面向对象程序设计实践（C++）实践报告

班级：2018211318

姓名：熊宇

学号：2018210074

专业：计算机科学与技术

**基础题目**

1. C++基础知识实验

编写C++程序完成“矩阵”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。

程序源代码：

*#include <iostream>*

*using namespace std;*

*//假定矩阵大小为4×5（整型）*

*typedef struct*

*{*

*int Num[4][5];*

*}Trc,\* Mar;*

*//定义矩阵初始化函数，可以从cin中输入矩阵元素;*

*void MarInit(Mar A);*

*//定义矩阵输出函数，将矩阵格式化输出到cout；*

*void MarPrint(Mar A);*

*//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；*

*void MarAdd(Mar A, Mar B, Mar C);*

*//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；*

*void MarSub(Mar A, Mar B, Mar C);*

*int main()*

*{*

*//矩阵空间采用new动态申请，保存在指针中*

*//动态申请三个矩阵：A1、A2、A3；*

*Mar A1 = new Trc;*

*Mar A2 = new Trc;*

*Mar A3 = new Trc;*

*//初始化A1、A2；*

*MarInit(A1);*

*MarInit(A2);*

*//计算并输出A3 = A1加A2，A3 = A1减A2；*

*MarAdd(A1, A2, A3);*

*MarPrint(A3);*

*MarSub(A1, A2, A3);*

*MarPrint(A3);*

*//释放矩阵空间。*

*delete A1;*

*delete A2;*

*delete A3;*

*system("pause");*

*return 0;*

*}*

*void MarInit(Mar A)*

*{*

*cout << "请输入四行五列矩阵：" << endl;*

*for (int i = 0;i < 4;i++)*

*{*

*for (int j = 0;j < 5;j++)*

*cin >> A->Num[i][j];*

*}*

*}*

*void MarPrint(Mar A)*

*{*

*cout << "当前矩阵为：" << endl;*

*for (int i = 0;i < 4;i++)*

*{*

*for (int j = 0;j < 5;j++)*

*cout << A->Num[i][j] << " ";*

*cout << endl;*

*}*

*}*

*void MarAdd(Mar A, Mar B, Mar C)*

*{*

*cout << "---执行矩阵加法---" << endl;*

*for (int i = 0;i < 4;i++)*

*{*

*for (int j = 0;j < 5;j++)*

*C->Num[i][j] = A->Num[i][j] + B->Num[i][j];*

*}*

*}*

*void MarSub(Mar A, Mar B, Mar C)*

*{*

*cout << "---执行矩阵减法---" << endl;*

*for (int i = 0;i < 4;i++)*

*{*

*for (int j = 0;j < 5;j++)*

*C->Num[i][j] = A->Num[i][j] - B->Num[i][j];*

*}*

*}*

输入及输出：

请输入四行五列矩阵：

99 89 79 69 58

48 38 28 17 27

37 47 56 66 76

86 95 85 75 65

请输入四行五列矩阵：

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

---执行矩阵加法---

当前矩阵为：

100 91 82 73 63

54 45 36 26 37

48 59 69 80 91

102 112 103 94 85

---执行矩阵减法---

当前矩阵为：

98 87 76 65 53

42 31 20 8 17

26 35 43 52 61

70 78 67 56 45

1. 类与对象实验

2\_1

编写C++程序完成“圆形”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；

（要求当用户不输入坐标数据时，能够默认为坐标原点（0，0））

（2） 定义一个圆形类，其属性包括圆心和半径；

（3） 创建两个圆形对象，提示用户输入圆心坐标和半径，判断两个圆是否相交，并输出结果；

（4） 观察圆形对象以及Point类成员的构造函数与析构函数的调用。

（提示及要求：1，可通过在构造与析构函数中加入输出提示信息的方式观察相关调用；可以使用system("pause")进行程序的暂停；2，能够理解并说明每一次构造与析构函数调用是哪个对象的调用，并观察和解释相关调用顺序及其原因）

程序源代码：

*#include <iostream>*

*#include <cmath>*

*using namespace std;*

*//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；*

*//（要求当用户不输入坐标数据时，能够默认为坐标原点（0，0））*

*double max(double a, double b)*

*{*

*if (a >= b)*

*return a;*

*else*

*return b;*

*}*

*class Point*

*{*

*private:*

*int x = 0;*

*int y = 0;*

*public:*

*Point(int x, int y)*

*{*

*cout << "Point类构造函数被执行" << endl;*

*this->x = x;*

*this->y = y;*

*}*

*~Point()*

*{*

*cout << "Point类析构函数被执行" << endl;*

*}*

*};*

*//定义一个圆形类，其属性包括圆心和半径；*

*class Circle*

*{*

*private:*

*Point center;*

*int route;*

*public:*

*Circle(int x, int y, int r):center(x,y),route(r)*

*{*

*cout << "Circle类构造函数被执行" << endl;*

*}*

*~Circle()*

*{*

*cout << "Circle类析构函数被执行" << endl;*

*}*

*};*

*class Judge*

*{*

*private:*

*Circle C1;*

*Circle C2;*

*public:*

*Judge(int x1, int y1, int r1, int x2, int y2, int r2) :C1(x1, y1, r1), C2(x2, y2, r2)*

