序号：

实验 3 继承与派生

班级 信安 20-2　　学号 20101110201　　姓名 李天昊

[实验目的]

1、了解继承在面向对象程序设计中的重要作用；
2、进一步理解继承与派生的概念；
3、学会通过继承派生出一个新类的方法。

[实验要求]

给出以下各实验内容的源程序代码，并把编译、运行过程中出现的问题以及解决方法填入实验报告中，按时上交。

[实验学时]2 学时。

[实验内容]

1、 声明一个人员类（Person ），包括 3 个数据成员：name（姓名）、age（年龄）、sex（性别）；2 个成员函数：构造函数和输出相关信息的函数 display()。利用单继承的方式声明一个学生（Student）派生类，其中增加 2 个数据成员：grade（年级）、score（总学分）；3 个成员函数：构造函数、输出函数 show()和增加学分的函数 add()。在定义派生类对象时给出初始化的数据，然后输出这些数据。

```
#include <iostream>
#include <string>

using namespace std;

class CPerson {
private:
    string m_name;   // 姓名
    size_t m_age;    // 年龄
    string m_sex;    // 性别
public:
    CPerson() = default;
    CPerson(string name, size_t age, string sex): m_name(name), m_age(age), m_sex(sex) {  }
    inline void Display(ostream &os = cout);
};
```

```
inline void CPerson::Display(ostream &os) {
    os << ">> name:    " << m_name << endl;
    os << " | age:      " << m_age << endl;
    os << " | sex:      " << m_sex << endl;
}


class CStudent: public CPerson {
private:
    size_t m_grade;         // 年级
    double m_score;         // 总学分
public:
    CStudent(string name, size_t age, string sex, size_t grade, double score =
0):
                CPerson(name, age, sex), m_grade(grade), m_score(score) {    }
    inline void Display(ostream &os = cout);
    inline void add(double a);

};

inline void CStudent::Display(ostream &os) {
    CPerson::Display();
    os << " | grade: " << m_grade << endl;
    os << " | score: " << m_score << endl;
}

inline void CStudent::add(double a) {
    m_score += a;
}

int main() {
    CStudent stu("David Lee", 18, "male", 1, 10);
    stu.Display();
    return 0;
}
```

2、分别定义 Teacher(教师)类和 Cadre(干部)类，采用多重继承方式由这两个类派生出新类 Teacher_Cadre(教师兼干部)。要求：

① 在两个基类中都包含姓名、出生日期(日期类 Date 的子对象)、性别、地址、电话等数据成员。

② 在 Teacher 类中还包含数据成员 title(职称)，在 Cadre 类中还包含数据成员 post(职务)， 在 Teacher_Cadre 类中还包含数据成员 wages(工资)。

③ 对两个基类中的姓名、出生日期、性别、地址、电话等数据成员用相同的名字，在引用这些数据成员时，指定作用域。

④ 在类体中声明成员函数，在类外定义成员函数。

⑤ 在派生类 Teacher_Cadre 的成员函数 show()中调用 Teacher 类中的 display()函数，输出姓名、出生日期、性别、职称、地址、电话，然后再用 cout 语句输出职务与工资。

```
#include <iostream>
#include <string>

using namespace std;

class CDate {
private:
	int d;
	int m;
	int y;
public:
	CDate(int _d, int _m, int _y): d(_d), m(_m), y(_y) {    }
	friend ostream & operator<<(ostream &os, CDate &d);
};

ostream & operator<<(ostream &os, CDate &date) {
	os << date.d << "-" << date.m << "-" << date.y;
	return os;
}

class CTeacher {
private:
	string m_name;          // 姓名
	CDate m_birthday;     // 出生日期
	string m_sex;           // 性别
```

```
    string m_address;     // 地址
    string m_phone_no;    // 电话号码
    string m_title;       // 职称
public:
    CTeacher(
        const string &name, const CDate &birth, string sex,
        const string &addr, const string &pho, const string &title):
        m_name(name),  m_birthday(birth),  m_sex(sex),  m_address(addr),
  m_phone_no(pho), m_title(title) {   }
protected:
    void Display(ostream &os = cout);
};

void CTeacher::Display(ostream &os) {
    os << " >> name:         " << m_name << endl;
    os << "    | birthday: " << m_birthday << endl;
    os << "    | sex:          " << m_sex << endl;
    os << "    | address:    " << m_address << endl;
    os << "    | phone:       " << m_phone_no << endl;
    os << "    | title:       " << m_title << endl;
}

class CCadre {
private:
    string m_name;          // 姓名
    CDate m_birthday;     // 出生日期
    string m_sex;           // 性别
    string m_address;     // 地址
    string m_phone_no;    // 电话号码
    string m_post;          // 职务
public:
    CCadre(
        const string &name, const CDate &birth, string sex, const string
  &addr,
        const string &pho, const string &post):
        m_name(name),  m_birthday(birth),  m_sex(sex),  m_address(addr),
  m_phone_no(pho), m_post(post) {   }
protected:
```

```
        string & get_post();
};

string & CCadre::get_post() {
        return m_post;
}

class CTeacher_Cadre: public CTeacher, public CCadre {
private:
        double m_wages;
public:
        CTeacher_Cadre(
                const string &name, const CDate &birth, string sex, const string
    &addr,
                const string &pho, const string &title, const string &post, double
    wages):
                CTeacher(name, birth, sex, addr, pho, title), CCadre(name, birth,
    sex, addr, pho, post), m_wages(wages) {    }
        void Show();
};

void CTeacher_Cadre::Show() {
        Display();
        cout << "    | post:          " << get_post() << endl;
        cout << "    | wages:         " << m_wages << endl;
}

int main() {
        CTeacher_Cadre tc("David", CDate(2002, 5, 28), "male", "Beijing,
    China", "13121515269", "title", "post", 12345.67);
        tc.Show();
        return 0;
}
```

[实验总结]给出对本次实验的总结。

1、进一步了解继承在面向对象程序设计中的重要作用；
2、加深了对继承与派生的概念的理解；
3、掌握通过继承派生出新类的方法。