YHim / hello-world

Branch: master hello-world / C++远征之继承篇 / 第三章-继承方式 / 公有继承 / **公有继承**.md Find file Copy path

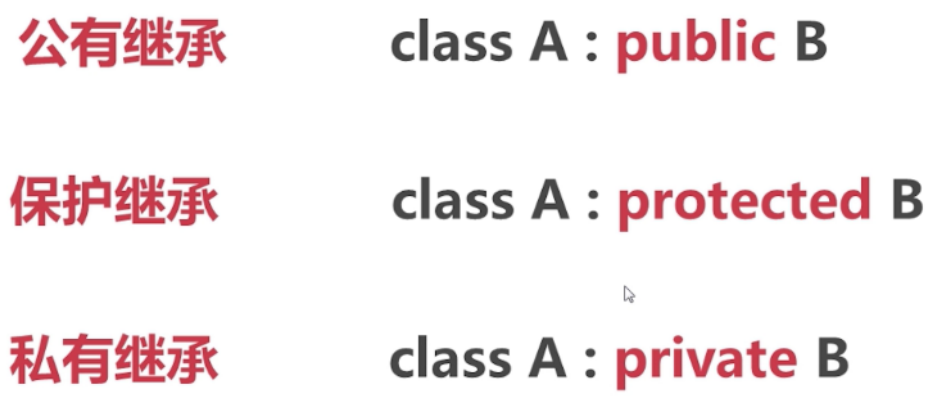
YHim 21 fa6fc74 a minute ago

1 contributor

46 lines (26 sloc) 1.13 KB

公有继承

继承有三种方式：



这里先讲公有继承。

例子：

```
class Person
{
public:
    void eat();
    string m_strName;
    int m_iAge;
};
```

```
class Worker:public Person
{
public:
    void eat();
    void work();
    string m_strName;
    int m_iAge;
    int m_iSalary;
};
```


使用时：

```
int main(void)
{
    Worker worker;
    worker.m_strName = “Merry”;
    worker.eat();

    return 0;
};
```


worker类的对象可以访问Person类的数据成员和成员函数。

---

protected访问限定符下的继承特性

例子：

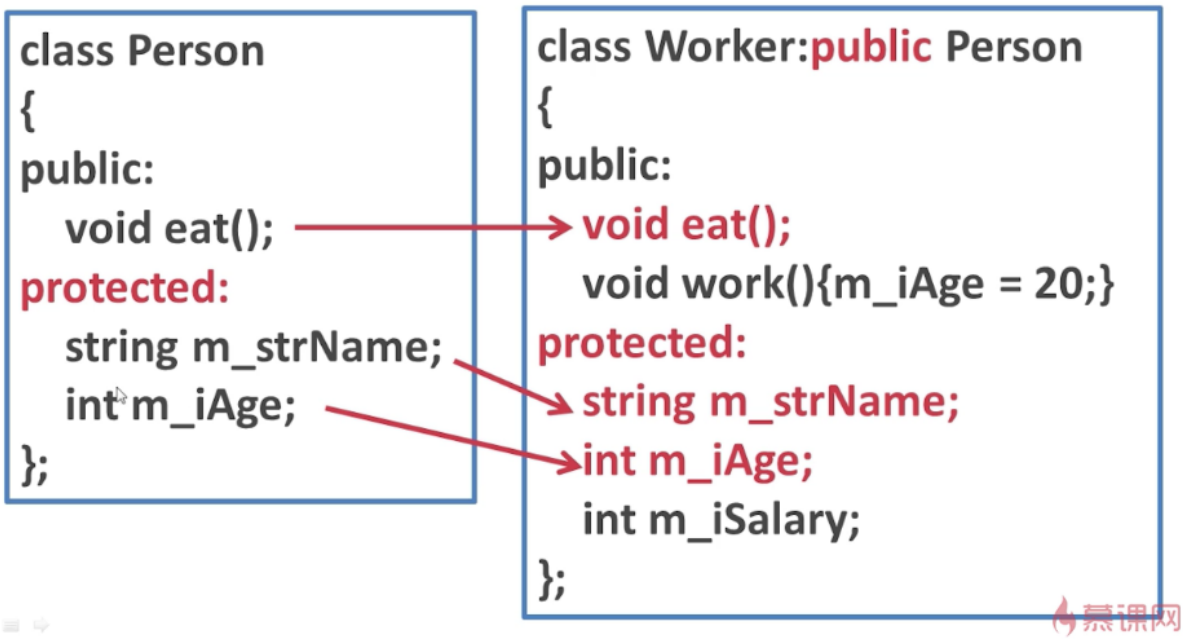
```
class Person
{
public:
    void eat();
protected:
    int m_iAge;
private:
    string m_strName;
};
```

```
int main(void)
{
    Person person;
    person.eat();
    person.m_iAge = 20;
    person.m_strName = “Jim”;

    return 0;
}
```