*{*

*cout << "Judge类构造函数被执行" << endl;*

*if ((abs(sqrt(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2))) > abs(r1 - r2)) && (abs(sqrt(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2))) < abs(r1 + r2)))*

*cout << "两圆相交" << endl;*

*else*

*{*

*if (((abs(sqrt(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2)))) > 0) && ((abs(sqrt(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2)))) < abs(r1 - r2)))*

*cout << "两圆不相交" << endl;*

*else*

*cout << "两圆相交" << endl;*

*}*

*}*

*~Judge()*

*{*

*cout << "Judge类析构函数被执行" << endl;*

*}*

*};*

*int main()*

*{*

*//创建两个圆形对象，提示用户输入圆心坐标和半径，判断两个圆是否相交，并输出结果；*

*cout << "请输入第一个圆的圆心坐标和半径，格式为：圆心横坐标 圆心纵坐标 半径" << endl;*

*int x1, y1, r1;*

*cin >> x1 >> y1 >> r1;*

*cout << "请输入第二个圆的圆心坐标和半径，格式为：圆心横坐标 圆心纵坐标 半径" << endl;*

*int x2, y2, r2;*

*cin >> x2 >> y2 >> r2;*

*Judge j(x1, y1, r1, x2, y2, r2);*

*system("pause");*

*return 0;*

*}*

输入及输出：

请输入第一个圆的圆心坐标和半径，格式为：圆心横坐标 圆心纵坐标 半径

2 2 2

请输入第二个圆的圆心坐标和半径，格式为：圆心横坐标 圆心纵坐标 半径

1 1 3

Point类构造函数被执行

Circle类构造函数被执行

Point类构造函数被执行

Circle类构造函数被执行

Judge类构造函数被执行

两圆相交

请按任意键继续. . .

Judge类析构函数被执行

Circle类析构函数被执行

Point类析构函数被执行

Circle类析构函数被执行

Point类析构函数被执行

思考：

观察圆形对象以及Point类成员的构造函数与析构函数的调用：

在输入两点坐标和两圆半径后；

当类中有成员是其他类的对象时，首先调用成员变量的构造函数，调用顺序和成员变量的定义顺序一致。成员变量的构造函数统统执行完毕后，再调用该类的构造函数；

即先调用j->C1->center的Point类构造函数，构造C1的圆心；

再调用j->C1的Circle类构造函数，构造C1这个圆；

之后调用j->C2->center的Point类构造函数，构造C2的圆心；

再调用j->C2的Circle类构造函数，构造C2这个圆；

最后调用j的Judge类构造函数，构造Judge来判断是否两圆相交；

析构函数的执行顺序是“倒关”的方式，即先与构造函数的执行顺序相反。

2\_2

编写C++程序完成“矩阵”类以下功能：

（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；（要求及提示：最好能实现对赋值操作符“=”的重载；注意检查“自赋值”、释放“旧元素”）

