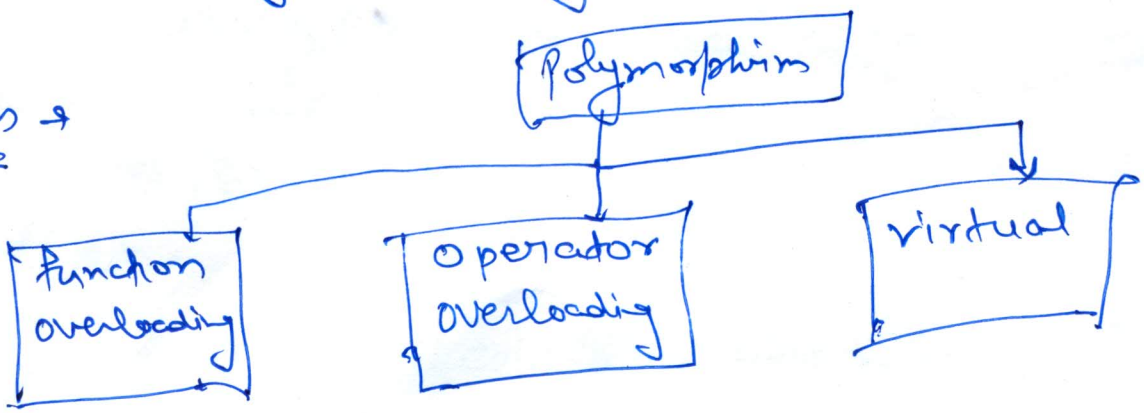
Polymorphism → poly morphism → form

compile time
(a) function overloading (b) operator overloading

Run time
(a) virtual (b) class

Types →



virtual function → When we use the same function name in both the base & derived classes, the function in baseclass declared as virtual using the keyword virtual preceding its normal declaration. When a function is made virtual, C++ determines which function to be used at run time based on the type of object pointed to by the base pointer, rather than type of pointer. Thus, by making the base pointer to point to different object, we can execute different versions of the virtual function

Program:
```
#include <iostream>.
    class Base.
{public:
void display () {cout << "\n display base"; }.
virtual void show () { cout << "\n show base"; }
};
class Derived : public base.
{public: void display(){cout<< "\n display derived";
        void show () { cout<< "\n show derived";
```

```
main()
{
baseB;
DerivedD;
Base *bptr;
cout << "\n bptr points to base \n";
bptr = &B;
bptr -> display(); //calls Base version
bptr -> show(); // calls Base version
cout << "\n\n bptr points to Derived \n";
bptr = &D;
bptr -> display(); // calls Base version;
bptr -> show(); //calls Derived version
return 0;
}
```

Pure virtual function

```
virtual void display() = 0;
```

Binding :
① early — static
② late — dynamic