**拷贝构造函数**

1. **定义并测试一个名为Rectangle的矩形类，其数据成员为矩形的长和宽，两个函数成员arear()和length()，分别计算并输出矩形的面积和周长。**

**（1） 添加构造函数和析构函数。其中构造函数中要求为形参加上默认值，默认值均为5。参考测试值如下：**

Rectangle Rec1(3,2)

Rectangle Rec2 (8)

Rectangle Rec3()

**（2） 添加拷贝构造函数。将Rec1复制给Rec4。**

2、**设计一个复数类，两个数据成员分别表示复数的实部（real）和虚部（imag），有2个构造函数分别在不同的情况下初始化对象，函数Print（）用于输出复数。类的成员函数如下：**

CComplex(double, double);

CComplex(CComplex &c);

CComplex add(CComplex & x);

void Print();

3、三维坐标中的一条直线可通过直线的两端点坐标（x1,y1,z1）和（x2,y2,z2）来描述。定义一个“线”类（Line）,其成员函数包括：

Void SetPoint1(int x,int y,int z); 设置第1个坐标

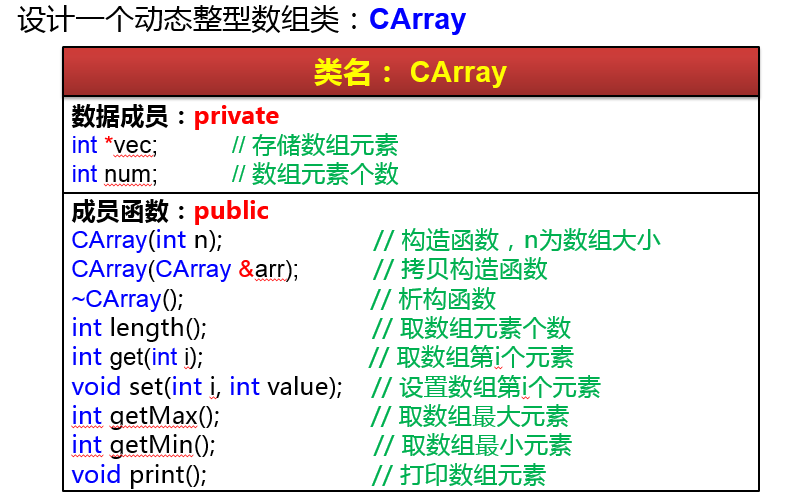
Void SetPoint2(int x,int y,int z); 设置第2个坐标

Void OutPutTwoPoint();输出端点的坐标

Double Length();求出直线的长度

请编写完整的程序，在主函数中完成上述所有成员函数的测试工作。

4、



参考代码：

（1）

#include<iostream.h>

class Rectangle

{

private:

float len,wid;

public:

Rectangle(float x=5,float y=5)

{

len=x;

wid=y;

}

Rectangle(Rectangle &r)

{

len=r.len;

wid=r.wid;

}

void length()

{

cout<<(wid+len)\*2<<endl;

}

void area()

{

cout<<len\*wid<<endl;

}

~Rectangle()

{}

};

void main()

{

Rectangle R1;

Rectangle R2(2);

Rectangle R3(2,3);

Rectangle R4(R3);

R1.length();

R1.area();

R2.length();

R2.area();

R3.length();

R3.area();

R4.length();

R4.area();

（2）

#include <iostream.h>

class CComplex

{

public:

CComplex(double, double);

CComplex(CComplex &c);

CComplex add(CComplex & x);

void Print();

private:

double real, imag;

};

CComplex::CComplex(double r=0.0, double i=0.0)

{

real = r; imag = i;

cout<<"调用两个参数的构造函数"<<endl;

}

CComplex::CComplex (CComplex &c)

{

real = c.real; imag = c.imag;

cout<<"调用拷贝构造函数"<<endl;

}

void CComplex::Print() // 显示复数值

{ cout << "(" << real << "," << imag << ")" << endl; }

void f(CComplex n) //对象作为函数参数

{

cout<<"n=";

n. Print();

}

CComplex CComplex::add(CComplex & x)

{

CComplex y(real+x.real ,imag+x.imag );

return y; //调用复数类的拷贝构造函数

}

void main(void)

{

CComplex a(3.0,4.0), b(5.6,7.9);

CComplex c(a); //调用复数类的拷贝构造函数

cout << "a = ";

a.Print();

cout << "c = ";

c.Print();

f(b); //对象作为函数实参传递给函数形参

c=a.add(b); //对象作为函数返回值，调用拷贝构造函数

c.Print ();

}

4、#include <iostream>

#include<cmath>

using namespace std;

class CArray

{

private:

int \*vec; // 存储数组元素

int num; // 数组元素个数

public:

CArray(int n); // 构造函数，n为数组大小

CArray(CArray &arr); // 拷贝构造函数

~CArray(); // 析构函数

int length(); // 取数组元素个数

int get(int i); // 取数组第i个元素

void set(int i, int value); // 设置数组第i个元素

int getMax(); // 取数组最大元素

int getMin(); // 取数组最小元素

void print(); // 打印数组元素

};

int CArray::length(){

return num;

}

int CArray::get(int i){

return vec[i];

}

void CArray::set(int i, int value){

vec[i]=value;

}

int CArray::getMax(){

int max=vec[0];

for(int i=1;i<num;i++)

{if(vec[i]>max)

max=vec[i];}

return max;

}

int CArray::getMin(){

int min=vec[0];

for(int i=1;i<num;i++)

{if(vec[i]<min)

min=vec[i];}

return min;

}

void CArray::print(){

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

cout<<vec[i]<<" ";

cout<<endl;

}

CArray::~CArray()

{

if(vec!=NULL)

delete[] vec;

}

CArray::CArray(int n):num(n)

{

vec=new int[num];

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

vec[i]=0;

}

CArray::CArray(CArray &arr):num(arr.num)

{

vec=new int[num];

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

vec[i]=arr.vec[i];

}

int main()

{

CArray a1(5);

a1.print();

int n=a1.length();

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

a1.set(i,rand()%90+10);

CArray a2=a1;

a1.print();

a2.print();

a1.set(3,100);

a1.print();

a2.print();

cout<<a1.getMax()<<endl;

cout<<a1.getMin()<<endl;

cout<<a2.getMax()<<endl;

cout<<a2.getMin()<<endl;

}