（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、pA2、pA3。

程序源代码：

*#include <iostream>*

*#include <cmath>*

*using namespace std;*

*//*

*class Martric*

*{*

*//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）。*

*private:*

*int line;*

*int row;*

*int\*\* data = NULL;*

*//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。*

*public:*

*Martric(int line, int row)*

*{*

*cout << "Martric类构造函数被执行" << endl;*

*this->line = line;*

*this->row = row;*

*this->data = new int\* [line];*

*for (int i = 0;i < line;i++)*

*this->data[i] = new int[row];*

*}*

*Martric(Martric& M)*

*{*

*cout << "Martric类拷贝构造函数被执行" << endl;*

*this->line = M.line;*

*this->row = M.row;*

*this->data = new int\* [line];*

*for (int i = 0;i < line;i++)*

*this->data[i] = new int[row];*

*for (int i = 0;i < this->line;i++)*

*{*

*for (int j = 0;j < this->row;j++)*

*this->data[i][j] = M.data[i][j];*

*}*

*}*

*void Input()*

*{*

*cout << "Martric类输入函数被执行" << endl;*

*for (int i = 0;i < this->line;i++)*

*{*

*for (int j = 0;j < this->row;j++)*

*cin >> this->data[i][j];*

*}*

*}*

*void Output()*

*{*

*cout << "Martric类输出函数被执行" << endl;*

*for (int i = 0;i < this->line;i++)*

*{*

*for (int j = 0;j < this->row;j++)*

*cout << this->data[i][j] << " ";*

*cout << endl;*

*}*

*}*

*void MarAdd(Martric& A, Martric& B, Martric& C)*

*{*

*cout << "---执行矩阵加法---" << endl;*

*for (int i = 0;i < A.line;i++)*

*{*

*for (int j = 0;j < A.row;j++)*

*C.data[i][j] = A.data[i][j] + B.data[i][j];*

*}*

*}*

*void MarSub(Martric& A, Martric& B, Martric& C)*

*{*

*cout << "---执行矩阵减法---" << endl;*

*for (int i = 0;i < A.line;i++)*

*{*

*for (int j = 0;j < A.row;j++)*

*C.data[i][j] = A.data[i][j] - B.data[i][j];*

*}*

*}*

*~Martric()*

*{*

*cout << "Martric类析构函数被执行" << endl;*

*for (int i = 0;i < this->line;i++)*

*{*

*delete[] this->data[i];*

*}*

*delete[] this->data;*

*}*

*};*

*int main()*

*{*

*//定义三个矩阵：A1、A2、A3；*

*int line, row;*

*cout << "请输入矩阵的行数与列数" << endl;*

*cin >> line >> row;*

*Martric A1(line, row);*

*Martric A2(line, row);*

*Martric temp(line, row);*

*//初始化A1、A2；*

*cout << "请初始化第一个矩阵" << endl;*

*A1.Input();*

*cout << "请初始化第二个矩阵" << endl;*

*A2.Input();*

*//计算并输出A3 = A1加A2，A3 = A1减A2；（要求及提示：最好能实现对赋值操作符“ = ”的重载；注意检查“自赋值”、释放“旧元素”）*

*A1.MarAdd(A1, A2, temp);*

*Martric A3 = temp;*

*A3.Output();*

*A1.MarSub(A1, A2, A3);*

*A3.Output();*

*//用new动态创建三个矩阵类的对象：pA1、pA2、pA3；*

*Martric\* pA1 = new Martric(line, row);*

*Martric\* pA2 = new Martric(line, row);*

*Martric\* pA3 = new Martric(line, row);*

*//初始化pA1、pA2；*

*pA1->Input();*

*pA2->Input();*

*//计算并输出pA3 = pA1加pA2，pA3 = pA1减pA2；*

*pA1->MarAdd(\*pA1, \*pA2, \*pA3);*

*pA3->Output();*

*pA1->MarSub(\*pA1, \*pA2, \*pA3);*

*pA3->Output();*

*//释放pA1、pA2、pA3。*

*pA1->~Martric();*

*pA2->~Martric();*

*pA3->~Martric();*

*system("pause");*

*return 0;*

*}*

输入及输出：

请输入矩阵的行数与列数

4 5

Martric类构造函数被执行

Martric类构造函数被执行

Martric类构造函数被执行

请初始化第一个矩阵

Martric类输入函数被执行

99 89 79 69 58

48 38 28 17 27

37 47 56 66 76

86 95 85 75 65

请初始化第二个矩阵

Martric类输入函数被执行

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

---执行矩阵加法---

Martric类拷贝构造函数被执行

Martric类输出函数被执行

100 91 82 73 63

54 45 36 26 37

48 59 69 80 91

102 112 103 94 85

---执行矩阵减法---

Martric类输出函数被执行

98 87 76 65 53

42 31 20 8 17

26 35 43 52 61

70 78 67 56 45

Martric类构造函数被执行

Martric类构造函数被执行

Martric类构造函数被执行

Martric类输入函数被执行

99 89 79 69 58

48 38 28 17 27

37 47 56 66 76

86 95 85 75 65

Martric类输入函数被执行

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

---执行矩阵加法---

Martric类输出函数被执行

100 91 82 73 63

54 45 36 26 37

48 59 69 80 91

102 112 103 94 85

---执行矩阵减法---

Martric类输出函数被执行

98 87 76 65 53

42 31 20 8 17

26 35 43 52 61

70 78 67 56 45

Martric类析构函数被执行

Martric类析构函数被执行

Martric类析构函数被执行

请按任意键继续. . .

