广州大学学生实验报告

2018年4月7日

**开课学院及实验室：**计算机科学与工程实验室电子楼412A室

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机 | **专业班级** | 软件171 | **姓名** | 谢金宏 | **学号** | 1706300001 |
| **实验课程名称** | | 面向对象程序设计实验 | | **成绩** | |  | |
| **实验项目名称** | | 继承与多态性 | | **指导老师** | | 陶文正 | |

# 实验目的及要求

1. 掌握继承、基类和派生类的概念。
2. 掌握初始化基类成员的方法。
3. 掌握派生类对基类的继承。
4. 学习虚函数和纯虚函数的定义与使用方式。
5. 理解抽象类的概念，学习如何用指针指向其他的派生类，实现多态性。
6. 掌握抽象类的定义与使用方式，并注意指针的用法。
7. 学习如何使用虚函数、纯虚函数、抽象类和实现类的多态性。

# 实验设备及平台

1. 硬件环境：计算机
2. 软件环境：G++ 7.2.0, Visual Studio Code

# 实验内容及步骤

1. **定义派生类的实例**

#include <string>

#include <iostream>

using namespace std;

class Student {

public:

Student() = default;

Student(string serial, string name, int sex, int year, int month, int day) :

serial\_(serial), name\_(name), sex\_(sex),

year\_(year), month\_(month), day\_(day) {};

void set\_serial(string serial) { serial\_ = serial; }

void set\_name(string name) { name\_ = name; }

void set\_sex(int sex) { sex\_ = sex; }

void set\_year(int year) { year\_ = year; }

void set\_month(int month) { month\_ = month; }

void set\_day(int day) { day\_ = day; }

string serial() { return serial\_; }

string name() { return name\_; }

int sex() { return sex\_; }

int year() { return year\_; }

int month() { return month\_; }

int day() { return day\_; }

protected:

string serial\_, name\_;

int sex\_, year\_, month\_, day\_;

};

class ExtendedStudent: public Student {

public:

ExtendedStudent() = default;

ExtendedStudent(

string serial, string name, int sex, int year,

int month, int day, int C, int English

) : Student(serial, name, sex, year, month, day) {

C\_ = C, English\_ = English;

}

void set\_C(int C) { C\_ = C; }

void set\_English(int English) { English\_ = English; }

int C() { return C\_; }

int English() { return English\_; }

void List() {

cout << "学号：" << serial() << endl;

cout << "姓名：" << name() << endl;

cout << "性别：" << (sex() == 0 ? "女" : "男") << endl;

cout << "出生日期：" << year() << "年" << day() << "月" << day() << "日" << endl;

cout << "C++：" << C() << endl;

cout << "英语：" << English() << endl;

}

protected:

int C\_, English\_;

};

template <typename T>

void input(T &t, const string hint)

{

cout << hint; cin >> t;

}

int main() {

int n;

cout << "请输入学生人数：" << endl;

cin >> n;

ExtendedStudent \*s = new ExtendedStudent[n];

for (int i=0; i<n; ++i) {

cout << "正在输入第" << i+1 << "位学生" << endl;

string serial, name;

int sex, year, month, day, C, English;

input(serial, string("学号："));

input(name, string("姓名："));

input(sex, string("性别（非零数字表示男性）："));

cout << "出生年月日：" << endl;

cin >> year >> month >> day;

input(C, string("C语言成绩："));

input(English, string("英语成绩："));

s[i] = ExtendedStudent(serial, name, sex, year, month, day, C, English);

}

cout << endl;

cout << "下面列出已输入学生的信息：" << endl;

for (int i=0; i<n; ++i) {

if (i) cout << endl;

cout << "第" << i+1 << "位学生" << endl;

s[i].List();

}

delete[] s;

}

以上程序提醒输入学生的数量和对应数量学生的信息；输入完毕后将列出输入学生的信息。程序使用了面向对象编程中的继承特性。

1. **多重继承的实例**

#include <iostream>

using namespace std;

class Base1 {

public:

    void show\_i() { cout << i << endl; }

protected:

    int i;

};

class Base2 {

public:

    void show\_j() { cout << j << endl; }

protected:

    int j;

};

class Derived : public Base1, public Base2 {

public:

    void set(int x, int y) { i = x, j = y; }

};

int main() {

    Derived obj;

    obj.set(5, 7);

    obj.show\_i();

