------------------

}

Void main()

{

A a\_obj;

a\_obj.setData(28);

Cout << a\_obj.getData () ;

}

↑↑↑↑ This is without constructuors↑↑↑↑↑

==================================================================================================

↓↓↓↓ This is with the constructors ↓↓↓↓

#include <iostream>

Class A

{ private :

Int age;

Public:

--------------------------------------------------

A (int x) // constructor

{ age = x }; // same name as class and dont return

—----------------------------------------------

Int getData ()  
{

Return age;

}

Void main()

{

A a\_obj ( 28) ;

Cout << a\_obj.getData();

}

* Why constructors??
* Programmer may forget to initialize the data members in the object after creating it.
* When there are many objects, then it would be a tedious job.
* Initialize and allocate memory to data members

Rules:

* Same Name as Class Name
* No return type

Types:

1. Non - parameterized or Default
2. Parametrized constructor
3. Copy constructor

**Non - Parameterized constructor**

#include <iostream>

Class A

{

Private :

int age;

Public :

—------------------------------------

A() // Non Parametrized constructor

{age = 0 } // same name as class and don’t return anything

This is non parametrized constructor because it doesnt contain any parameters and used to set the default value in member variable.

*Lets look VS CODE file name oops1.cpp.*

Key Takeaway from this program

* In order to create non parametrized constructor, we will create the constructor with no arguments
* And inside the body of constructors , we can leave it blank or we can give them certain values so that they doesnt contain any garbage values

**Parameterized Constructor**

* It takes some arguments

Eg:

#Include <iostream>

Class A

{

Private :

Int age ;

Public:

—----------------------------------------

A (int x) // Parameterized constructor

{age = x;} // same name as class and doesn’t return anything

—----------------------------------------

// The value will be passed by the user as argument

Int getdata ()

{

Return Age;}

}

Void main ()

{

A a\_obj(28);

Cout << a\_obj.getData ();

}

Lets hop on to **VS CODE** file name : oop2.cpp

**Copy Constructor**

Example:

1. You go to shop and ask for shopkeeper
2. I want the same shoe my friend has
3. Shopkeeper says “ how will i know your friend’s shoe”
4. What will i do
5. I will go to my frn and ask for shoe and show to shopkeeper
6. Now the shopkeeper will understand my requirement and he will give the copy of shoe
7. *Similarly, in C++ , the copy constructor are used to create new object from the existing object*

Example:

#include <iostream>

using namespace std;

class A

{

private:

int age;

public:

//----------------------------

A(int x)

{

age = x; // Parameterized constructor

}

//----------------------------

A( A &a\_obj1)

{

age = a\_obj1.age;

}

int getData()

{

return age;

}

};

int main()

{

A a\_obj1 (28) ; // Parameterized constructor

A a\_obj2 ( a\_obj1 ); // Copy constructor

cout << a\_obj2.getData();

}

We are using the & to say the compiler that we are using that variable as reference

HOP TO **VS CODE** :

Oops3.cpp

**Overloaded Constructor**

Example: // this code is also in the file oops4.cpp

#include <iostream>

using namespace std;

// class A

// {

// private:

// int age;

// public:

// //-------------------------------

// A () // Non -Parameterized Constructor

// { age =0;}

// //-------------------------------

// A (int x) // Parameterized Constructor

// { age =x ;}}

// //-------------------------------

// A ( &obj1) // Copy Constructor

// { age = a\_obj1.age}

// //-------------------------------

// };

// Instead of creating separate non parameterized constructor , we will assign the value only in the parameterized constructor

// So the upper code becomes like this:

class A

{

private:

int age;

public:

//-------------------------------

A (int x=0) // initializing with zero instead of initializing in no parameterized constructor // Parameterized Constructor

{ age =x ;}

//-------------------------------

A (A &a\_obj1) // Copy Constructor

{ age = a\_obj1.age;}

//-------------------------------

};

int main()

{

A a\_obj1;

return 0;

}

Lets hop to **VS code** oops4.cpp

WAP TO TAKE PHONE DETAILS AS INPUT AND STORE THEM IN OBJECT AND USE CONSTRUCTORS

Phone Details:- // (mobile phone details)

1. Name
2. RAM
3. Processor
4. Battery

// hop to vscode Project1.cpp

**OPERATOR OVERLOADING**

* We used to have bulk phone
* It was not portable that was problem
* After years of rev. We came to button touch & screen touch mobiles
* So we have overloaded the old calling feature of the phone to smartphone with better and latest features.
* Similarly, we were facing the problem in programming
* The problem was the operator was only able to perform within the predefined data types like integers, floats, double , string
* But it failed when user defined data types ( like st)
* if we want to add two ints , it can do but when we want to add the two Objects, it loses its functionality
* We want to make the “+” operator smart enough like the smartphone

OPERATOR OVERLOADING IS NOTHING BUT MAKING IT WORK FOR THE USER DEFINED DATA TYPES LIKE OBJECTS AND STRUCTURES.  
  