Martric类析构函数被执行

Martric类析构函数被执行

Martric类析构函数被执行

Martric类析构函数被执行

1. 继承与派生实验

编写C++程序完成“形状”的以下功能：

（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：矩形类和圆形类；

（3） 从矩形类派生正方形类；

（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；

（提示及要求：1，可通过在构造与析构函数中加入输出提示信息的方式观察相关调用；可以使用system("pause")进行程序的暂停；2，能够理解并说明每一次构造与析构函数调用是哪个对象的调用，并观察和解释相关调用顺序及其原因）

（6） 对不同对象计算面积。

程序源代码：

*#include <iostream>*

*#include <iomanip>*

*#define Pi 3.14159265358979323846264338328*

*using namespace std;*

*//声明一个基类Shape（形状），其中包含一个方法来计算面积；*

*class Shape*

*{*

*protected:*

*int type;//0为矩形 1为圆形*

*int edge1, edge2;//矩形*

*int r;//半径*

*public:*

*Shape(int type, int edge1, int edge2, int r)*

*{*

*cout << "Shape类构造函数被执行" << endl;*

*this->type = type;*

*this->edge1 = edge1;*

*this->edge2 = edge2;*

*this->r = r;*

*}*

*void Aera()*

*{*

*double area = 0;*

*if (type == 0)*

*area = edge1 \* edge2;*

*else if (type == 1)*

*area = Pi \* r \* r;*

*cout.setf(ios::fixed);*

*cout << "面积为" << fixed << setprecision(2) << area << endl;*

*}*

*~Shape()*

*{*

*cout << "Shape类析构函数被执行" << endl;*

*}*

*};*

*//从Shape派生两个类：矩形类和圆形类；*

*class Rectangular : public Shape*

*{*

*public:*

*Rectangular(int type, int edge1, int edge2) :Shape(type, edge1, edge2, r)*

*{*

*cout << "Rectangular类构造函数被执行" << endl;*

*this->type = type;*

*this->edge1 = edge1;*

*this->edge2 = edge2;*

*};*

*~Rectangular()*

*{*

*cout << "Rectangular类析构函数被执行" << endl;*

*}*

*};*

*class Circle : public Shape*

*{*

*public:*

*Circle(int type, int r) :Shape(type, edge1, edge2, r)*

*{*

*cout << "Circle类构造函数被执行" << endl;*

*this->type = type;*

*this->r = r;*

*};*

*~Circle()*

*{*

*cout << "Circle类析构函数被执行" << endl;*

*}*

*};*

*//从矩形类派生正方形类；*

*class Square :public Rectangular*

*{*

*public:*

*Square(int type, int edge1, int edge2) :Rectangular(type, edge1, edge2)*

*{*

*cout << "Square类构造函数被执行" << endl;*

*this->type = type;*

*this->edge1 = edge1;*

*this->edge2 = edge2;*

*}*

*~Square()*

*{*

*cout << "Square类析构函数被执行" << endl;*

*}*

*};*

*int main()*

*{*

*//创建派生类的对象，观察构造函数、析构函数调用次序；*

*//建一个Square 一个Circle*

*Square S(0, 2, 2);*

*Circle C(1, 1);*

*//对不同对象计算面积。*

*S.Aera();*

*C.Aera();*

*system("pause");*

*return 0;*

*}*

输入及输出：

Shape类构造函数被执行

Rectangular类构造函数被执行

Square类构造函数被执行

Shape类构造函数被执行

Circle类构造函数被执行

面积为4.00

面积为3.14

请按任意键继续. . .

Circle类析构函数被执行

Shape类析构函数被执行

Square类析构函数被执行

Rectangular类析构函数被执行

Shape类析构函数被执行

思考：

理解并说明每一次构造与析构函数调用是哪个对象的调用，并观察和解释相关调用顺序及其原因：

首先我们要明确，构造函数的调用顺序为：基类构造函数、对象成员构造函数、派生类本身的构造函数，而析构函数的调用顺序为：派生类本身的析构函数、对象成员析构函数、基类析构函数（与构造顺序正好相反）；

这里派生顺序为Shape->Rectangular->Square

->Cirlce

所以在创建Square类对象时，依次调用Shape类、Rectangular类、Square类的构造函数；在创建Circle类对象时，依次调用Shape类、Circle类构造函数；

而在析构函数执行时，依次调用Circle类、Shape类析构函数；Square类、Rectangular类、Shape类析构函数。

1. I/O流实验

编写C++程序完成猜价格游戏的以下功能：

（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。

（提示及要求：1，要求使用C++的输入输出方式（cin, cout），不能使用C语言的printf等；2，注意检查输入的合法性）

程序源代码：

*#include <iostream>*

*#include <time.h>*

*using namespace std;*

*//检查输入的合法性*

*bool isint(char s[])*

*{*

*for (int i = 0; s[i] != '\0'; i++)*

*if (s[i]<1||s[i]>255||!isdigit(s[i]))*

*return false;*

*return true;*

*}*

*int main()*

*{*

*//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）；*

*srand((int)time(NULL));*

*int price = rand() % 1000 + 1;*

*char input[100];*

*int guess = 0;*

*//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；*

*cout << "请输入你猜测的价格（1-1000的整数）" << endl;*

*while (guess != price)*

*{*

*while (cin >> input && !isint(input))*

*{*

*cout << "请输入1-1000范围内的整数" << endl;*

*getchar();*

*}*

*//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；*

*guess = atoi(input);*

*if (guess < price)*

*cout << "你猜的价格太低了" << endl;*

*else if (guess > price)*

*cout << "你猜的价格太高了" << endl;*

*}*

*//直到猜对为止，并给出提示。*

*cout << "你猜对了" << endl;*

*system("pause");*

*return 0;*

*}*

输入及输出：

请输入你猜测的价格（1-1000的整数）

abdjwsnd

请输入1-1000范围内的整数

500

你猜的价格太高了

250

你猜的价格太高了

125

你猜的价格太高了

62

你猜的价格太低了

93

你猜的价格太低了

109

你猜的价格太低了

116

你猜的价格太低了

120

你猜的价格太低了

122

你猜的价格太低了

123

你猜的价格太低了

124

你猜对了

5、重载实验

5\_1

虚函数

针对题目3的“形状”类，编写C++程序完成以下功能：

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

（2） 将【形状】中的基类定义为抽象类，比较与【形状（A）】程序的差异。

程序源代码：

#include <iostream>

#include <iomanip>

#define Pi 3.14159265358979323846264338328

using namespace std;

//声明一个基类Shape（形状），其中包含一个方法来计算面积；

class Shape

{

public:

virtual float area() = 0;//这是一个纯虚函数

Shape()

{

cout << "Shape类构造函数被执行" << endl;

}

~Shape()

{

cout << "Shape类析构函数被执行" << endl;

}

};

//从Shape派生两个类：矩形类和圆形类；

class Rectangular : public Shape

{

private:

float length, wide;

public:

Rectangular(float len, float wid)

{

cout << "Rectangular类构造函数被执行" << endl;

length = len;

wide = wid;

};

~Rectangular()

{

cout << "Rectangular类析构函数被执行" << endl;

}

float area()

{

return length \* wide;

}

};

class Circle : public Shape

{

private:

float R;

public:

Circle(float r)

{

cout << "Circle类构造函数被执行" << endl;

R = r;

};

~Circle()

{

cout << "Circle类析构函数被执行" << endl;

}

float area()

{

return Pi \* R \* R;

}

};

//从矩形类派生正方形类；

class Square :public Rectangular

{

private:

float edge;

public:

Square(float e) :Rectangular(e, e)//构造直接基类

{

cout << "Square类构造函数被执行" << endl;

edge = e;

}

~Square()

{

cout << "Square类析构函数被执行" << endl;

}

//因为基类里面已经有area函数，这里不再写

};

int main()

{

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

//将【形状】中的基类定义为抽象类，比较与【形状（A）】程序的差异。

float a, b, r, e;

cout << "请输入矩形的长和宽：" << endl;

cin >> a >> b;

Rectangular T(a, b);

//我们不能实例化抽象类，那样编译器会报错，抽象类只提供一个接口，他不清楚子类如何对基类中的纯虚函数的实现

Shape\* t = new Rectangular(a, b);

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

cout << "矩形面积为：" << T.area() << endl;

cout << "矩形面积为：" << t->area() << endl;

cout << endl;

cout << "请输入圆的半径：" << endl;

cin >> r;

Circle c(r);

cout << "圆的面积为：" << c.area() << endl;

cout << endl;

cout << "请输入正方形的边长：" << endl;

cin >> e;

Square s(e);

cout << "正方形的面积为：" << s.area() << endl;

cout << endl;

system("pause");

return 0;

}

输入及输出：

请输入矩形的长和宽：

7 8

Shape类构造函数被执行

Rectangular类构造函数被执行

Shape类构造函数被执行

Rectangular类构造函数被执行

矩形面积为：56

矩形面积为：56

请输入圆的半径：

5

Shape类构造函数被执行

Circle类构造函数被执行

圆的面积为：78.5398

请输入正方形的边长：

4

Shape类构造函数被执行

Rectangular类构造函数被执行

Square类构造函数被执行

正方形的面积为：16

请按任意键继续. . .

Square类析构函数被执行

Rectangular类析构函数被执行

Shape类析构函数被执行

Circle类析构函数被执行

Shape类析构函数被执行

Rectangular类析构函数被执行

Shape类析构函数被执行

5\_2

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。

程序源代码：

#include <iostream>

#include <cmath>

using namespace std;

//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；

//（要求当用户不输入坐标数据时，能够默认为坐标原点（0，0））

class Point

{

private:

int x = 0;

int y = 0;

public:

//构造函数和析构函数

Point()

{

cout << "Point类构造函数被执行" << endl;

}

Point(int x, int y)

{

cout << "Point类构造函数被执行" << endl;

this->x = x;

this->y = y;

}

~Point()

{

cout << "Point类析构函数被执行" << endl;

}

//检查函数：输出点坐标

void Output()

{

cout << "横坐标为：" << x << "，纵坐标为：" << y << endl;

}

//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。

//++p，--p

Point operator ++();

Point operator --();

//p++，p--

Point operator ++(int);

Point operator --(int);

};

Point Point::operator ++()

{

Point p;

p.x = x + 1;

p.y = y + 1;

x += 1;

y += 1;

return p;

}

Point Point::operator --()

{

Point p;

p.x = x - 1;

p.y = y - 1;

x -= 1;

y -= 1;

return p;

}

Point Point::operator ++(int)

{

Point p;

p.x = x;

p.y = y;

x += 1;

y += 1;

return p;

}

Point Point::operator --(int)

{

Point p;

p.x = x;

p.y = y;

x -= 1;

y -= 1;

return p;

}

int main()

{

Point A(0, 0);

Point B(7, 9);

Point C(25, 28);

cout << "A的初始坐标为：";

A.Output();

cout << "B的初始坐标为：";

B.Output();

cout << "C的初始坐标为：";

C.Output();

cout << endl;

//p++，p--

A = B++;

cout << "执行 p++ 操作后，A的坐标为：";

A.Output();

cout << "执行 p++ 操作后，B的坐标为：";

B.Output();

A = C--;

cout << "执行 p-- 操作后，A的坐标为：";

A.Output();

cout << "执行 p-- 操作后，C的坐标为：";

C.Output();

cout << endl;

//++p，--p

A = ++B;

cout << "执行 ++p 操作后，A的坐标为：";

A.Output();

cout << "执行 ++p 操作后，B的坐标为：";

B.Output();

A = --C;

cout << "执行 --p 操作后，A的坐标为：";

A.Output();

cout << "执行 --p 操作后，C的坐标为：";

C.Output();

cout << endl;

system("pause");

return 0;

}

输入及输出：

Point类构造函数被执行

Point类构造函数被执行

Point类构造函数被执行

A的初始坐标为：横坐标为：0，纵坐标为：0

B的初始坐标为：横坐标为：7，纵坐标为：9

C的初始坐标为：横坐标为：25，纵坐标为：28

Point类构造函数被执行

Point类析构函数被执行

Point类析构函数被执行

执行 p++ 操作后，A的坐标为：横坐标为：7，纵坐标为：9

执行 p++ 操作后，B的坐标为：横坐标为：8，纵坐标为：10

Point类构造函数被执行

Point类析构函数被执行

Point类析构函数被执行

执行 p-- 操作后，A的坐标为：横坐标为：25，纵坐标为：28

执行 p-- 操作后，C的坐标为：横坐标为：24，纵坐标为：27

Point类构造函数被执行

Point类析构函数被执行

Point类析构函数被执行

执行 ++p 操作后，A的坐标为：横坐标为：9，纵坐标为：11

执行 ++p 操作后，B的坐标为：横坐标为：9，纵坐标为：11

Point类构造函数被执行

Point类析构函数被执行

Point类析构函数被执行

执行 --p 操作后，A的坐标为：横坐标为：23，纵坐标为：26

执行 --p 操作后，C的坐标为：横坐标为：23，纵坐标为：26

实验笔记

在本次基础实验中，我学习了C++的许多基础知识并加以了运用，了解了C++深拷贝浅拷贝的区别、了解了纯虚函数虚基类、多态的性质和使用方法及注意事项，收获颇丰。

**综合题目**

1. 设计思路

本代码的模块化依托于程序的界面切换，图片资源文件存储在Qt的资源文件内，另设definition.h进行宠物小精灵、用户的结构化定义。这里解释一下，我是在total.h 里定义了UserList和numOfUser来将其作为整个项目的全局变量，在程序内需要通过用户信息txt文件对其进行初始化，所以使用了友元函数friend void InitialUserListInfo() 来导入用户信息，他是user的友元，其他非类成员函数，全都是编写过程中书写的检测、调试输出函数。界面主要有：

