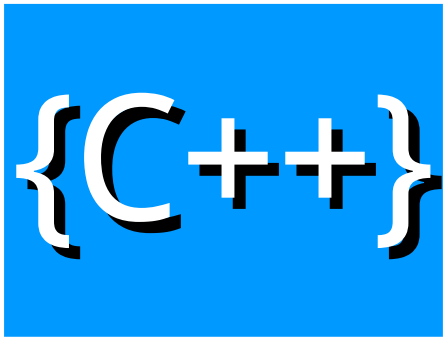
Polymorphism

Week 9

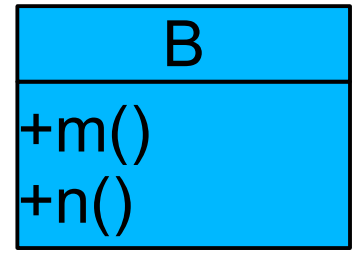


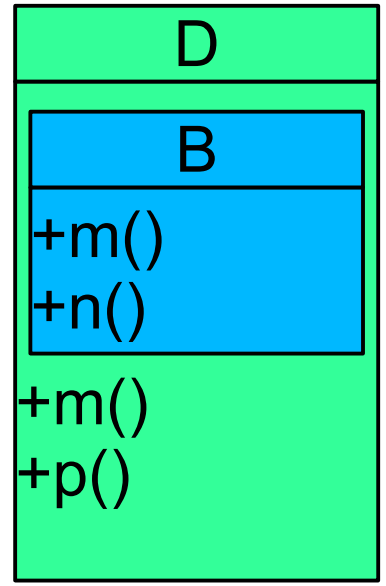
Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

Accessing Members of Base and Derived Classes

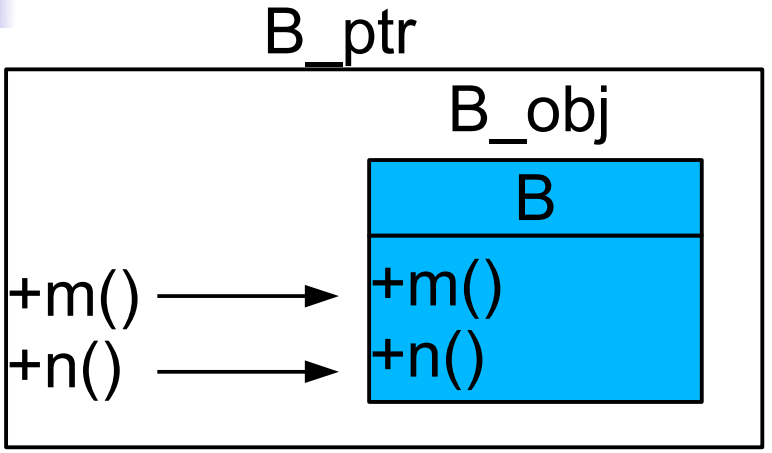
```
class B {
public:
   void m();
   void n();
   ...
} // class B

class D: public B {
public
   void m();
   void p();
   ...
} // class D
```

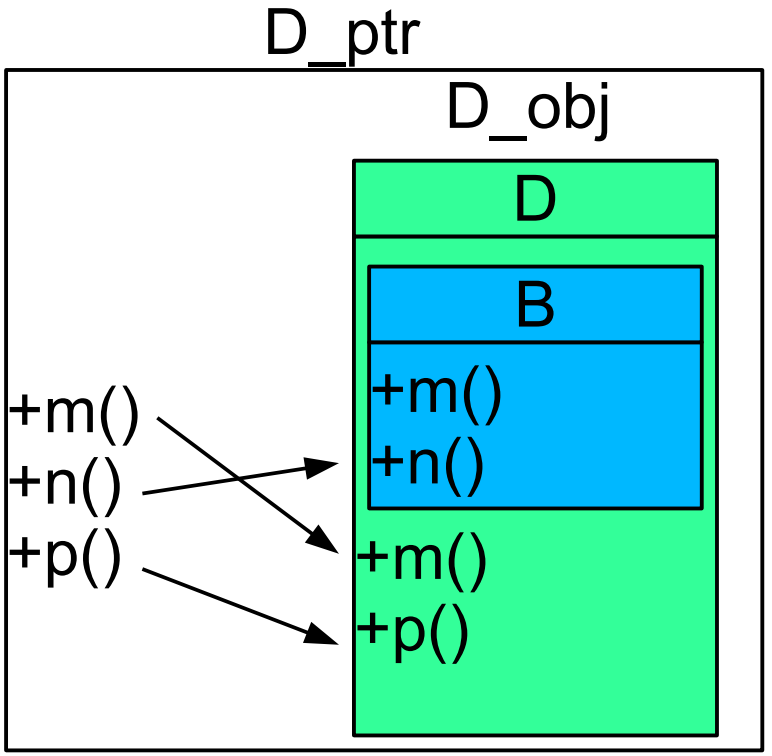
Class `D` *redefines* method `m()`





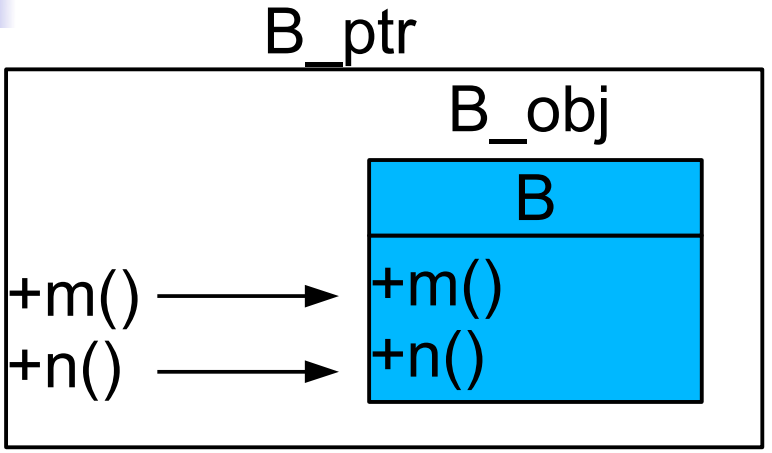
Accessing Members of Base and Derived Classes

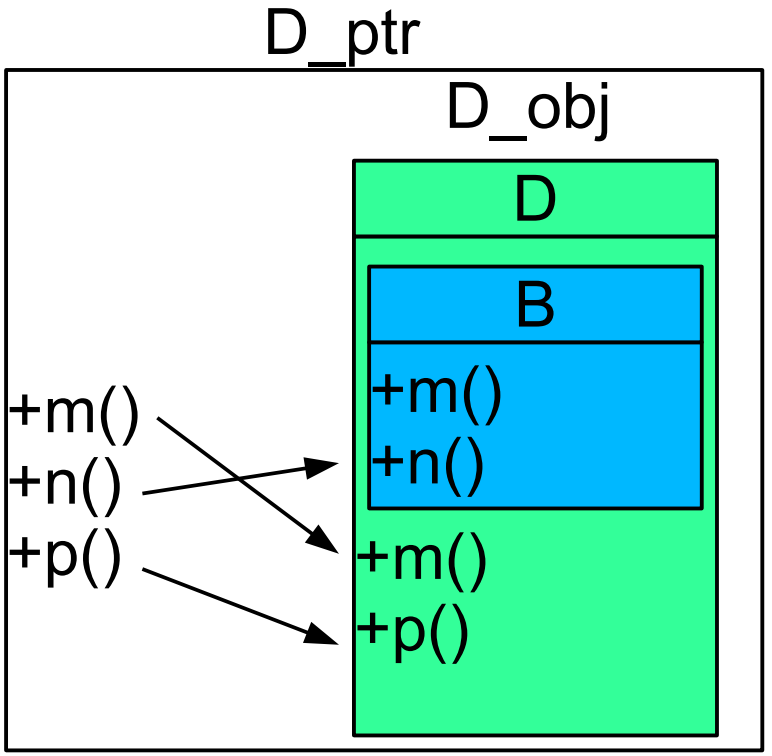





```
B B_obj;
D D_obj;
B *B_ptr = &B_obj;
D *D_ptr = &D_obj;
```

Accessing Members of Base and Derived Classes



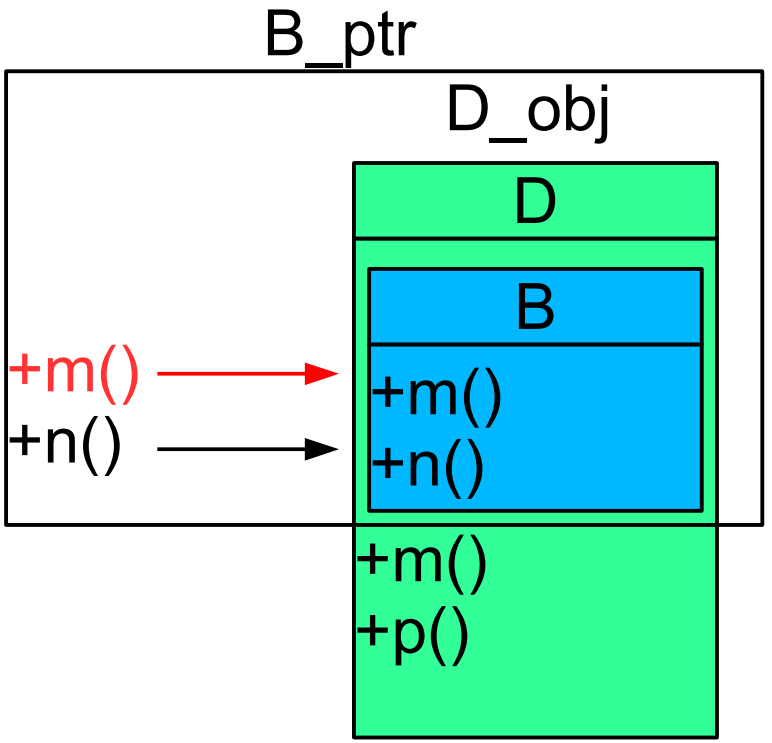



```
B B_obj;
D D_obj;
B *B_ptr = &B_obj;
D *D_ptr = &D_obj;
```

The following are legal:

```
B_obj.m() //B's m()
B_obj.n()

D_obj.m() //D's m()
D_obj.n() //B's n()
D_obj.p()


B_ptr->m() //B's m()
B_ptr->n()

D_ptr->m() //D's m()
D_ptr->n() //B's n()
D_ptr->p()
```

Accessing Members of Base and Derived Classes




```
D D_obj;
```

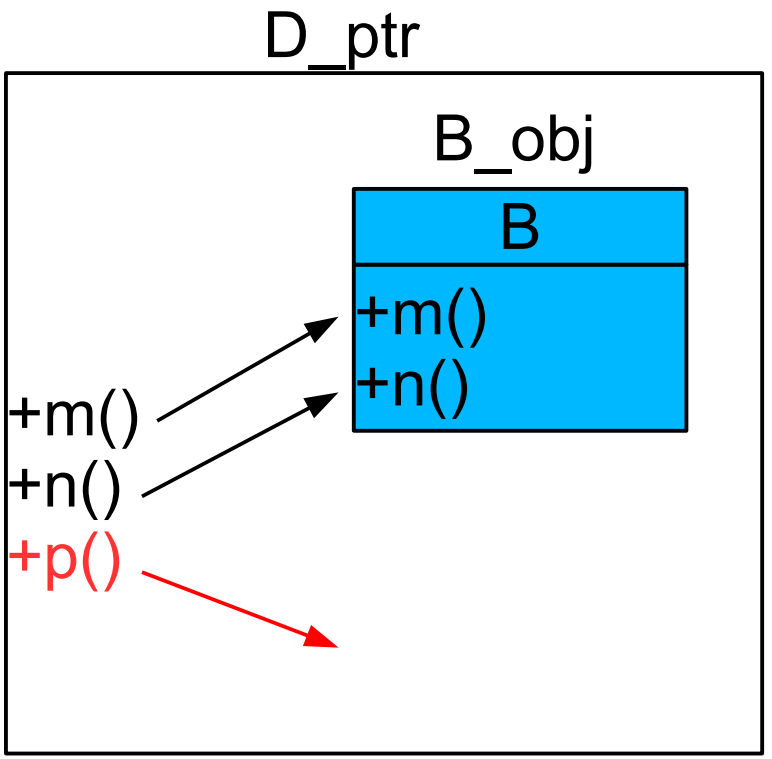
- The following are legal:

```
B *B_ptr = &D_obj;
B_ptr->m(); //B's m()
B_ptr->n();
```

- The follwing are not legal:

```
B_ptr->p();
```

Accessing Members of Base and Derived Classes



- The following are *not* legal:

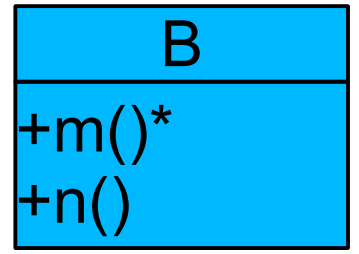
```
D *D_ptr = &B_obj;
```

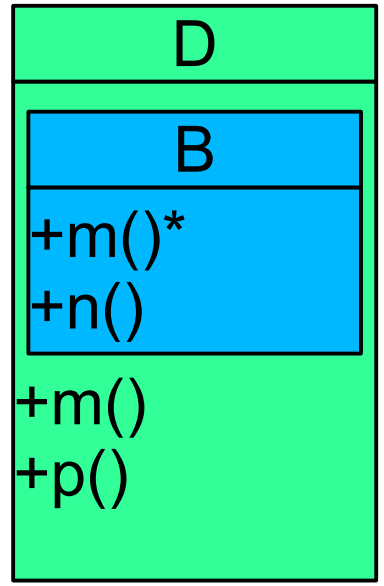
Virtual functions

* is virtual function

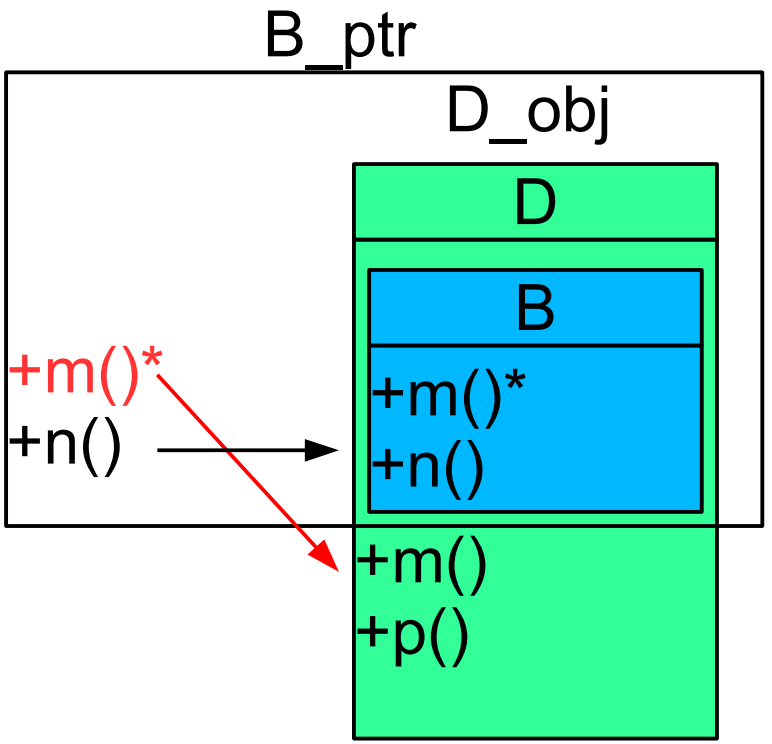
```
class B {
public:
   virtual void m();
   void n();
   ...
} // class B

class D: public B {
public
   void m();
   void p();
   ...
} // class D
```

Class `D` *redefines* method `m()`



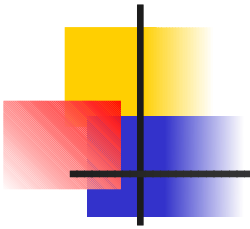


Virtual functions



The following are legal:

```
D D_obj;
B *B_ptr = &D_obj;

B_ptr->m()  //D's m()
B_ptr->n()  //B's n()
```



Polymorphism

- The ability to declare functions/methods as **`virtual`** is one of the central elements of polymorphism in *C++*

- *Polymorphism:*– from the Greek “having multiple forms”
  - In programming languages, the ability to assign a different meaning or usage to something in different contexts

Assignment 9

- 沿用 Assignment 6 的規定，必要時可以修改 class Monster ，以符合本次作業要求
- 本作業必須使用 Polymorphism 來實作
- 撰寫類別 Hedgehog ，俱備反擊能力，可以使得攻擊方損血，扣血量爲攻擊方當次攻擊效果的 1/4
  - 需要撰寫成員函式 getCounterAttack() 取得反擊效果
  - 若有必要可以在 class Monster() 加入 getCounterAttack() ，並設定爲 virtual function

Assignment 9

- 撰寫類別 Player ，可以擁有三隻怪物（可重複類型）
  - 成員變數
    - Name ，代表玩家名字
    - MonsterList ，代表對戰清單，大小爲 3
  - 建構子具有一個參數，用於設定 Player 名字
    - 建構子需要從三種寵物 (Dragon, Unicorn, Hedgehog) 隨機挑選，建立完整的對戰清單
  - 成員函式
    - getName() ，取得玩家名字

Assignment 9

- getCurrentMonster()，從 MonsterList 依序取得要進行對戰的寵物，寵物死亡則跳過，所有寵物都已經死亡，則傳回 nullptr
- showAliveMonsters()，顯示所有存活的怪物狀態

- 主程式對戰規則
  - 產生兩個 Player
  - 每回合隨機決定誰先攻，Player 從怪物清單中取得要對戰的寵物，以進行對戰
  - 若有一方沒有任何寵物可以對戰，則遊戲結束
  - 每回合對戰時，需顯示出適當的訊息

```
for(;;){
        cout<<"message"<<endl;


        system("pause");
        system("cls");
}
```