



暨南大学  
JINAN UNIVERSITY

# 本科实验报告

课 程 名 称: C++程序设计实验

课 程 编 号: 08060168

学 生 姓 名: 张印祺

学 号: 2018051948

学 院: 信息科学技术学院

系: 计算机科学系

专 业: 网络工程

指 导 教 师: 张晓刚

教 师 单 位: 计算机科学系

开 课 时 间: 2019~ 2020 学 年 度 第 二 学 期

暨南大学教务处

2020 年 6 月 25 日

# 操作系统原理实验      课程实验项目目录

学生姓名：张印祺    学号：2018051948

序号	实验项目 编号	实验项目名称	*实验项目 类型	成绩	指导教师
1	实验一	熟悉 C++程序	验证		张晓刚
2	实验二	类和对象（一）	设计		张晓刚
3	实验三	类和对象（二）	设计		张晓刚
4	实验四	类的继承	设计		张晓刚
5	实验五	多态性	设计		张晓刚
6	实验六	运算符重载	设计		张晓刚
7	实验七	模板和异常处理	设计		张晓刚
8	实验八	流与文件	设计		张晓刚

\*实验项目类型：演示性、验证性、综合性、设计性实验。

\*此表由学生按顺序填写。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 熟悉 C++程序 指导教师 张晓刚  
实验项目编号 实验一 实验项目类型 验证 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 3 月 19 日 下 午 ~ 3 月 19 日 下 午

## 一、实验目的

- 1) 初步了解 Visual C++的集成开发环境;
- 2) 熟悉 C++程序的基本结构;
- 3) 掌握 C++程序的数据输入输出方法

## 二、实验内容

- 1) 编写程序计算 2 个或 3 个整数的最大数。具体要求: 用重载函数的方法来计算最大数; 函数原型为: `int max( int a, int b)` 和 `int max( int a, int b, int c)`。
- 2) 用 `setw`、`cout` 和 `for` 循环编写打印输出下面图形的程序。

```
      *
    ***
  *****
*****
*****
  *****
    ***
      *
```

- 3) 某校教师的课酬计算方法是: 教授 100 元/小时, 副教授 80/小时, 讲师 60/小时, 助教 40/小时。编写计算教师课酬的程序, 从键盘输入教师的姓名、职称、授课时数, 然后 输出该教师应得的课酬。

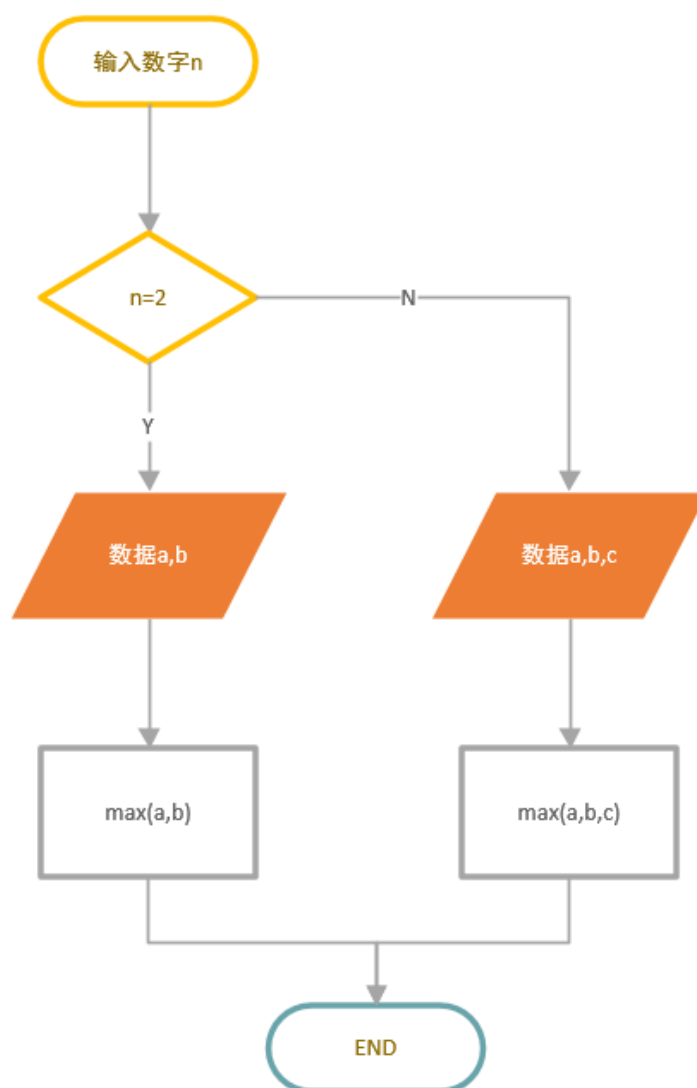
## 三、问题分析

对于问题 1), 我们可以按照方法重载的定义, 定义两个函数, 一个三变量, 一个 两变量, 然后提示输入变量的个数, 再输入数字获得 `max` 值。 对于

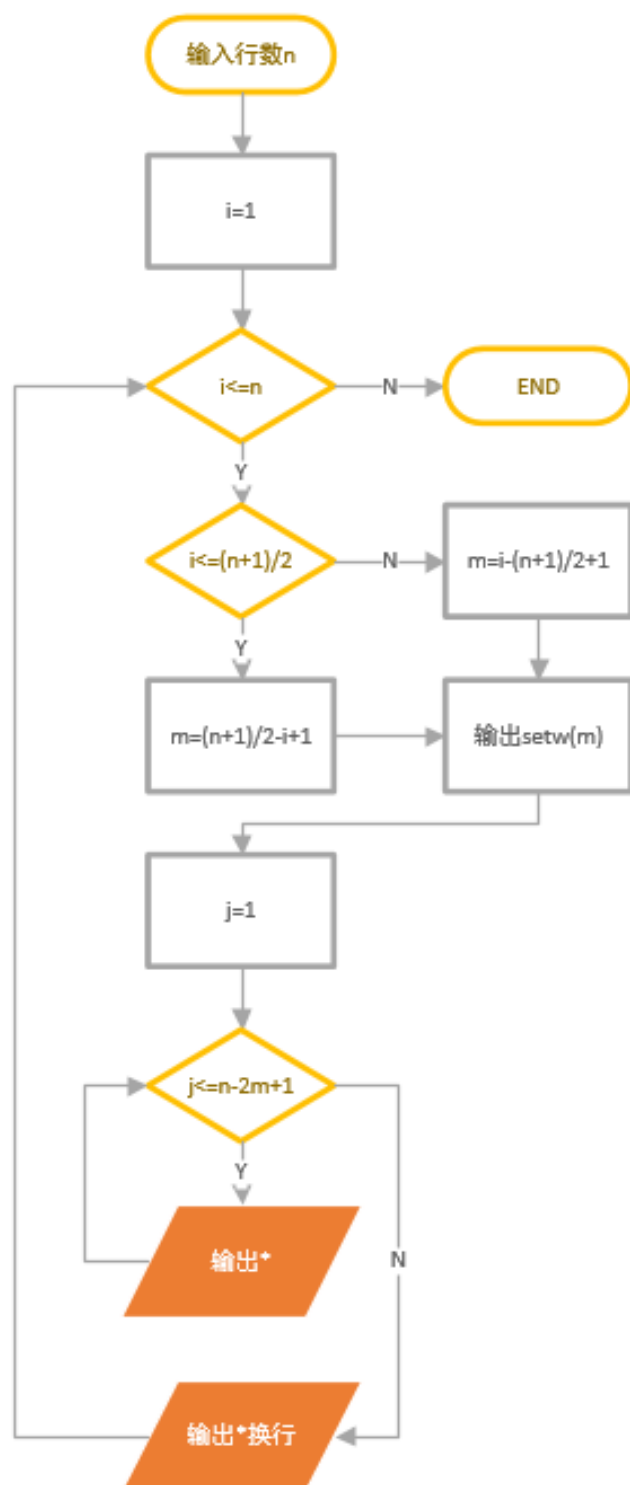
问题 2), 我们可以观察得到, 第一行有 3 个空格, 一个\*, 第二行有 2 个空格, 3 个\*, 第三行有 1 个空格.....由此我们可以得到规律, 我们设总行数为  $n$ , 第  $i$  行有  $\text{abs}((n+1)/2-i)$  对于问题 3), 我们可以定义一个教师类, 然后在类里定义一个方法, 根据职称确定每位教师的薪资, 再由授课时长乘以薪资计算教师的总工资

