

实验3 类与对象的应用 1

实验目的：

- 1 掌握类和对象的概念、定义和使用方法。
- 2 掌握不同特性对象成员的访问方法，以及 this 指针的概念和使用方法。
- 3 掌握构造函数和析构函数的使用方法。

实验内容：

1. 定义个人的活期储蓄账户类 SavingsAccount, 数据成员包括：账号(id)、余额(balance)、年利率(rate)等信息，成员函数包括显示账户信息(show)、存款(deposit)、取款(withdraw)、结算利息(settle)等操作。

(1)提示

利息的计算方式：一年中每天的余额累积起来再除以一年的总天数，得到一个日均余额，再乘以年利率。

1	10	20
0	100	200

$$0*(10-1)+100*(20-10)$$

余额发生变动,先算累加

为了简便，类中所有日期均用一个整数来表示，该整数是一个以日为单位的相对日期。例如如果以开户日为 1，那么开户日后的第 3 天就用 4 来表示，这样通过将两个日期相减就可得到两个日期相差的天数。

```
class SavingsAccount
{
    int id; //帐号
```

```

double balance; //余额

double rate; //年利率

int lastDate; //上次变更余额的日期

double accumulation; //余额按日累加之和

double accumulate(int date); //获得到指定日期为止的存款金额按日累积值

public:

    SavingsAccount (int date, int id, double rate); //构造函数

    void deposit(int date, double amount); //存入现金, date 为日期, amount 为金额

    void withdraw(int date, double amount); //取出现金

    void settle(int date); //结算利息, 每年 1 月 1 日调用一次该函数

    void show(); //输出账户信息

    int getId() {return id;}

    double getBalance () { return balance;}

    double getRate() {return rate;}

};

```

(2)要求

- 完成上述成员函数的定义;
- 定义类对象, 测试程序的正确性

定义两个账户 s0 和 s1, 年利率都是 1.5%, 随后分别在第 5 天和第 45 天向账户 s0 存入 5 千元和 5500 元, 在第 25 天向账户 s1 存入 1 万元, 在第 60 天从账户 s1 取出 4 千元。账户开户后第 90 天是银行的计息日。分别输出 s0 和 s1 两个账户的信息(账号、余额)。

➤ 代码

```

#include <iostream>

using namespace std;

const double rate = 0.015;

class SavingsAccount
{
private:
    int id;                // 帐号
    double balance;        // 余额
    double rate;           // 年利率
    int lastDate;          // 上次变更余额的日期
    double accumulation;   // 余额按日累加之和
public:
    SavingsAccount(int id, int date, double rate); // 构造函数
    double accumulate(int date); // 获得到指定日期为止的存款金额按日累积值
    void deposit(int date, double amount);        // 存入现金, date 为日期, amount 为金额
    void withdraw(int date, double amount);        // 取出现金
    void settle(int date);                         // 结算利息, 每年1月1日调用一次该函数
    void show();                                   // 输出账户信息
    int getId() { return id; }
    double getBalance() { return balance; }
    double getRate() { return rate; }
};

SavingsAccount::SavingsAccount(int _id, int _lastDate, double _rate) : id(_id), balance(0),
rate(_rate), lastDate(_lastDate), accumulation(0){};

double SavingsAccount::accumulate(int date){
    accumulation += balance * (date - lastDate);
    lastDate = date;
    return accumulation;
}

void SavingsAccount::deposit(int date, double amount){
    accumulate(date);
    balance += amount;
    cout << date << "日" << "已存入" << amount << "元" << endl;
    cout << "当前余额为" << balance << "元" << endl;
}

void SavingsAccount::withdraw(int date, double amount){
    accumulate(date);
    balance -= amount;
}

```

```

    cout << date << "日" << "已取出" << amount << "元" << endl;
    cout << "当前余额为" << balance << "元" << endl;
}

void SavingsAccount::settle(int date){
    double interest = (accumulate(date)/365) * rate;
    cout << "当前利息为" << interest << "元" << endl;
}

void SavingsAccount::show(){
    cout << "账号:" << id << endl;
    cout << "当前余额为:" << balance << "元" << endl;
}

int main(){
    SavingsAccount sa0(1, 0, rate);
    SavingsAccount sa1(2, 0, rate);
    sa0.deposit(5, 5000);
    sa0.deposit(45, 5500);
    sa1.deposit(25, 10000);
    sa1.withdraw(60, 4000);
    sa0.settle(90);
    sa1.settle(90);
    sa0.show();
    sa1.show();
    return 0;
}


```

➤ 运行结果

```

(pt2) PS D:\code\Experimental_Report\CPP\面向对象基本知识\E3_c1> cd "d:\code\Experimental_Report\CPP\面向对象基本知识\E3_c1\" ; if ($?) { g++ SavingsAccount.cpp -o SavingsAccount } ; if ($?) { .\SavingsAccount }
5日已存入5000元
当前余额为5000元
45日已存入5500元
当前余额为10500元
25日已存入10000元
当前余额为10000元
60日已取出4000元
当前余额为6000元
当前利息为27.637元
当前利息为21.7808元
账号:1
当前余额为:10500元
账号:2
当前余额为:6000元

