实验UNIT 08　多态性

《程序设计》课程组

第8讲上机练习

实验目的：

1. 学习使用运算符重载的方法；
2. 掌握使用虚函数实现动态多态性。

第8讲上机练习

实验任务：

1. 课堂练习：多态性两个实例程序

2. 编程练习：运算符重载，虚函数

第8讲上机练习

- 实验步骤提示：

1. 为每个题目建立一个新的控制台项目文件；
2. 向其中提交程序所需的头文件、源程序文件；
3. 选择菜单“生成解决方案”编译源程序；
4. 执行程序，观察输出结果是否正确观察输出结果是否正确，如果有错误，可以执行第5步；
5. 使用debug功能：跟踪观察数组的数组元素值、指针及其指向对象的值变化是否正确？

现在开始课堂练习！

练习内容：多态性综合实例练习

请上机练习以下的两个实例！

程序实例1

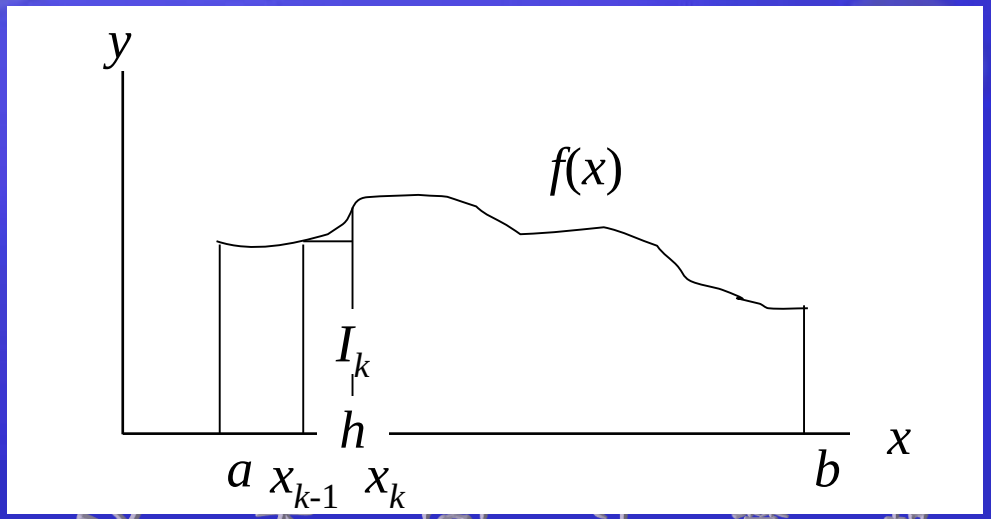
——变步长梯形积分算法求解函数的定积分

程序实例—变步长梯形积分算法求解函数的定积分——算法基本原理

- 我们只考虑最简单的情况，设被积函数是一个一元函数，定积分表达式为：

$$I = \int_a^b f(x)\mathrm{d}x$$

- 积分表示的意义是一元函数f(x)在区间a到b之间与x轴所夹的面积



算法基本原理（续）

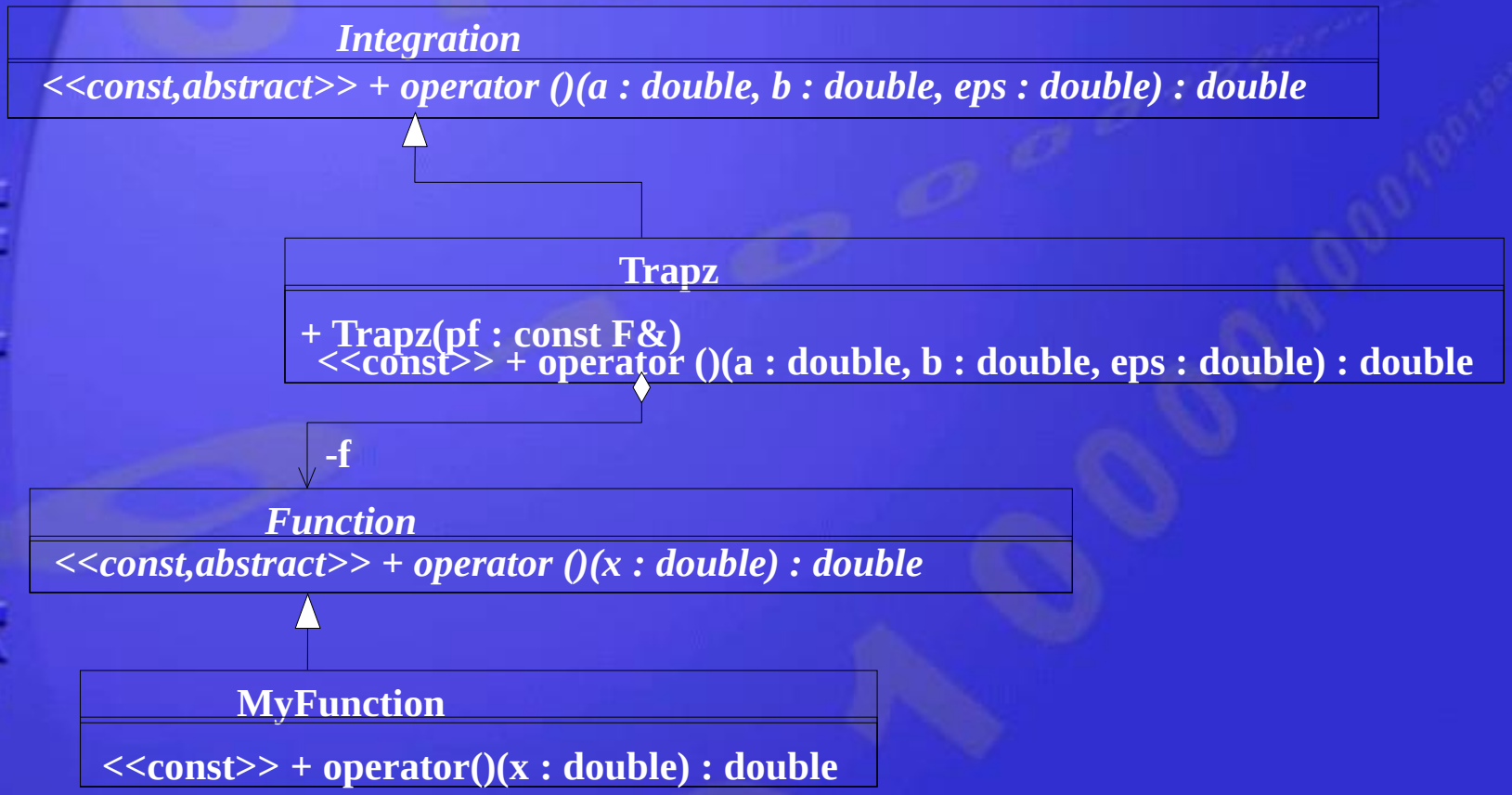
- 在每个小区间上都用小的梯形面积来近似原函数的积分，当小区间足够小时，我们就可以得到原来积分的近似值。每个小区间的面积值公式：

$$T_n = \sum_{k=0}^{n-1} \frac{h}{2}[f(x_k) + f(x_{k+1})]$$

- 实际计算中步长h逐次减半，反复利用上述求积公式进行计算，直到所求得的积分结果满足要求的精度为止。并得到递推公式：

$$T_{2n} = \frac{1}{2}T_n + \frac{h}{2}\sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}})$$

程序设计分析


Integration
<<const,abstract>> + operator ()(a : double, b : double, eps : double) : double
Trapz
+ Trapz(pf : const F&)
<<const>> + operator ()(a : double, b : double, eps : double) : double
-f
Function
<<const,abstract>> + operator ()(x : double) : double
MyFunction
<<const>> + operator()(x : double) : double

