

第8章 取其精华 发挥优势—继承

1、模拟人的行走、听、说、写

2、为什么需要继承

3、派生类的定义

4、基类与派生类

5、三种继承方式

6、派生类的构造与析构函数

7、点、圆、圆柱体继承设计

8、从U盘到MP3继承设计

三种派生类继承方式

👉 公有继承public

👉 私有继承private

👉 保护继承protected

公有继承方式(public)

👉 采用公有继承方式创建的派生类

👉 对基类各种成员访问权限如下：

👉 基类公有成员相当于派生类的公有成员，即派生类可以象访问

自身公有成员一样访问从基类继承的公有成员

👉 基类保护成员相当于派生类的保护成员，即派生类可以象访问

自身的保护成员一样，访问基类的保护成员

👉 派生类内部成员无法直接访问基类的私有成员

// 公有继承举例

class Person

{protected:

char Name[10];

int Age;

char Sex;

public:

void Register(char *name, int age, char sex) //设置数据成员

{ strcpy(Name, name);

Age = age;

Sex=(sex=='m'? 'm':'f');

}

void ShowMe()

//输出数据成员

{ cout<<Name<<"\t"<<Sex<<"\t"<<Age<<"\t"; }

};

class Employee: public Person

//雇员类定义

{ char Dept[20];

//工作部门

float Salary;

//月薪

public:

Employee()

{ EmployeeRegister("XXX",0,'m',"XXX",0); }

void EmployeeRegister(char *name, int age, char sex, char *dept, float salary);

void ShowEmp();

//显示雇员信息

};

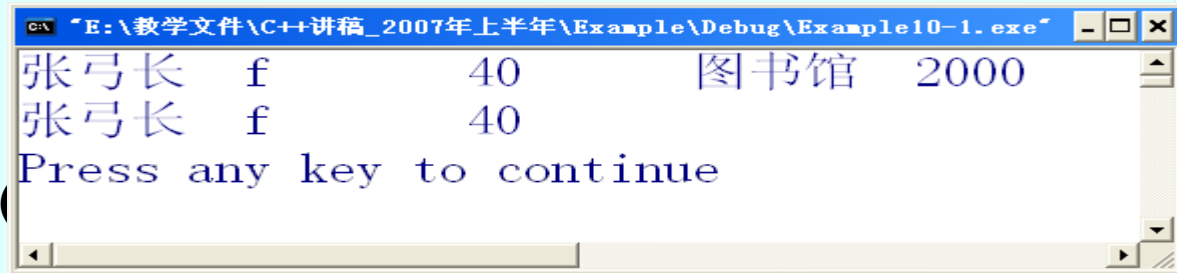
```

void Employee::EmployeeRegister(char *name, int age, char sex, char *dept, float
salary)
{
    Register(name,age,sex);//改成直接操作基类数据成员?
    strcpy(Dept, dept);
    Salary = salary;
}

void Employee::ShowEmp ()
{
    cout<<Name<<"\t"<<Sex<<"\t"<<Age<<"\t"; //将基类protected改为private?
    cout<<Dept<<"\t"<<Salary;
}

int main()//主函数
{
    Employee emp;
    emp.EmployeeRegister(
    emp.ShowEmp();
    cout<<endl;
    emp.ShowMe();
    cout<<endl;
    return 0;
}

```



```

C:\ "E:\教学文件\C++讲稿_2007年上半年\Example\Debug\Example10-1.exe"
张弓长  f      40      图书馆  2000
张弓长  f      40
Press any key to continue

```

私有继承方式(private)

👉 派生类对基类各种成员访问权限如下：

👉 基类公有成员和保护成员都相当于派生类的私有成员，派生类只能通过自身的函数成员访问他们

👉 对于基类的私有成员，无论派生类内部成员或派生类使用者都无法直接访问。

修改前面程序为私有继承

```
int main()//主函数
{
    Employee emp;
    emp.EmployeeRegister("张弓长",40,'f',"图书馆",2000);
    emp.ShowEmp();
    emp.ShowMe();
    return 0;
}
```

👉 主函数中5句话

👉 前三句正常通过

👉 第4句违反继承规则，Showme() 是派生类的私有成员，只能成员函数访问，对象不能访问

```

#include<iostream.h>
#include<string.h>
class Person
{protected:      char      Name[10];           //人员类定义
                  int       Age,Sex;           //姓名
                                                    //年龄、性别
public:   void Register(char *name, int age, char sex) //设置数据成员
        {
            strcpy(Name, name);
            Age = age;
            Sex=(sex=='m'? 'm':'f');
        }
        void ShowMe() {cout<<Name<<"\t"<<Sex<<"\t"<<Age<<"\t";}
};

class Employee: private Person      //雇员类定义
{
    char Dept[20];                  //工作部门
    float Salary;                   //月薪
public:
    Employee()
    { EmployeeRegister("XXX",0,'m',"XXX",0); }
    void EmployeeRegister(char *name, int age, char sex, char *dept, float salary);
    void ShowEmp();                 //显示雇员信息
    char* GetEmployeeName() { return Name; } //取姓名
    char GetEmployeeSex()   { return Sex; }  //取性别
    int GetEmployeeAge()    { return Age; }  //取年龄
};

