面向对象程序设计与实践（1）

实验报告

北 京 邮 电 大 学

计 算 机 学 院

班级：2014211314

学号：2014211529

班内序号：16

姓名：袁振宇

2015 年 9 月

目 录

实验一 简单 C＋＋程序设计

一、 实验目的：

1、 熟悉C＋＋编程环境，掌握在Dev-C++开发环境下编写、编译、调试和执行C＋＋程序的方法。

2、 掌握C＋＋基本语法、数据类型和程序控制结构，能够编写简单C＋＋程序。

二、 实验内容和要求：

1、猜价格游戏

编写 C++程序完成以下功能：

（1）假定有一件商品，程序用随机数指定该商品的价格（1-1000 的整数）；

（2）提示用户猜价格，并输入：若用户猜的价格比商品价格高或低，对用户作出相应的提示；

（3）直到猜对为止，并给出提示。

源代码：

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

int main()
{
    int price;                                  //"price"是用户猜测的价格

    srand(time(0));
    int PRICE = 1 + rand()%1000;          //"PRICE"是商品实际的价格
    cout << PRICE << endl;

    do
    {
        cout << "Please input the price you guess:";
        cin >> price;
        if(price >1000)
              cout << "Please input an integer less than 1000!\n" << endl;
```

```
        else
        {
            if(price < PRICE)
                cout << "Too lower! Please try again.\n" << endl;
            if(price > PRICE)
                cout << "Too higher! Please try again.\n" << endl;
            if(price == PRICE)
                cout << "Your are right!" << endl;
        }
    }while(price != PRICE);

    system("pause");
    return 0;
}
```

感想：对于 C/C++而言，通常产生的随机数的方法是调用以下两个函数：

srand(time(0));设定随机数种子， 参数内是使用当前时间作为种子

rand(); 产生一个随机数还需包含头文件 ctime 和 cstdlib。

实验二 类与对象

一、实验目的：

1、 理解面向对象程序设计的基本思想。

2、 掌握面向对象程序设计的重要概念——类和对象。

3、 掌握用类分析问题的基本方法，并用C＋＋编程实现。

二、实验内容和要求：

1、矩形

编写 C++程序完成以下功能：

（1）定义一个 Point 类，其属性包括点的坐标，提供计算两点之间距离的方法；

（2）定义一个矩形类，其属性包括左上角和右下角两个点，提供计算面积的方法；

（3）创建一个矩形对象，提示用户输入矩形左上角和右下角的坐标；

（4）观察矩形对象以及 Point 类成员的构造函数与析构函数的调用；

（5）计算其面积，并输出。

源代码：

```
#include<iostream>
#include<stdlib.h>
#include<cmath>
using namespace std;
//Point 类
class Point
{
public:
    Point(int x1, int y1, int x2, int y2);
    ~Point();
    void distance(int x1, int y1, int x2, int y2);
private:
    int m_iX1, m_iY1, m_iX2, m_iY2;
    double m_iDistance;
```

```
};
Point::Point(int x1, int y1, int x2, int y2)
{
    m_iX1 = x1;
    m_iY1 = y1;
    m_iX2 = x2;
    m_iY2 = y2;
    cout << "构造函数:Point(int x1, int y1, int x2, int y2)" <<
endl;
}
Point::~Point()
{
    cout << "析构函数：  ~Point()" << endl;
}
void Point::distance(int x1, int y1, int x2, int y2)
{
    int xx = fabs(x1 - x2);
    int yy = fabs(y1 - y2);
    xx *= xx;
    yy *= yy;
    m_iDistance = sqrt(xx+yy);
    cout << "两点间的距离是:" << m_iDistance << endl;
}
//矩形类
class Rectangle
{
public:
    Rectangle(int x1, int y1, int x2, int y2);
    ~Rectangle();
      void getArea(int x1, int y1, int x2, int y2);
private:
    int m_iX1, m_iY1, m_iX2, m_iY2;
    long m_lArea;
};
```

```
Rectangle::Rectangle(int x1, int y1, int x2, int y2)
{
    m_iX1 = x1;
    m_iY1 = y1;
    m_iX2 = x2;
    m_iY2 = y2;
    cout << "构造函数：Rectangle(int x1, int y1, int x2, int y2)"
<< endl;
}
Rectangle::~Rectangle()
{
    cout << "析构函数：  ~Rectangle()" << endl;
}
void Rectangle::getArea(int x1, int y1, int x2, int y2)
{
    int xx = fabs(x1 - x2);
    int yy = fabs(y1 - y2);
    m_lArea = xx * yy;
      cout << "矩形的面积是:" << m_lArea << endl;
}
/*----------------------------------------------------------------------*/

int main()
{
    cout << "请输入左上角点(X1, Y1):";
    int X1, Y1;
    cin >> X1 >> Y1;
    cout << "请输入右下角点(X2，Y2):";
    int X2, Y2;
    cin >> X2 >> Y2;