#### 四、 实验步骤

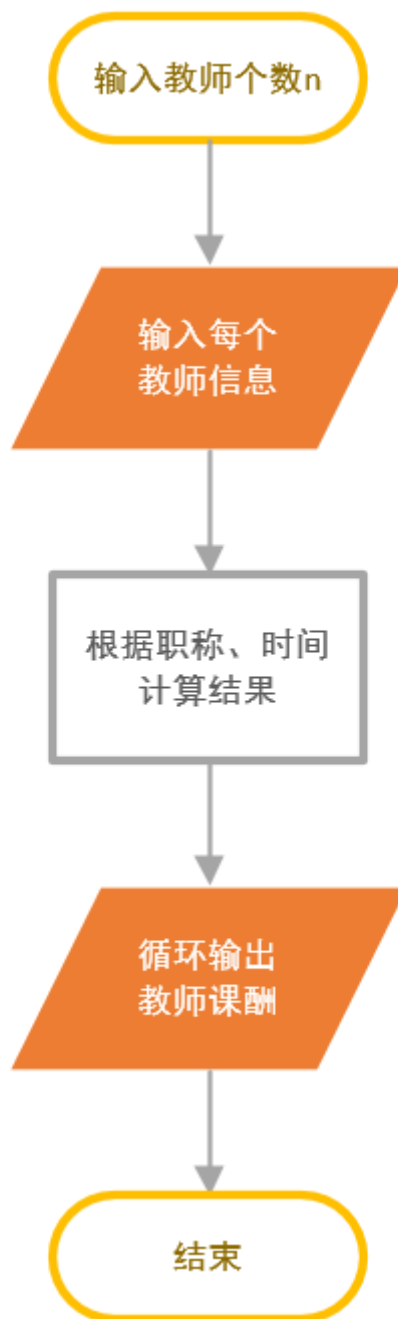
##### (1) 程序流程图



(2) 程序流程图



### (3) 程序流程图



## 五、 源程序

(1)

```
#include <iostream>
using namespace std;
int max(int a, int b)
{
    return a > b ? a : b;
}
```

```

int max(int a, int b, int c)
{
    return max(a, b) > c ? max(a, b) : c;
}
int main()
{
    int a, b, c, flag;
    bool fr = true;
    while (fr)
    {
        cout << "两数判断请输入\"2\", 三数判断请输入\"3\"" << endl;
        cin >> flag;
        switch (flag)
        {
            case 2:
                cout << "请输入两个数: " << endl;
                cin >> a >> b;
                cout << "最大数是: " << max(a, b) << endl;
                break;
            case 3:
                cout << "请输入三个数: " << endl;
                cin >> a >> b >> c;
                cout << "最大数是: " << max(a, b, c) << endl;
                break;
            default:
                break;
        }
        cout << "继续使用请输入\"1\", 退出请输入\"0\"" << endl;
        cin >> fr;
    }
    return 0;
}

```

(2)

```

#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int sp, height;
    bool flag = true;
    while (flag)
    {

```

```

        cout << "请输入你的图高度: " << endl;
        cin >> height;
        for (int i = 1; i <= height; i++)
        {
            if (i <= (height + 1) / 2)
                sp = (height + 1) / 2 - i + 1;
            else
                sp = i - (height + 1) / 2 + 1;
            cout << setw(sp);
            for (int j = 1; j <= height - 2 * sp + 1; j++)
                cout << '*';
            cout << '*' << endl;
        }
        cout << "继续使用请输入"1", 退出请输入"0" << endl;
        cin >> flag;
    }
    return 0;
}

```

(3)

```

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

int Num;

struct teacher
{
    string name;
    double rate;
    double salary;
    double hours;
    string title;
};

void setName(teacher teacher[])
{
    for (int i = 0; i < Num; i++)
    {
        cout << "请输入姓名、职称: ";
        cin >> teacher[i].name >> teacher[i].title;
        cout << "该教师输入结束" << endl;
    }
}

```



```

}

void setRate(teacher teacher[])
{
    for (int i = 0; i < Num; i++)
    {
        if (teacher[i].title == "教授")
            teacher[i].rate = 100;
        else if (teacher[i].title == "副教授")
            teacher[i].rate = 80;
        else if (teacher[i].title == "讲师")
            teacher[i].rate = 60;
        else
            teacher[i].rate = 40;
    }
}

void setHours(teacher teacher[])
{
    for (int i = 0; i < Num; i++)
    {
        cout << "请输入" << teacher[i].name << "授课时数 (h) :";
        cin >> teacher[i].hours;
        cout << endl;
    }
}

void setSalary(teacher teacher[])
{
    for (int i = 0; i < Num; i++)
    {
        teacher[i].salary = teacher[i].hours * teacher[i].rate;
    }
}

void Pr(teacher teacher[])
{
    for (int i = 0; i < Num; i++)
    {
        cout << teacher[i].name << "    应    发    工    资    :
" << teacher[i].salary << "元" << endl;
    }
}

```

```

int main()
{
    string _init;
    cout << "请输入教师数量: ";
    cin >> Num;
    cout << endl;
    teacher teacher[Num];
    cout << "首次使用请输入“init”，否则输入任意字符。" << endl;
    cin >> _init;
    if (_init == "init")
    {
        setName(teacher);
        setRate(teacher);
    }
    setHours(teacher);
    setSalary(teacher);
    Pr(teacher);
    return 0;
}

```

## 六、实验结果

(1)

```

两数判断请输入“2”，三数判断请输入“3”
2
请输入两个数:
12
34
最大数是: 34
继续使用请输入“1”，退出请输入“0”
1
两数判断请输入“2”，三数判断请输入“3”
3
请输入三个数:
12
34
56
最大数是: 56
继续使用请输入“1”，退出请输入“0”

```

(2)

```

请输入你的图高度:
7
    *
   ***
  *****
 *****
 *****
   ***
    *
继续使用请输入“1”，退出请输入“0”

```

(3)

```
请输入教师数量：3

首次使用请输入"init"，否则输入任意字符。
init
请输入姓名、职称：张三
副教授
该教师输入结束
请输入姓名、职称：李四
教授
该教师输入结束
请输入姓名、职称：王五
讲师
该教师输入结束
请输入张三授课时数(h)：12

请输入李四授课时数(h)：32

请输入王五授课时数(h)：11

张三应发工资：960元
李四应发工资：3200元
王五应发工资：660元
```

## 七、实验总结

第一题和第三题没有什么问题，但是在第二题在做的时候可以使用一个循环完成。

心得体会：经过本次实验，配置了 VS Code 的开发环境，同时配置了 CLion 的开发环境，理解了方法重载的用法，更加熟练地使用结构体，同时对于 math 库函数有了更深刻的理解。最后，对于结构化开发有了更好的理解：尽量将程序的耦合性降低，将各个功能模块尽量拆分，程序之间不要有太多的相互调用关系。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 类和对象（一） 指导教师 张晓刚  
实验项目编号 实验二 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 4 月 12 日 下 午 ~ 4 月 12 日 下 午

## 一、实验目的

- 1) 了解类与对象的概念，掌握类及类中成员函数的定义及使用方法。
- 2) 掌握对象的定义和使用方法。
- 3) 了解构造函数、析构函数和拷贝构造函数的作用、特点及使用方法

## 二、实验内容

- 1) 定义一个 `FDAccount` 类，用以描述一个定期存折(fixed deposit),实现现金支取。余额合计。信息显示等。存折基本信息包括帐号，账户名称，存款余额，存取期限（以月为 单位），存款利率（以百分点为单位）等。
- 2) 设计一个 `Student` 类，其属性包括：学号 `sno`、姓名 `name` 以及性别 `sex`，分别利用 `Student` 类构造函数和拷贝构造函数建立对象，打印每个 `Student` 类对象的信息。要求分别 编写浅拷贝构造函数和深拷贝构造函数两种不同实现版本的程序。

## 三、问题分析

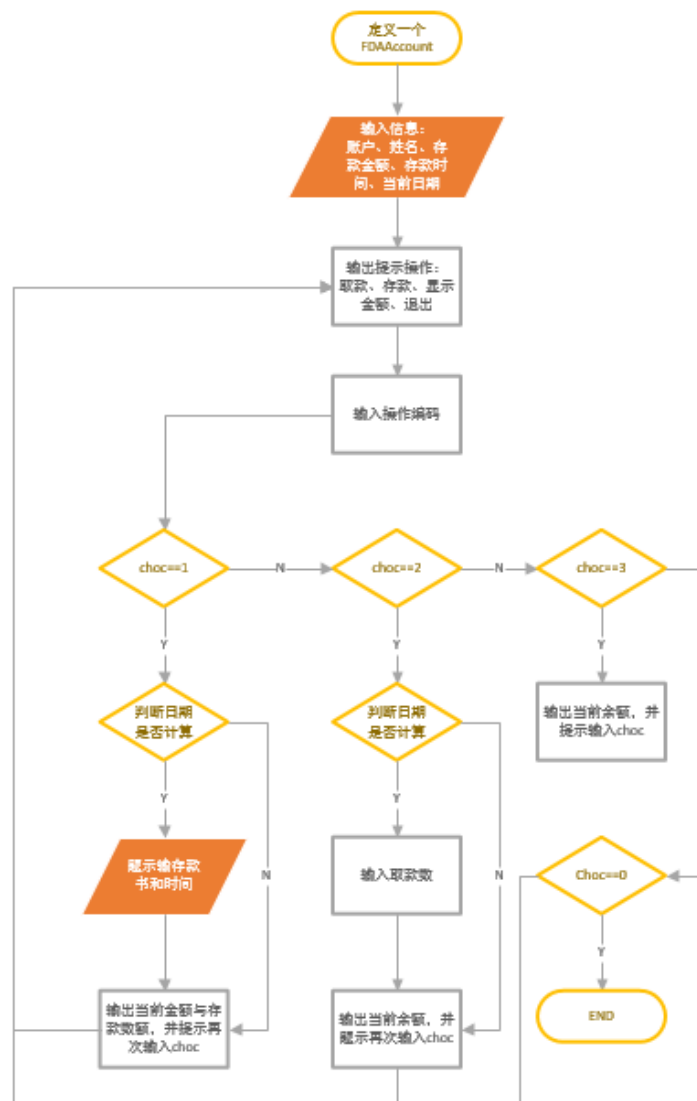
- 1) 对于第一个问题，先根据给定的内容定义 `FDAccount` 类，里面定义账号、账户名称、存款余额、存款期限和存款利率作为该类的私有成员变量。定义两个方法，实现计算当前余额 以及对于私有成员变量的显示。由于是私有成员变量，所以在类中的方法必须有 `set` 和 `get` 来获取和设置余额等变量。在课堂上与老师讨论的时候，老师认为需要做一个判断，就是当前是否达到上次的存取期限，由此做一个点，如果未达到存取期限