When we make operators (+,-,\*,/) work for *user-defined* types like objects and structures. This is known as **operator overloading.**

Example: oops5.cpp

// TRADUTIONAL ADDING THE VARIABLES OF OBJECTS

// OPERATOR OVERLOADING

// WAP TO add the weight using the traditional method

#include <iostream>

using namespace std;

class A

{

private:

int weight;

public:

A(int x = 0)

{

weight = x;

}

A addWeight(A w2)

{

A temp;

temp.weight = weight + w2.weight;

return temp;

}

void getweight()

{

cout << "Weight " << weight;

}

};

int main()

{

int a = 63, b = 74, c = 0;

c = a + b;

/\* here the c = a+b is the operation between the pre defined data types. but when we want to add the user defined data types

it loses is capability

We have two approach to add the two objects

1. Traditional Approach

-►in traditinal approach we create an addweight function to add

2. Operator Overloading Approach

-► In this menthod we do the operator overtloading

\*/

A person1(63);

A person2(74);

A total;

total = person1.addWeight(person2); // either this or this ↓↓↓↓

// total = person2.addWeight(person1);

total.getweight();

}

Example : OOPS6.cpp

// OPERATOR OVERLOADING

// WAP TO add the weight using the operator overloading method

#include <iostream>

using namespace std;

class A

{ private:

int weight;

public:

A(int x = 0)

{ weight = x; }

A operator + (A w2)

{ A temp;

temp.weight = weight + w2.weight ;

return temp; }

void getweight(); };

int main()

{ int a = 63, b = 74, c = 0;

c = a + b;

A person1(63);

A person2(74);

A total;

total = person1 + person2;

total.getweight(); }

void A ::getweight()

{ cout << "Weight " << weight;}

Goto VS code

oops7.cpp

WAP to overload ++ Pre increment operator

#Include <iostream>

Void main()

Int a =63;

++a;

Person(63);

++Jon; (wrong)

}

Class Person

{

Private:

Int wieght;

Public:

A (int x=0)

{ weight =x;}

**PRE-INCREMENT OPERATOR OVERLOADING**

**void operator ++()**

**{**

**++weight;**

**}**

Void operator ++()

{

++weight;

}

};

**LETS HOP TO VS CODE**

Oops8.cpp

**POST-INCREMENT OPERATOR OVERLOADING**

oops9.cpp

For post increment we pass the int as the argument

// post increment

void operator ++(int)

{

weight++;

}

**INHERITANCE**

**What is inheritance?**

►Grandfather←Father←Child

►► Parent Class ← Child Class

* THe class from which we are inheriting is called parent class / **base class**
* THe class which is inheriting the class is called child class / **derived class**

*So, inheritance is a mechanism in which one class acquires the property of the another class.*

**Why inheritance ?**

* You are writing a test, u r preparing for test/exam
* Your university modifies the curriculum and told you to write the exam again, i would be in definite problem
* To conduct exam again, it is

1. Waste of time
2. Resource,cost
3. Not feasible

* **PROGRAMMING**
* Module (Initial) → Testing → Module (Final)

If someone says you to modify the initial module, it will be the problems as pointed above.

So, in this situation , INHERITANCE helps

M1

↑

M1 + feature

We can create a new class by inheriting the old class + feature

INHERITANCE APPROACH

Base Model

* + Body
* + Wheels
* + Seats
* + Doors
* + Windows
* + AC
* + Fuel Tank
* + Suspension
* + Transmission

↓ ↓ ↓

X Model Y Model Z model

* + Petrol - + Diesel - + Diesel
* + PS - + PS - + PW + PS
* + PW - + PW - ABS +SR
* + ABS - + ABS - AutoTrans

↑↑↑↑Car Variants ↑↑↑↑

We created the base model

And we inheriting the Base model and inheriting the base class to derived class to various version.