protected和private还可以作为访问限定符。在这个例子中，使用时红色语句是错误的。

```
void Person::eat()
{
    m_strName = “Jim”;
    m_iAge = 10;
}
```


当去实现成员函数eat()时，在函数当中去访问名字、年龄这两个数据成员的时候，是可以正常访问的。

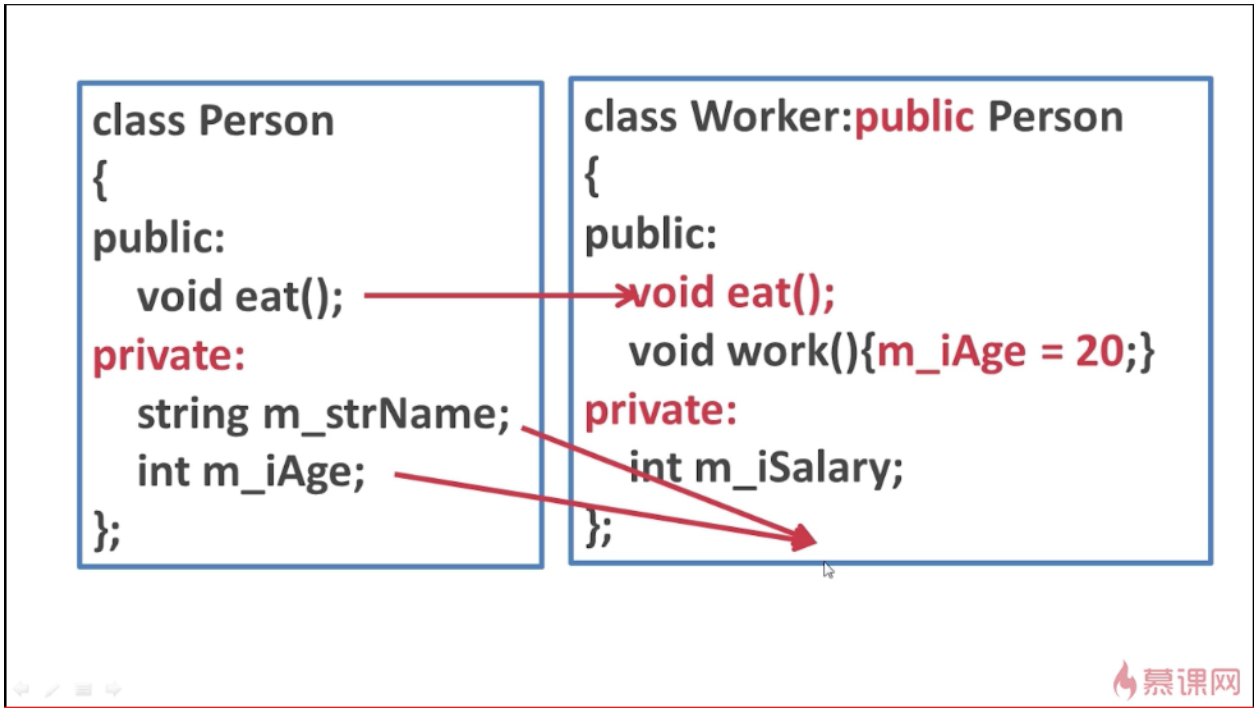


此时，Worker继承了Person，public访问限定符下的数据就会继承到public下面，protected下面的数据就会继承到protected下面。

---

private访问限定符下的继承特性

例子：



private下面的数据成员被继承到了Worker下面的不可见位置，而不是private下面。因此，在Worker中使用m_iAge就是不对的。

总结

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

YHim / hello-world

Branch: master hello-world / C++远征之继承篇 / 第三章-继承方式 / 公有继承 / **公有继承-例子**.md Find file Copy path