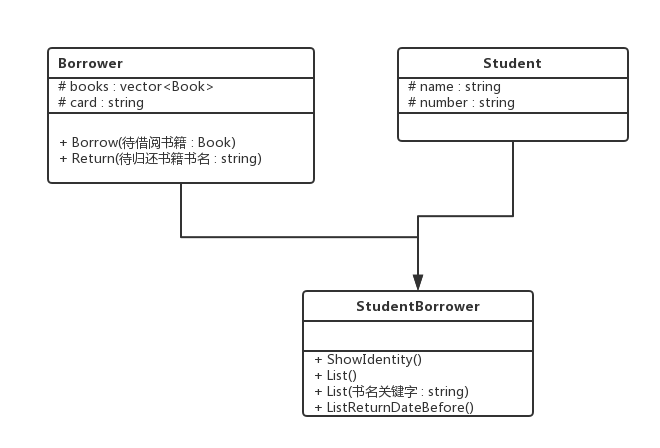
    obj.show\_j();

}

以上程序是多重继承的一个例子。

1. **抽象并编程现实生活中图书馆借阅的相关内容**

设有“借阅者”类，主要功能是记录借书证号以及借阅书籍，并记录借还书的日期等；“学生”类，记录学生的姓名和学号；设计“学生借阅者”类，它是“借阅者”类和“学生”类的派生类。在此基础上设计函数，显示借阅者的相关信息。



#include <algorithm>

#include <numeric>

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <vector>

using namespace std;

// Date结构 - 表示特定日期

// 构造器：

// Date(int day, int month, int year) - 以指定的年月日构造日期

// 方法：

// Show() - 从标准输出流输出日期

// toString() - 返回代表当前日期的字符串

// 重载等于号操作符以及小于号操作符

struct Date {

static const int kDaysInMonth[13];

    int day, month, year;

Date() {

year = 2018;

month = rand()%12+1;

day = rand()%kDaysInMonth[month]+1;

}

Date(int day, int month, int year) {

month = max(1, month), month = min(12, month); // 保证日期合法，下同

day = max(1, day), day = min(kDaysInMonth[month], day);

this->year = year, this->month = month, this->day = day;

}

    void Show() {

        cout.fill('0');

        cout << setw(4) << year << '-' << setw(2) << month << '-' << setw(2) << day;

    }

string toString() {

stringstream ss; ss.fill('0');

ss << setw(4) << year << '-' << setw(2) << month << '-' << setw(2) << day;

string ret; ss >> ret;

return ret;

}

bool operator == (const Date d) const {

return (day == d.day) && (month == d.month) && (year == d.year);

}

bool operator < (const Date d) const {

if (year != d.year) return year < d.year;

if (month != d.month) return month < d.month;

return day < d.day;

}

bool operator <= (const Date d) const {

return \*this < d || \*this == d;

}

};

const int Date::kDaysInMonth[13] = {

0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31,

};

// Book结构 - 表示借阅的特定书籍

// 数据成员：

// name - 书籍名称

// borrow\_date, return\_date - 借出日期和应还日期

struct Book {

    string name;

    Date borrow\_date, return\_date;

};

// Borrower类 - 借阅者

// 数据成员：

// card - 借书证号

// books - 借阅的书籍

// 方法：

// Borrow(Book&) - 借阅给出的书籍

// Return(string) - 归还指定名称的书籍

class Borrower {

public:

void Borrow(const Book &);

bool Return(const string);

protected:

string card\_;

vector<Book> books\_;

};

void Borrower::Borrow(const Book &book) {

books\_.push\_back(book);

}

bool Borrower::Return(const string name) {

bool already\_returned = false;

for (auto it=books\_.begin(); it!=books\_.end(); ++it)

{

if (it->name == name) {

books\_.erase(it);

already\_returned = true;

}

}

return already\_returned;

}

// Student类 - 学生

// 数据成员：

// name - 姓名

// number - 学号

class Student {

public:

Student(string name, string number) {

name\_ = name, number\_ = number;

}

protected:

string name\_;

string number\_;

};

// StudentBorrower - 学生借阅者

// 继承自Borrower, Student

// 方法：

// ShowIdentity() - 显示借阅者的身份信息

// List() - 列出借阅的所有书籍

// List(string contain\_string) - 列出借阅书籍中书名包含contain\_string的书籍

// ListReturnDateBefore(Date date) - 列出借阅书籍中归还日期在指定日期之前的书籍

class StudentBorrower : public Borrower, public Student {

public:

StudentBorrower(string name, string number, string card) : Student(name, number) {

card\_ = card;

}

void ShowIdentity();

void List();

void List(string book\_name);

void ListReturnDateBefore(Date);

private:

void List(const vector<unsigned>&);

};

void StudentBorrower::ShowIdentity() {

cout << "借阅者：" << name\_ << " (" << number\_ << ")" << endl;

cout << "借阅证号：" << card\_ << endl;

}

void StudentBorrower::List(const vector<unsigned> &idx) {

unsigned len = 0;

for (int i : idx) {

len = max(unsigned(books\_[i].name.size()), len);