**Why Inheritance?**

1. Reduce Duplicate Code
2. Code Reuse
3. Better Organization of Code

**INHERITANCE EXAMPLE**

#Example 1:

#include <iostream>

using namespace std;

class rectangle

{

public:

int length;

int breadth;

void show() // base class

{ cout << length<< endl;

cout << breadth<< endl; }

};

// CREATING THE DERIVED CLASS

class cuboid:public rectangle // publicly inheriting the rectangle class

{

public :

int height;

void display()

{ cout << height << endl;}

};

int main()

{

rectangle r;

r.length= 10;

r.breadth = 20;

r.show();

// INHERITED CLASS

cuboid c;

c.length = 10;

c.breadth = 20;

c.height = 30;

c.show();

c.display();

}

#Example 2:

WAP in which cuboid class inherit rectangle class and calculate area and volume

// project2.cpp

**CONSTRUCTORS AND INHERITANCE**

**CONCEPT #1:**

class base

{

Public:

base()

{ cout<< “Default of base class “; }

Base (int b )

{ cout << “ Parameterized of Base class : ; }

};

class derived : public base

{

//Empty

};

Void main ()

{

derived d1; // *this will call default*

derived d2(9); // error happens  *because only the default constructor can be called where there is no constructor in derived class*

// IF THERE IS NO CONSTRUCTOR IN DERIVED CLASS, THEN THE DERIVED CLASS WILL CALL THE CONSTRUCTORS FROM THE BASE CLASS

}

***NOTE : IF WE DON’T SPECIFY THE CONSTRUCTOR, THEN THE DERIVED CLASS WILL USE APPROPRIATE CONSTRUCTOR FROM THE BASECLASS. ( APPLICABLE TO ONLY DEFAULT CONSTRUCTOR)***

Goto vs code oops10.cpp

**CONCEPT #2:**

class base  
{

public :

base(){ cout << “DEFAULT of base class “ <<endl;}

base(int b) { cout<< “ Parameterized of base class << endl ;}

};

class derived:public base

{

public :

derived () { cout << “default of derived class” << endl ; }

derived () { cout << “ parameterized of base class “ << endl ; }

};

void main ()

{ derived d1; //  *this will first call the base default msg and derived base message first*

derived d2(9); // this will also call the base default msg first and derived parameterized msg // **it will not call the parameterized constructor even if we are passing the parameters in it . the derived class cant see the parameterized constructor of the base class .**

}

*NOTE : - 1st Default constructor of base class, then default constructor of derived class is called*

*NOTE :- 2nd Parameterized constructor of base class is not called when para. constructor is present in derived class.*

Hop to vs code **oop11.cpp**

**CONCEPT #3:**

**How we can call the parameterized constructor of base class?**

class base  
{

public :

base(){ cout << “DEFAULT of base class “ <<endl;}

base(int **b\_arg**) { cout<< “ Parameterized of base class << **b\_arg** << endl ;}

};

class derived:public base

{

public :

derived () : *base()*  // ***this is how we call the constructor of base class***

{ cout << “Default of derived class “<< endl ;}

derived (int **d\_arg**) : *base (d\_arg)*

{ cout << “ parameterized of derived class “ << endl ; }

};

Void main ()

{

Derived d1;

Derived d2(9); // first the parameterized constructor of base class will be called, then, parameterized constructor of derived class is called.

}

**HOP TO VS CODE : oop12.cpp**

**OVERRIDING MEMBER FUNCTION**

class base

{

public :

void msg() { cout << “base class” ;}

};

class derived : public base

{

public:

void msg() { cout << “ derived class”;}

};

void main ()

{

base b ;

b.Msg();

derived c;

c.Msg();

}

// *If we create the new function in the derived class with the same name as base class and if we call the function* in this case msg () *, the function from the derived class is called instead of base class because*

*“ THE DERIVED CLASS IS MADE SO THAT WE CAN USE THE MODIFICATION FOR OUR COMFORT, and here we have modified in the derived class. As a result, the function from the derived class is called.  
this is called function overriding.*

NOTE : REDEFINING THE FUNCTIONALITY OF BASE CLASS INTO DERIVED CLASS, THEN IF WE CREATE OBJECT OF DERIVED CLAS

1. What if i want to call the base function in the base class and i dont want to create the object of the base class
2. FOR that , you can call using the following way

class base

{

public :

void msg() { cout << “base class” ;}

};

class derived : public base

{

public:

void msg() {

cout << “ derived class”;

**base::Msg(); *// calling***

}

};

void main ()

{

derived c;

c.Msg();

}

HOP TO VS CODE **oops13.cpp**

**Relations:**

1. isA Car



lets see a example

class Suzuki

{

public:

vois chechis ()

