

电子科技大学信息与软件工程学院

标 准 实 验 报 告

(实验) 课程名称: 面向对象程序设计 C++

电子科技大学教务处制表

```
class Person
{
    protected:
        string name;
        int age;
        char gender;
```

```

        static int counter;        // 统计总人数
public:
    Person();
    Person(string, int, char); //通过参数初始化成员
    ~Person();
    void set(string, int, char); //设置人员基本信息
    void show();                //显示人员基本信息
    int get_counter();          //获取人员总数
};

```

(2) 编程实现 Person 类各函数，分析其余各类人员的特征，设计合理的继承结构，并在 Person 类的基础上派生出其余各类人员，每个类需要有构造函数和析构函数、设置人员信息、显示人员基本信息及获取人员总数的成员函数。

(3) 分别采用 public、protected、private 三种继承方式，观察不同继承方式对基类成员在派生类中访问属性的影响。

(4) 在主函数中测试各类的功能。

七、实验器材（设备、元器件）：

PC 计算机、Windows 系列操作系统 、Visual Studio2013 软件

八、实验步骤：

- 1) 创建工程；
- 2) 添加头文件；
- 3) 在头文件中定义基类及各派生类；
- 4) 实现各类的成员函数；
- 5) 在 cpp 文件，编写主程序；
- 6) 编译链接，并运行程序，验证各类的功能；
- 7) 修改基类成员的访问控制属性，验证基类成员的各种访问控制属性在派生类的可见性；
- 8) 修改继承方式，验证不同继承方式对基类成员在派生类访问权限的影响。

九、实验程序及结果分析：

实验程序：

Person.h 头文件

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

class Person                                //类 Person
{
    protected:
        char *name;
        int age;
        char gender;                        //性别，用'T'表示男，'k'表示女
        static int counter;                // 统计总人数
    public:
        Person() {                          //构造函数
            name=new char;
            name[0]='\0';
            age=0;
            gender='\0';
            counter+=1;
        }
        Person(char* na, int ag , char ge){//通过参数初始化成员
            name=na;
            age=ag;
            gender=ge;
            counter+=1;
        }
        virtual void set(char* na, int ag, char ge){    //设置人员基本信息
```

```

        name=na;

        age=ag;

        gender=ge;
    }

    virtual void show(){          //显示人员基本信息

        cout<<"人员基本信息为: "<<endl;

        cout<<"姓名: "<<name<<endl;

        cout<<"年龄: "<<age<<endl;

        cout<<"性别: "<<gender<<endl;

        cout<<"\n"<<endl;

    }

    int get_counter();//获取人员总数
~Person(){}

};

int Person::counter=0;

int Person::get_counter(){          //返回静态数据成员 counter;

    return counter;

}

class Teacher:public Person    {

    private:

        int wn;

        char *title;

    public:

        //Teacher(char      *a="",int      b=0,char      c='\0',int      d=0,char
*e=""):Person(a,b,c),wn(d),title(e){}

        Teacher(char *a,int b,char c,int d,char *e):Person(a,b,c),wn(d),title(e){}

        ~Teacher(){}

        void set(char* na, int ag, char ge,int w,char *ti){          //设置老师基本信

```

息

```
        name=na;

        age=ag;

        gender=ge;

        wn=w;

        title=ti;

    }

    void show(){

        cout<<"该老师基本信息为: "<<endl;

        cout<<"姓名: "<<name<<endl;

        cout<<"年龄: "<<age<<endl;

        cout<<"性别: "<<gender<<endl;

        cout<<"工号: "<<wn<<endl;

        cout<<"职称: "<<title<<endl;

        cout<<"\n"<<endl;

    }

    int get_counter();//获取人员总数

};

int Teacher::get_counter(){           //返回静态数据成员 counter;

    return counter;

}

class Undergraduate:protected Person {

    private:

        int sn;

        char *tie;

    public:

        //Teacher(char    *a="",int    b=0,char    c='\0',int    d=0,char

        *e=""):Person(a,b,c),wn(d),title(e){}
```

```

        Undergraduate(char *a,int b,char c,int d,char *e):Person(a,b,c),sn(d),tie(e){}
        ~Undergraduate(){}

        void set(char* na, int ag, char ge,int s,char *ti){           //设置老师基本信
息
            name=na;
            age=ag;
            gender=ge;
            sn=s;
            tie=ti;
        }

        void show(){
            cout<<"该本科生基本信息为: "<<endl;
            cout<<"姓名: "<<name<<endl;
            cout<<"年龄: "<<age<<endl;
            cout<<"性别: "<<gender<<endl;
            cout<<"学号: "<<sn<<endl;
            cout<<"系别: "<<tie<<endl;
            cout<<"\n"<<endl;
        }

        int get_counter();//获取人员总数
};

int Undergraduate::get_counter(){           //返回静态数据成员 counter;
    return counter;
}

class Postgraduate:private Person {
    private:
        int sn;
        char *tie;

```

```

        char *tutor;

    public:
        //Teacher(char      *a="",int      b=0,char      c='\0',int      d=0,char
        *e=""):Person(a,b,c),wn(d),title(e){}

        Postgraduate(char      *a,int      b,char      c,int      d,char      *e,char
        *f):Person(a,b,c),sn(d),tie(e),tutor(f){}