```



```
Employee::void EmployeeRegister(char *name, int age, char sex, char *dept, float salary)
```

```
{ Register(name,age,sex);
```

```
strcpy(Dept, dept);
```

```
Salary = salary;
```

```
}
```

```
void Employee::ShowEmp()
```

```
{ cout<< Name<<"\t";
```

```
cout<<Age<<"\t";
```

```
cout<<Sex<<"\t";
```

```
cout<<Dept<<"\t"<<Salary<<endl;
```

```
}
```

```
int main()
```

```
{
```

```
Employee emp;
```

```
emp.EmployeeRegister("张弓长",40,'m', "图书馆",2000);
```

```
emp.ShowEmp();
```

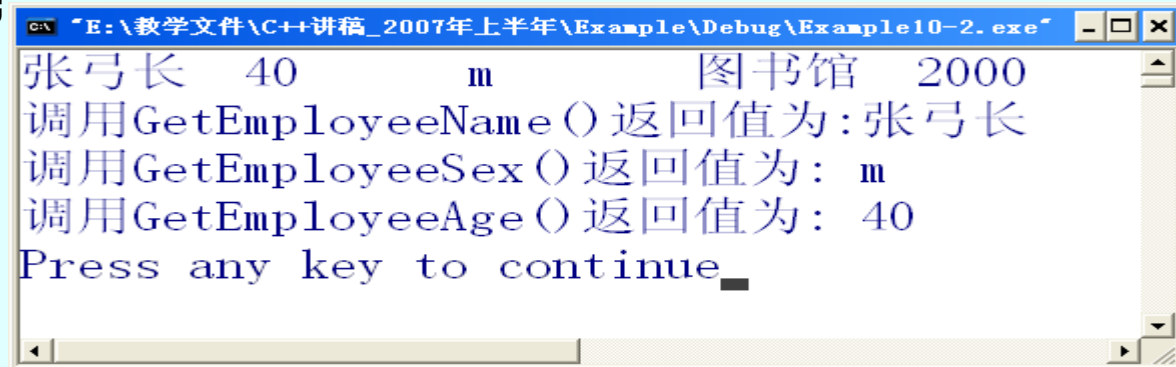
```
cout<<"调用GetEmployeeName()返回值为:"<<emp.GetEmployeeName()<<endl;
```

```
cout<<"调用GetEmployeeSex()返回值为: "<<emp.GetEmployeeSex()<<endl;
```

```
cout<<"调用GetEmployeeAge()返回值为: "<<emp.GetEmployeeAge()<<endl;
```

```
return 0;
```

```
}
```



```
E:\教学文件\C++讲稿_2007年上半年\Example\Debug\Example10-2. exe
张弓长 40 m 图书馆 2000
调用GetEmployeeName() 返回值为: 张弓长
调用GetEmployeeSex() 返回值为: m
调用GetEmployeeAge() 返回值为: 40
Press any key to continue
```

保护继承方式(protected)

- 采用保护继承方式创建的派生类
- 对基类各种成员访问权限如下：
 - 基类的公有成员和保护成员都相当于派生类的保护成员，派生类可以通过自身的成员函数或其子类的成员函数访问他们
 - 对于基类的私有成员，无论派生类内部成员或派生类使用者都无法直接访问

基类Person的定义

```
#include<iostream.h>
#include<string.h>
class Person
{
protected:
    char    Name[20];
    char    Sex;
    int     Age;

public:
    void    Register(char *name, int age, char sex)
    {
        strcpy(Name, name);
        Age = age;
        Sex = (sex == 'm'? 'm': 'f');
    }
    void    ShowMe() { cout << Name << '\t' << Age << '\t' << Sex << endl; }
};
```

派生类学生类保护继承定义

```
class Student : protected Person
```

```
{
```

```
    int Number;
```

```
    char ClassName[10];
```

```
public:
```

```
    void Register(char *classname, int number, char *name, int age, char sex)
```

```
    { strcpy(ClassName, classname);
```

```
      Number = number;
```

```
      strcpy(Name, name);
```

//正确, 引用基类的保护成员

```
      Age = age;
```

//正确, 引用基类的保护成员

```
      Sex = (sex == 'm'? 'm': 'f'); //正确, 引用基类的保护成员
```

```
    }
```

```
void ShowStu()
```

```
{
```

```
    cout << Number << '\t' << ClassName << '\t';
```

```
    ShowMe();
```

```
}
```

```
};
```

保护继承方式测试

```
int main()
{
    Student stu;
    stu.Register("计算机51",85071011,"张弓长",18,'m');
    stu.ShowStu();
    // stu.ShowMe(); //错误，对象不能访问保护成员
    return 0;
}
```

继承方式小结

派生方式	基类中的 访问限定	在派生类中对基类 成员的访问限定	外部函数 如main()
公有继承	public	public	可以直接访问
	protected	protected	不可以直接访问
	private	不可以直接访问	不可以直接访问
私有继承	public	private	不可以直接访问
	protected	private	不可以直接访问
	private	不可以直接访问	不可以直接访问
保护继承	public	protected	不可以直接访问
	protected	protected	不可以直接访问
	private	不可以直接访问	不可以直接访问

感谢收看！