{ //……………………… }

void engine()

{ //……………………… }

void suspension()

{ //……………………… }

void transmission()

{ //……………………… }

void doors()

{ //……………………… }

};

class DODO:public Suzuki

{

public:

void chechis() // function overriding

{ //…………modified…………… }

void doors() // function overriding

{ //…………modified…………… }

void ABS // added new feature

{ //…………modified…………… }

};

**TAKEAWAY : In isA relationship we use inheritance bcz we dont want to create the class from the scratch**

1. hasA Relationship

class Suzuki

{

public:

vois chechis ()

{ //……………………… }

void engine()

{ //……………………… }

void suspension()

{ //……………………… }

void transmission()

{ //……………………… }

void doors()

{ //……………………… }

};

class Antoinette // company that used to produce engine for the plane but now it // //is creating the engine for the cars

{

void V8\_Engine( )

{

//………………………………………….

}

void V6\_Engine( )

{

//………………………………………….

}

};

***Now i want to create my own car such that two use some specific feature from above two classes***

1. Now instead of inheriting we create the object of the above two classes

class DODO

{

private:

Suzuki design\_obj;

Antoinette anto\_obj;

public:

void addChechis()

{ design\_obj.chechis();}

void addEngine()

{ anto\_obj.V8\_Engine();}

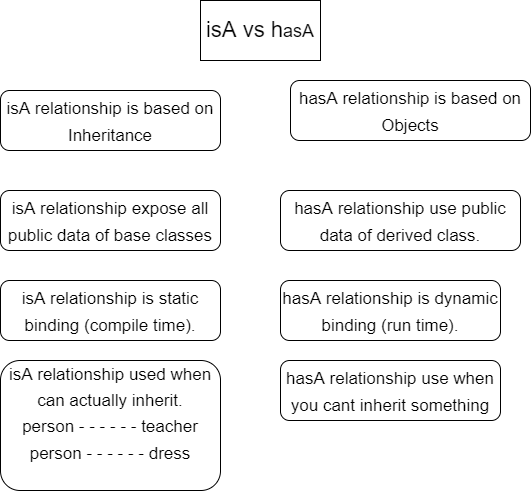
};

*The benefit of doing this is we donot expose all data to the DODO class*

*So, we can avoid unnecessary exposure of the things*

Lets suppose **Suzuki** gives me access to create by own car by using its features

1. When i use **isA relationship** , i can use all the features freely
2. But, when i use **hasA relationship,** we use objects thats useful for us



Ways of Inheritance

Private class Parent

Protected

Public

Private class child:public Parent

Protected

Public

Private class grandchild:public child

Protected

Public

1. Child and grandchild would not be able to access Private
2. Child and grandchild would be able to access both protected and public
3. Protected would be inherited as Protected  
   AND  
   Public would be inherited as Public in Child and Grandchild

| Protected Inheritance:-  Protected and Public in Parent class would be inherited as a Protected in in Child and GrandChild classes |
| --- |

NOTE :- While inheriting as protected, **public**  will also be inherited as protected.

Polymorphism

Base class pointer and Derived Class Object?

Base\* ptr = NULL;

ptr = new Derived();

Here there may be confusion that the pointer of base class and the object is of derived class

does it throw any error

bcz the pointer of base class is well compatible with the obj of der.class

Basic Car

↑

Advance Car

* pointer is of basic car
* obj is of advance car

we are sitting in basic car and saying to advance car that this is also advance car

Can we access the feature of ferrari ?

=> no

class basicCar

{

public:

void body();

void door();

void windows();

vid tyres);

}

void advVar: public basicCar

{

public:

void ABS();

void PS();

void EngineV8();

void AT();

}

void main()

{

basicCar \*ptr;

ptr = new advVar();

}

//-**Even if the pointer points to obj of adv car, it cant use the feature of //advance car**

**// this ptr can only use the feature of base class**

**/\* //right**

*ptr->body();*

*ptr->doors();*

*ptr->windows();*

*ptr->tyres();*

*// wrong*

ptr->ABS();

ptr->PS();

ptr->EngineV8();

ptr->AT();

\*/

}

Virtual Function

Virtual means existing in appearance but not in reality

Virtual funcion means fun. existing in class can’t be used

**=> Program that appears to be calling a function of one class may in reality be calling a function of different class.**

class Derv1:public base

{

public:

void show()

{

cout << “Derived1”;

}

class Derv2:public base

{

public:

void show()

{

cout << “Derived2”;