不允许用户取钱,后显示 当前的余额以及时间。在取款的时候,如果用户取款数额大于存款数额提示钱不够。

2) 对于第二个问题,我们需要定义两个类,第一个类是浅拷贝的,第二个类是深拷贝的, 在浅拷贝实现版本中, name 属性为固定大小的 char 数组;在深拷贝实现版本中, name 属性为指向 char 的指针类型,需要在构造函数中为 person 对象的 name 属性分配动态内存,在析构函数中释放掉申请的动态内存。注意不要让同一块动态内存被释放多次。

#### 四、实验步骤

##### (1) 程序流程图



## (2) 程序流程图



## 五、 源代码

(1)

```
#include <iostream>
#include <stdlib.h>
using namespace std;

class FDAccount
{
public:
    FDAccount(char *ID, char *depositor, double amount, int period, double rate); //amount 当前存入
    double fetch(char *ID, char *depositor, double amount);
    //取钱
```

```

        void update(); //刷新余额
        void show(); //显示账户所有信息
        int getTerm() { return term; }
        double getAmount() { return balance; }

protected:
    double interest_rate;

private:
    char *accounts;
    char *name;
    double balance; //Deposited
    int term;        //DDL
};
FDAccount::FDAccount(char *ID, char *depositor, double amount, i
nt period, double rate)
{
    name = depositor;
    accounts = ID;
    if (amount < 0 || rate < 0)
    {
        cout << "ERROR" << endl;
        exit(1);
    }
    balance = amount;
    term = period;
    interest_rate = rate;
}
double FDAccount::fetch(char *ID, char *depositor, double amount)
{
    cout << "Account:" << accounts << endl
        << "Name:" << name << endl
        << "Balance:" << amount << endl;
    balance -= amount;
    return balance;
}
void FDAccount::update()
{
    balance = balance + balance * (interest_rate / 100.0) * (ter
m / 12.0);
}
void FDAccount::show()
{
    cout << "账户信息:" << endl;

```

```

        cout << "账户:" << accounts << endl;
        cout << "名称:" << name << endl;
        cout << "期限:" << term << endl;
        cout << "利率:" << interest_rate << endl;
        cout << "账户余额:" << balance << endl;
    }
    int main()
    {
        FDAccount depositor("8888", "测试用例", 888888, 24, 3.00);
        depositor.show();
        cout << endl;
        cout << "输入上次存款时间为多少个月前? " << endl;
        int x;
        cin >> x;
        if (x < depositor.getTerm())
        {
            cout << "还未到期" << endl;
            depositor.show();
            return 0;
        }
        else
        {
            cout << "已到期可以取出" << endl;
            depositor.update();
            depositor.show();
        label1:
            int y = 0;
            cout << "取现金额:";
            cin >> y;
            if (y > depositor.getAmount())
            {
                cout << "余额不足,重新输入" << endl;
                goto label1;
            }
            else
            {
                depositor.fetch("8888", "测试用例", y);
                cout << endl;
                depositor.show();
            }
        }
        return 0;
    }
}

```



(2)

```
//Student1 浅拷贝, Student2 深拷贝
#include <iostream>
#include <string.h>
using namespace std;
class Student1
{
public:
    Student1(char *sName = "noName", int sid = 999, char sex
_ = 'F')
    {
        cout << "调用构造函数" << endl;
        strcpy(name, sName);
        sno = sid;
        sex = sex_;
    }
    Student1(Student1 &p)
    {
        cout << "调用浅拷贝" << endl;
        strcpy(name, p.name);
        sno = p.sno;
        sex = p.sex;
    }
    void printinfo()
    {
        cout << name << ' ' << sno << ' ' << sex << endl;
    }

private:
    int sno;
    char name[10];
    char sex; //F or M
};
class Student2
{
public:
    Student2(char *sName = "noName", int sid = 123, char _se
x = 'M')
    {
        cout << "调用构造函数" << endl;
        name = new char[strlen(sName) + 1];
        if (name != 0)
        {
```

```

        strcpy(name, sName);
        sno = sid;
        sex = _sex;
    }
}
Student2(Student2 &p)
{
    cout << "调用深拷贝" << endl;
    name = new char[strlen(p.name) + 1];
    if (name != 0)
    {
        strcpy(name, p.name);
        sno = p.sno;
        sex = p.sex;
    }
}
void printinfo()
{
    cout << name << ' ' << sno << ' ' << sex << endl;
}

void getAddr()
{
    cout << this->name << "Add:";
    cout << &(this->name) << endl;
}
int sno;
char *name;
char sex; //F or M
};
int main()
{
    Student1 zhang("张");
    zhang.printinfo();
    Student1 wang(zhang);
    wang.printinfo();
    Student2 zhang1("张");
    zhang1.printinfo();
    zhang1.getAddr();
    Student2 wang1(zhang1);
    wang1.printinfo();
    wang1.getAddr();
}

```

## 六、 实验结果

```
调用构造函数
张 999 F
调用浅拷贝
张 999 F
调用构造函数
张 123 M
张Add:0x61fde8
调用深拷贝
张 123 M
张Add:0x61fdc8
```

```
账户信息：
账户:8888
名称:测试用例
期限:24
利率:3
账户余额:888888

输入上次存款时间为多少个月前？
0
还未到期
账户信息：
账户:8888
名称:测试用例
期限:24
利率:3
账户余额:888888
```

## 七、 实验总结

- 1、 了解了类与对象的相关只是, 熟练掌握了类以及其中成员函数的定义和使用。
- 2、 了解了构造函数、析构函数、拷贝函数所用, 深入了解了深拷贝与浅拷贝的区别与其底层原因。
- 3、 了解如何优化程序可读性, 尽量将程序的耦合性降低, 将各个功能模块尽量拆分, 程序之间不要有太多的相互调用关系, 同时规范化代码命名与 code style 还有注释可以将自己的代码质量进一步提高。
- 4、 掌握了面向对象与面向过程的区别, 针对于面向对象程序应该更加关注代码的封装, 将代码中绝大部分的内容给隐藏起来, 只留出部分有用的接口供给用户使用。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 类和对象（二） 指导教师 张晓刚  
实验项目编号 实验三 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 4 月 12 日 下 午 ~ 4 月 12 日 下 午

## 一、实验目的

- 1) 巩固类和对象的定义及使用方法。
- 2) 了解静态成员的概念、特点及使用方法。
- 3) 了解友元的概念、特点及使用方法。
- 4) 掌握组合类的概念，了解组合类对象构造和析构的过程。

## 二、实验内容

- 1) 商店销售某一商品，商店每天公布统一的折扣(discount)，同时允许销售人员在销售时灵活掌握售价(price)，在此基础上，对一次购 10 件以上者，还可以享受 9.8 折优待。现已知当天 3 个销货员的销售情况为：

销货员号 (num)	销货件数(quantity)	销货单价 (price)
101	5	23.5
102	12	24.56
103	100	21.5

编写程序，计算出当日此商品的总销售款 sum,以及每件商品的平均售价，要求用静态数据成员和静态成员函数。

- 2) 定义单链表类，完成单链表的相关操作：

- (1) 显示输出一个已经生成的链表；
- (2) 对一个空表插入链表项，插入的新表被放在链表的头部，即前插入；
- (3) 对一个空表追加链表项，追加的新链表项被放在表尾部；

- (4) 两个链表相连接，既将一个链表接在另一个链表的尾部；
- (5) 将一个链表的数据项逆向输出；
- (6) 求一个链表的数据项数即长度。

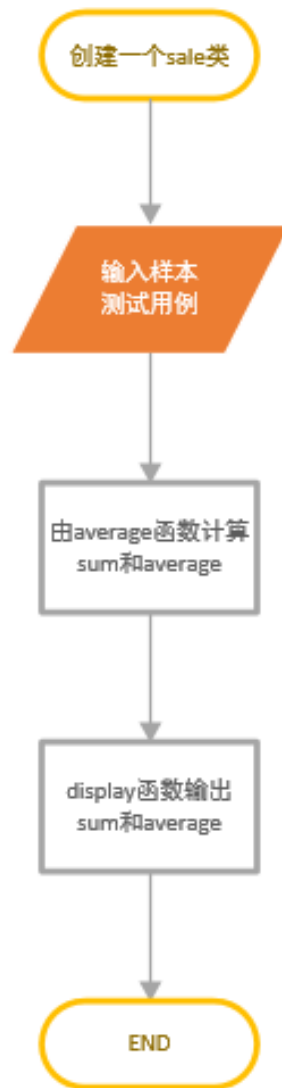
### 三、问题分析

1) 正如题目所说，设置一个类，然后将折扣 `discount`, 总销售款 `sum` 和商品销售总件数 `n` 声明为此类的静态数据成员，由于该变量是该类的共同特征，所以设为静态数据成员，再定义静态成员函数 `Average`(求平均价)和 `display`(输出结果)，求平均价格的函数是将所有销货员的销货件数乘以单价加和除以销货件数，展示函数输出平均价格以及总销售款。

2) 正如题目所说的，首先定义链表项类，包含一个 `int` 类型数据域和指针域；然后再定义单链表类，并链表类作为链表项类的友元类，来完成上述的单链表类相关操作，显示输出一个已生成的链表就是根据链表的首元素一个一个输出；前插入则是直接在链表头插入；后插入需要遍历直到链表尾部再执行插入操作；连接则是一个一个后插入；逆向输出则是用的倒置，头尾相反，其实也可以利用堆栈实现；求链表的长度则是遍历所有的链表项，一次加一。