源程序及说明

——例8-7 变步长梯形积分法求解函数的定积分

- 我们求一个测试函数在某给定区间的积分值，对整个程序进行了测试，误差为eps为10-7。
- 测试函数：

$$I = \int_1^2 \frac{\log(1+x)}{1+x^2} \mathrm{d}x$$

- 整个程序分为三个独立的文档：
  Trapzint.h文件包括类的定义
  Trapzint.cpp文件包括类的成员函数实现。
  文件intmain.cpp是程序的主函数，主函数中定义了函数类Fun和梯形积分类Trapz的对象

```
//Trapzint.h  文件一，类定义
class Function {              //抽象类Function的定义
public:
    virtual double operator () (double x) const = 0; //纯虚函数重载运算符()
    virtual ~Function() { }
};

class MyFunction: public Function {        //公有派生类MyFunction定义
public:
    virtual double operator()(double x) const;        //覆盖虚函数
};

class Integration {           //抽象类Integration定义
public:
    virtual double operator () (double a, double b, double eps) const = 0;
    virtual ~Integration() { }
};
```

例8-7（续）

```
class Trapz: public Integration    {          //公有派生类Trapz定义
public:
    Trapz(const Function &f) : f(f) {}        //构造函数
    virtual double operator ()(double a, double b,double eps) const;
private:
    const Function &f;    //私有成员，Function类对象的指针
};

//Trapzint.cpp  文件二，类实现
#include "Trapzint.h"    //包含类的定义头文件
#include <cmath>
double MyFunction::operator () (double x) const {//被积函数
    return log(1.0 + x) / (1.0 + x * x);
}
double Trapz::operator () (double a,double b,double eps) const {
//积分运算过程，重载为运算符()
    bool done = false;                  //是Trapz类的虚函数成员
    int n = 1;
    double h = b - a;
    double tn = h * (f(a) + f(b)) / 2;//计算n = 1时的积分值
    double t2n;
```

例8-7（续）

```
do {
        double sum = 0;
        for(int k = 0; k < n; k++) {
                double x = a + (k + 0.5) * h;
                sum += f(x);
        }
        t2n = (tn + h * sum) / 2.0;//变步长梯形法计算
        if (fabs(t2n - tn) < eps)
                done = true;       //判断积分误差
        else {     //进行下一步计算
                tn = t2n;
                n *= 2;
                h /= 2;
        }
    } while (!done);
    return t2n;
}
```

例8-7（续）

```
//8_7.cpp  文件三，主函数
#include "Trapzint.h"     //类定义头文件
#include <iostream>
#include <iomanip>
using namespace std;

int main() {        //主函数
    MyFunction f;           //定义MyFunction类的对象
    Trapz trapz(f);         //定义Trapz类的对象
    //计算并输出积分结果
    cout << "TRAPZ Int: " << setprecision(7) << trapz(0, 2, 1e-7) << endl;
    return 0;
}
```