    //计算两点间距离
    Point poi(X1, Y1, X2, Y2);                    //调用构造函数
    poi.distance(X1, Y1, X2, Y2);
```

```
    poi.~Point();
    //计算矩形面积
    Rectangle rec(X1, Y1, X2, Y2);              //调用构造函数
    rec.getArea(X1, Y1, X2, Y2);
    rec.~Rectangle();

    system("pause");
    return 0;
}
```

2、友元

编写 C++程序完成以下功能：

（1）定义一个 Boat 和 Car 两个类，他们都具有私用属性——重量；

（2）编写一个函数，计算两者的重量和。

double TotalWeight(Boat& b, Car& c);

源代码：

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

class Car;                                          //声明 Car 类

//Boat 类
class Boat
{
public:
    friend double TotalWeight(Boat &b, Car &c);     //声明友元函数
    Boat(double weighe1);
    ~Boat(){};
private:
    double m_dWeight1;
};
Boat::Boat(double weight1)
```

```
{
    m_dWeight1 = weight1;
}

//Car 类
class Car
{
public:
    friend double TotalWeight(Boat &b, Car &c);          //声明友元函数
    Car(double weight2);
    ~Car(){};
private:
     double m_dWeight2;
};
Car::Car(double weight2)
{
    m_dWeight2 = weight2;
}

double TotalWeight(Boat &b, Car &c)
{
    return b.m_dWeight1 + c.m_dWeight2;
}
int main()
{
    cout << "请输入两物品的重量:";
    double w1, w2;
    cin >> w1 >> w2;

    Boat B(w1);
    Car   C(w2);

    cout << "两物品的重量和是:" << TotalWeight(B, C) << endl;
```

```
    system("pause");
    return 0;
}
```

感想：

本实验让我初次体验了类和友元函数的用法，体会到类中各成员与结构体的不同，以及构造函数和析构函数的作用。

实验三 数组与指针

一、实验目的：

1、 理解面向对象程序实际的数据结构。

2、 掌握C＋＋中利用数组和指针组织数据的方法。

二、试验内容和要求

1、 矩阵（一）

编写C++程序完成以下功能：

（1）假定矩阵大小为 4×5（整型数组表示）；

（2）定义矩阵初始化函数，可以从 cin 中输入矩阵元素；

（3）定义矩阵输出函数，将矩阵格式化输出到 cout；

（4）定义矩阵相加的函数，实现两个矩阵相加的功能,结果保存在另一个矩阵中；

（5）定义矩阵相减的函数，实现两个矩阵相减的功能,结果保存在另一个矩阵中；

（6）定义三个矩阵：A1、A2、A3；

（7）初始化 A1、A2；

（8）计算并输出：A3 = A1 加 A2，A3 = A1 减 A2。

源代码：

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

void initMatrix(int a[][5]);
void getMatrix(int a[][5]);
void sumMatrix(int a1[][5], int a2[][5], int b[][5]);
void subMatrix(int a1[][5], int a2[][5], int b[][5]);

int main()
{
    int A1[4][5], A2[4][5];
```

```
    int A3[4][5];

    initMatrix( A1 );
    initMatrix( A2);
    sumMatrix(A1, A2, A3);
    subMatrix(A1, A2, A3);

    system("pause");
    return 0;
}

void initMatrix(int a[][5])
{
    cout << "Please input a 4*5 matrix:" << endl;
    for(int i = 0; i <= 3; i ++)
        for(int j = 0; j <= 4; j++)
            cin >> a[i][j];
}

void getMatrix(int a[][5])
{
    for(int i = 0; i <= 3; i ++)
    {
        for(int j = 0; j <= 4; j++)
            cout << setw(6) << a[i][j];
        cout << endl;
    }
}

void sumMatrix(int a1[][5], int a2[][5], int b[][5])
{
    for(int i = 0; i <= 3; i ++)
        for(int j = 0; j <= 4; j++)
            b[i][j] = a1[i][j] + a2[i][j];
```

```
    cout << "The sum of tow matrixes is:" << endl;
    getMatrix( b );
}

void subMatrix(int a1[][5], int a2[][5], int b[][5])
{
    for(int i = 0; i <= 3; i ++)
        for(int j = 0; j <= 4; j++)
            b[i][j] = a1[i][j] - a2[i][j];

    cout << "The subtraction of tow matrixes is:" << endl;
     getMatrix( b );
}
```

2、 矩阵（二）

编写C++程序完成以下功能：

（1）假定矩阵大小为 4×5（整型）；

（2）矩阵空间采用 new 动态申请，保存在指针中；

（3）定义矩阵初始化函数，可以从 cin 中输入矩阵元素；

（4）定义矩阵输出函数，将矩阵格式化输出到 cout；

（5）定义矩阵相加的函数，实现两个矩阵相加的功能,结果保存在另一个矩阵中；

（6）定义矩阵相减的函数，实现两个矩阵相减的功能,结果保存在另一个矩阵中；

（7）动态申请三个矩阵：A1、A2、A3；

（8）初始化 A1、A2；

（9）计算并输出 A3 = A1 加 A2，A3 = A1 减 A2；

（10） 释放矩阵空间。

源代码：

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