```



2. 对象作为函数参数

(1)提示

由于类是一个数据类型，也可以将类作为参数传递给函数，参数传递遵循传值（或传地址）的方式，这同所有其他的数据类型是相同的。类对象作形参有 3 种方式：

- 对象本身做参数（传值），传对象副本
- 对象引用做参数（传地址），传对象本身
- 对象指针做参数（传地址），传对象本身

注意：当函数参数是类类型时，调用函数时用实参初始化形参，要调用拷贝构造函数。通常默认的拷贝构造函数就可以实现实参到形参的复制，若类中有指针类型时，用户必须定义拷贝构造函数，实现实参到形参的复制。参考程序如下：

```
#include <iostream.h>

#include <string.h>

#include <stdlib.h>

class CStrSub

{   char *str;

public:

    CStrSub(char *s);

    CStrSub(const  CStrSub &);

    ~ CStrSub();

    void set(char *s);

    void show()

    {   cout<<str<<endl;   }

};
```

```

CStrSub:: CStrSub(char *s)
{ str=new char[strlen(s)+1];
  if(!str)
  {   cout<<"申请空间失败！ "<<endl;
      exit(-1);
  }
  strcpy(str,s);
}

```

```

CStrSub:: CStrSub(const  CStrSub &temp)
{ str=new char[strlen(temp.str)+1];
  if(!str)
  {   cout<<"申请空间失败！ "<<endl;
      exit(-1);
  }
  strcpy(str,temp.str);
}

```

```

CStrSub::~ ~ CStrSub( )
{ if(str!=NULL) delete [ ]str;  }

```

```

void CStrSub::set(char *s)
{
    delete [ ]str;

    str=new char[strlen(s)+1];

    if(!str)
    {
        cout<<"申请空间失败！ "<<endl;

        exit(-1);
    }

    strcpy(str,s);
}

```

```

CStrSub input(CStrSub temp)
{
    char s[20];

    cout<<"输入字符串： "<<endl;

    cin>>s;

    temp.set(s);

    return temp;
}

```

```

void main()
{
    CStrSub a("hello");

    a.show( );

    CStrSub b=input(a);
}

```

```
a.show( );  
  
b.show( );  
  
}
```

(2)要求

- 修改 input (CStrSub temp) 函数，对象引用、对象指针作为函数参数时，程序执行结果与对象作为函数参数有什么不同。

■ 需要修改的地方

➤ 引用做参数(只需修改一处)

修改 1:input 函数

```
// 引用做参数  
CStrSub &input(CStrSub &temp)  
{  
    char s[20];  
    cout << "输入字符串: " << endl;  
    cin >> s;  
    temp.set(s);  
    return temp;  
}
```

➤ 指针做参数(需要修改了两处)

修改 1:input 函数

```
CStrSub *input(CStrSub *temp)  
{  
    char s[20];  
    cout << "输入字符串: " << endl;  
    cin >> s;  
    temp->set(s);  
    return temp;  
}
```

修改 2:input 函数的调用

```
CStrSub *b = input(&a);  
a.show();  
b->show();
```


■ 运行结果

- 1.对象本身做参数（传值），传对象副本

```
(pt2) PS D:\code\Experimental_Report> cd "d:\code\Experimental_Report\CPP\面向对象基本知识\E3_
\c2\" ; if ($?) { g++ c2.cpp -o c2 } ; if ($?) { .\c2 }
hello
输入字符串:
谭棵
hello
谭棵
```

传入的对象副本,没有调用拷贝构造,a 的值没有变

- 2.对象引用做参数（传地址），传对象本身

```
(pt2) PS D:\code\Experimental_Report> cd "d:\code\Experimental_Report\CPP\面向对象基本知识\E3_
\c2\" ; if ($?) { g++ c2.cpp -o c2 } ; if ($?) { .\c2 }
hello
输入字符串:
谭棵
谭棵
谭棵
```

传入的对象本身的引用,调用了拷贝构造,a 的值被赋值为 b

- 3.对象指针做参数（传地址），传对象本身

```
(pt2) PS D:\code\Experimental_Report> cd "d:\code\Experimental_Report\CPP\面向对象基本知识\E3_
\c2\" ; if ($?) { g++ c2.cpp -o c2 } ; if ($?) { .\c2 }
hello
输入字符串:
谭棵
谭棵
谭棵
```

传入的对象本身的指针,调用了拷贝构造,a 的值被赋值为 b

三、实验总结

- 更加熟悉构造方法的参数列表构造的方法
- 对于实验二,明白了函数使用值传递传对象副本,使用引用传对象本身和使用指针传对象本身之间的区别,主要区别是拷贝构造的调用
- 更加熟悉了对象作为函数参数传递时的内存机制
- 理解了类对象作为函数参数的三种传递方式：传值（对象本身）、传引用、传指针以及拷贝构造函数在对象作为函数参数传递过程中的作用