运行结果：

```
TRAPZ Int: 0.5548952
```

例8-7（续）

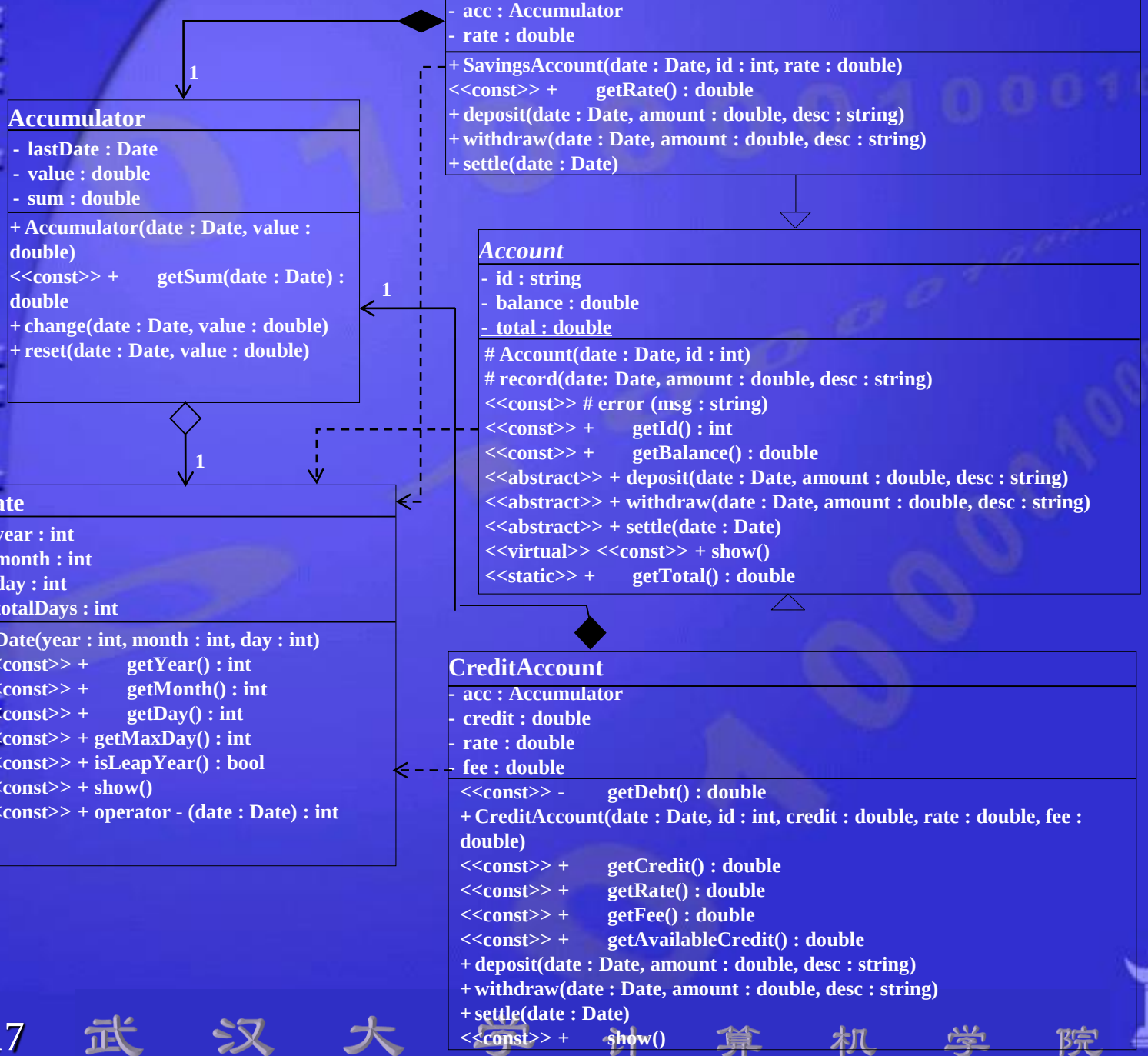
综合实例

——对个人银行账户管理程序的改进

综合实例

——对个人银行账户管理程序的改进

- 本例在第七章例7-10的基础上，对Account类做了如下改进：
  - ✓ 将show函数声明为虚函数，因此通过指向CreditAccount类实例的Accout类型的指针来调用show函数时，被实际调用的将是为CreditAccount类定义的show函数，这样，如果创建一个Account指针类型的数组，使各个元素分别指向各个账户对象，就可以通过一个循环来调用它们的show函数；
  - ✓ 在Account类中添加deposit、withdraw、settle这3个函数的声明，且将它们都声明为纯虚函数，这使得通过基类的指针可以调用派生类的相应函数，而且无需给出它们在基类中的实现。经过这一改动之后，Account类就变成了抽象类。


SavingsAccount
- acc : Accumulator
- rate : double
+ SavingsAccount(date : Date, id : int, rate : double)
<<const>> + getRate() : double
+ deposit(date : Date, amount : double, desc : string)
+ withdraw(date : Date, amount : double, desc : string)
+ settle(date : Date)
Accumulator
- lastDate : Date
- value : double
- sum : double
+ Accumulator(date : Date, value : double)
<<const>> + getSum(date : Date) : double
+ change(date : Date, value : double)
+ reset(date : Date, value : double)
Account
- id : string
- balance : double
- total : double
Account(date : Date, id : int)
record(date: Date, amount : double, desc : string)
<<const>> # error (msg : string)
<<const>> + getId() : int
<<const>> + getBalance() : double
<<abstract>> + deposit(date : Date, amount : double, desc : string)
<<abstract>> + withdraw(date : Date, amount : double, desc : string)
<<abstract>> + settle(date : Date)
<<virtual>> <<const>> + show()
<<static>> + getTotal() : double
Date
- year : int
- month : int
- day : int
- totalDays : int
+ Date(year : int, month : int, day : int)
<<const>> + getYear() : int
<<const>> + getMonth() : int
<<const>> + getDay() : int
<<const>> + getMaxDay() : int
<<const>> + isLeapYear() : bool
<<const>> + show()
<<const>> + operator - (date : Date) : int
CreditAccount
- acc : Accumulator
- credit : double
- rate : double
- fee : double
<<const>> - getDebt() : double
+ CreditAccount(date : Date, id : int, credit : double, rate : double, fee : double)
<<const>> + getCredit() : double
<<const>> + getRate() : double
<<const>> + getFee() : double
<<const>> + getAvailableCredit() : double
+ deposit(date : Date, amount : double, desc : string)
+ withdraw(date : Date, amount : double, desc : string)
+ settle(date : Date)
<<const>> + show()
1
1
1

