实验六 继承（三）—— 多继承及继承的应用

> 张林鹏_2021032449

一、实验目的

1. 理解多继承的概念, 熟悉多继承的定义及应用;
2. 理解多继承方式下的二义性产生原因, 熟悉解决二义性的方法;
3. 进一步熟悉继承的综合应用.

二、实验内容

程序1: exp_601.cpp

1. 编译该程序, 会出现编译错误, 其原因是在定义Z类的`print()`成员函数中的`show()`函数调用时出现了二义性. 解决方法是:
2. 将`show(); //显示x的值`改为: `X::show()`.
3. 将`show(); //显示y的值`改为: `Y::show()`.
4. 调试成功后输出结果过为:

```
x=3  y=4  z=5
x=10  y=20  z=30
```

5. 成员函数`void set_xyz(int a,int b,int c)`中`set_x(a)`将派生类 **Z** 的对象 **x** 成员变量设置为 **a**,

   `set_y(b)`的作用是将派生类 **Z** 对象的 **y** 成员变量设置为 **b**.

程序2: exp_602.cpp

6. 编译运行程序的输出结果是:

```
Constructing base a=2
Constructing base1 b=3
```

```
Constructing base2 c=4
Constructing derived d=5
```

7. 从输出结果中可看出, 间接基类 `base` 被调用了两次, 产生原因是 `derived` 的两个直接基类 `base1`, `base2` 有用共同基类 `base`, 产生了二义性.

8. 将程序中:

   i. `class base1:public base` 改为: `class base1:virtual public base`;
   ii. `class base2:public base` 改为: `class base2:virtual public base`;
   iii. `derived(int x1,int x2,int x3,int x4):base1(x1,x2),base2(x1,x3)` 改为: `derived(int x1,int x2,int x3,int x4):base(x1),base1(x1,x2),base2(x1,x3)`,

   编译运行输出结果为:

```
Constructing base a=2
Constructing base1 b=3
Constructing base2 c=4
Constructing derived d=5
```

9. `virtual public base` 表示间接基类 `base` 为虚基类.

