CSE 10/01/2022

Limitations of struct:

i) cannot declare public/private property
ii) cannot define operator overloading

So we use <u>class</u> in Cpp.

Constructor:

i) constructor does not have any return
ii) Constructor should be inside public scope
iii) Constructor name should be exactly same with class name.

Operator overloading: When we create user defined objects and try to use operator on those objects, we need to define operators, how they should behave. Its called operator overloading.

Example: C1 = 1 + 2i and C2 = 2 + 3i

If we want to evaluate C1 + C2 then we need to define how + operator should work

CSE 20/1/2022

Inheritance:

prohibit ~~o~~ → private and cannot access from outside but it can be inherited into another classes.

private → cannot be inherited into other classes.

class vehicle
    fuel()
    capacity()
    apply breaks()

super class (parent)
sub class (child)

class car/bus/truck :

↳ can be inherited here

```
class parent
  {
   public:
        int id_p;
  };
class child : ~~pare~~ public parent{
   public:
        int_c;
};
```

```
void main () {
    child obj;
    obj_c = 10;
    obj_p = 15;  → ~~pro~~ if publicly inherited no error
                 → if private/protected  " then error
                   and need function to assign value.
```

#

```
class A {
  public:
      int x;
  protected:
      int y;
  private:
      int y
```

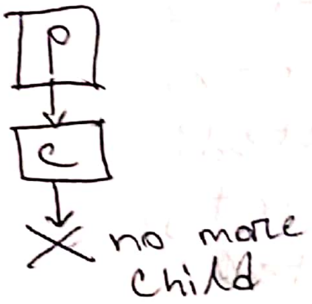
```
class B : public A {
   x // public
   y // protected
}
```

```
class C : ~~public~~ protected A {
   x // protected
   y // protected
}
```

```
class D : private A {
   x // private
   y // private
}
```

Base class member access specified	Type of inheritance		
	public	protected	private
Public	public	protected	private
Protected	protected	protected	private
Private	×	×	×

single inheritance:



```
class vehicle {
  public:
    vehicle() {
      cout<<"It is a vehicle."<<endl;

void main() {
  car obj;
```

```
Class B {
  int a;
  public:
    int b = 10;
    void get_ab() { a = b;}
        ~~a=b;} b=10;}~~
    int get_a() { return a;}
    void show a() { cout <<'a'<< a << endl;}
```

```
clss A : public B {
    int c;
    public:
     void multip() { c = b * get_a(); }
     void display(){
        cout << "a" << get_a();
        cout << "b" << b;
        cout << "c" << c;
void main(){
   A a;
   a.get_ab();
   a.multip();
   a.show_a();
   a.display();
```

Output

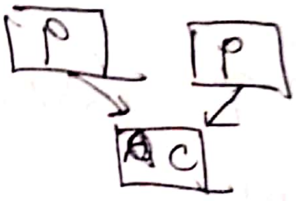
```
a 5
a 5
b 10
c 50
```

```
   a.b = 20;
   a.multip();
   a.display():
```

Output

```
a 5
b 20
c 100
```

Multiple inheritance:

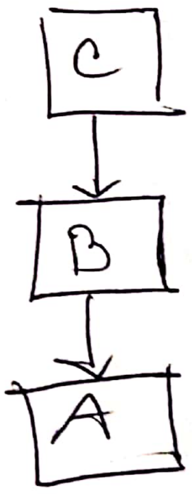


```
class vehicle{
  public:
      vehicle(){cout<<"vehicle"<endl;}
};
class fourw{
  public:
     fourw(){cout<<"4 wheel"<<endl;}
class  car: public vehicle, public fourw{
   .......
   }
```

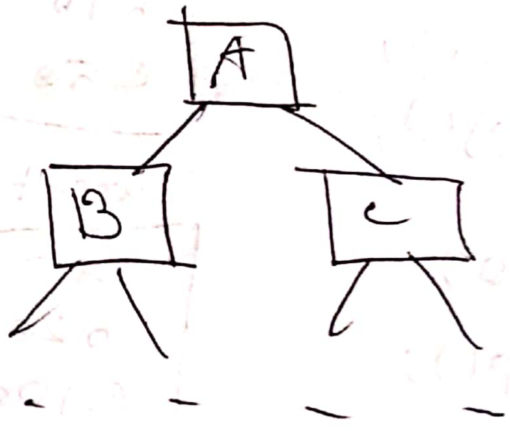
output

```
vehicle
4 wheel
```

multi layer:



hiarachical:



hybrid: