 YHim 21

fa6fc74 3 minutes ago

1 contributor

17 lines (12 sloc) 460 Bytes

## 保护继承私有继承

保护继承

基类成员访问属性	继承方式	派生类成员访问属性
private成员	protected	无法访问
protected成员		protected
public成员		protected



私有继承

基类成员访问属性	继承方式	派生类成员访问属性
private成员	private	无法访问
protected成员		private
public成员		private



-----  
线段类中只能访问到坐标类的公有数据成员/成员函数——“Has a”关系，私有继承是Has a的一种

## 总结

private不能继承、类外不能访问。protected能继承，类外不能访问。public能继承、类外能访问。权限：类外访问 < 继承 < 类内访问。

Branch: master

Find fileCopy path

hello-world / C++远征之继承篇 / 第三章-继承方式 / 保护继承私有继承 / 保护继承私有继承-例子.md

YHim 21

fa6fc74 3 minutes ago

1 contributor

224 lines (186 sloc) 3.74 KB

# 保护继承私有继承-例子

## public继承

public成员可以一直public继承下去，被继承后属于public，可以被对象直接访问。protected成员可以一直public继承下去，被继承后属于protected，不可以被对象直接访问。private成员不能被public继承。

## protected继承

public成员可以一直protected继承下去，被继承后属于protected，不可以被对象直接访问。protected成员可以一直protected继承下去，被继承后属于protected，不可以被对象直接访问。private成员不能被protected继承。

## private继承

public成员只能被private继承一次，被继承后属于private，不可以被对象直接访问。protected成员只能被private继承一次，被继承后属于private，不可以被对象直接访问。private成员不能被private继承。

要求：

```
/*
*****
保护继承和私有继承
要求：1. Person类，数据成员：m_strName，成员函数：构造函数、play()
      2. Soldier类，数据成员：m_iAge，成员函数：构造函数、work()
      3. Infantry步兵类，成员函数：attack()
*/
*****
*/
```

Person.h

```
#include <string>
using namespace std;

class Person
{
public:
    Person();
    void play();
protected:
    string m_strName;
};
```

Person.cpp

```
#include "Person.h"
#include <iostream>
```

```
using namespace std;

Person::Person()
{
    m_strName = "Merry";
}

void Person::play()
{
    cout << "Person--play()" << endl;
    cout << m_strName << endl;
}
```

#### Soldier.h

```
//soldier士兵
#include "Person.h"

class Soldier : public Person
{
public:
    Soldier();
    void work();
protected:
    int m_iAge;
};
```

#### Soldier.cpp

```
#include <iostream>
#include "Soldier.h"
using namespace std;

Soldier::Soldier()
{
}

void Soldier::work()
{
    m_strName = "Jim";
    m_iAge = 20;
    cout << m_strName << endl;
    cout << m_iAge << endl;
    cout << "Soldier--work()" << endl;
}
```

#### Infantry.h

```
//infantry步兵
#include "Soldier.h"

class Infantry : public Soldier
{
public:
    void attack();
};
```

#### Infantry.cpp

```
#include <iostream>
#include "Infantry.h"
using namespace std;

void Infantry::attack()
{
    cout << "Infantry--attack()" << endl;
}
```