YHim 21 fa6fc74 2 minutes ago

1 contributor

330 lines (272 sloc) 4.65 KB

公有继承-例子

要求：

```
/*********************************************************************/
/*
  继承方式：公有继承
    要求：
      1. 定义Person类，要求含有m_strName和m_iAge两个数据成员及构造函数和析构函数、eat函数
      2. 定义Worker类，要求共有继承Person类，含有数据成员m_iSalary、构造函数、析构函数、work函数
*/
/*********************************************************************/
```

Person.h

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

class Person
{
public:
        Person();
        ~Person();
        void eat();
        string m_strName;
        int m_iAge;
};
```

Person.cpp

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

Person::Person()
{
        cout << "Person()" << endl;
}

Person::~Person()
{
        cout << "~Person()" << endl;
}

void Person::eat()
{
        cout << "eat()" << endl;
}
```

Worker.h

```
#include "Person.h"

class Worker : public Person
{
```

```
public:
        Worker();
        ~Worker();
        void work();
        int m_iSalary;
};
```

Worker.cpp

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

Worker::Worker()
{
        cout << "Worker()" << endl;
}

Worker::~Worker()
{
        cout << "~Worker()" << endl;
}

void Worker::work()
{
        cout << "work()" << endl;
}
```

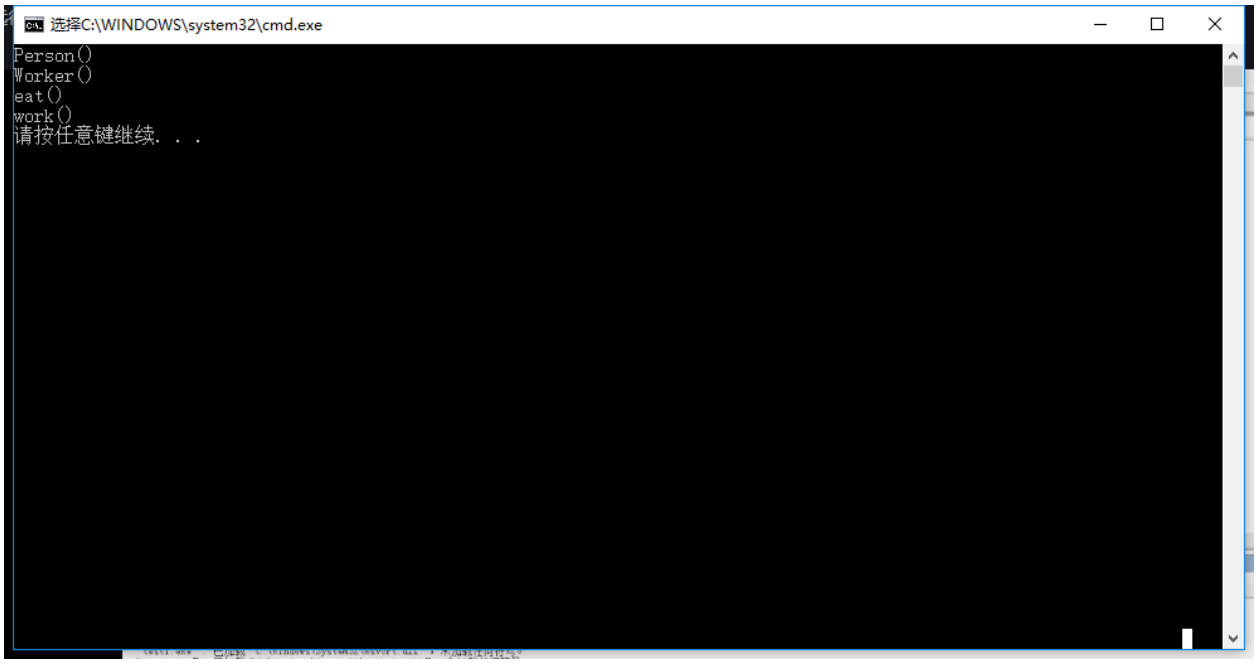
demo.cpp

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

int main()
{
        Worker *p = new Worker();//从堆中实例化对象
        p->m_strName = "Jim";
        p->m_iAge = 10;
        p->eat();
        p->m_iSalary = 1200;
        p->work();

        delete p;
        p = NULL;

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

将demo.cpp改成：

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

int main()
{
        Worker worker;//从栈中实例化
        worker.m_strName = "Jim";
        worker.m_iAge = 10;
        worker.eat();
        worker.m_iSalary = 1200;
        worker.work();

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

运行结果：



说明了在公共继承的情况下，父类的public访问限定符下的数据成员和成员函数被继承到了子类的public访问限定符下。

Person.h

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

class Person
{
public:
        Person();
        ~Person();
        void eat();
protected://+
        string m_strName;
private://+
        int m_iAge;
};
```