## 四、实验步骤

### (1) 程序流程图



## (2) 程序流程图



## 五、源程序

(1)

```
#include <iostream>
using namespace std;
class sale
{
private:
    int num;
    int quantity;
    double price;
    static double sum;
    static double discount;
    static int count;

public:
    sale(int n, int q, double p) : num(n), quantity(q), price(p){};
    static double Average(sale *, int n);
    static void display(sale *, int n);
};

double sale::discount = 0.98;
double sale::sum = 0;
int sale::count = 0;
double sale::Average(sale *s, int n)
{
    sale *p;
    for (p = s; p < s + n; p++)
    {
        if (p->quantity > 10)
        {
            p->price *= discount;
        }
        sum += (p->price) * (p->quantity);
        count += p->quantity;
    }
    return sum / count;
}

void sale::display(sale *s, int n)
{
    cout << "Average:" << sale::Average(s, n) << endl;
    cout << "Sum:" << sale::count << endl;
}
```



```

int main(int argc, const char *argv[])
{
    sale s[3] = {sale(101, 5, 23.5), sale{102, 12, 24.56}, sale(1
03, 100, 21.5)};
    sale::display(s, 3);
    return 0;
}

```

(2)

```

#include <iostream>
using namespace std;
class List;
class Node
{
public:
    friend class List;
    Node(int d = 0)
    {
        data = d;
        next = 0;
    }

private:
    Node *next;
    int data;
};

class List
{
public:
    List() { list = 0; }
    List(int d) { list = new Node(d); }
    int print();
    void insert(int d = 0);
    int append(int d = 0);
    void connect(List &il);
    void reverse();
    int length();

private:
    Node *end();
    Node *list;
};

int List::print()

```

```

{
    if (list == 0)
    {
        cout << "NULL!" << endl;
        return 0;
    }
    Node *p = list;
    while (p)
    {
        cout << p->data << " ";
        p = p->next;
    }

    return 0;
}
void List::insert(int d)
{
    Node *p = new Node(d);
    p->next = list;
    list = p;
}
int List::append(int d)
{
    Node *p = new Node(d);
    if (list == 0)
        list = p;
    else
        (end())->next = p;
    return 0;
}
Node *List::end()
{
    Node *p, *ptr;
    for (p = ptr = list; ptr;)
    {
        p = ptr;
        ptr = ptr->next;
    }
    return p;
}
void List::connect(List &il)
{
    Node *p = il.list;
    while (p)

```

```

        {
            append(p->data);
            p = p->next;
        }
    }
}

void List::reverse()
{
    Node *p, *ptr, *t;
    ptr = 0;
    p = list;
    list = end();
    while (p != list)
    {
        t = p->next;
        p->next = ptr;
        ptr = p;
        p = t;
    }
    list->next = ptr;
}

int List::length()
{
    int count = 0;
    Node *p = list;
    for (; p; p = p->next)
        count++;
    return count;
}

int main(int argc, const char *argv[])
{
    List list1;
    list1.print();
    for (int i = 0; i < 20; i += 2)
        list1.insert(i);
    cout << "下面输出前插入的 list1" << endl;

    list1.print();
    List list2;
    cout << "\n";
    list2.print();
    for (int i = 0; i < 10; i += 3)
        list2.append(i);
    cout << "下面输出后插入的 L2" << endl;
}

```

```

        list2.print();
        cout << "\n";
        list1.connect(list2);
        cout << "下面输出合并的 L1" << endl;
        list1.print();
        list2.reverse();
        cout << "\n";
        cout << "下面输出反转后的 L2" << endl;
        list2.print();
        cout << "\n";
        cout << "下面输出 L1 的长度: ";
        cout << list1.length() << endl;

        return 0;
    }

```

## 六、实验结果

```

Average:21.4814
Sum:117

```

```

NULL!
下面输出前插入的list1
18 16 14 12 10 8 6 4 2 0
NULL!
下面输出后插入的L2
0 3 6 9
下面输出合并的L1
18 16 14 12 10 8 6 4 2 0 0 3 6 9
下面输出反转后的L2
9 6 3 0
下面输出L1的长度: 14

```

## 七、实验总结

经过本次实验更进一步加深了对于类和对象的定义，同时可以进一步更好的利用类和对象的定义。同时可以更好的利用类和对象。了解了静态成员的概念、特点和使用方法，在一些对象的共性方面可以使用静态成员和友元等方式，允许一个类调用另一个类的私有成员，加深了对于对象构造、组合类等概念的了解。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 类的继承 指导教师 张晓刚  
实验项目编号 实验四 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 5 月 12 日 下 午 ~ 5 月 12 日 下 午

## 一、实验目的

- 1) 理解类的继承概念、各种继承方式及其特点。
- 2) 理解并掌握派生类对象的构造和析构过程。
- 3) 理解并掌握单继承，多继承的定义方式以及多继承情况下的二义性及解决办法。
- 4) 了解虚基类的定义、原理和用法。

## 二、实验内容

- 1) 定义一个继承与派生关系的类体系，在派生类中访问基类成员。先定义一个点类，包含 x, y 坐标数据成员，显示函数和计算面积的函数成员；以点为基类派生一个圆类，增加表示半径的数据成员，重载显示和计算面积的函数；定义一个线段类，以两个点类对象作数据成员，定义显示、求面积及长度函数，线段类采用聚合方式，因为有两个端点，不能用派生。
- 2) 由汽车类派生出轿车类和卡车类，再由轿车类和卡车类多重派生出皮卡类。所谓皮卡指的是轿车的后备箱改为卡车似的后厢，可以兼运少量货物。汽车类可以说明为虚基类，以避免在皮卡类中出现两组汽车类的数据，请与未说明为虚基类的情况对比。

## 三、问题分析

- 1) 正如题目所说，设置一个点类，在点类里面设置两个成员变量

x 和 y，再设置两个方法，求面积以及相关的信息，设置其友元类为线类，再创建一个圆类继承点类，重载里面求取面积的操作以及输出信息的操作，再定义一个线类，重载其中求取面积以及输出相关信息的函数，附加一个求取长度的函数即可，输出的时候可以调用点类的信息。

2) 正如题目所说的，里面的有两种操作，有无定义为虚基类，定义为虚基类的。先定义一个汽车类为虚基类的，由其派生出轿车类以及卡车类，再由二者派生出皮卡类。对于这里的汽车类设定由品牌、价格和车轮数三个成员变量，一个输出其中信息的方法，轿车类多出一个座位数的变量，以及重载汽车类的输出方法，卡车类多出一个载重量以及重载其中的输出方法，皮卡类不需要多设置其中的变量，但需要重载输出方法；同理设立一个不为虚基类的汽车类派生两个类，轿车类以及卡车类，再由此派生出皮卡类；输出结果，后修改公共项，再输出结果；非虚基类也是如此，对比可以知道非虚基类里面存在两组数据，也就是其中的品牌和车轮数目是两个，虚基类只存在一组数据，也就是品牌和车轮数目是公共的。

#### 四、实验步骤

(1)

```
#include <iostream>
#include <cmath>
using namespace std;
#define PI 3.14159
class Point
{
    friend class Line;

protected:
    double x, y;

public:
    Point()
    {
        x = 0;
        y = 0;
    }
}
```

```

    Point(double xv, double yv)
    {
        x = xv;
        y = yv;
    }
    double Area() { return 0; }
    void Show() { cout << "x=" << x << ' ' << "y=" << y << endl;
}
};
class Circle : public Point
{
protected:
    double radius;

public:
    Circle()
    {
        x = 0;
        y = 0;
        radius = 0;
    }
    Circle(double xv, double yv, double vv) : Point(xv, yv)
    { //调用基类构造函数
        radius = vv;
    }
    Circle(Circle &cir) : Point(cir)
    { //按赋值兼容规则 cir 可为 Point 实参
        radius = cir.radius;
    }
    Circle &operator=(Circle &cir)
    {
        this->Point::operator=(cir); //在派生类中定义重载的拷贝赋值
操作符有固定的 标准格式
        radius = cir.radius;
        return *this;
    }
    double Area() { return PI * radius * radius; }
    void Show()
    {
        cout << "x=" << x << ' ' << "y=" << y << " radius=" << r
adius << endl; //访问基类的数据成员
    }
};
class Cylinder : public Circle

```

```

{
    double high;

public:
    Cylinder()
    {
        x = 0;
        y = 0;
        radius = 0;
        high = 0;
    }
    Cylinder(double xv, double yv, double vv, double kv) : Circle(xv, yv, vv)
    { //调用基类构造 函数
        high = kv;
    }
    Cylinder(Cylinder &cyl) : Circle(cyl)
    { //按赋值兼容规则 cyl 可为 Cylinder 实参
        high = cyl.high;
    }
    Cylinder &operator=(Cylinder &cyl)
    {
        this->Circle::operator=(cyl); //在派生类中定义重载的拷贝赋值操作符有固定 的标准格式
        high = cyl.high;
        return *this;
    }
    double ceArea() { return 2 * PI * radius * high; }
    double quArea() { return ceArea() + 2 * Area(); }
    double volume() { return Area() * high; }
    void Show()
    {
        cout << "x=" << x << ' ' << "y=" << y << ' ' << "radius=" << radius << ' ' << "high=" << high << endl; //访问基类的数据成员
    }
};

class Line
{
    Point start, end; //对象成员
public:
    Line() {} //对象成员初始化
    Line(double xv1, double yv1, double xv2, double yv2) : start(xv1, yv1), end(xv2, yv2) {}
};