}

class base

{

public:

void show()

{

cout << “Base”;

}

void main()

{

Derv1 dv1;

Derv2 dv2;

base \*ptr;

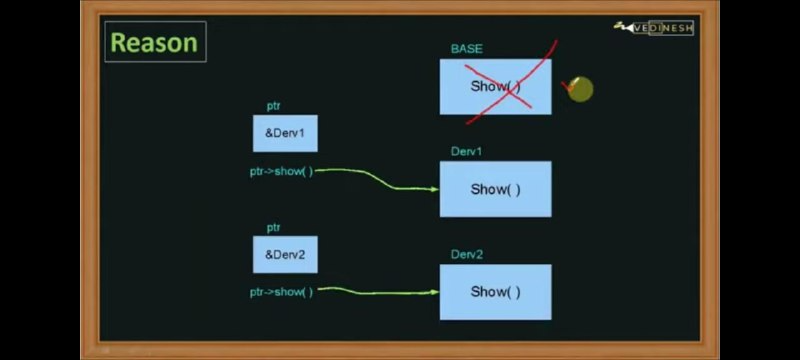
ptr = &dv1;

ptr->show();

}

Let’s hop to **VS CODE oops14.cpp**

**// after adding virtual infront of show function in base class, it can now access the drv1 & drv2 show function**

****

* First, the drv1 and drv2 can’t access their own function through ptr;
* Now, to make them access their own function, we make the base show function as virtual as below  
  syntax : virtual void show(){ cout << “base”;}
* Now ,the base show() function becomes virtual and compiler will understand and show the drv1/drv2 show function

Why Virtual Function?

Example:

* U m owner of candy shop
* Today is aniv.
* you hv dec to gib candy to kids who are regular
* now whenever kids appear, u will decide which candy to gib to them
* like blue to girl red to boy
* u cant predict how many boys/girls ll appear

Such example also comes in programming, where we cannot predict which object will come next and at run time compiler will decide

Thats why we use virtual function

This is called **Late Binding.**

Compiler defer the decision until the program is running

And at runtime when it come to know which class is pointed by PTR, then appropriate function would be called.

This is called Dynamic Binding / Late Binding

class person { public:

virtual void give() { cout << “Bun “; } };

class girl:public person{ public:

void give() { cout << “Blue Bun “; } };

class boy:public person{ public:

void give() { cout << “Red Bun “; } };

void main ()

{

boy b1;

girl g1;

person\*ptr = NULL; // initializing with null

ptr = &b1;

ptr->give();

ptr = &g1;

ptr->give();

}

Abstract Class

Abstract class is used when we never want to instantiate an object of BASE class.

*When we want to esure that object of base class is never instantiated is never generated , then we use abstract class*

**WE WANT TO ESNURE THAT THE OBJECT OF BASE CLASS is never generated.**

**we just have to write virtual infront of the function which is common to allother derived class of this class and write = 0**

syntax: virtual void give() = 0;

class Person{

public:

virtual void give() =0; // 0 means we dont want to instantiate

};

class boy: public Person {

public:

void give(){ cout << “brown bun”;}};

class girl: public Person{

public:

void giv(){cout <<”pink bun”}};

Pure virtual function

=> when the put the virtual function to 0, it becomes the pure virtual function,

WHEN WE MAKE THE VIRTUAL FUNCTION IT DOESNT HAVE BODY

And if a pure virtual function exists in a class then that class becomes **abstract class**

e

**WHY ABSTRACT CLASS?**

*It exists only to act as parent of derived class*

* *To ensure that ,the base class will do nothing and only act as a parent , we make it abstract class*
* *To ensure that the object of parent/ base class never exists*

Let’s see the main function

void main(){

boy b1;

girl g1’

Person \*ptr = NULL;

ptr=&b1;

ptr->give();

ptr=&g1;