程序: exp_603.cpp & person.h & teacher.h & student.h & score.h

- exp_603.cpp

```cpp
#include <bits/stdc++.h>
#include "score.h"
using namespace std;

void input_base(score *p, int n);          // 学生基本数据输入
void input_score(score *p, int n, int m); // 学生成绩输入
void print_base(score *p, int n);          // 学生基本数据输出
void print_score(score *p, int n, int m); // 学生成绩输出
score &average(score *p, int n, int m);    // 普通函数：平均成绩计算
void sort(score *p, int n, int m);         // 普通函数：按平均成绩排序

int main()
{
    int n, m;
    cout << "学生人数：";
    cin >> n;
    cout << "考试科数：";
    cin >> m;
    score *p, aver;
    p = new score[n]; // 动态分配内存单元—动态数组
    if (p == NULL)
```

```
    {
        cout << "内存分配失败" << endl;
        return 0;
    }
    int ch;
    do
    {
        cout << "\n\n      请选择:\n";
        cout << "  1. 输入学生基本数据\n";
        cout << "  2. 输入学生成绩\n";
        cout << "  3. 计算课程平均成绩\n";
        cout << "  4. 输出学生基本数据\n";
        cout << "  5. 输出学生成绩\n";
        cout << "  6. 按平均成绩排序\n";
        cout << "  0. 退出\n";
        cout << "\n    输入你的选择:";
        cin >> ch;
        switch (ch)
        {
        case 1:
            input_base(p, n);
            break;
        case 2:
            input_score(p, n, m);
            break;
        case 3:
            aver = average(p, n, m);
            break;
        case 4:
            print_base(p, n);
            break;
        case 5:
            print_score(p, n, m);
            aver.print_score();
            break;
        case 6:
            sort(p, n, m);
            break;
        case 0:
            break;
        }
    } while (ch);
    delete[] p; // 释放内存
}

void input_base(score *p, int n) // 学生基本数据输入
{
    int i, id, y1, m1, d1, y2, m2, d2;
    char name[11], sex[3], dpt[20];
    cout << "\n  请输入学生基本数据:";
    for (i = 0; i < n; i++)
    {
        cout << "第" << i + 1 << "个学生:\n";
```

```
        cout << "学号:";
        cin >> id;
        cout << "姓名:";
        cin >> name;
        cout << "性别:";
        cin >> sex;
        cout << "出生年:";
        cin >> y1;
        cout << "出生月:";
        cin >> m1;
        cout << "出生日:";
        cin >> d1;
        cout << "所在系:";
        cin >> dpt;
        cout << "入学年:";
        cin >> y2;
        cout << "入学月:";
        cin >> m2;
        cout << "入学日:";
        cin >> d2;
        p[i].set_person(name, sex, y1, m1, d1);  // 完成函数的调用
        p[i].set_student(id, sex, y2, m2, d2); // 完成函数的调用
    }
}

void input_score(score *p, int n, int m)
{
    int i, j;
    float x[M];
    for (i = 0; i < n; i++)
    {
        cout << p[i].get_id() << p[i].get_name() << "的成绩：" << endl;
        for (j = 0; j < m; j++)
        {
            cout << "第" << j + 1 << "科成绩：";
            cin >> x[j];
        }
        p[i].set_score(x, i); // 完成函数的调用
    }
}

void print_base(score *p, int n)
{
    int i;
    for (i = 0; i < n; i++)
        p[i].print_base();
    cout << endl;
}

void print_score(score *p, int n, int m)
{
    int i;
    for (i = 0; i < n; i++)
```

```
        p[i].print_score();
}

score &average(score *p, int n, int m) // 用返回引用的方法
{
    int i, j;
    float s[M] = {0};
    static score aver; // 返回的对象必须是静态的
    for (j = 0; j < m; j++)
    {
        for (i = 0; i < n; i++)
            s[j] = s[j] + p[i].get_score(j);
        s[j] = s[j] / n;
    }
    aver.set_person("平均成绩", " ", 0, 0, 0);
    aver.set_score(s, j); // 完成函数的调用
    return aver;
}

void sort(score *p, int n, int m) // 选择法排序：完成空白处的内容
{
    score t;
    float a;
    int i, j, k;
    for (i = 0; i < n - 1; i++)
    {
        a = p[i].get_aver();
        k = i;
        for (j = i + 1; j < n; j++)
            if (p[j].get_aver() < a)
            {
                a = p[j].get_aver();
                k = j;
            }
        if (i != k)
        {
            t = p[i];
            p[i] = p[k];
            p[k] = t;
        }
    }
}
```

- person.h

```
#ifndef PERSON_H
#define PERSON_H

#include "date.h"
#include <string>
#include <string.h>
```

```
using namespace std;

class person
{
protected:
    char name[11];
    char sex[2];
    date birthday;

public:
    person(void); // 无参构造函数
    void set_person(char *na, char *s, int y, int m, int d);
    char *get_name(void)
    {
        return name;
    } // 完成成员函数的定义
    char *get_sex(void)
    {
        return sex;
    } // 完成成员函数的定义
    int get_year(void)
    {
        return birthday.get_year();
    } // 完成成员函数的定义
    int get_month(void)
    {
        return birthday.get_month();
    }
    int get_day(void)
    {
        return birthday.get_day();
    } // 完成成员函数的定义
    void print(void);
};

person::person(void) // 无参构造函数
{
    strcpy(name, "无名氏");
    strcpy(sex, "男");
    birthday.set_date(1980, 1, 1);
}

void person::set_person(char *na, char *s, int y, int m, int d)
{
    strcpy(this->name, na);
    strcpy(this->sex, s);
    birthday.set_date(y, m, d);
} // 完成成员函数的定义

void person::print(void)
{
    cout << "姓名:" << name << endl;
    cout << "性别:" << sex << endl;
```

```
    cout << "出生日期:" << birthday.get_year() << "年";
    cout << birthday.get_month() << "月";
    cout << birthday.get_day() << "日" << endl;
}

#endif
```