```



```

        double GetLength() { return sqrt((start.x - end.x) * (start.
x - end.x) + (start.y - end.y) * (start.y - end.y)); }
        double Area() { return 0; }
        void Show()
        {
            cout << "start point:\n";
            start.Show();
            cout << "end point:\n";
            end.Show();
        }
};
int main()
{
    Point pt(0, 0);
    Circle cl1(100, 100, 10), cl2(cl1), cl3;
    Cylinder h1(50, 50, 20, 30), h2(h1), h3;
    Line ln1(0, 0, 100, 100), ln2;
    cout << "点面积: " << pt.Area() << endl;
    pt.Show();
    cout << "cl1 圆面积: " << cl1.Area() << endl;
    cl1.Show();
    cout << "cl2 圆面积: " << cl2.Area() << endl;
    cl2.Show();
    cl3 = cl1;
    cout << "cl3 圆面积: " << cl3.Area() << endl;
    cl3.Show();
    cout << "h1 底面积: " << h1.Area() << endl;
    cout << "h1 侧面积: " << h1.ceArea() << endl;
    cout << "h1 全面积: " << h1.quArea() << endl;
    cout << "h1 体积: " << h1.volume() << endl;
    h1.Show();
    cout << "h2 底面积: " << h2.Area() << endl;
    cout << "h2 侧面积: " << h2.ceArea() << endl;
    cout << "h2 全面积: " << h2.quArea() << endl;
    cout << "h2 体积: " << h2.volume() << endl;
    h2.Show();
    h3 = h1;
    cout << "h3 底面积: " << h3.Area() << endl;
    cout << "h3 侧面积: " << h3.ceArea() << endl;
    cout << "h3 全面积: " << h3.quArea() << endl;
    cout << "h3 体积: " << h3.volume() << endl;
    h3.Show();
    cout << " 线 面 积 : " << ln1.Area() << '\t' << " 线 长
度: " << ln1.GetLength() << endl;

```

```

        ln1.Show();
        ln2.Show();
        return 0;
    }

```

(2)

```

#include <iostream>
#include <string>
using namespace std;
class automobile
{
protected:
    string model;           //型号
    double cylinders_capability; //排量
    int wheels;             //轮数
    double price;           //价格
public:
    automobile(string mod = "#", double cl = 0, int wh = 4, double pr = 0)
    {
        model = mod;
        cylinders_capability = cl;
        wheels = wh;
        price = pr;
    }
    void display()
    {
        cout << " 汽车型号 : " << model << '\t' << " 排量 : " << cylinders_capability << "升" << endl;
        cout << " 轮数 : " << wheels << '\t' << " 价格 : " << price << endl;
    }
};

class car : public virtual automobile
{
protected:
    int seats;             //座位数
    int safeguards;        //保护措施数量
public:
    car(string mod = "#", double cl = 0, int wh = 4, double pr = 0, int st = 5, int saf = 0) : automobile(mod, cl, wh, pr)
    {
        seats = st;
        safeguards = saf;
    }
};

```

```

    }
    void display()
    {
        automobile::display();
        cout << " 座位数: " << seats << '\t' << " 保护措施数量: " << safeguards << endl;
    }
};

class truck : public virtual automobile
{
protected:
    double carrying_capacity; //载重量
public:
    truck(string mod = "#", double cl = 0, int wh = 4, double pr = 0, double cc = 0) : automobile(mod, cl, wh, pr)
    {
        carrying_capacity = cc;
    }
    void display()
    {
        automobile::display();
        cout << " 货物载重量: " << carrying_capacity << " 吨 " << endl;
    }
    void display1()
    {
        cout << " 货物载重量: " << carrying_capacity << " 吨 " << endl;
    }
};

class car_truck : public car, public truck
{
public:
    car_truck(string mod = "#", double cl = 0, int wh = 4, double pr = 0, int st = 5, int saf = 0, double cc = 0) : car(mod, cl, wh, pr, st, saf), truck(mod, cl, wh, pr, cc), automobile(mod, cl, wh, pr) {}
    void display()
    {
        car::display();
        truck::display();
    }
    void reset()
    { //从类 car 修改类 automobile 中的 mode, 实际仅有一个 automobile 基

```

类

```
        car::model = "!";
    }
};
class car1 : public automobile
{
protected:
    int seats;          //座位数
    int safeguards;     //保护措施数量
public:
    car1(string mod = "#", double cl = 0, int wh = 4, double pr
= 0, int st = 5, int saf = 0) : automobile(mod, cl, wh, pr)
    {
        seats = st;
        safeguards = saf;
    }
    void display()
    {
        automobile::display();
        cout << " 座位数: " << seats << '\t' << " 保护措施数量:
" << safeguards << endl;
    }
};
class truck1 : public automobile
{
protected:
    double carrying_capacity; //载重量
public:
    truck1(string mod = "#", double cl = 0, int wh = 4, double p
r = 0, double cc = 0) : automobile(mod, cl, wh, pr)
    {
        carrying_capacity = cc;
    }
    void display()
    {
        automobile::display();
        cout << " 货物载重量: " << carrying_capacity << " 吨
" << endl;
    }
    void display1()
    {
        cout << " 货物载重量: " << carrying_capacity << " 吨
" << endl;
    }
}
```

```

};
class car_truck1 : public car1, public truck1
{
public:
    car_truck1(string mod = "#", double cl = 0, int wh = 4, double pr = 0, int st = 5, int saf = 0, double cc = 0) : car1(mod, cl, wh, pr, st, saf), truck1(mod, cl, wh, pr, cc) {}
    void display()
    {
        car1::display();
        truck1::display();
    }
    void reset()
    {
        car1::model = "!";
    } //仅修改类car的基类automobile的model,类truck的基类automobile
    的model未改
};
int main()
{
    car_truck ct("##", 1.6, 4, 8.4, 5, 3, 0.6);
    ct.display();
    ct.reset();
    ct.display(); //显示出两个同样的model,实际是一个
    car_truck1 ct1;
    ct1.display();
    ct1.reset();
    ct1.display(); //显示出有两个不同的model
    return 0;
}

```

## 五、实验结果

```
点面积: 0
x=0 y=0
cl1 圆面积: 314.159
x=100 y=100 radius=10
cl2 圆面积: 314.159
x=100 y=100 radius=10
cl3 圆面积: 314.159
x=100 y=100 radius=10
h1 底面积: 1256.64
h1 侧面积: 3769.91
h1 全面积: 6283.18
h1 体积: 37699.1
x=50 y=50 radius=20 high=30
h2 底面积: 1256.64
h2 侧面积: 3769.91
h2 全面积: 6283.18
h2 体积: 37699.1
x=50 y=50 radius=20 high=30
h3 底面积: 1256.64
h3 侧面积: 3769.91
h3 全面积: 6283.18
h3 体积: 37699.1
x=50 y=50 radius=20 high=30
线面积: 0      线长度: 141.421
start point:
x=0 y=0
end point:
x=100 y=100
start point:
x=0 y=0
end point:
x=0 y=0
```

```
汽车型号：##    排量：1.6升
轮数：4 价格：8.4
座位数：5      保护措施数量：3
汽车型号：##    排量：1.6升
轮数：4 价格：8.4
货物载重量：0.6吨
汽车型号：!     排量：1.6升
轮数：4 价格：8.4
座位数：5      保护措施数量：3
汽车型号：!     排量：1.6升
轮数：4 价格：8.4
货物载重量：0.6吨
汽车型号：#     排量：0升
轮数：4 价格：0
座位数：5      保护措施数量：0
汽车型号：#     排量：0升
轮数：4 价格：0
货物载重量：0吨
汽车型号：!     排量：0升
轮数：4 价格：0
座位数：5      保护措施数量：0
汽车型号：#     排量：0升
轮数：4 价格：0
货物载重量：0吨
```

## 六、实验总结

经过本次实验更进一步加深了对于类和对象的定义，同时可以进一步更好的利用类和对象的定义。同时可以更好的利用类和对象。了解了静态成员的概念、特点和使用方法，在一些对象的共性方面可以使用静态成员和友元等方式，允许一个类调用另一个类的私有成员，加深了对于对象构造、组合类等概念的了解。

加深了对于虚基类的了解，虚基类的存在是为了防止多继承的时候出现交叉项出现了。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 多态性 指导教师 张晓刚  
实验项目编号 实验五 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 5 月 12 日 下 午 ~ 5 月 12 日 下 午

## 一、实验目的

- 1) 了解多态性的相关概念。
- 2) 了解并掌握虚函数的定义、原理及使用方法。
- 3) 了解纯虚函数和抽象类的概念及使用方法。

## 二、实验内容

1) 先建立一个职工类 Employee, 包含数据成员: ID(职工编号), name(职工姓名), 并且以它为基类, 派生出经理类 Manager、技术人员类 Technician 和销售类类 Saleman。在 经理类中增加数据成员 salary(基本工资)和 comm(奖金), 在技术人员类中增加数据成员 wage(每小时工资数)和 hours(月工作小时数), 在销售人员类中增加数据成员 salary(基本 工资)和 profit(销售利润)。

要求:

(1) 定义类时, 所有类中必须包含构造函数、析构函数、修改和获取所有数据成员的函数, 以及虚函数计算职工工资和输出职工信息。工资计算方法为: 经理 (基本工资+奖金), 技术人员 (月工作小时数\*每小时工资数), 销售人员 (基本工资+销售利润\*10%)。

(2) 编写主函数, 定义一个指向 Employee 的指针, 分别指向已经建立的 Manager、Technician 和 Saleman 对象, 尝试分别测试它们的计算工资函数和输出职工信息函数。

2) 下列 Shape 类是一个表示形状的抽象类, area 为求图形面积的函数, show 为显示图 形信息的函数。total 则是一个通用的求不同形状图形面积总和的函数。

```
class Shape
```



```
//抽象形状类
{
public:
    virtual double area()=0;
    virtual void show()=0;
};
double total(Shape *s[], int n){ ... ... }
```

要求:

- (1) 从 Shape 类派生出三角形类(Triangle)、矩形类(Rectangle)以及圆形类(Circle), 并给出具体的求面积函数和显示图形信息的函数。
- (2) 写出计算不同形状图形面积总和的函数 Total 的函数体。

### 三、问题分析

1) 正如题目所说, 设置一个职工类, 里面定义一些诸如 id 和名字的成员变量, 设置职工类 为纯虚函数, 即其中的方法不需要填写, 再由职工类派生出经理、技术人员和销售人员。由题目给出的方式计算每个职位对应的薪资, 由于题目要求可以修改成员变量, 所以对应的每个类都需要设置 get 和 set 函数, 而题目也需要要求有析构函数, 所以在其中也需要定义析 构函数。在主函数中设立职员类的指针后, 分别对应各个类后指向对应类以及输出。

2) 正如题目所说的, 先定义一个 shape 类为抽象类, 里面两个方法为纯虚函数, 再由 shape 类派生出三角形类、矩形类和圆形类, 由他们对应的面积公式设置一个方法计算其面积, 而 在其中显示每个类相应的信息, 比如圆心、半径以及面积。而在其中设立一个方法, 计算所 以其中对象的面积之和, 简单来说就是在主函数中设立一个 shape 类型的指针数组, 然后在 建立对象的时候, 使指针指向对象, 然后把这个指针数组传入 total 函数, 对于每个对象都调用其中求面积的函数即可。

### 四、源代码

(1)

```
#include <iostream>
using namespace std;
#include <string.h>
```

```

class Employee //职工类
{
public:
    Employee(char *, char *); //构造函数
    ~Employee() {}           //析构函数
    virtual double pay() = 0;
    virtual void print() = 0;

protected:
    char name[20]; //姓名
    char id[20];   //工号
};

Employee::Employee(char *name, char *id)
{
    strcpy(this->name, name);
    strcpy(this->id, id);
}

class Manager : public Employee
{
public:
    Manager(char *name, char *id, double salary, double more) :
Employee(name, id) { this->salary = salary; }
    ~Manager() {}
    double pay() { return salary + more; }
    void print() { cout << endl
                    << "name:" << name << " ID:" << id << "
                    pay:" << pay() << endl; }

private:
    double salary;
    double more;
};

class Saleman : public Employee
{
public:
    Saleman(char *name, char *id, double salary, double profit)
: Employee(name, id)
    {
        this->salary = salary;
        this->profit = profit;
    }
    ~Saleman() {}
    double pay() { return (salary + 0.1 * profit); }
    void print() { cout << endl

```

```

        << "name:" << name << " ID:" << id << "
    pay:" << pay() << endl; }

private:
    double salary;
    double profit;
};

class Technician : public Employee
{
public:
    Technician(char *name, char *id, int hours, double wage) : E
mployee(name, id)
    {
        this->hours = hours;
        this->wage = wage;
    }
    ~Technician() {}
    double pay() { return hours * wage; }
    void print()
    {
        cout << endl
            << "name:" << name << " ID:" << id << " pay:" <<
pay() << endl;
    }

private:
    int hours;
    double wage;
};

int main()
{
    Manager manager("张三", "101", 9000, 123);
    Technician technician("李四", "112", 8 * 29, 30);
    Saleman saleman("王五", "999", 6000, 10000);
    saleman.print();
    manager.print();
    technician.print();
    return 0;
}

```

(2)

```
#include <iostream>
#include "math.h"
using namespace std;
class Shape
{
public:
    virtual double area() = 0;
    virtual void show() = 0;
};
class Triangle : public Shape
{
protected:
    double a, b, c;

public:
    Triangle() : a(0), b(0), c(0) {}
    Triangle(double aa, double bb, double cc) : a(aa), b(bb), c(
cc) {}
    double area() override
    {
        double p = 0.5 * (a + b + c);
        return sqrt(p * (p - a) * (p - b) * (p - c));
    }
    void show() override
    {
        cout << "Tri a,b,c" :
" << a << " " << b << " " << c << endl;
        cout << "Tri's area" << area() << endl;
    }
};
class Rectangle : public Shape
{
protected:
    double a, b;

public:
    Rectangle() : a(0), b(0) {}
    Rectangle(double aa, double bb) : a(aa), b(bb) {}
    double area() override { return a * b; }
    void show() override
    {
        cout << "Rect a,b:" << a << " " << b << endl;
    }
};
```

```

        cout << "Rect's area" << area() << endl;
    }
};
class Circle : public Shape
{
protected:
    double a, b, r;

public:
    Circle() : a(0), b(0), r(0) {}
    Circle(double aa, double bb, double rr) : a(aa), b(bb), r(rr) {}
    double area() override { return r * r * 3.14; }
    void show() override
    {
        cout << "C'S: "
              << "(" << a << ", " << b << ")" << endl;
        cout << "radius: " << r << endl;
        cout << "area:" << area() << endl;
    }
};
double total(Shape *s[], int n)
{
    int i = 0;
    double sum = 0;
    for (; i < n; i++)
    {
        sum += s[i]->area();
    }
    return sum;
}
int main()
{
    Shape *p[3];
    Triangle tr = Triangle(3, 4, 5);
    Rectangle re = Rectangle(3, 4);
    Circle ci = Circle(3, 4, 5);
    tr.show();
    re.show();
    ci.show();
    p[0] = &tr;
    p[1] = &re;
    p[2] = &ci;
    cout << "total: " << total(p, 3) << endl;
}

```

```
        return 0;  
    }
```

## 五、实验结果

```
name:王五  ID:999  pay:7000  
  
name:张三  ID:101  pay:9000  
  
name:李四  ID:112  pay:6960
```

```
Tri a,b,c: 3 4 5  
Tri's area6  
Rect a,b:3 4  
Rect's area12  
C'S: (3,4)  
radius: 5  
area:78.5  
total: 96.5
```

## 六、实验总结

加深了对于类和对象的定义以及运用的理解。

了解了什么是抽象类，抽象类是不允许实例化的，因为是抽象类，没有具体的实现方法，在继承之后必须把所有的方法实现之后才能进行实例化。

了解了纯虚函数，纯虚函数与抽象类有着千丝万缕的联系，纯虚函数可以不实现，不实现的话无法使用。但是子类继承的话就必须要把没有实现的纯虚函数给实现了。

抽象类就是把一些由共性的类，提取他们的共性集合作为他们的父类，在第二题中将面积变量以及计算面积返回图形特征数据、导入图形数据，那我们就把他作为一个父类。

更好的了解了多态性，允许你将父对象设置成为一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。多态性在 Object Pascal 和 C++

中都是通过虚函数实现的。同时其中涉及到一个关键字 `override`，在使用 `override` 之后可以在编译过程中编译器检查父类中是否有你重写的方法，确认方法是重载而非重写。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 运算符重载 指导教师 张晓刚  
实验项目编号 实验六 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 6 月 1 日 下 午 ~ 6 月 1 日 下 午

## 一、实验目的

- 1) 熟悉和掌握运算符重载的规则。
- 2) 掌握几种常用的运算符重载的方法。

## 二、实验内容

1) 设计一个日期类 Date, 包括年(year)、月(month)和日(day)私有数据成员。要求实现日期类的基本运算, 主要包括:

- (1) 重载关于一日期加上天数的加法运算符+;
- (2) 重载关于一日期减去天数的减法运算符-;
- (3) 重载关于两个日期相减的减法运算符-;
- (4) 重载输出运算符<<与输入运算符>>。

**提示:** 闰年及非闰年的天数不一样, 可以定义全局二维数组存放每月天数; 另外, 程序可以把日期转换为天数或将天数转换为日期, 它们都可以以一个固定日期作为参考。

2) 编写程序, 设计一个时钟类 Timer, 包含小时(hours)、分钟(minutes)和秒(seconds)私有数据成员, 要求实现时间倒计时的功能。

**提示:** 可以通过重载关于时钟类的前置或后置自减运算符(-- )以及输入输出运算符来实现时间倒计时。

## 三、问题分析

1) 正如题目所说, 设置一个日期类 Date, 其中设立年、月和日为其私有数据成员, 里面提示有提及, 把设立两个方法, 一个把日期转换成天数, 一个把天数转化成日期, 在这里我是一公元 0 年 1 月 1 日为基准的, 这样在公元后的时间不会出现负数, 其中实现并不困



难，只是有些繁重，就是根据年、月和日去相加，闰年的 2 月多加一天。关于运算符的重载，也并不困难，因为老师有讲过模版，+和-其实通过日期转化时间和时间转化日期可以轻松的实现。而输出和输入也按照格式来就好了。

2) 正如题目所说的，先定义一个 Timer 类，里面包含小时、分钟以及秒作为其私有数据成员，实现倒计时功能，只需要对自减运算符进行重载即可，然后判断是否到零，是的话时或者分退一，如果全零则停止。而在主函数里我使用了 sleep ( ) 函数实现计时功能。

## 四、实验步骤

(1)

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

class Date
{
    int year;
    int month;
    int day;
    int leap(int);
    int md2d(int leap, int month, int day);

public:
    Date(int y, int m, int d);
    Date();
    friend ostream &operator<<(ostream &, Date &);
    //某日期加上天数
    Date operator+(int days);
    //某日期减去天数
    Date operator-(int days);
    //两日期相差的天数
    int operator-(Date &b);

private:
    int y2d(int year)
    {
        return 365 + leap(year);
    }
};

const int day_tab[2][12] = {{31, 28, 31, 30, 31, 30, 31, 31, 30,
    31, 30, 31},
```

```

        {31, 29, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31}};

int Date::leap(int y)
{
    return (y % 4 == 0 && y % 100 != 0 || y % 400 == 0);
}

int Date::md2d(int leap, int month, int day)
{
    for (int i = month - 2; i >= 0; i--)
    {
        day += day_tab[leap][i];
    }
    return day;
}