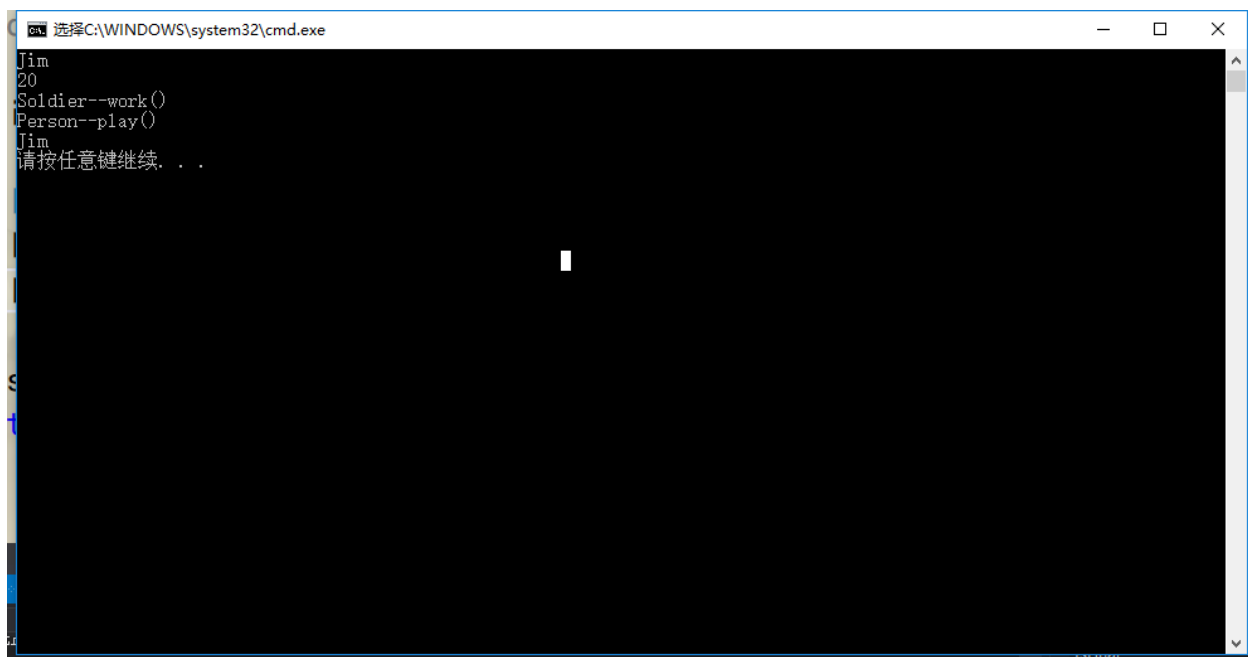
#### demo.cpp

```
#include <iostream>
#include <stdlib.h>
#include "Soldier.h"

int main()
{
    Soldier soldier;
    soldier.work();
    soldier.play();

    system("pause");
    return 0;
}
```

运行结果：



```
选择C:\WINDOWS\system32\cmd.exe
Jim
20
Soldier--work()
Person--play()
Jim
请按任意键继续. . .
```

Soldier.h

```
//soldier士兵
#include "Person.h"

class Soldier : protected Person//change
{
public:
    Soldier();
    void work();
protected:
    int m_iAge;
};
```

demo.cpp

```
#include <iostream>
#include <stdlib.h>
#include "Soldier.h"

int main()
{
    Soldier soldier;
    soldier.work();//正常
    soldier.play();//出现问题

    system("pause");
    return 0;
}
```

当进行protected继承之后，在基类当中的play()无法被子类的对象直接调用。

-----

Infantry.cpp

```
#include <iostream>
#include "Infantry.h"
using namespace std;

void Infantry::attack()
{
    m_strName = "Jim";//+
    cout << m_strName << endl;//+
    cout << "Infantry--attack()" << endl;
}
```

demo.cpp

```
#include <iostream>
#include <stdlib.h>
#include "Infantry.h"

int main()
{
    Infantry soldier;
    soldier.attack();

    system("pause");
    return 0;
}
```

运行结果：

A screenshot of a Windows command prompt window. The title bar shows the path "C:\WINDOWS\system32\cmd.exe". The window content shows the output of the program: "Jim", "Infantry--attack()", and "请按任意键继续. . .". The cursor is positioned at the end of the last line.

说明了Person类当中的public下和protected下的数据成员和成员函数被继承到了Soldier的protected的下面。

-----

Soldier.h

```
//soldier士兵
#include "Person.h"

class Soldier : private Person//change
{
public:
    Soldier();
```

```
        void work();  
protected:  
        int m_iAge;  
};
```

程序报错，m\_strName不能够在Infantry这个类的成员函数(attack())当中去直接访问。

## 小总结

继承都是继承所有的数据成员和成员函数，只是能不能访问的问题。