} // 统计待列出的书籍中最长书名的长度

cout << "#" << " | " << "标题" << " | " << "借书日期" << " | " << "应还日期" << endl;

for (unsigned i=0; i<idx.size(); ++i) {

cout.fill('0'); cout << right << setw(2) << i+1 << " | ";

cout.fill(' '); cout << left << setw(len) << books\_[idx[i]].name << " | ";

cout << books\_[idx[i]].borrow\_date.toString() << " | ";

cout << books\_[idx[i]].return\_date.toString() << endl;

}

}

void StudentBorrower::List() {

if (!books\_.size()) return;

vector<unsigned> idx(books\_.size());

iota(idx.begin(), idx.end(), 0);

List(idx);

}

void StudentBorrower::List(string contain\_string) {

vector<unsigned> idx;

for (unsigned i=0; i<books\_.size(); ++i) {

if (books\_[i].name.find(contain\_string) != string::npos) idx.push\_back(i);

}

if (idx.size()) List(idx);

else cout << "借阅的书籍中书名没有包含“" << contain\_string << "”的书籍" << endl;

}

void StudentBorrower::ListReturnDateBefore(Date date) {

vector<unsigned> idx;

for (unsigned i=0; i<books\_.size(); ++i) {

if (books\_[i].return\_date < date) idx.push\_back(i);

}

if (idx.size()) List(idx);

else cout << "借阅的书籍中没有应还日期在" << date.toString() << "之前的书籍" << endl;

}

int main()

{

// 随机生成书籍信息

Book books[10] {

{"钢铁是怎样炼成的", Date(), Date()},

{"青年近卫军", Date(), Date()},

{"大卫科波菲尔", Date(), Date()},

{"活着", Date(), Date()},

{"人性的弱点", Date(), Date()},

{"静静的顿河", Date(), Date()},

{"黄金时代", Date(), Date()},

{"再见，冥王星", Date(), Date()},

{"你好，乔安", Date(), Date()},

{"CLR via C#", Date(), Date()},

};

// 模拟借书过程

StudentBorrower borrower("Lightyears", "1706300001", "GZDX-1706300001");

borrower.ShowIdentity();

for (int i=0; i<10; ++i) {

borrower.Borrow(books[i]);

}

cout << endl;

cout << "列出借阅的所有书籍：" << endl;

borrower.List();

cout << endl;

cout << "列出借阅书籍中书名包含“的”字的书籍：" << endl;

borrower.List("的");

cout << endl;

cout << "列出借阅书籍中还书日期在10月30日之前的书籍：" << endl;

borrower.ListReturnDateBefore(Date(30, 10, 2018));

}

以上程序模拟借阅者从图书馆借书的过程，并根据预设指令打印借阅者的相关信息。程序产生的输出示例如下：

借阅者：Lightyears (1706300001)

借阅证号：GZDX-1706300001

列出借阅的所有书籍：

# | 标题 | 借书日期 | 应还日期

01 | 钢铁是怎样炼成的 | 2018-06-18 | 2018-11-11

02 | 青年近卫军 | 2018-06-05 | 2018-07-02

03 | 大卫科波菲尔 | 2018-11-15 | 2018-06-06

04 | 活着 | 2018-02-28 | 2018-02-16

05 | 人性的弱点 | 2018-08-08 | 2018-04-07

06 | 静静的顿河 | 2018-04-25 | 2018-03-30

07 | 黄金时代 | 2018-05-14 | 2018-10-24

08 | 再见，冥王星 | 2018-03-25 | 2018-12-27

09 | 你好，乔安 | 2018-12-07 | 2018-10-11

10 | CLR via C# | 2018-12-12 | 2018-08-06

列出借阅书籍中书名包含“的”字的书籍：

# | 标题 | 借书日期 | 应还日期

01 | 钢铁是怎样炼成的 | 2018-06-18 | 2018-11-11

02 | 人性的弱点 | 2018-08-08 | 2018-04-07

03 | 静静的顿河 | 2018-04-25 | 2018-03-30

列出借阅书籍中还书日期在10月30日之前的书籍：

# | 标题 | 借书日期 | 应还日期

01 | 青年近卫军 | 2018-06-05 | 2018-07-02

02 | 大卫科波菲尔 | 2018-11-15 | 2018-06-06

03 | 活着 | 2018-02-28 | 2018-02-16

04 | 人性的弱点 | 2018-08-08 | 2018-04-07

05 | 静静的顿河 | 2018-04-25 | 2018-03-30

06 | 黄金时代 | 2018-05-14 | 2018-10-24

07 | 你好，乔安 | 2018-12-07 | 2018-10-11

08 | CLR via C# | 2018-12-12 | 2018-08-06

1. 虚函数的简单使用

# 思考问题及课后练习

# 总结