Date::Date(int y, int m, int d)
{
    year = y;
    month = m;
    day = d;
}

Date::Date()
{
    cout << "年:";
    cin >> year;
    cout << "月:";
    cin >> month;
    cout << "日:";
    cin >> day;
}

Date Date::operator+(int days)
{
    days += md2d(leap(year), month, day);
    //0<days<=365+leap
    int y = year;
    while (days > y2d(y))
    {
        days -= y2d(y);
        y++;
    }
}

```

```

while (days <= 0)
{
    y--;
    days += y2d(y);
}
//d2md
int m = 1;
int d1;
while ((d1 = days - day_tab[leap(y)][m - 1]) > 0)
{
    days = d1;
    m++;
}
Date dt(y, m, days);
return dt;
}

```

```

Date Date::operator-(int days)
{
    return *this + (-days);
}

```

```

int Date::operator-(Date &b)
{
    int d1 = md2d(leap(year), month, day);
    int d2 = md2d(leap(b.year), b.month, b.day);
    int ds = d1 - d2;
    int y1 = year;
    int y2 = b.year;
    while (y1 > y2)
    {
        y1--;
        ds += y2d(y1);
    }
    while (y1 < y2)
    {
        ds -= y2d(y1);
        y1++;
    }
    return ds;
}

```

```

ostream &operator<<(ostream &os, Date &d)
{

```

```

        os << d.year << "年" << d.month << "月" << d.day << "日";
        return os;
    }
    void add()
    {
        Date d1;
        int day;
        cout << "加上天数";
        cin >> day;
        Date d2 = d1 + day;
        cout << d2 << endl;
    }
    void minu()
    {
        Date d1;
        int day;
        cout << "减去天数";
        cin >> day;
        Date d2 = d1 - day;
        cout << d2 << endl;
    }
    void cha()
    {
        Date d1;
        Date d2;
        cout << "相差的天数" << d1 - d2 << endl;
    }
    int main()
    {
        cout << "Date" << endl;
        //某日期加上天数、某日期减去天数、两日期相差的天数
        add();
        minu();
        cha();
    }
}

```

(2)

```

#include <iostream>
#include <unistd.h>
#include <Windows.h>
using namespace std;
class Timer
{

```

```

private:
    int hour, minute, second;

public:
    Timer(){};
    Timer(int h, int m, int s) : hour(h), minute(m), second(s) {}
    friend ostream &operator<<(ostream &os, Timer &s);
    friend ostream &operator>>(ostream &os, Timer &s);
    Timer operator--();
    int getHour() { return hour; }
    int getMinute() { return minute; }
    int getSecond() { return second; }
    void setHour(int h) { hour = h; }
    void setMinute(int m) { minute = m; }
    void setSecond(int s) { second = s; }
    void show() {}
};

Timer Timer::operator--()
{
    if (second > 0)
    {
        second--;
    }
    else if (minute > 0)
    {
        second = 59;
        minute--;
    }
    else
    {
        minute = 59;
        second = 59;
        hour--;
    }
    return *this;
}

ostream &operator<<(ostream &os, Timer &t)
{
    os << t.getHour() << ":";
    os << t.getMinute() << ":";
    os << t.getSecond() << endl;
    return os;
}

```

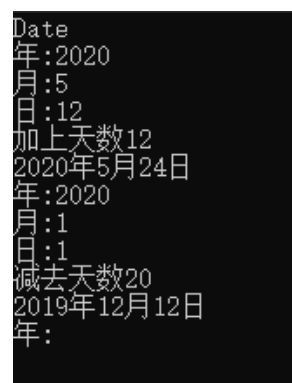
```

istream &operator>>(istream &is, Timer &t)
{
    cout << "按顺序输入时 分 秒" << endl;
    int i;
    is >> i;
    t.setHour(i);
    is >> i;
    t.setMinute(i);
    is >> i;
    t.setSecond(i);
    //cout<<endl;
    return is;
}

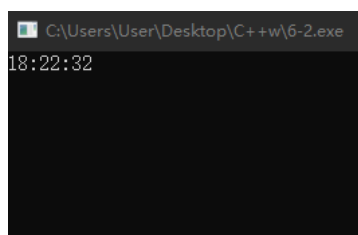
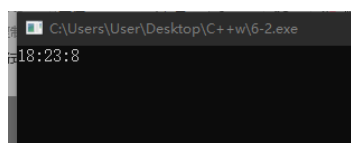
int main(int argc, const char *argv[])
{
    Timer t = Timer();
    cin >> t;
    while (1)
    {
        if (t.getHour() == 0 && t.getMinute() == 0 && t.getSecond() == 0)
        {
            t.setHour(24);
            t.setMinute(0);
            t.setSecond(0);
        }
        --t;
        system("cls");
        cout << t;
        sleep(1);
    }
}

```

## 五、实验结果



Date  
 年:2020  
 月:5  
 日:12  
 加上天数12  
 2020年5月24日  
 年:2020  
 月:1  
 日:1  
 减去天数20  
 2019年12月12日  
 年:



## 六、实验总结

加深了对于类和对象的定义以及运用的理解,对于面向对象程序设计的封装性有了一个更好的认识以及操作手段。

第一题中的运算符重载应当将日期与天数相互转换的方法实现的话,那么在运算符重载过程中会出现许多的冗余。

我们可以重定义或重载大部分 C++ 内置的运算符。这样就能使用自定义类型的运算符。

重载的运算符是带有特殊名称的函数,函数名是由关键字 `operator` 和其后要重载的运算符符号构成的。与其他函数一样,重载运算符有一个返回类型和一个参数列表。其实他的本质和函数重载十分相似,都是对于原本方法进行重载,区别在于运算符重载是重写编译器中的运算符,而函数重载是重写预定义函数或是我们自己先前写的函数。

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 模板与异常处理 指导教师 张晓刚  
实验项目编号 实验七 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 6 月 18 日 下 午 ~ 6 月 18 日 下 午

## 一、实验目的

- 1) 正确理解模板的概念。
- 2) 掌握函数模板和类模板的声明和使用方法。
- 3) 学习简单的异常处理方法。

## 二、实验内容

- 1) 编写一个使用数组类模板 `Array` 对数组进行排序、求最大值和求元素和的程序，并采用相关数据进行测试。
- 2) 设计一个只能容纳 10 个元素的队列，如果入队元素超出了队列容量，就抛出一个队列已满的异常；如果队列已空还要取出元素，就抛出一个队列已空的异常。

## 三、问题分析

1) 正如题目所说，设置一个数组类模版 `Array`，在 `Array` 类里面附有几个方法，一个是对数组进行排序，这里采用的是 `c++` 自带的 `sort` 方法，求最大值用的是打擂台的方法，求和则是直接加和，数据测试十分顺利。

2) 正如题目所说的，先定义一个 `Queue` 类，然后在此类里面定义一个长度为 10 的数组，然后有一个 `count` 记录当前元素的个数，另外有进队和出队操作，如果进队则先判断是否元素个数大于 10，大于则抛出异常“队列已满”，否则进队；如果出队的时候里面没有元素则抛出异常“队列已空”，否则出队，我还定义了一个显示队列的方法以及析构函数。



## 四、实验步骤

(1)

```
//
//  main.cpp
//  7.1
//
//  Created by Inch.Z on 2020/6/10.
//  Copyright ? 2020 Inch.Z. All rights reserved.
//

#include <iostream>
using namespace std;
#include <string.h>
#include <algorithm>
template <class ElemType>

class Array
{
private:
    int length;
    ElemType *que;

public:
    Array()
    {
        cout << "请输入数组元素个数: " << endl;
        cin >> length;
        que = new ElemType[length];
        cout << "请输入元素: " << endl;
        for (int i = 0; i < length; i++)
            cin >> que[i];
    }
    void sortArray()
    {
        sort(que, que + length);
        cout << "排序已完成! 排序后的序列为:" << endl;
        for (int i = 0; i < length; i++)
            cout << que[i] << " ";
        cout << "" << endl;
    }
    ElemType max()
    {
```

```

        ElemType m = que[0];
        for (int i = 0; i < length - 1; i++)
            if (que[i] < que[i + 1])
                m = que[i + 1];
        return m;
    }
    ElemType sum()
    {
        ElemType m = que[0];
        for (int i = 1; i < length; i++)
            m += que[i];
        return m;
    }
};

int main(int argc, const char *argv[])
{
    Array<int> array;
    int choc = 0;
    cout << "数组操作码: 1、排序 2、找最大值 3、求和 0、退出" << endl;
    cin >> choc;
    while (choc)
    {
        switch (choc)
        {
            case 1:
                array.sortArray();
                cin >> choc;
                continue;
            case 2:
                cout << "MAX: " << array.max() << endl;
                cin >> choc;
                continue;
            case 3:
                cout << "SUM: " << array.sum() << endl;
                cin >> choc;
                continue;
            default:
                cout << "ERROR! PLEASE RETRY!" << endl;
                cout << "数组操作码: 1、排序 2、找最大值 3、求和 0、退出"
" << endl;
                cin >> choc;
                continue;
        }
    }
}

```

```
}
```

(2)

```
//  
//  main.cpp  
//  7.2  
//  
//  Created by Inch.Z on 2020/6/10.  
//  Copyright © 2020 Inch.Z. All rights reserved.  
#include <iostream>  
using namespace std;  
class Queue  
{  
private:  
    int *data;  
    int head, tail, count;  
  
public:  
    Queue()  
    {  
        data = new int[10];  
        tail = -1;  
        count = 0;  
        head = 0;  
    }  
    ~Queue()  
    {  
        delete[] data;  
    }  
    void pop()  
    {  
        if (count < 0)  
            return;  
        head = (head + 1) % 10;  
        --count;  
    }  
    void push(int e)  
    {  
        if (count < 10)  
        {  
            tail = (tail + 1) % 10;  
            data[tail] = e;  
            ++count;  
        }  
    }  
}
```