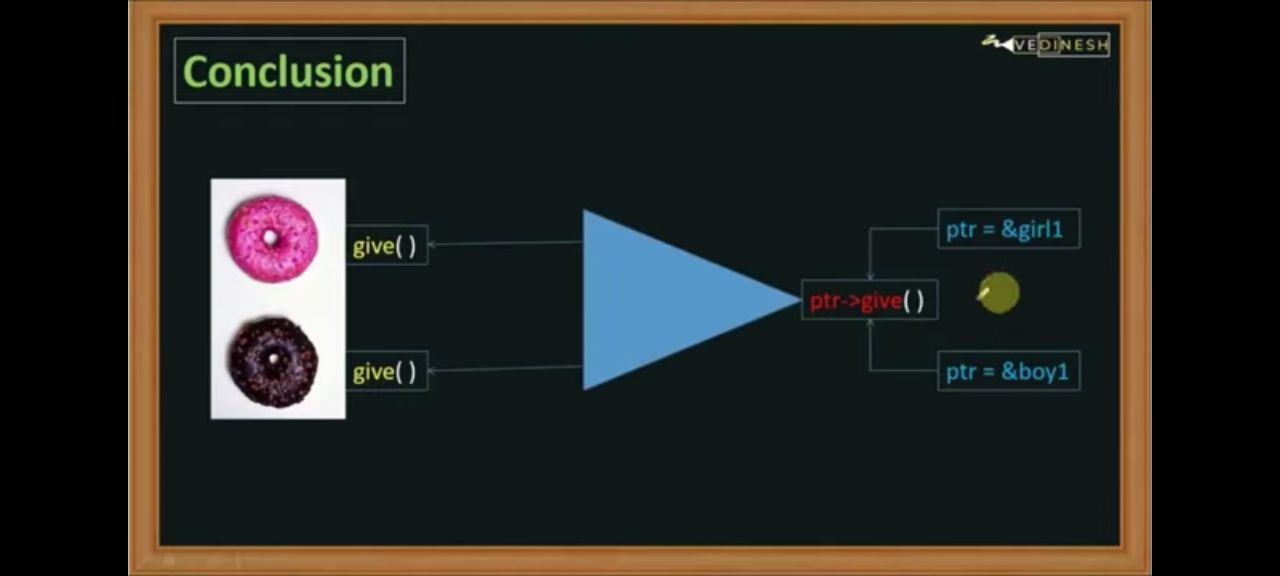
ptr->give();

ptr=&b1;

ptr->give();

}

HOP TO VS CODE oops15.cpp



when we call the function , it will select the functionality acc to the reference stored inside the pointer variable.

Moreover, giver() function is performing more than one task.

1. in class girl, it is selecting the pink bun and in class boy it is selecting the brown bun.

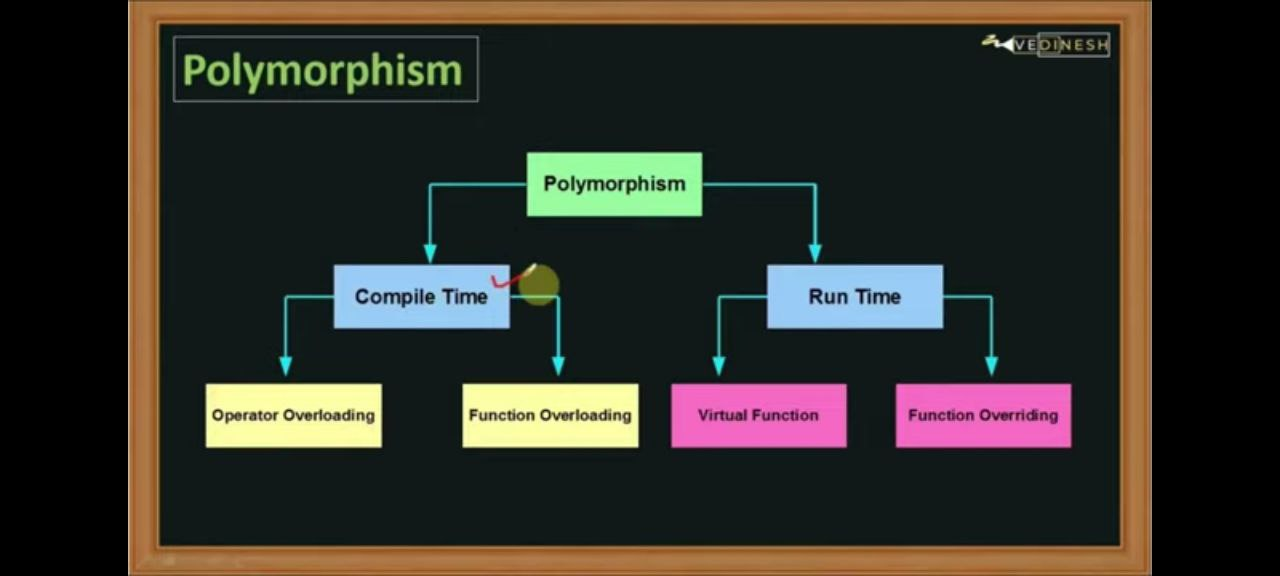
So, This ability to perform more than one task of a same function is polymorphism.