1. 主界面

主界面设置了项目信息文本框，登录和注册按钮——进入登录和注册界面，此界面不设置socket通信。

1. 注册界面

注册界面设置用户名、密码文本框，注册和返回按钮——点击注册和返回主界面。该界面初始化就与服务器建立连接，退出界面即断开。点击注册按钮后，发送给服务器Regis 昵称 密码，服务器根据Regis标志字段执行注册检验操作，如果用户名已经被注册，反馈IncorrectName ；否则返回Registersuccess 分配的小精灵编号 昵称，同时存进用户账户信息文件、对战信息文件。此处为了方便通信，我将内部设置的宠物小精灵用十进制数字表示，服务器和客户端都根据这个数字编码来识别宠物小精灵。客户端收到注册成功信息后，自动跳转到登录界面；收到该用户名已经被注册信息后，自动返回主界面。这里我参考了一个人性化设计，离开注册界面再次打开注册界面，之前的全部输入会被清空；而离开登录界面再次打开，只会清空密码输入，保留用户名输入。

1. 登录界面

登录界面设置用户名、密码文本框，登录和返回按钮——点击登录和返回主界面。该界面初始化就与服务器建立连接，退出界面即断开。点击登录按钮后，发送给服务器Login 昵称 密码，服务器根据Login标志字段执行登录检验操作，如果账号密码正确，就返回 LoginSuccess + 昵称；否则返回 FailLogin 。客户端收到登录成功信息后，自动跳转到用户界面；收到登录失败信息后，会清空密码输入，保留用户名输入，自动返回主界面。

1. 用户界面

用户界面设置在线好友展示文本框——主要展示在线好友和离线好友，背包按钮——打开背包界面展示用户拥有的宠物小精灵同时输出用户的宠物小精灵信息，胜率及金银牌勋章查询按钮——打开胜率及金银牌勋章查询界面，升级赛、决斗赛按钮——打开选择精灵界面来进行升级赛、决斗赛战斗，会根据点击的按钮传送kind参数来通知下一步骤是进行升级赛还是决斗赛。还有注销按钮——注销用户，将用户状态更改为离线，更新UserList，并更新用户信息文件。

1. 背包界面

利用QLabel初始化标签，然后根据精灵的GetName函数确定图片路径，在资源文件里找到图片利用pixmap实现位置贴图即可，同时QDebug输出宠物小精灵信息。

1. 胜率及金银铜牌勋章查询界面

填写昵称的文本框——输入任意用户的昵称；然后点击查询按钮，就可以在下方的展示文本框输出用户的勋章和高级勋章以及胜率——胜率以胜场数/总场数的形式出现，每次关闭界面前都会清空用户输入和显示。

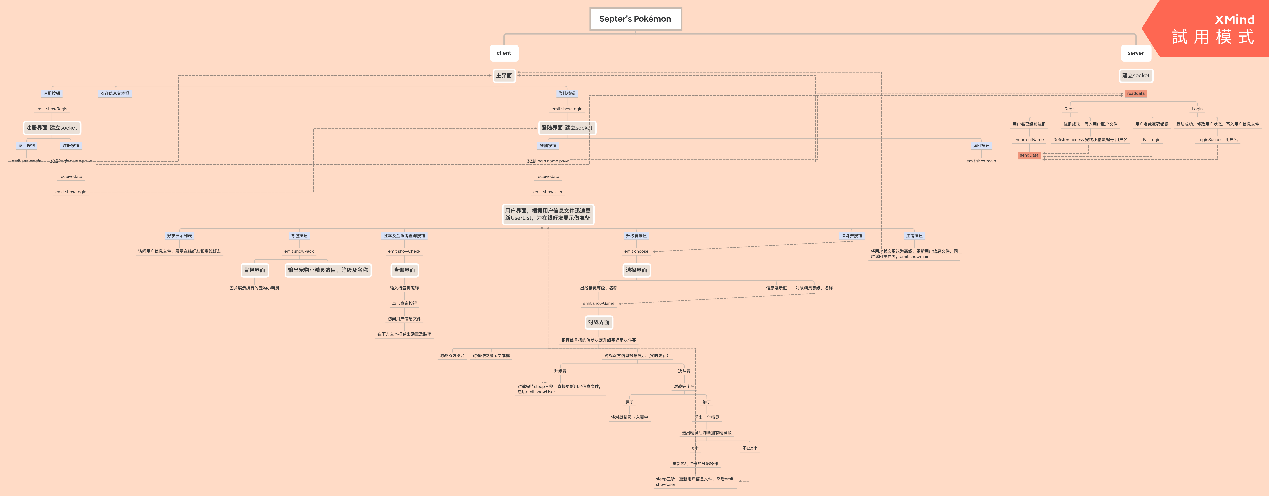
1. 选择出战及对战精灵界面

选择界面包括信息展示文本框——其中包含了当前用户可以选择出战的宠物小精灵的等级和名称，可以选择对战的精灵的等级和名称；还有输入出战宠物小精灵等级和种类输入文本框、对战宠物小精灵等级和种类输入文本框；输入完毕后点击【我选好了】按钮就跳转到对战界面开始作战了。