```

        else
        {
            throw "队列已满! ";
        }
    }
int top()
{
    if (count > 0)
    {
        return data[head];
    }
    else
    {
        throw "队列已空! ";
    }
}
void output()
{
    cout << "队列的元素为: " << endl;
    for (int i = head; i != tail + 1; i = (i + 1) % 10)
    {
        cout << data[i] << " ";
    }
    cout << endl;
}
};
int main(int argc, const char *argv[])
{
    Queue q;
    int choc = 0;
    cout << "请选择功能: 1、入队 2、出队 0、退出" << endl;
    cin >> choc;
    while (choc)
    {
        if (choc == 1)
        {
            cout << "请输入要入队的值: " << endl;
            int d;
            cin >> d;
            try
            {
                q.push(d);
            }

```

```

        catch (const char *msg)
        {
            cerr << msg << endl;
        }
    }
    else if (choc == 2)
    {
        try
        {
            cout << "队首的元素为: " << q.top() << endl;
            q.pop();
        }
        catch (const char *msg)
        {
            cerr << msg << endl;
        }
    }
    else if (choc != 0)
    {
        cout << "输入错误! " << endl;
    }
    q.output();
    cout << "请选择功能: 1、入队 2、出队 0、退出" << endl;
    cin >> choc;
}
}

```

## 五、实验结果

```

请输入数组元素个数:
5
请输入元素:
3
4
5
1
2
数组操作码: 1、排序 2、找最大值 3、求和 0、退出
2
MAX: 2
3
SUM: 15
1
排序已完成! 排序后的序列为:
1 2 3 4 5

```

```
请选择功能：1、入队 2、出队 0、退出
1
请输入要入队的值：
5
队列的元素为：
5
请选择功能：1、入队 2、出队 0、退出
1
请输入要入队的值：
13
队列的元素为：
5 13
请选择功能：1、入队 2、出队 0、退出
1
请输入要入队的值：
7
队列的元素为：
5 13 7
请选择功能：1、入队 2、出队 0、退出
2
队首的元素为：5
队列的元素为：
13 7
请选择功能：1、入队 2、出队 0、退出
```

## 六、实验总结

通过本次实验加深了对于类和对象的定义以及运用的理解，掌握了一些常用函数模板和类模板的声明和使用以及异常处理的一些基本方式。

**模板：**模板是泛型编程的基础，泛型编程即以一种独立于任何特定类型的方式编写代码。模板是创建泛型类或函数的蓝图或公式。库容器，比如迭代器和算法，都是泛型编程的例子，它们都使用了模板的概念。每个容器都有一个单一的定义，比如 向量，我们可以定义许多不同类型的向量，比如 `vector<int>` 或 `vector<string>`。

**异常：**异常是程序在执行期间产生的问题。C++ 异常是指在程序运行时发生的特殊情况，比如尝试除以零的操作。

异常提供了一种转移程序控制权的方式。C++ 异常处理涉及到三个关键字：try、catch、throw。

throw: 当问题出现时，程序会抛出一个异常。这是通过使用 throw 关键字来完成的。使用 throw 语句在代码块中的任何地方抛出异常。throw 语句的操作数可以是任意的表达式，表达式的结果的类型决定了抛出的异常的类型。

catch: 在想要处理问题的地方，通过异常处理程序捕获异常。catch 关键字用于捕获异常。catch 块跟在 try 块后面可以指定想要捕捉的异常类型，这是由 catch 关键字后的括号内的异常声明决定的。

try: try 块中的代码标识将被激活的特定异常。它后面通常跟着一个或多个 catch 块。如果有一个块抛出一个异常，捕获异常的方法会使用 try 和 catch 关键字。try 块中放置可能抛出异常的代码，try 块中的代码被称为保护代码

# 暨南大学本科实验报告专用纸

课程名称 C++程序设计实验 成绩评定             
实验项目名称 流与文件 指导教师 张晓刚  
实验项目编号 实验八 实验项目类型 设计 实验地点             
学生姓名 张印祺 学号 2018051948  
学院 信息科学技术学院 系 计算机科学系 专业 网络工程  
实验时间 2020 年 6 月 18 日 下 午 ~ 6 月 18 日 下 午

## 一、实验目的

- 1) 理解 C++ 的输入输出的含义及其实现方法。
- 2) 掌握标准输入输出流的应用，包括格式输入输出。
- 3) 掌握对文件的输入输出操作。

## 二、实验内容

- 1) 编写一个程序拷贝文本文件，在拷贝文件过程中，改变所有字母的大小写(即大写字母转为小写字母，小写字母转为大写字母)。
- 2) 设计一个图书类 Book，包括书名、出版社名称、作者姓名、图书定价等数据成员，从键盘输入 10 本图书的各项数据，并将这 10 本图书的相关数据写入磁盘文件 book.dat 中，然后从 book.dat 中读取图书数据，计算所有图书的总价，显示每本图书的详细信息，每本图书的信息显示在一行上。

## 三、问题分析

- 1) 正如题目所说，使用流文件，先打开一个文件，然后输入一些大些或者小写,然后再打开一个文件做拷贝用，在拷贝的过程中，判断是否到达文件的结尾，如果否则判断是大写或者小写，然后转换，转换



成功后可以输出也可以直接在文件打开，这里选择在文件打开。

2) 正如题目所说的，先定义一个 Book 类，里面包括私有成员变量书名、出版社名称、作者 姓名以及图书定价，其中我还设置了获取价格的方法，由于需要计算图书的总价，定义了一个展示图书详细信息的方法。然后在主函数中，先试输入了一本书的信息，然后把这本书写入流文件，再进入下一本书；在展示的时候，把每本书的信息从流文件按顺序读取，然后每次都加上这本书的价格，输出每本书的信息以及总的价格。

#### 四、实验步骤

(1)

```
//
//  main.cpp
//  8.1
//
//  Created by 张印祺 on 2020/6/10.
//  Copyright © 2020 张印祺. All rights reserved.
//

#include <iostream>
#include <fstream>
#include <ctype.h>
using namespace std;

int main(int argc, const char *argv[])
{
    ofstream intt;
    intt.open(".\\input.txt");
    if (!intt)
    {
        cout << "不能打开输入文件! " << endl;
        return 0;
    }
    intt << "Abc"
        << " " << 90 << endl;
```

```

    ifstream in;
    in.open(".\\input.txt");
    ofstream out;
    out.open(".\\output.txt");
    if (!out)
    {
        cout << "不能打开输出文件! " << endl;
        return 0;
    }
    char ch;
    while (!in.eof())
    {
        ch = in.get();
        if (islower(ch))
            ch = toupper(ch);
        else
            ch = tolower(ch);
        out.put(ch);
    }
    in.close();
    in.close();
    out.close();
}

```

(2)

```

//
//  main.cpp
//  8.2
//
//  Created by 张印祺 on 2020/6/10.
//  Copyright © 2020 张印祺. All rights reserved.
//
#include <iostream>
#include <fstream>
#include <string.h>
#include <cstring>
using namespace std;
class Book
{
private:
    char name[20];
    char publisher[20];
    char editor[20];
    float price;

```

```

public:
    Book() {}
    Book(char *n, char *p, char *e, float pri)
    {
        strcpy(name, n);
        strcpy(publisher, p);
        strcpy(editor, e);
        price = pri;
    }
    float getPrice()
    {
        return price;
    }
    void display()
    {
        cout << name << "\t" << publisher << "\t" << editor << "
\t" << price << endl;
    }
};

int main()
{
    int num;
    cout << "输入书的数目: " << endl;
    cin >> num;
    fstream ioFile;
    char Name[20], Publisher[20], Editor[20];
    float pri;
    ioFile.open("../book.dat", ios::out);
    for (int i = 0; i < num; i++)
    {
        cout << "书名:";
        cin >> Name;
        cout << "出版社:";
        cin >> Publisher;
        cout << "作者:";
        cin >> Editor;
        cout << "价格:";
        cin >> pri;
        Book b(Name, Publisher, Editor, pri);
        ioFile.write((char *)&b, sizeof(b));
    }
    ioFile.close();
}

```

```

        ioFile.open("../book.dat", ios::in | ios::binary);
        cout << "书名\t"
              << "   出版社\t"
              << "   作者\t"
              << "   价格" << endl;
        Book book;
        float pric = 0;
        for (int i = 0; i < num; i++)
        {
            ioFile.read((char *)&book, sizeof(book));
            pric += book.getPrice();
            book.display();
        }
        cout << "总价格为: " << pric << endl;
        ioFile.close();
    }
}

```

## 五、实验结果

8 > input.txt	8 > output.txt
1    Abc	1    aBC
2    90	2    90

书名	出版社	作者	价格
111	222	qqq	123
222	333	www	234
333	eee	ttt	56
总价格为: 413			

## 六、实验总结

加深了对流与文件的理解，掌握了标准的输入输出流格式，以及对于文件的输入和输出。

要在 C++ 中进行文件处理，必须在 C++ 源代码文件中包含头文件 `<iostream>` 和 `<fstream>`。同时，要确保文件的正确处理的话最重要的一点是及时关闭已完成文件，也就是说有 `open` 就必须要有 `close`。

`open()` 成员函数的第一参数指定要打开的文件名称和位置，第二个参数定义文件被打开的模式。其中各种参数使用就不再做过多赘述，在使用时有需要的话可以查看文档。