Anything that exists in more than one form is polymorphism

TAKEAWAY:

When one thing exists in more than one form and when that same functionality behaves diffly acc to input value, so this feature of shopwing more than one form is polymorphism

Types of polymorphism



Polym;

compltm

opera over fun ovrld

runtim

virtual fun fun overr(when base fun is over by der )

Virtual Destructor

We have already seen that destructor are used to deallocate memory. that aareo\used to clear to class obj and class mem

And des. are called when obj is out of scope / deleting the obj explicitly

class base{

public:

~base(){cout << “ Base class destroyed”;}};

class derived:public base{

public:

~derived(){cout << “ derived class destroyed”;}};

void main(){

base \*b1 = new derived; // pointer of base class and obj of der class

delete b1; // deleting the b1 object // only base will be destroyed

// we are only deleting half of the object

}

// make base class func as virtual.

class base{

public:

virtual ~base(){cout << “ Base class destroyed”;}};

TAKEAWAY:

if we want to delete the whole obj we need to make our base class function virtual, otherwise only the half will be destroyed.

HOP TO vs code oops16.cpp

Friend Function and Class

* u r solving puzzle alone
* one of ur frn join me to solve puzzle , u would be glad bcz , rduce of work load, speed of code, less chances of mistake , high standard of output, better work mgmt

**WHy friend function**

Class

Private : int x;

protected : int y; Function\_Outside();

public : int z;

so there is function outside the class and trying to accesss inside the classs, but the function is a normal function , it cant access all the function of the class. It can only access the public function of the class

LETS MAKE THE FUNCTION AS A FRIEND

Class

Private : int x;

protected : int y; Friend\_Funciton();

public : int z;

Now the friend function can access the private and protected part of the class

Many of C++ programmers think that frn function violates the function of class

But many porgarmmer think that with frn functnlty we are losing the functionality of OOP from class

From my perspective, Cpp programmers have give the full control of the code

**Other than CPP, no programming language contains the frn function**

class Alpha

{

private:

int a1;

public:

Alpha(int arg =0)

{a1= arg;}// constructor

friend void Fun(); /// this is how to define the frn function

};

class Beta

{

private:

int b1;

public:

Alpha(int arg =0)

{b1= arg;}// constructor

friend void Fun(); /// this is how to define the frn function

};

void Fun()

{

Alpha a(8);

Beta b(2);

int x = a.a1 + b.b1;

cout << “sum” << x;

}

void main()

{

Fun();

}

**TAKEAWAY: This friend function is connecting two or more classes ( acting as a fridge.)**

Lets hop to VS code for practical demonmstration/

oops17.cpp

Operator Overloading

WAP to implement the overloading of compilation operator.

hop to vs code  **oops18.cpp**

Friend Class

class Suzi class Jena ( no frns)

class Suzi class Jena (frns)

* can access and use features and functionalities of each other

How to create a friend Class???

class Alpha

{

private:

int a1;

public:

Alpha( int arg =0 )

{

a1 = arg;

friend class Beta;

};

class Beta

{

private:

int b1;

public:

Beta( int arg = 0 )

{

b1 = arg;

}

int Sum () // member function

{

Alpha alpha\_obj(3); // it is accessing the alpha class

int sum = alpha\_obj.a1 + b1; // this cannot be done in main or any other class if the class haven't been made the friend by writing th syntax  
{

return sum;

}

};

#include <iostream>

using namespace std;

class Alpha

{

private:

int a1;

public:

Alpha(int arg =0)

{

a1 =arg;

}

friend class Beta;

};

class Beta

{

private:

int b1;

public:

Beta(int arg =0)

{

b1 =arg;

}

;

int sum()

{

Alpha alpha\_obj(1);

int sum= alpha\_obj.a1+b1; //here beta is inaccessible if the class Beta is not declared as a friend in the class Alpha

return sum;

}

};

int main()

{

Beta beta\_obj(7);

beta\_obj.sum();

return 0;

}

// Here we dont have to interact with the various class in the main itself

TAKEAWAY: Now all member functions of beta can access private data of Alpha

oops19.cpp

Static Member and Function

class Alpha

{

private:

int a;

int b;

public:

Alpha()

{

a = 5;

b = 5;

}

};

void main()

{

Alpha a1; // when we are creating a1 we are reserving some memory for   
 // a and b variables

Alpha a2; // similary for a2

}

// Both these objects will reserve separate space of memory.

// They are not interlinkec to each other even if they are of the same class.