```
void initMatrix(int *[]);
void getMatrix(int *[]);
void sumMatrix(int *[], int *[], int *[]);
void subMatrix(int *[], int *[], int *[]);

//定义全局变量i、j作为矩阵的下标
int    i, j;

int main()
{
    /*----------------------用new动态申请三块内存  ----------------------*/

    int **Ap1 = new int*[4];
    if(Ap1 == NULL)
        return 0;
    for(i = 0; i <= 3; i ++)
    {
        Ap1[i] = new int[5];
        if(Ap1[i] == NULL)
            return 0;
    }

    int **Ap2 = new int*[4];
    if(Ap2 == NULL)
        return 0;
    for(i = 0; i <= 3; i ++)
    {
        Ap2[i] = new int[5];
        if(Ap2[i] == NULL)
            return 0;
    }

    int **Ap3 = new int*[4];
```

```
if(Ap3 == NULL)
        return 0;
for(i = 0; i <= 3; i ++)
{
     Ap3[i] = new int[5];
     if(Ap3[i] == NULL)
          return 0;
}

/*----------------------------------------------------------------*/

initMatrix( Ap1 );
initMatrix( Ap2 );
sumMatrix(Ap1, Ap2, Ap3);
subMatrix(Ap1, Ap2, Ap3);

/*----------------------delete三块动态内存  ------------------------*/

for(i = 0; i <= 3; i ++)
        delete []Ap1[i];
delete []Ap1;
Ap1 = NULL;

for(i = 0; i <= 3; i ++)
        delete []Ap2[i];
delete []Ap2;
Ap2 = NULL;

for(i = 0; i <= 3; i ++)
        delete []Ap3[i];
delete []Ap3;
Ap3 = NULL;

system("pause");
```

```
    return 0;
}

void initMatrix(int *a[])
{
    cout << "Please input a 4*5 matrix:" << endl;
    for(i = 0; i <= 3; i ++)
        for(j = 0; j <= 4; j++)
            cin >> a[i][j];
}

void getMatrix(int *a[])
{
    for(i = 0; i <= 3; i ++)
    {
        for(j = 0; j <= 4; j++)
            cout << setw(6) << a[i][j];
        cout << endl;
    }
}

void sumMatrix(int *a1[], int *a2[], int *b[])
{
    for(i = 0; i <= 3; i ++)
        for(j = 0; j <= 4; j ++)
            b[i][j] = a1[i][j] + a2[i][j];

    cout << "The sum of tow matrixes is:" << endl;
    getMatrix( b );
}

void subMatrix(int *a1[], int *a2[], int *b[])
{
    for(i = 0; i <= 3; i ++)
```

```
        for(j = 0; j <= 4; j++)
            b[i][j] = a1[i][j] - a2[i][j];

    cout << "The subtraction of tow matrixes is:" << endl;
     getMatrix( b );
}
```

3、矩阵（三）

编写C++程序完成以下功能：

（1）用类来实现矩阵，定义一个矩阵的类，属性包括：

- 矩阵大小，用 lines, rows（行、列来表示）；
- 存贮矩阵的数组指针，根据矩阵大小动态申请（new）。

（2）矩阵类的方法包括：

- 构造函数，参数是矩阵大小，需要动态申请存贮矩阵的数组；
- 析构函数，需要释放矩阵的数组指针；
- 拷贝构造函数，需要申请和复制数组；
- 输入，可以从 cin 中输入矩阵元素；
- 输出，将矩阵格式化输出到 cout；
- 矩阵相加的函数，实现两个矩阵相加的功能,结果保存在另一个矩阵类，但必须矩阵大小相同；
- 矩阵相减的函数，实现两个矩阵相减的功能,结果保存在另一个矩阵类，但必须矩阵大小相同。

（3）定义三个矩阵：A1、A2、A3；

（4）初始化 A1、A2；

（5）计算并输出 A3 = A1 加 A2，A3=A1 减 A2；

（6）用 new 动态创建三个矩阵类的对象：pA1、pA1、pA3；

（7）初始化 pA1、pA2；

（8）计算并输出 pA3=pA1 加 pA2，pA3=pA1 减 pA2；

（9）释放 pA1、pA1、pA3。

源代码：