例8-8（续）

```
//date.h
#ifndef __DATE_H__
#define __DATE_H__
class Date {          //日期类
private:
    int year;                 //年
    int month;                //月
    int day;                  //日
    int totalDays;            //该日期是从公元元年1月1日开始的第几天
public:
    Date(int year, int month, int day);//用年、月、日构造日期
    int getYear() const { return year; }
    int getMonth() const { return month; }
    int getDay() const { return day; }
    int getMaxDay() const;              //获得当月有多少天
    bool isLeapYear() const {                       //判断当年是否为闰年
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
    }
    void show() const;                              //输出当前日期
    int operator - (const Date& date) const {//计算两个日期之间差多少天
        return totalDays - date.totalDays;
    }
};
#endif //__DATE_H__
```

例8-8（续）

```
//accumulator.h
#ifndef __ACCUMULATOR_H__
#define __ACCUMULATOR_H__
#include "date.h"
class Accumulator {        //将某个数值按日累加
private:
    Date lastDate;         //上次变更数值的时期
    double value;          //数值的当前值
    double sum;            //数值按日累加之和
public:
    double getSum(const Date &date) const {
        return sum + value * (date - lastDate);
    }
    //该类其它成员函数的原型和实现与例7-10完全相同，不再重复给出
};
#endif //__ACCUMULATOR_H__
```

例8-8（续）

```
//account.h
#ifndef __ACCOUNT_H__
#define __ACCOUNT_H__
#include "date.h"
#include "accumulator.h"
#include <string>

class Account {                 //账户类
private:
    std::string id;             //帐号
    double balance;             //余额
    static double total;        //所有账户的总金额
protected:
    //供派生类调用的构造函数，id为账户
    Account(const Date &date, const std::string &id);
    //记录一笔帐，date为日期，amount为金额，desc为说明
    void record(const Date &date, double amount, const std::string &desc);
    //报告错误信息
    void error(const std::string &msg) const;
```

例8-8（续）

```
public:
    const std::string &getId() const { return id; }
    double getBalance() const { return balance; }
    static double getTotal() { return total; }
    //存入现金，date为日期，amount为金额，desc为款项说明
    virtual void deposit(const Date &date, double amount, const std::string
    &desc) = 0;
    //取出现金，date为日期，amount为金额，desc为款项说明
    virtual void withdraw(const Date &date, double amount, const std::string
    &desc) = 0;
    //结算（计算利息、年费等），每月结算一次，date为结算日期
    virtual void settle(const Date &date) = 0;
    //显示账户信息
    virtual void show() const;
};
//SavingsAccount和CreditAccount两个类的定义与例7-10完全相同
//不再重复给出
#endif //__ACCOUNT_H__
```