// Both of them occupy separate memory space

TAKEAWAY::  **EACH OBJECT WILL CREATE SEPARATE COPY OF ITSELF IN THE MEMORY**

Static Members

class Alpha

{

private:

int a;

int b;

public:

Alpha()

{

a = 5;

b = 5;

stat++;

}

static int stat;

};

void main()

{

Alpha a1;

Alpha a2;

}

// we’ve already seen that a1 and a2 will occupy separate space in memory.

// but a static member reserve a memory only once

// it doesnt reserve the memory multiple times.

// **STATIC MEMBER NEVER RESERVE A SEPARATE SPACE IT ONLY OCCUPIES SPACE ONCE AND THAT SPACE IS USED BY ALL THESE OBJECTS**

**//even we are creating the two objects only one memory is being used**

**Note :-**  Static member would be allocated only once.

**Note :- And that memory is shared by both the objects**

**Note :- Static data members belon a class and common to all objects.**

// We have created the static member inside of class but it required adeclaration outside a class.So syntax int Alpha::stat = 0;

**int n\_number(int number\_num,int pos) // this selects out the number of the position marked**

**{**

**// 123456789 is supplied // position 5 // return value must be 5**

**cout << "Passed Number is : " << number\_num << endl ;**

**cout << "Passed Position is : " << pos << endl ;**

**int a;**

**a = 10;**

**long int c;**

**int rem;**

**c = pow(10, pos) + 0.48 ; // adding 0.48 because of integer truncation**

**cout << "C is : " << c << endl;**

**rem = number\_num % c;**

**cout << "Rem is : " << rem;**

**return rem;**

**}**

**// after we create the object a1 ,constructor is called , the value of stat will rise to 1 from 0**

**// when we create the 2nd object , ,constructor is called , the value of stat will rise to 2 from 1.**

// Now lets see if we want to print the value of this static member then how we can do that

// for that there are multiple concepts.

1st // with the help of object

cout << a1.stat; // will print the value of statcic variable // which should be 2

// bcz when we create 1st ovj , value =1 and when 2nd obj val = 2;

// due to the post increment operator

2nd // with the help of object

cout a2.stat;

3rd // with the help of class

cout << Alpha::stat;

hop to **VS CODE oops20.cpp**

**Static Member Function**

// static member function is a special function which can only access static member variables.

It can only access and perform some operation on static member only.

If you try to use another variable like a and b it will throw error.

// in static member funciton we are incrementing the value

// static member variable is initialized to zero with the help of the statement   
*int Alpha :: stat = 0;*

*// so static member function can directly access the static member variable.*

class Alpha

{

private:

int a;

int b;

public:

Alpha()

{

a = 5;

b = 5;

}

static int stat;

static int getStat() // static member function

{

stat ++ ;

return stat;}

};

int Alpha :: stat = 0;

void main()

{

cout << Alpha:: getStat(); // calls function and stats value goes to 1.

Alpha a1; // inititalized with 5 and 5 to a & b

Alpha a2; // inititalized with 5 and 5 to a & b

cout << Alpha:: getStat(); // again trying to access the static member funciton // it will again increment value from 1 to 2; and return the value 2;

cout <<a1.getStat(); // will again increment the value of stat and make it to 3.

cout << a2.getStat(); // will again increment the value of stat and make it to 4.

}

**NOTE : STATIC MEMBERS CAN ONLY ACCESS STATIC MEMBERS.**

Hop to vs code oops\_21.cpp

**MINI ATM PROJECT**

vs code

Respected Madam,

With due respect, I would like to bring to your kind attention that I have been assigned to the post of Cashier in Urbar Nepal Pvt.Ltd.I am totally disappointed, and it was quite embarrassing for me that for the last few months, I have not received my Salary on time. And that’s the kind of thing I’ve never experienced in my working life.

You must be aware that people plan their bills according to their budget. I have to provide financial support to my family because of work , and I need to submit my rent bills and other expenses within the first week of every month. I am getting delayed paying my bills. At least , we should be getting the date of salary release so that we can manage things accordingly. Moreover, the landlord of the mulpani godown has also expressed that they should either be pre-informed or be paid on time.

Furthermore, I have previously expressed this matter to the other members, but there has been no answer until now. Thus, I decided to write an official complaint letter in the hope of receiving a more expeditious response.

I believe that the company acknowledges the significance of its representatives, and I hope that our values and needs will continuously be upheld by the organization.Thank You!

Yours truly,

Nabaraj Dhungel