```
#include<iostream>
```

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

int i, j;

//构造矩阵类
class Matrix
{
public:
      int **m_iPtr;
      Matrix();
    Matrix(int line, int row);
    Matrix(Matrix & A);
    ~Matrix();
    void initMatrix();
    void getMatrix(Matrix & A);
    void sumMatrix(Matrix & A1, Matrix & A2, Matrix & B);
    void subMatrix(Matrix & A1, Matrix & A2, Matrix & B);
private:
    int m_iLine, m_iRow;
};

Matrix::Matrix()              //默认构造函数，不带参数，动态申请矩阵
{
    cout << "Please input lines and rows:";
    cin >> m_iLine >> m_iRow;
    m_iPtr = new int *[m_iLine];
    if(NULL != m_iPtr)
    {
        for(i = 0; i <= m_iLine - 1; i++)
                m_iPtr[i] = new int[m_iRow];
    }
}
```

```
Matrix::Matrix(int line, int row)     //构造函数，参数是矩阵的行和列
{
    m_iLine = line;
    m_iRow    = row;
    m_iPtr = new int *[m_iLine];
    if(NULL != m_iPtr)
    {
        for(i = 0; i <= m_iLine - 1; i++)
                m_iPtr[i] = new int[m_iRow];
    }
}

Matrix::Matrix(Matrix &A)             //拷贝构造函数，申请和复制矩阵
{
    m_iPtr = new int *[A.m_iLine];
    for(i = 0; i <= A.m_iLine-1; i++)
        m_iPtr[i] = new int[A.m_iRow];
    m_iLine = A.m_iLine;
    m_iRow    = A.m_iRow;
}

Matrix::~Matrix()                          //析构函数 ，释放内存
{
    for(i = 0; i <= m_iLine-1; i++)
            delete []m_iPtr[i];
    delete []m_iPtr;
    //m_iPtr = NULL;
}

void Matrix::initMatrix()              //初始化数组，读入矩阵元素
{
    cout << "Please input a " << m_iLine << "*" << m_iRow << "
matrix:" << endl;
```

```
    for(i = 0; i <= m_iLine-1; i ++)
        for(j = 0; j <= m_iRow-1; j++)
            cin >> m_iPtr[i][j];
}

void Matrix::getMatrix(Matrix &A)        //输出矩阵元素
{
    for(i = 0; i <= m_iLine-1; i ++)
    {
        for(j = 0; j <= m_iRow-1; j++)
            cout << setw(6) << m_iPtr[i][j];
        cout << endl;
    }
}

void Matrix::sumMatrix(Matrix &A1, Matrix &A2, Matrix &B)
    //计算两矩阵之和
{
    for(i = 0; i <= m_iLine-1; i ++)
        for(j = 0; j <= m_iRow-1; j++)
            B.m_iPtr[i][j] = A1.m_iPtr[i][j] + A2.m_iPtr[i][j];

    cout << "The sum of tow matrixes is:" << endl;        //输出新矩阵
    getMatrix( B );
}

void Matrix::subMatrix(Matrix &A1, Matrix &A2, Matrix &B)
//计算两矩阵之差
{
    for(i = 0; i <= m_iLine-1; i ++)
        for(j = 0; j <= m_iRow-1; j++)
            B.m_iPtr[i][j] = A1.m_iPtr[i][j] - A2.m_iPtr[i][j];

    cout << "The subtraction of tow matrixes is:" << endl;
```

```
    getMatrix( B );                                    //输出新矩阵
}

int main()
{
    int m, n;

    Matrix A1;                    //调用默认构造函数
    Matrix A2(A1);                //调用拷贝函数
    Matrix A3(A1);                //调用拷贝函数

    A1.initMatrix();                    //初始化矩阵A1
    A2.initMatrix();                    //初始化矩阵A2

    A3.sumMatrix(A1, A2, A3);          //计算矩阵A1，A2之和
    A3.subMatrix(A1, A2, A3);           //计算矩阵A1，A2之差

    system("pause");
    return 0;
}
```

感想：

拷贝构造函数极大地方便了程序员对类的使用，以及用new 和delete动态申请内存以及它们和C语言中malloc 、free的区别，使用时应加以区分。

实验四 继承与派生

一、实验目的：

1、 理解面向对象程序设计中继承与派生的概念。

2、 掌握C＋＋中采用类的继承与派生解决问题的方法。

二、实验内容和要求：

1、 形状（一）

编写C++程序完成以下功能：

（1）声明一个基类 Shape（形状），其中包含一个方法来计算面积；

（2）从 Shape 派生两个类矩形和圆形；

（3）从矩形派生正方形；

（4）分别实现派生类构造函数、析构函数和其他方法；

（5）创建派生类的对象，观察构造函数、析构函数调用次序；

（6）不同对象计算面积。

源代码：

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

#define PI 3.1415

class Shape                                          // 基 类
Shape
{
public:
    Shape()
    {
      cout << "Shape()" << endl;
    }
    ~Shape()
    {
      cout << "~Shape()" << endl;
    }
```

