- classes are the central feature of cpp that supports oop. classes are called user defined types.

Classes are used to specify the form of an object. contains the data and the methods to manipulate objects.

class is a blue print for a data type.

```
class Box {

    public:
        double l; double b;
        double h;
    };
```

<u>objects</u>

```
Box BOX1;
Box BOX2;
```

All objects will have their own copy of data members!

public data members can be accessed using (.) operator. where as private and protected can't be.

<u>class member func</u> : Member function of a class is like any other func defined inside a class

To define the function outside the class we can use (::) scope resolution operator.

<u>Access modifiers</u>: public private protected

The default access for members and classes is private.

- only the class and friend funcs can access private mems.
- protected are similar to private but thes can also be accessed in the derived classes.

constructor is called when a new object is created] spl func in
Destructor is called when an objected is deleted] class.

Friend func has access to private and protected.

<u>Inline func</u>: If a func is defined inline, the compiler places a copy of code of func at each point where func is called at compile time.

compiler can ignore keyword inline if func defined is more than a line. If an inline func is edited, it requires all clients of func to recompile again else continues with old func.

Every object in cpp has access to its own address through this pointer. Friend funcs do not have this pointer. because it is not a member of class. only member func have this pointer.

- When a member of a class is declared static, nomatter how many objects are created only one copy of static mem is created.

Static data is initialised to '0'.

Static member funcs - independent of particular object.

Static member funcs can access, (No this pointer allowed)

1) Static data mems
2) Static mem funcs
3) other funcs from outside class.

These can help in knowing if object is created or not.

Inheritance:

Defining a class (derived) from an other class (parent)(base)

Helps in, fast implementation
Reuse code
Maintaining code
Readability

Inheritance implements 'IS-A' relationship.

If no access specifier is used, publicly inherited.

Access	public	protected	private
Same ~~public~~	✓	✓	✓
Derived	✓	✓	✗
outside	✓	✗	✗

A derived class inherits all except,

- const, dest and copy const. of base class
- over loaded operators of base class
- friend functions of base class

Public Inheritance:

Public → public
Protected → protected
private can't be accessed directly.
but can access through other public and protected mems.

protected Inheritance

public → protected
protected → protected.

Private Inheritance

Public → private
Protected → private.

Multiple Inheritance: Multilevel Hierachial.

$$\begin{tikzcd} A \arrow[dr, dash] & & B \arrow[dl, dash] \\ & C & \end{tikzcd}$$

C++ func overloading / operator overloading

Cpp allows us to define more than one func or operator with the same name in the same scope.

overloaded declaration is a declaration that is declared with same name as previous but with diff arguments and obviously diff implem-entation.

operators which can be overloaded are,

+ - / % ^ new new[] delete delete[] etc.

can not be overloaded are,

:: . * ?:

Polymorphism

Polymphism occurs when there is a hierarchy of classes and they are related by inheritance.

Here, it means that a call to member function will occur but it causes another function to be executed depending on type of object ~~to be exe.~~ that invoked function.

- Static resolution : when there are 2 funcs with same name but the function is set once by the compiler as the version defined in the base class. Also called static linkage.

Since the func call is set previously, (ie), during compilation it is called early binding.

This can be solved by declaring the func as virtual. Indirectly telling that no static linkage is required. (In base class).

That's how polymorphism is used, 2 funcs with same name in diff classes, sometimes same arguments too but the (object has invoked matters.

This is called dynamic binding or late binding.

pure virtual funcs:

A function is defined in the base class with virtual keyword and if we don't have anything meaniful to be written we can leave it by just naming.

Virtual ______() $\underline{\underline{= 0}}$ ✓

Has no body. and this kind of a virtual func is called as pure virtual function.

Abstraction: providing only the essential info to the outside world and hiding their background details / implementation.

– programming / design technique that relies on separation of interface and implementation.

Eg: pow( ), sort( ) funcs.

Here, we use classes to define our own Abstract Data types

Access label specifiers helps in data abstraction.

Benefits: 1) class internals are protected from user-level errors. which might corrupt state of object.

Encapsulation

$$\begin{tikzcd} & \text{code} \arrow[dl] \arrow[dr] & \\ \text{Prog statements} & & \text{prog code.} \end{tikzcd}$$

– functions – data that gets effected by statements.

Encapsulation is about binding of data and code together (i.e), funcs that manipulate data and keeping away from outside interface. not leading to any misuse.

This led to the concept of data hiding.

Both abstraction and encapsulation can be implemented with help of user defined types, classes.

This can be applied even to Virtual funcs not only data mems.

Interfaces: Helps in adding new apps to system easily even after defined already.

Abstract class is a class with atleast one pure virtual func

app interfaces are implemented using abstract class.

The purpose of ABC (abstract class) is to provide an appropriate base class from which other classes can inherit.

We can not instantiate an object for ABC. Leads to compilation error. This can only serve as an interface

classes that can be used to instantiate objects are called concrete classes.