例8-8（续）

```
//account.cpp
//仅下面的函数定义与例7-10不同，其它皆相同，不再重复给出
void SavingsAccount::settle(const Date &date) {
    if (date.getMonth() == 1) {      //每年的一月计算一次利息
        double interest = acc.getSum(date) * rate
            / (date - Date(date.getYear() - 1, 1, 1));
        if (interest != 0)
            record(date, interest, "interest");
        acc.reset(date, getBalance());
    }
}

//8_8.cpp
#include "account.h"
#include <iostream>
using namespace std;
int main() {
    Date date(2008, 11, 1);          //起始日期
    //建立几个账户
    SavingsAccount sa1(date, "S3755217", 0.015);
    SavingsAccount sa2(date, "02342342", 0.015);
```

```
CreditAccount ca(date, "C5392394", 10000, 0.0005, 50);
    Account *accounts[] = { &sa1, &sa2, &ca };
    const int n = sizeof(accounts) / sizeof(Account*);//账户总数
    cout << "(d)deposit (w)withdraw (s)show (c)change day (n)next month
    (e)exit" << endl;
    char cmd;
    do {
        //显示日期和总金额
        date.show();
        cout << "\tTotal: " << Account::getTotal() << "\tcommand> ";
        int index, day;
        double amount;
        string desc;
        cin >> cmd;
        switch (cmd) {
        case 'd': //存入现金
                cin >> index >> amount;
                getline(cin, desc);
                accounts[index]->deposit(date, amount, desc);
                break;
```

例8-8（续）

例8-8（续）

```
    case 'w':    //取出现金
                 cin >> index >> amount;
                 getline(cin, desc);
                 accounts[index]->withdraw(date, amount, desc);
                 break;

case 's':        //查询各账户信息
                 for (int i = 0; i < n; i++) {
                          cout << "[" << i << "] ";
                          accounts[i]->show();
                          cout << endl;
                 }
                 break;
        case 'c': //改变日期
                 cin >> day;
                 if (day < date.getDay())
                          cout << "You cannot specify a previous day";
                 else if (day > date.getMaxDay())
                          cout << "Invalid day";
                 else
                          date = Date(date.getYear(), date.getMonth(), day);
                 break;
```

```
                case 'n': //进入下个月
                if (date.getMonth() == 12)
                        date = Date(date.getYear() + 1, 1, 1);
                else
                        date = Date(date.getYear(), date.getMonth() + 1, 1);
                for (int i = 0; i < n; i++)
                        accounts[i]->settle(date);
                break;
        }
    } while (cmd != 'e');
    return 0;
}
```

例8-8（续）

```
2008-11-1      #S3755217 created
2008-11-1      #02342342 created
2008-11-1      #C5392394 created
(d)deposit (w)withdraw (s)show (c)change day (n)next month (e)exit
2008-11-1      Total: 0       command> c 5
2008-11-5      Total: 0       command> d 0 5000 salary
2008-11-5      #S3755217      5000    5000     salary
2008-11-5      Total: 5000    command> c 15
2008-11-15     Total: 5000    command> w 2 2000 buy a cell
2008-11-15     #C5392394      -2000   -2000    buy a cell
2008-11-15     Total: 3000    command> c 25
2008-11-25     Total: 3000    command> d 1 10000 sell stock 0323
2008-11-25     #02342342      10000   10000    sell stock 0323
2008-11-25     Total: 13000   command> n
2008-12-1      #C5392394      -16     -2016    interest
2008-12-1      Total: 12984   command> d 2 2016 repay the credit
2008-12-1      #C5392394      2016    0        repay the credit
2008-12-1      Total: 15000   command> c 5
2008-12-5      Total: 15000   command> d 0 5500 salary
2008-12-5      #S3755217      5500    10500    salary
2008-12-5      Total: 20500   command> n
2009-1-1       #S3755217      17.77   10517.8 interest
2009-1-1       #02342342      15.16   10015.2 interest
2009-1-1       #C5392394      -50     -50     annual fee
2009-1-1       Total: 20482.9 command> s
[0] S3755217    Balance: 10517.8
[1] 02342342    Balance: 10015.2
[2] C5392394    Balance: -50    Available credit:9950
2009-1-1       Total: 20482.9 command> e
```

运行结果

例8-8（续）

本次课堂练习结束！

第8讲上机任务

编程练习：

8-6、请编写一个抽象类Shape，在此基础上派生出类Rectangle和Circle，二者都有计算对象面积的函数getArea()、计算对象周长的函数getPerim()。

8-7、对类Point重载“++”（自增）、“--”（自减）运算符，要求同时重载前缀和后缀的形式。

请自行完善主函数测试代码！

本 讲 结 束

有问题吗?