```
    double getArea()
    {
        return 0;
    }
};

class Rectangle: public Shape                    //  派  生  类
Rectangle
{
public:
    Rectangle(double l, double w)
    {
        m_dLen  = l;
        m_dWide = w;
        cout << "Rectangle()" << endl;
    }
    ~Rectangle()
    {
        cout << "~Rectangle()" << endl;
    }
    double getArea()
    {
        return m_dLen * m_dWide;
    }
private:
    double m_dLen, m_dWide;
};

class Circle: public Shape                       //派生类 Circle
{
public:
    Circle(double r)
    {
        m_dR = r;
```

```
        cout << "Circle()" << endl;
    }
    ~Circle()
    {
        cout << "~Circle()" << endl;
    }
    double getArea()
    {
          return PI * m_dR * m_dR;
    }
private:
    double m_dR;
};

class Square: public Rectangle
{
public:
    Square(double x):Rectangle(x, x)
    {
         x = m_dX;
        cout << "Square()" << endl;
    }
    ~Square()
    {
        cout << "~Square()" << endl;
    }
private:
     double m_dX;
};

int main()
{
    double l, w;
    cout << "请输入矩形的长和宽：";
```

```
    cin >> l >> w;
    Rectangle Rec(l, w);
     Shape *p1 = &Rec;
    cout << "矩形的面积是：" << Rec.getArea() << endl;
    cout << "矩形的面积是：" << p1 -> getArea() << endl;

    double r;
    cout << "请输入圆的半径：";
    cin >> r;
    Circle Cir(r);
    cout << "圆的面积是：" << Cir.getArea() << endl;

    double x;
    cout << "请输入正方形的边长：";
    cin >> x;
    Square Squ(x);
    cout << "正方形的面积是：" << Squ.getArea() << endl;

    system("pause");
    return 0;
}
```

2、 形状（二）——虚函数

（1）将【形状（一）】 中的基类计算面积的方法定义为虚函数，比较与【形状（一）】程序的差异；

（2）将【形状（一）】中的基类定义抽象类，比较与【形状（一）】程序的差异。

源代码 1：

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

#define PI 3.1415
```

```
class Shape                                          //基类 Shape
{
public:
    Shape()
    {
        cout << "Shape()" << endl;
    }
    ~Shape()
    {
        cout << "~Shape()" << endl;
    }
    virtual double getArea()                         //定义虚函数
   {
        return 0;
   }
};

class Rectangle: public Shape                        //派生类 Rectangle
{
public:
    Rectangle(double l, double w)
    {
        m_dLen  = l;
        m_dWide = w;
        cout << "Rectangle()" << endl;
    }
    ~Rectangle()
    {
        cout << "~Rectangle()" << endl;
    }
    double getArea()
    {
        double area = m_dLen * m_dWide;
```

```
        return area;
    }
private:
    double m_dLen, m_dWide;
};

class Circle: public Shape                          //派生类 Circle
{
public:
    Circle(int r)
    {
        m_dR = r;
        cout << "Circle()" << endl;
    }
    ~Circle()
    {
        cout << "~Circle()" << endl;
    }
    double getArea()
    {
        double area = PI * m_dR * m_dR;
          return area;
    }
private:
    double m_dR;
};

class Square: public Rectangle
{
public:
    Square(double x):Rectangle(x, x)
    {
        cout << "Square()" << endl;
    }
```

```
    ~Square()
    {
        cout << "~Square()" << endl;
    }
};

int main()
{
    double l, w;
    cout << "请输入矩形的长和宽：";
    cin >> l >> w;
    Rectangle Rec(l, w);
    Shape *p1 = &Rec;
    cout << "矩形的面积是：" << Rec.getArea() << endl;
    cout << "矩形的面积是：" << p1 -> getArea() << endl;

    double r;
    cout << "请输入圆的半径：";
    cin >> r;
    Circle Cir(r);
    cout << "圆的面积是：" << Cir.getArea() << endl;

    cout << "请输入正方形的边长：";
    cin >> l;
    Square Squ(l);
    cout << "正方形的面积是：" << Squ.getArea() << endl;

    system("pause");
    return 0;
}
```

源代码 2：

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

#define PI 3.1415

class Shape                                          //基类 Shape
（抽象类）
{
public:
    Shape()
    {
       cout << "Shape()" << endl;
    }
    ~Shape()
    {
       cout << "~Shape()" << endl;
    }
    virtual double getArea() = 0;                    //定义纯虚函数
};

class Rectangle: public Shape                        //派生类 Rectangle
{
public:
    Rectangle(double l, double w)
    {
       m_dLen  = l;
       m_dWide = w;
       cout << "Rectangle()" << endl;
    }
    ~Rectangle()
    {
       cout << "~Rectangle()" << endl;
    }
```

