计算机学院　高级语言程序设计　课程实验报告

实验题目：类与对象应用（二）		学号：202200400053
日期：2024-03-14	班级：　2202	姓名：　王宇涵
Email：1941497679@qq.com		

实验目的：

1. 会使用类内的特殊成员函数
2. 掌握类的组合应用
3. 认识并能绘制 UML 图

实验软件和硬件环境：

软件环境 ：VSCODE

硬件环境 ：Legion Y7000P

实验步骤与内容：

1. 设计一个用于人事管理的 People 类。要求包括：构造、复制构造和析构函数、内联成员函数、带默认形参值的成员函数、类的组合。（即习题 4-10）

构造、复制构造和析构函数:

```
public:
    // 构造函数
    People(int num = 0, char s = '\0', const Date& bd = Date(), const std::string& i = "") :
        number(num), sex(s), birthday(bd), id(i) {}

    // 拷贝构造函数
    People(const People& other) : number(other.number), sex(other.sex), birthday(other.birthday), id(other.id) {}

    // 析构函数
    ~People() {}

    // 获取编号
    int getNumber() const {
        return number;
    }
```

内联成员函数:

```
// 内联成员函数：显示人员信息
inline void displayInfo() const {
    std::cout << "Number: " << number << std::endl;
    std::cout << "Sex: " << sex << std::endl;
    std::cout << "Birthday: ";
    birthday.display();
    std::cout << std::endl;
    std::cout << "ID: " << id << std::endl;
}
```

带默认形参值的成员函数

```
// 设置性别
void setSex(char s = '男') {
    sex = s;
}
```

类的组合 ：

```
// 人员类
class People {
private:
    int number;
    char sex;
    Date birthday;
    std::string id;
```

测试案例:

```
int main() {
    Date dob(1990, 5, 25); // 出生日期
    People person(1, 'M', dob, "1234567890"); // 创建人员对象

    // 显示人员信息
    person.displayInfo();

    return 0;
}
```

输出:

```
Number: 1
Sex: M
Birthday: 1990-5-25
ID: 1234567890
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp2\实验二代码\output>
```

2. **（1）类的组合，给第 4 章 PPT 中的例 4-4 程序加入 Line 和 Point 类的析构函数（含输出信息），分析程序运行结果。**

```
int main() {
    Point myp1(1, 1), myp2(4, 5);   //建立Point类的对象
    Line line(myp1, myp2);  //建立Line类的对象
    Line line2(line);   //利用复制构造函数建立一个新对象
    cout << "The length of the line is: ";
    cout << line.getLen() << endl;
    cout << "The length of the line2 is: ";
    cout << line2.getLen() << endl;
    return 0;
}
```

```
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling constructor of Line
Calling the destructor of Point
Calling the destructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Line
The length of the line is: 5
The length of the line2 is: 5
Calling the destructor of Line
Calling the destructor of Point
Calling the destructor of Point
Calling the destructor of Line
Calling the destructor of Point
Calling the destructor of Point
Calling the destructor of Point
Calling the destructor of Point
```

逐行分析结果 ：

1. 首先创建了两个对象 myp1, myp2 并没有调用复制构造函数
2. 又创建了对象 Line, 此时是有参构造函数, 将 myp1 和 myp2 分别赋值给了 Line 的成员变量, 调用了复制构造函数, 而将实参转化为形参的过程中也调用了复制构造函数, 因此输出了 4 次 Point 的复制构造函数, 并输出一次 Line 的构造函数, 形参使用完释放内存, 因此调用了 Point 的析构函数
3. 将 line 赋值给了 line2, 将 line 的 point 赋给 line2, 因此先调用了 point 的复制构造函数, 再输出调用了 line2 的复制构造函数,
4. 输出此时的长度, 均为 5

5. 函数结束, 开始逆序依此调用 line2, line, myp2, myp1 的析构函数, line 的析构函数包括了两个 point 的析构函数, 因此先输出两次 line, point, point, 最后输出两次 point 函数

（2）给例 4-7 增加 private 成员，成员函数。试试改名为 c 语言扩展名.c ，看能编译通过吗？

增加 private, 成员函数

```
class Student {
private:
    int num;        // 学号
    std::string name;   // 姓名
    char sex;       // 性别
    int age;        // 年龄

public:
    // 构造函数
    Student(int n = 0, const std::string& nm = "", char s = ' ', int a = 0) :
        num(n), name(nm), sex(s), age(a) {}

    // 成员函数用于设置学生信息
    void setInfo(int n, const std::string& nm, char s, int a) {
        num = n;
        name = nm;
        sex = s;
        age = a;
    }
```

改扩展名, 无法编译成功

实验二代码 > C 4_7.c > Student

```
1   //4_7.cpp
2   #include <iostream>
3   #include <iomanip>
4   #include <string>
5   using namespace std;
6
7   struct Student {    //学生信息结构体
8   public :
9       int num;        //学号
10      string name;    //姓名，字符串对象，将在第6章详细介绍
11      char sex;       //性别
12      int age;        //年龄
13  };
14
15  int main() {
16      Student stu = { 97001, "Lin Lin", 'F', 19};
17      cout << "Num:  " << stu.num << endl;
18      cout << "Name: " << stu.name << endl;
19      cout << "Sex:  " << stu.sex << endl;
20      cout << "Age:  " << stu.age << endl;
21      return 0;
22  }
```