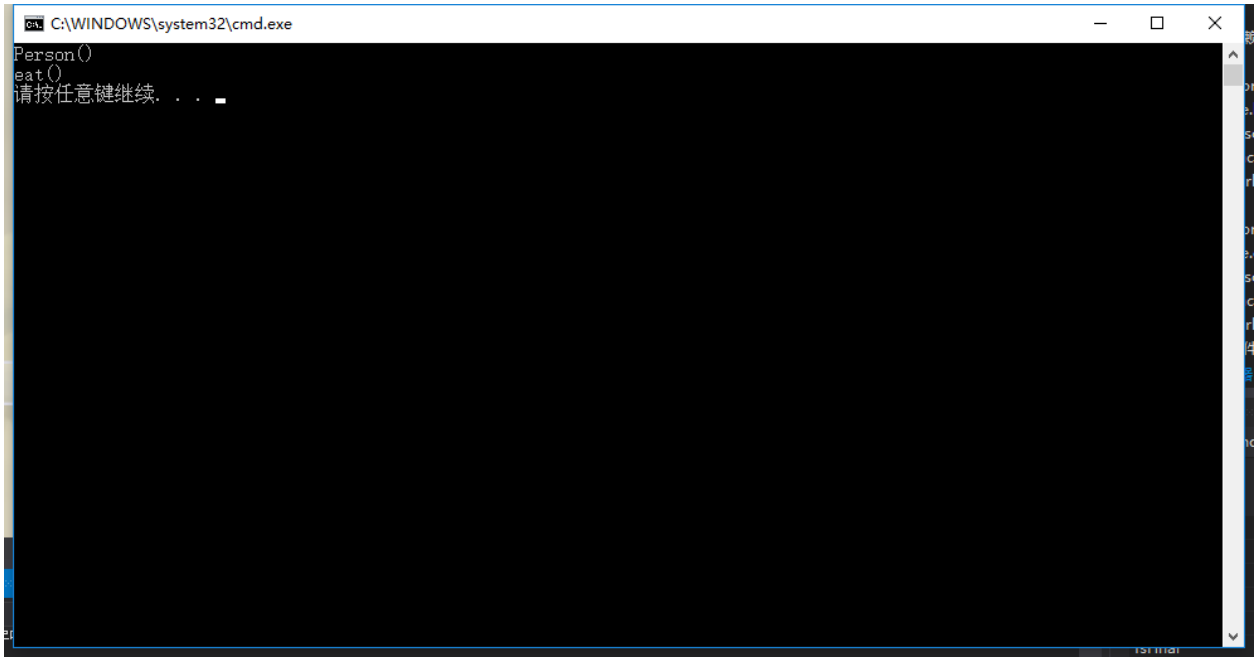
Person.cpp

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

Person::Person()
{
        cout << "Person()" << endl;
}

Person::~Person()
{
        cout << "~Person()" << endl;
}

void Person::eat()
{
        m_strName = "Jim";//+
        m_iAge = 20;//+
        cout << "eat()" << endl;
}
```

demo.cpp

```
//#include "Worker.h"
#include <iostream>
#include <stdlib.h>
#include "Person.h"//+
using namespace std;

int main()
{
        /*Worker worker;
        worker.m_strName = "Jim";
        worker.m_iAge = 10;
        worker.eat();
        worker.m_iSalary = 1200;
        worker.work();*/
        //ctrl+k+c注释掉多行代码

        Person person;

        person.eat();

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

运行结果：



说明了在protected下面和private下面定义的数据成员在eat()的成员函数当中访问效果是等价的。

再将demo.cpp改成：

```
//#include "Worker.h"
#include <iostream>
#include <stdlib.h>
#include "Person.h"//+
using namespace std;

int main()
{
        /*Worker worker;
        worker.m_strName = "Jim";
        worker.m_iAge = 10;
        worker.eat();
        worker.m_iSalary = 1200;
        worker.work();*/
        //ctrl+k+c注释掉多行代码

        Person person;
```

```
        person.m_strName = "Merry";
        person.m_iAge = 20;
        //person.eat();

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

程序报错，说明作为protected的数据成员来说，没有办法在main函数当中通过外部对象直接访问到。同样的，也不能访问private的数据成员。

接下来说说公有继承之后父类的protected下的数据成员，虽然被子类继承，却无法直接使用的情况。 Person.h

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

class Person
{
public:
        Person();
        ~Person();
        void eat();
protected:
        string m_strName;
        int m_iAge;
};
```

Worker.cpp

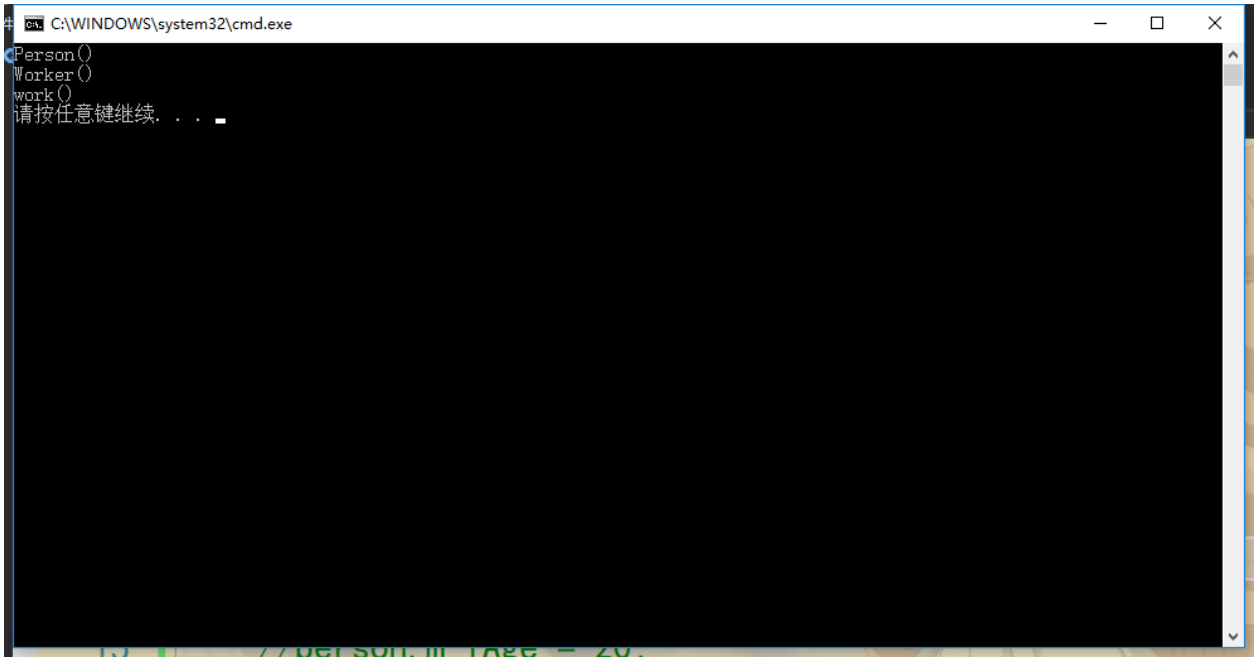
```
#include "Worker.h"
#include <iostream>
using namespace std;

Worker::Worker()
{
        cout << "Worker()" << endl;
}

Worker::~Worker()
{
        cout << "~Worker()" << endl;
}

void Worker::work()
{
        m_strName = "Jim";//+
        m_iAge = 30;//+
        cout << "work()" << endl;
}
```

demo.cpp

```
#include "Worker.h"
#include <iostream>
#include <stdlib.h>
//#include "Person.h"
using namespace std;

int main()
{
        Worker worker;
        worker.work();
        //ctrl+k+u解除掉注释过的多行代码

        //Person person;
        //person.m_strName = "Merry";
        //person.m_iAge = 20;
        //person.eat();

        system("pause");
```

```
        return 0;
}
```

运行结果：



说明了Person当中的protected数据成员被继承到了Worker的protected下面。

Person.h改成：

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

class Person
{
public:
        Person();
        ~Person();
        void eat();
private://change
        string m_strName;
        int m_iAge;
};
```

运行失败。