```
    double getArea()
    {
        double area = m_dLen * m_dWide;
        return area;
    }
private:
    double m_dLen, m_dWide;
};

class Circle: public Shape                          //派生类 Circle
{
public:
    Circle(int r)
    {
        m_dR = r;
        cout << "Circle()" << endl;
    }
    ~Circle()
    {
        cout << "~Circle()" << endl;
    }
    double getArea()
    {
        double area = PI * m_dR * m_dR;
          return area;
    }
private:
    double m_dR;
};

class Square: public Rectangle
{
public:
    Square(double x):Rectangle(x, x)
```

```
    {
        cout << "Square()" << endl;
    }
    ~Square()
    {
        cout << "~Square()" << endl;
    }
};

int main()
{
    double l, w;
    cout << "请输入矩形的长和宽：";
    cin >> l >> w;
    Rectangle Rec(l, w);
    Shape *p1 = &Rec;
    cout << "矩形的面积是：" << Rec.getArea() << endl;
    cout << "矩形的面积是：" << p1 -> getArea() << endl;

    double r;
    cout << "请输入圆的半径：";
    cin >> r;
    Circle Cir(r);
    cout << "圆的面积是：" << Cir.getArea() << endl;

    cout << "请输入正方形的边长：";
    cin >> l;
    Square Squ(l);
    cout << "正方形的面积是：" << Squ.getArea() << endl;

    system("pause");
    return 0;
}
```

感想：

本实验通过三道题让我们学习了继承与派生的实现，基类和派生类构造函数和析构函数的调用顺序，以及虚函数和抽象类的用法。抽象类(包含纯虚函数的类)不能创建对象,而包含虚函数的可以创建对象；纯虚函数只有定义没有实现，而虚函数既有定义又有实现*/

实验五 多态性

一、实验目的：

1、 理解面向对象程序设计中多态性的概念。

2、 理解C＋＋中同名函数之间的管理方法。

3、 掌握C＋＋中多态性的应用。

二、实验内容和要求：

1、 对Point类重载＋＋和－－运算符

编写C++程序完成以下功能：

（1）Point 类的属性包括点的坐标（x，y）；

（2）实现 Point 类重载＋＋和－－运算符：

- ++p，--p，p++，p--。
- ＋＋和－－分别表示 x，y 增加或减少 1。

源代码：

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

class Point
{
public:
   Point (int x, int y);
   ~Point() {};
   Point &operator++();              //前置++
   Point  operator++(int);           //后置++,用int进行区分
   Point &operator--();              //前置--
   Point  operator--(int);           //后置-- 用int进行区分
   void get_XY();
private:
```

```
    int m_iX, m_iY;
};

Point::Point(int x, int y)
{
    m_iX = x;
    m_iY = y;
}

Point &Point::operator++()
{
    m_iX ++;
    m_iY ++;
    return *this;
}

Point Point::operator++(int)
{
    Point old(*this);                          //定义临时变量
old，采用直接赋值的方法，使用默认拷贝构造函数
    this -> m_iX ++;
    this -> m_iY ++;
    //return *this;
    return old;                    //实现返回this++之前的值
}

Point &Point::operator--()
{
    m_iX --;
    m_iY --;
    return *this;
}

Point Point::operator--(int)
```

```
{
   Point old(*this);                          //定义临时变量
old，采用直接赋值的方法，使用默认拷贝构造函数
   this -> m_iX --;
   this -> m_iY --;
   //return *this;              //实现返回 this-- 之前的值
   return old;
}

void Point::get_XY()
{
   cout << "(x, y) = ";
   cout << "(" << m_iX << ", ";
   cout << m_iY << ")" << endl;
}

int main()
{
   int x, y;

   cout << "请输入 x , y :";
   cin >> x >> y;
   Point poi1(x, y);
   cout << " ++point1 : ";
   (++poi1).get_XY();

   //cout << "请输入 x , y :";
   //cin >> x >> y;
   Point poi2(x, y);
   cout << " point2++ : ";
   (poi2++).get_XY();

   //cout << "请输入 x , y :";
   //cin >> x >> y;
```

```
    Point poi3(x, y);
    cout << " --point3 : ";
    (--poi3).get_XY();

    //cout << "请输入 x , y :";
    //cin >> x >> y;
    Point poi4(x, y);
    cout << " point4-- : ";
    (poi4--).get_XY();

    system("pause");
    return 0;
}
```

感想：

本题练习了运算符重载，当运算符前置时，函数返回当前对象的值，故用引用。而运算符后置时，函数当返回进行运算前的值，故不用引用。本题还让我体会到了this指针的妙用。

实验六 流式 IO

一、实验目的：

1、 理解C＋＋的IO流类库的概念和结构。

2、 掌握C＋＋流类库的简单应用。

二、实验内容和要求：

1、 流式IO（一）

编写C++程序完成以下功能：

（1）使用 ofstream 向一个文本文件中输出各种类型的数据，并打开文件观察结果：

- 整数、无符号整型、长整型、浮点型、字符串、……

（2）用十进制、八进制、十六进制方式向文本文件中输出整数；

（3）使用控制符和成员函数来控制输出的格式：

- set（） precision() ...