（3）解释例 4-8 的运行结果。

```
ExamInfo course1("English", 'B');
ExamInfo course2("Calculus", true);
ExamInfo course3("C++ Programming", 85);
```

在 main 函数中，创建了三个 ExamInfo 对象，分别表示英语课程的等级制成绩为 'B'，微积分课程的是否通过为 true，以及 C++ 编程课程的百分制成绩为 85，通过传入数据的类型不同而初始化不同的

mode 和联合体变量，然后调用了每个对象的 show 方法来显示考试信息。

3. 分析附件实验 2 素材中的程序运行结果

（1） 构造函数重载及参数初始化表，c9-3.cpp

```
int main()
 {
  Box box1;
  cout<<"The volume of box1 is "<<box1.volume()<<endl;
  Box box2(15,30,25);
  cout<<"The volume of box2 is "<<box2.volume()<<endl;
  return 0;
 }
```

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp2\实验二代
The volume of box1 is 1000
The volume of box2 is 11250
```

解释: box1 调用了默认参数构造函数，三个参数分别为 10, 10, 10;
Box2 调用了含参构造函数，三个参数分别为 15, 30, 25
相乘得到结果

（2） 使用默认参数的构造函数，c9-4.cpp。

```
int main()
 {
  Box box1;
  cout<<"The volume of box1 is "<<box1.volume()<<endl;
  Box box2(15);
  cout<<"The volume of box2 is "<<box2.volume()<<endl;
  Box box3(15,30);
  cout<<"The volume of box3 is "<<box3.volume()<<endl;
  Box box4(15,30,20);
  cout<<"The volume of box4 is "<<box4.volume()<<endl;
  return 0;
 }
```

```
The volume of box1 is 1000
The volume of box2 is 1500
The volume of box3 is 4500
The volume of box4 is 9000
```

解释 ：输入参数会依此填补从左到右的默认参数，因此输出不同.

问：与 c9-3 比，它还能加入一个默认构造函数 Box() 吗？

答：不能

4. 写出课本第 4 章 UML 图 4-9 line 端的重数 1--*的解释

答: 表示 Line 类一个对象可以对应多个 Point 类对象

5. 练习课本 4.8 中的综合实例，个人银行账户管理程序。（类的设计）

```
class SavingAccount
{
private:
    int id;                 //账户名
    double balance;         //余额
    double rate;            //年利率
    int lastDate;                               //上次算过利息的最后一天
    double accumulation;
    void record(int date, double amount);    //指定天数的总金额
    double accumulate(int date) const        //accumulation是用来算利息的
    {
        return accumulation + balance * (date - lastDate);
    }
public :
    SavingAccount(int date, int id, double rate);
    int getId()
    {
        return id;
    }
    double getBalance()
    {
        return balance;
    }
    double getRate()
    {
        return rate;
    }
    void deposit(int date, double amount);
    void withdraw(int date, double amount);
    void settle(int date);
    void show();
};
```

```cpp
int main()
{
    SavingAccount zhanghu0(1, 100000, 0.015);            //创建两个账户，年利率百分之1.5
    SavingAccount zhanghu1(1, 111111, 0.015);

    zhanghu0.deposit(5, 5000);                      //第五天存进5000
    zhanghu1.deposit(25, 10000);
    zhanghu0.deposit(45, 5500);
    zhanghu1.withdraw(60, 4000);                     //第60天取出4000
    zhanghu0.settle(90);                 //第90天结算
    zhanghu1.settle(90);
    zhanghu0.show();
    zhanghu1.show();

    return 0;
}
```

```
1         #100000         is created
1         #111111         is created
5         #100000         5000    5000
25        #111111         10000   10000
45        #100000         5500    10500
60        #111111         -4000   6000
90        #100000         27.64   10527.6
90        #111111         21.78   6021.78
#100000 Balance:10527.6
#111111 Balance:6021.78
```

6. 分析 testcpymov.cpp 文件

```cpp
int main(){
    A a1,a2;
    fun(a1);  //此处参数传递时，调用A的复制构造函数吗？
  a2=fun1(a1); //调用（移动）赋值运算符重载
}
```

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\
in fun()
in fun1()
copy constructor
```

分析: 创建 a1,a2 未调用复制构造函数
调用 fun()函数时, 传递了 a1 的引用, 因此也没有发生实参到形参的转化, 没有调用复制构造函数
调用 fun1()函数时, 传入的 a1 赋值给了临时变量 t, 因此调用了复制构造函数, 返回了 t 赋值给了 a2, 此时调用的是等号运算符重载, 因此不打印输出

结论分析与体会：

本次实验具有一定的挑战性, 大大加深了我对于类和对象的理解, 更加深刻地体会到了构造函数和析构函数的调用方法, 顺序等.
同时自己通过设计类和方法, 掌握了类的组合应用, 也对面向对象的程序设计增强了兴趣, 体会到了该思想的宝贵之处.
此外,我还学到了 UML 图的相关知识, 并学会识别和绘制图, 本次实验实是收获良多.

就实验过程中遇到的问题及解决处理方法，自拟 1－3 道问答题：

1. 不明白构造函数, 析构函数的调用顺序如何处理?

答: 通过程序的单步调试来观察, 并复习理论课知识加深理解

2. 有组合的情况下, 构造函数和析构函数的顺序分别是什么?

答 : 先调用成员变量的构造函数再调用自身的构造函数, 先调用自身的析构函数再调用成员变量的析构函数