7-1

公有继承：基类的公有成员和保护成员的访问属性在派生类中不变，而基类的私有成员不可直接访问

私有继承：基类的公有成员和保护成员以私有成员身份出现在派生类中，而基类的私有成员在派生类中不可直接访问

保护继承：基类的公有成员和保护成员都以保护成员的身份出现在派生类中，而基类的私有成员不可直接访问

7-2

一般次序：

（1）调用基类构造函数，调用顺序按照它们被继承时声明的顺序（从左向右）

（2）对派生类新增成员对象初始化，调用顺序按照它们在类中声明的顺序

（3）执行派生类的构造函数体中的内容

7-3

```
class A;

class B;

B b;

b.fn1();

b.A::fn2();
```

7-4

虚基类：多继承情况下，声明虚基类后，虚基类的成员在进一步派生过程中和派生类一起维护一起维护同一个内存数据副本。

作用：将共同基类设置为虚基类，可以使从不同路径继承过来的同名数据成员在内存中只有一个副本，同一个函数名也只有一个映射，解决了同名成员的唯一标识问题。

7-5

```
class Shape;
class Rectangle:public Shape{
 public:
    Rectangle();
    double gerArea();
    ~Rectangle();
};
class Circle:public Shape{
  public:
    Circle();
    double gerArea();
    ~Circle();
};
class Square:public Rectangle{
  public:
    Square();
    ~Square();
};
```

7-8

```
class Document{
public:
```

```
    Document();
    ~Document();
private:
    int name;
};
class Book:public Document{
 public:
    Book();
    ~Book();
 private:
    int pageCount;
}
```

7-9

```
class Base{
 public:
  int fn1();
  int fn2();
};
class Derived:private Base{
  public:
    int fn1{return Base::fn1();}
```

```
    int fn2{return Base::fn2();}
};

Derived d;
d.fn1();
```

7-10

```
class Object{
public:
  Object(){}
  int show(){cout<<weight;}
  ~Object(){}
private:
  int weight;
};
class Box:public Object{
 public:
  Box(int a,int b,int c)： object(a),height(b),width(c）{}
  ~Box(){}
 private:
  Object height;
  Object width;
};
```

构造函数调用顺序：object --> height --> width，

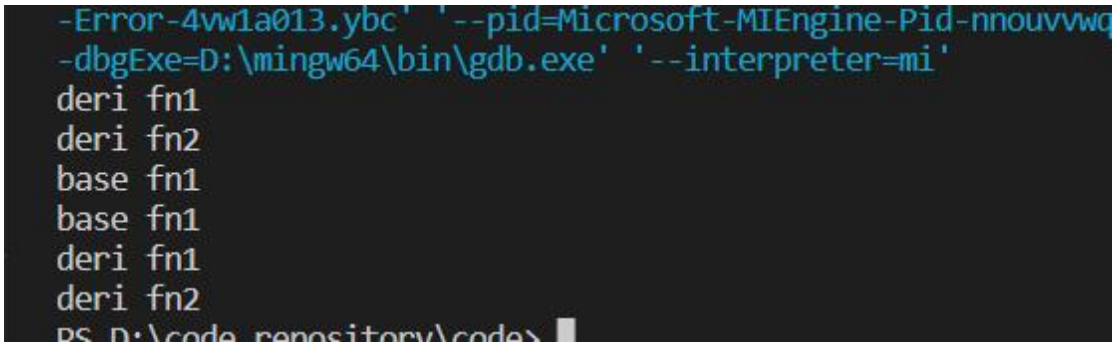
析构函数调用顺序与构造函数调用顺序完全相反

7-11

```
#include <iostream>
#include <string>
using namespace std;
class BaseClass{
public:
    void fn1(){cout<<"base fn1"<<endl;}
    void fn2(){cout<<"base fn1"<<endl;}
};
class DerivedClass:public BaseClass{
public:
    void fn1(){cout<<"deri fn1"<<endl;}
    void fn2(){cout<<"deri fn2"<<endl;}
};
int main(){
    DerivedClass d;
    BaseClass a;
    BaseClass *p=&a;
    DerivedClass *q=&d;
    d.fn1();
```

```
    d.fn2();

    p->fn1();

    p->fn2();

    q->fn1();

    q->fn2();

}
```

运行结果：



分析：类指针调用的函数是类里的；派生类 Derived 类的对象因为该派生类里有同名成员，所以隐藏了基类里的函数，调用的是派生类中的函数。