源代码：

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

int main()
{
   ofstream output;

   output.open("text1.txt", ios::out); //打开文件用于输出
   if(output.fail())                   //判断文件是否打开成功
   {
      cout << "File does not exit." << endl;
      cout << "Exit programe." << endl;
      return 0;
```

```
    }

    /*-------------------------- 向 文 件 中 写 入 数 据
-------------------------*/
    int i;
    unsigned int ui;
    long  li;
    float f;
    double d;
    char  c;
    string str;

    cout << "Please input an integer:";
    cin  >> i;
    cout << "Please input an unsigned integer:";
    cin  >> ui;
    cout << "Please input a long int:";
    cin  >> li;
    cout << "Please input a float:";
    cin  >> f;
    cout << "Please input a double:";
    cin  >> d;
    cout << "Please input a char:";
    cin  >> c;
    cout << "Please input a string:";
    //cin  >> str;
    getline(cin, str);

    output << setw(10) << i;
    output << setw(10) << ui;
    output << setw(10) << li;
    output << setw(10) << setprecision(8) << showpoint << f;
//用 showpoint 使当小数部分全为 0 时可输出小数部分而不只是整数
部分
```

```
    output << setw(10) << setprecision(8) << showpoint << d;
    output << setw(10) << c;
    output << setw(10) << str;

    output << endl;
    output << "十进制：   i = " << dec << i << endl;
    output << "八进制:   i = " << oct << i << endl;
    output << "十六进制: i = " << hex << i << endl;

    output.close();                              //关闭文件

    system("pause");
    return 0;
}
```

2、 流式IO（三）

编写 C++程序完成以下功能：

（1）输入一个文本文件名；

（2）打开文件名，在该文件的每一行前面加上一个行号，保存在另外一个文本文件中。

源代码：

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

int main()
{
    ifstream input;
    input.open("text2.1.txt");
```

```
    ofstream output;
    output.open("text2.2.txt");
    if(input.fail()||output.fail())   //判断文件是否打开成功
    {
        cout << "File does not exit." << endl;
        cout << "Exit programe." << endl;
        return 0;
    }

    string str;
    int i = 1;
    while(getline(input, str))
    {
        output << setw(4) << i << "  " << str << endl;
        i ++;
    }

    input.close();
    output.close();

    system("pause");
    return 0;
}
```

感想：

本题让我练习了C++中文件的操作和格式化输出，通过文件不同的打开方式可对文件进行相应的操作。

在格式化输出中，用 showpoint 有效解决了当浮点型小数部分全为0时只输出整数部分值的问题。

实验七 C＋＋程序设计应用

一、实验目的：

3、 掌握应用面向对象程序设计方法解决实际问题的能力。

4、 综合运用C＋＋编程的能力。

二、实验内容和要求：

1、 电话本

编写 C++程序完成以下功能：

（1）实现简单电话本功能，用姓名来搜索电话号码；

（2）用户输入姓名，程序查找并输出结果；

（3）用户可以通过输入，添加姓名和电话号码；

（4）用户可以删除姓名和电话号码；

（5）电话本可以保存在指定文件中；

（6）电话可被从指定文件中读入到内存。

源代码：

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

class Contact
{
public:
    Contact();                          //构造函数
    ~Contact() {};                      //默认析构函数
    void Item();                        //提示用户进行相应的操作
    void Search();                      //查找联系人
    void Renew();                       //更新联系人
    void Add();                         //添加联系人
```

```
    void Del();                                 //删除联系人
    void Exit();                                //退出程序
private:
    string m_sName[10];                          //姓名
    string m_sPhonum[10];                       //电话号码
    int m_iN;                                   //电话本行数
};

Contact::Contact()
{
    cout << "\t\t\t\t      Contacts Book" << endl;
    cout << "\t\t\t\t<1> Search contacts" << endl;
    cout << "\t\t\t\t<2> Renew    contacts" << endl;
    cout << "\t\t\t\t<3> Add        contacts" << endl;
    cout << "\t\t\t\t<4> Delete contacts" << endl;
    cout << "\t\t\t\t<5> Exit      program"    << endl;
     fstream file1;
     file1.open("contactBook.txt", ios::in | ios::out);
     //以输入输出方式打开文件 contactBook.txt
     if(file1.fail())
     //判断文件是否打开成功
    {
        cout << "File does not find." << endl;
        cout << "Exit programe." << endl;
        exit(0);
    }

     string str;
     m_iN = 0;
     while(getline(file1,str))
          m_iN ++;