        ~Postgraduate(){}

        void set(char* na, int ag, char ge,int s,char *ti,char *tu){           //设置老师
        基本信息

            name=na;
            age=ag;
            gender=ge;
            sn=s;
            tie=ti;
            tutor=tu;
        }

        void show(){
            cout<<"该研究生基本信息为: "<<endl;
            cout<<"姓名: "<<name<<endl;
            cout<<"年龄: "<<age<<endl;
            cout<<"性别: "<<gender<<endl;
            cout<<"学号: "<<sn<<endl;
            cout<<"系别: "<<tie<<endl;
            cout<<"导师: "<<tutor<<endl;
            cout<<"\n"<<endl;
        }

        int get_counter();//获取人员总数

};

int Postgraduate::get_counter(){           //返回静态数据成员 counter;

```



```
        return counter;
    }
```

如上程序所示：我设计了一个 **Person** 类，在这个类中包含各类人员共有的基本信息，如姓名、年龄、性别(关于性别我是使用一个字符来进行区分男女性别，其中字符“l”表示男性，字符“k”表示女性)以及静态数据成员 **counter**，用于统计学校人员总数，并先对其进行初始化。

然后我定义了一个构造函数对相关变量进行初始化，并实现 **counter** 加一的操作，接着对构造函数进行重载。

之后便是构造设置人员基本信息的函数和显示人员基本信息的函数，这两个函数我都设计成了虚函数，即在前缀加入 **virtual** 标识，以方便在派生类中进行重载。

最后定义了一个获取成员总数，即获取 **counter** 值的函数以及析构函数。

接下来便是定义 **Teacher**、**Undergraduate**、**Postgraduate** 三个派生类，这三个类分别采用 **public**、**protected**、**private** 继承的方法，其中 **Teacher** 类中添加了私有变量“工号”和“职称”。**Undergraduate** 类中添加了私有变量“学号”和“系别”。**Postgraduate** 类中添加了私有变量“学号”、“系别”和“导师”。并且在三个派生类中都重载了 **set()**和 **show()**函数，并定义了获取人员总数 **counter** 的函数。

Person.cpp 主程序

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

int main(){
    Person S("danyi",20,'l');
```

```

S.show();

S.set("dy",21,'k');

S.show();


Teacher t("yangnan",20,'l',1,"teather");

t.show();

t.set("yn",30,'k',648,"professor");

t.show();


Undergraduate u("tangjiaqi",20,'k',2016008,"ruanjian");

u.show();

u.set("tjj",30,'l',2016009,"jingji");

u.show();


Postgraduate p("hsj",20,'k',2016010,"ruanjian","ly");

p.show();

p.set("hushijie",30,'l',2016011,"jingji","luo");

p.show();


Person *s1=&t;

s1->set("yue",20,'l');

s1->show();


cout<<"人员总数为: "<<endl;

cout<<s1->get_counter()<<endl;

return 0;

}

```

如上程序所示，我分别用 Person 类、Teacher 类、Undergraduate 类、Postgraduate 类定义了四个实例对象，使用的方法是调用构造函数进行实例化。并且分别进行

的打印和设置基本信息的操作。最后使用基类定义了一个指针用来指向对象 t 的地址，再调用该指针引用的成员函数 set() 和 show()。最后进行输出输出结果如下图所示：

```
C:\Users\n'n\Desktop\My bad\C++\C++
人员基本信息为：
姓名： danyi
年龄： 20
性别： 1

人员基本信息为：
姓名： dy
年龄： 21
性别： k

该老师基本信息为：
姓名： yangnan
年龄： 20
性别： 1
工号： 1
职称： teather

该老师基本信息为：
姓名： yn
年龄： 30
性别： k
工号： 648
职称： professor

该本科生基本信息为：
姓名： tangjiaqi
年龄： 20
性别： k
学号： 2016008
系别： ruanjian

该本科生基本信息为：
姓名： tjj
年龄： 30
性别： 1
学号： 2016009

该研究生基本信息为：
姓名： hsj
年龄： 20
性别： k
学号： 2016010
系别： ruanjian
导师： ly

该研究生基本信息为：
姓名： hushijie
年龄： 30
性别： 1
学号： 2016011
系别： jingji
导师： luo

该老师基本信息为：
姓名： yue
年龄： 20
性别： 1
工号： 648
职称： professor

人员总数为：
4
请按任意键继续. . .
```

(由于图片过长，所以我分了两端表示输出结果)

如上图所示：每一个对象定义后便进行一次输出，再调用了 set() 函数后，再一次调用 show() 函数进行输出。然后输出成员总数为四人。

//修改访问属性

```
private:
    char *name;
    int age;
    char gender;           //性别，用'T'表示男，'k'表示女
    static int counter;    // 统计总人数
```

//修改继承方式

```
class Teacher:public Person
class Undergraduate:public Person
class Postgraduate:public Person
```

结果分析：

修改之前的基类成员变量为保护段成员，派生类无论是 `public`、`protected` 或 `private` 继承都能访问基类的成员变量，但是只要基类成员变量设置为私有后，派生类就算公有继承也无法访问基类私有段成员。

十、实验结论：

继承是面向对象语言的一个重要机制，通过继承可以在一个一般类的基础上建立新类，被继承的类称为基类，实验中，在修改基类访问属性之前，基类成员变量是能够被访问输出的，派生类无论是 `public`、`protected` 或 `private` 继承都能访问基类的成员变量，但是只要基类成员变量设置为私有后，派生类就算公有继承也无法访问基类私有段成员。

而且在定义了基类 `set()` 和 `show()` 函数为虚函数后，用基类定义的指针指向派生类的对象地址后，便能够通过指针来访问派生类中的函数。

十一、总结及心得体会：

通过本次实验，我了解到了继承的关系，也了解了是虚函数的使用方法和作用。也对静态方面有了更深的理解静态类，静态变量，静态方法，这些东西都是属于它父级的全局性的东西，如静态类在 namespace 下共用，以及静态变量和静态方法，都是属于 namespace 级别的共享数据，不存在继承不继承的问题。

十二、对本实验过程及方法、手段的改进建议：

我认为本实验充分的利用了一些 C++ 的基本程序操作手段，无修改意见。

报告评分：

指导教师签字：