- date.h

```
#ifndef DATE_H
#define DATE_H
#include <bits/stdc++.h>

class date
{
private:
    int year, month, day; // 年、月、日三个私有成员
public:
    date(void)
    {
        year = 1980;
        month = 1;
        day = 1;
    }
    void set_date(int y, int m, int d)
    {
        year = y;
        month = m;
        day = d;
    } // 完成成员函数的定义
    int get_year(void)
    {
        return year;
    } // 完成成员函数的定义
    int get_month(void)
    {
        return month;
    } // 完成成员函数的定义
    int get_day(void)
    {
        return day;
    } // 完成成员函数的定义
};

#endif
```

- student.h

```
#ifndef STUDENT_H
#define STUDENT_H
```

```cpp
#include "person.h"
#include "date.h"

class student : public person
{
public:
    int id;
    char department[20];
    date enterdate;

public:
    student(void);
    void set_student(int n, char *s, int y, int m, int d);
    int get_id(void)
    {
        return id;
    } // 完成成员函数的定义
    char *get_department(void)
    {
        return department;
    } // 完成成员函数的定义
    int get_enteryear(void)
    {
        return enterdate.get_year();
    } // 完成成员函数的定义
    int get_entermonth(void)
    {
        return enterdate.get_month();
    } // 完成成员函数的定义
    int get_enterday(void)
    {
        return enterdate.get_day();
    } // 完成成员函数的定义
    void print(void);
    void print_base();
};

student::student(void)
{
    strcpy(name, "无名氏");
    strcpy(sex, "男");
    birthday.set_date(1980, 1, 1);
    id = 0;
    strcpy(department, "计算机");
    enterdate.set_date(2000, 9, 1);
}
void student::set_student(int n, char *s, int y, int m, int d)
// n、s、y、m、d分别为id、department、enterdate提供值
{
    id = n;
    strcpy(department, s);
    enterdate.set_date(y, m, d);
} // 完成成员函数的定义
```

```
void student::print(void)
{
    cout << "学号:" << id << endl;
    person::print();
    cout << "系(专业):" << department << endl;
    cout << "进校日期:" << enterdate.get_year() << "年";
    cout << enterdate.get_month() << "月";
    cout << enterdate.get_day() << "日" << endl;
}
void student::print_base()
{
    cout << setw(8) << get_id();
    cout << setw(12) << get_name();
    cout << setw(4) << get_sex();
    cout << setw(6) << get_year() << "-" << get_month() << "-" << get_day();
    cout << setw(20) << get_department();
    cout << setw(6) << get_enteryear() << "-" << get_entermonth();
    cout << "-" << get_enterday() << endl;
}

#endif
```

- score.h

```
#ifndef SCORE_H
#define SCORE_H

#include "student.h"
const int M = 10;
class score : public student
{
private:
    float sc[M], aver;
    int m;

public:
    score(void);                        // 无参构造函数
    void set_score(float x[], int n); // 提供成绩
    float get_score(int i)              // 得到第i科成绩
    {
        return sc[i];
    }                       // 完成成员函数的定义
    float get_aver(void) // 得到平均成绩
    {
        return aver;
    } // 完成成员函数的定义
    void print(void);
    void print_score(void);
};

score::score(void) // 无参构造函数
```

```
{
    strcpy(name, "无名氏");
    strcpy(sex, "男");
    birthday.set_date(1980, 1, 1);
    id = 0;
    strcpy(department, "计算机");
    enterdate.set_date(2000, 9, 1);
    int i;
    m = M;
    for (i = 0; i < m; i++)
        sc[i] = 0;
    aver = 0;
}
void score::set_score(float x[], int n) // 提供成绩:完成成员函数的定义
{
    int i;
    float sum = 0;
    m = n;
    for (i = 0; i < m; i++)
    {
        sc[i] = x[i];
        sum += x[i];
    }
    aver = sum / m;
}

void score::print(void) // 重载输出print()
{
    student::print();
    int i;
    for (i = 0; i < m; i++)
        cout << "  " << sc[i];
    cout << "  " << aver << endl;
}
void score::print_score(void)
{
    int j;
    cout << setw(8) << get_id();
    cout << setw(12) << get_name();
    for (j = 0; j < m; j++)
        cout << setw(6) << get_score(j);
    cout << " " << setw(6) << get_aver() << endl;
}

#endif
```