    fstream file;
```

```
    file.open("contactBook.txt", ios::in | ios::out);
    for(int i=0; i <= m_iN; i ++)
        file >> m_sName[i] >> m_sPhonum[i];

    file1.close();
    file.close();
    Item();
}

void Contact::Item()
{
    cout << "\t\tPlease input operator 1 ~ 5 you want to do:";
    int ope;
    cin >> ope;
    switch(ope)
    {
        case 1: Search(); break;
        case 2: Renew();  break;
        case 3: Add();    break;
        case 4: Del();    break;
        case 5: Exit();   break;
    }
}

void Contact::Search()
{
    string name;
    cout << "\t\tPlease input name you want to search:";
    cin >> name;

    int is = 0;
    int i, m=0;
    for(i=0; i <= m_iN-1; i ++)
    {
```

```
        if(m_sName[i] == name){
            is = 1;
            m = i;
            break;
        }
    }

    if(1 == is)
        cout << "\t\tFound!    " << m_sName[m] << "'s phone number is:"
<< m_sPhonum[m] << endl;
    if(0 == is)
         cout << "\t\tNot found!" << endl;

    Item();
}

void Contact::Renew()
{
    string name;
    string num;
    cout << "\t\tPlease input name you want to renew:";
    cin >> name;

    ofstream file;
    file.open("contactBook.txt");

    int is = 0;
    int i, m=0;
    for(i=0; i <= m_iN; i ++)
    {
        if(m_sName[i] == name){
            is = 1;
            m = i;
            break;
```

```
        }
    }

     int ope;
    if(is == 1)
    {
        cout << "\t\tFound this contact!" << endl;
        cout << "\t\t<1>Renew name\n\t\t<2>Renew phone number\n\t\t<3>Renew both name and phone number" << endl;
        cin  >> ope;
        switch(ope){
            case 1:
                cout << "\t\tPlease input new name:";
                cin >> name;
                m_sName[m] = name;
                break;
            case 2:
                cout << "\t\tPlease input new phone number:";
                cin >> num;
                m_sPhonum[m] = num;
                break;
            case 3:
                cout << "Please input new name and new phone number:";
                cin >> name >> num;
                m_sName[m]    = name;
                m_sPhonum[m] = num;
                break;
        }
        cout << "\t\tSuccessful!" << endl;
    }

    if(is == 0)
        cout << "\t\tNot found!" << endl;
```

```
    for(i = 0; i <= m_iN-1; i++)
        file << m_sName[i] << " " <<    m_sPhonum[1] << endl;

    file.close();
    Item();
}

void Contact::Add()
{
    string name;
    string num;
    cout << "\t\tPlease input the name and phone number you want to add:";
    cin >> name >> num;
    m_sName[m_iN] = name;
    m_sPhonum[m_iN] = num;
    m_iN++;

    ofstream file;
    file.open("contactBook.txt");

    for(int i = 0; i <= m_iN-1; i++)
        file << m_sName[i] << " " <<    m_sPhonum[i] << endl;

    cout << "\t\tSuccessful!" << endl;

    file.close();
    Item();
}

void Contact::Del()
{
    string name;
    string num;
```

```
cout << "\t\tPlease input name you want to delete:";
cin >> name;

ofstream file;
file.open("contactBook.txt");

int is = 0;
int i, m=0;
for(i=0; i <= m_iN; i ++)
{
    if(m_sName[i] == name){
        is = 1;
        m = i;
        break;
    }
}
if(is == 1)
{
      for(i = m; i <= m_iN-2; i++)
      {
         m_sName[i] = m_sName[i+1];
         m_sPhonum[i] = m_sPhonum[i+1];
      }
}
if(is == 0)
      cout << "\t\tNot found!" << endl;

for(i = 0; i <= m_iN-1; i++)
      file << m_sName[i] << " " <<    m_sPhonum[i] << endl;

cout << "\t\tSuccessful!" << endl;

  file.close();
Item();
```

```
}

void Contact::Exit()
{
    cout << "\t\t\t\tExit program!" << endl;
    exit(0);
//退出程序
}

int main()
{
    Contact con;                                    //创建联系人对象
    system("pause");
    return 0;
}
```

感想：

本题是一道 C++的综合练习题，通过文件考察了我的综合能力，起初看到题目并没有想出有效的方法解决问题，随着思路的延伸，开始拨云见日，在类的私有属性中，用数组来存储联系人的姓名和电话号码，继而相应的添加，查找，删除，更新等操作都转化为对数组的操作，只需要将信息从文件读取和输入即可，由于信息事先存在文件中，为了确定数组的大小，需先用 getline()读一遍文件，确定文件有多少行信息（每条信息占一行，包含姓名和电话号码）。本题还体会到了 string 类的优良特性，这是 C 语言所不具备的。