一、选择题

1. B
2. D
3. A
4. B
5. C
6. D

二、填空题

1. 类的成员包括数据成员和成员函数。
2. 释放对象所占的内存空间并完成善后处理工作的是析构函数。
3. 对于常量数据成员和引用数据成员的初始化只能通过初始值列表来完成。
4. 拷贝构造函数以类的对象的引用作为参数。
5. 当对象生成以后，系统就为这个对象定义了一个this指针，它指向这个对象的地址。
6. 编程题

1.#include<iostream>

#include<fstream>

#include<string>

using namespace std;

class matrix{

int a; //矩阵的维数

double \*arr; //数组

public:

matrix();

matrix(int n ,double \*m);

~matrix();

void setmatrix(int n); //自行输入矩阵元素

matrix operator\*(const matrix &b); //实现矩阵的乘法运算

friend ostream& operator<<(ostream& os,const matrix& m); //实现矩阵的输出

};

matrix::matrix() //定义无参构造函数

{

a = 1;

arr = new double[1];

\*arr = 1;

}

matrix::matrix(int n,double \*m) //定义有参构造函数

{

a = n;

arr = new double[n\*n];

for (int i = 0; i < n \* n; i++)

{

\*(arr + i) = \*(m + i);

}

}

matrix::~matrix() //析构函数的定义

{

delete[]arr;

}

void matrix::setmatrix(int n) //定义自行输入元素的构造函数

{

delete []arr;

a = n;

arr = new double[n\*n];

for (int i = 0; i < n ; i++)

{

for (int j = 0; j < n; j++)

{

cout << "请输入第" << i + 1 << "行" << "第" << j + 1 << "列的元素" << "\t";

cin >> \*(arr + i \* n + j);

cout << endl;

}

}

}

matrix matrix::operator \*(const matrix& b) //重载乘法运算符

{

if (a = b.a)

{

double \*matr1=new double[a\*a];

for(int t=0;t<a\*a;t++)

\*(matr1 + t) = 0;

for (int i = 0; i < a; i++)

{

for (int j = 0; j < a; j++)

{

for (int k = 0; k < a; k++)

{

\*(matr1+i\*a+j)+=(\*(arr + i \* a + k))\*(\*(b.arr + k \* a + j));

}

}

}

matrix\* p = new matrix(a, matr1);

return \*p;

}

else

{

cout << "两个矩阵不是同一类型，无法计算" << endl;

return \*this;

}

}

ostream& operator<<(ostream& os, const matrix& m) //重载输出运算符

{

for (int i = 0; i < m.a; i++)

{

for (int j = 0; j < m.a; j++)

{

os<< \*(m.arr + i \* m.a + j) << "\t";

}

os << endl;

}

return os;

}

int main(){

ifstream in("C:\\Users\\hp\\source\\repos\\2.3 1\\data.txt"); //从文本文档中读取数据

double m1[16]{ 0 }, m2[16]{ 0 }; //定义两个数组存放读取到的数据

if (in) //判断是否读取到数据

{

while (!in.eof())

{

for (int i = 0; i < 16; i++)

{

in >> m1[i];

}

for (int i = 0; i < 16; i++)

{

in >> m2[i];

}

}

in.close();

}

else

cout << "读取失败！" << endl;

matrix a(4,m1), b(4,m2); //定义两个矩阵对象

cout << a \* b;

return 0;

}

2.#include<iostream>

using namespace std;

class Rectangle; //声明矩形类

class Point //定义点类

{

friend Rectangle;

int m\_x;

int m\_y;

public:

Point() { m\_x = 0, m\_y = 0; }

Point(int x, int y)

{

m\_x = x;

m\_y = y;

}

~Point(){}

int print\_x() { return m\_x; }

int print\_y() { return m\_y; }

};

class Rectangle //定义矩形类

{

Point p1;

Point p2;

public:

Rectangle(const Point& m, const Point& n) { p1 = m, p2 = n; }

~Rectangle(){}

int size() { return abs((p1.m\_x - p2.m\_x) \* (p1.m\_y - p2.m\_y)); }

int perimeter() { return (abs(p1.m\_x - p2.m\_x)+ abs(p1.m\_y - p2.m\_y))\*2; }

};

int main() {

int x1,x2,y1, y2;

cout << "输入矩形的左上角点坐标:";

cin >> x1 >> y1;

cout<< "输入矩形的右下角点的坐标:";

cin >> x2 >> y2;

Point p1(x1, y1),p2(x2,y2);

Rectangle r(p1,p2);

cout << "矩形的面积为:" << r.size() << endl;

cout << "矩形的周长为:" << r.perimeter() << endl;

return 0;

}