1. 战斗界面

对战界面包含最顶端文本框——显示是升级赛还是决斗赛；两个QLabel贴图——显示对战双方图片；对战双方各数据输出文本框。在对战过程中，对方攻击力折损自身防御力的50%对自身生命值造成伤害，生命值损伤值直接加入对方经验值。对战结束后，停顿三秒直接回到用户界面。

1. 设计图解



（高清图见附件）

1. 实验笔记

在本次课设综合题的实现过程中，我体会到了C++内存泄漏的可怕之处，更体会到了找内存泄漏的可怕之处。对多态的运用也体会较深，在试图访问派生类的成员时，盲目指针会直接访问到基类的对象，因此使用基类纯虚函数、派生类重写的方式进行数值的获取。

这是我严格意义上的第一次单人课设，在完成他的过程中有艰难困苦，也有满满的成就感；在实现过程中自己去设计、自己去实现、自己去调试，无论是功能的实现还是页面的不断美化，都让我感受到了满满的成就感和收获，我会永远记得自己的第一次单人课设——C++宠物小精灵。

此外，在实现本次课设综合题的时候，我先是用了一定的时间学习了socket编程、Qt开发。现将部分典型代码调试过程总结如下：

* 1. 忽视指针的实际含义

程序代码：

Elf \*ex;

switch(rp)

{

case 1:

ex=new Chavaran;

ex->level=1;

temp.PetVec.push\_back(ex);

delete ex;

break;

… …

}

程序报错：在后面调用小精灵的GetLevel()函数报错如下

pure virtual method called

terminate called without an active exception

原因：将ex加入UserList后直接delete，忽视了vector存储的是指针，指针指向的空间，一旦delete就将空间释放，会出错

* 1. 错误领悟Qt的资源文件含义

在加入宠物小精灵、页面背景等文件进入资源文件后，我错误地以为可以将用户账户文件、信息文件同样加入资源文件，后来频频出现打开文件失败。

原因：失败的文件打开都是以写入方式打开文件，原来Qt的资源文件是不可以写的。转念一想也对，如果可以直接修改资源文件，在开发过程中岂不是乱套了。

* 1. 类继承的虚方法警告

程序报错：warning: 'Chavaran' has no out-of-line virtual method definitions; its vtable will be emitted in every translation unit

原因：如果在头文件中定义了一个类并且具有vtable（它具有虚方法或者它来自具有虚方法的类），则它必须始终在类中具有至少一个外联虚拟方法。 如果没有这个，编译器会将vtable和RTTI复制到每个.o文件中，其中#include标题，膨胀.o文件大小和增加链接时间。

也就是说，当类中定义了虚函数时，编译器会自动生成虚函数表vtable，用来对应每一个虚函数。那么，如果所有虚拟方法都是内联的（在头文件中定义），那么编译器不知道哪个翻译单元（.cpp文件）在其中发出vtable，因此它会在所有这些中发出一个副本，并且链接器会处理用它。这会在目标文件中产生额外的工作和并且是.o文件变得更庞大。另一方面，如果虚拟函数是在线外定义的（在.cpp中），则可以在那里发射vtable，因此只会发出一个副本。

所以知道以上原因过后，解决方法就很简单了，因为我是在类中定义了虚函数并且直接在类定义内部实现这些虚函数，所以将虚函数自动变成了内联函数，那么只需要把虚函数的实现挪到类的外面去实现就可以啦。

* 1. 关于全局变量的长期的不规范写法

本项来自于QT Creator 4.8.0的代码提示功能，让我意识到我的一个长期不规范行为。

之前定义全局变量:

1、在1.cpp文件直接写例如 int test;

2、如果2.cpp需要使用test时，一般我会直接在2.cpp直接加extern int test；

其实这是不规范的，

规范的写法：在1.h中写extern int test，然后在2.cpp包含1.h

如果int test只在1.cpp中使用，应该定义为static int test；

* 1. 关于对战时实时更改双方生命值的问题

刚开始实现后，整个对战界面忽闪忽闪的白屏，后来才知道是因为太快导致反应不过来，解决的办法是：

QApplication::processEvents()//将来不及处理的操作先放着在后台慢慢处理