作业（理论课）

第二次作业

1.【问题描述】

设计并测试一个名为Rectangle的矩形类，其属性为矩形的左下角与右上角两个点的坐标，根据坐标能计算矩形的面积。

【输入形式】

程序参考的输入（提示“Input Point A:”输入左下角的坐标；提示“Input Point B:”输入左下角的坐标）：

Input Point A:0 0

Input Point B:8.2 4.5  
【输出形式】

程序参考的输出：

Rectangle Area:36.9  
【样例输入】

Input Point A:0 0

Input Point B:8.2 4.5  
【样例输出】

Rectangle Area:36.9

【样例说明】  
【评分标准】

代码部分：

#include <iostream>

using namespace std;

class rectangle

{

private:

float a,b,c,d,area;

public:

void get()

{

cout<<"Input Point A:";

cin>>a >>b;

cout<<"Input Point B:";

cin>>c >>d;

}

float mj(float a,float b,float c,float d)

{

area=(d-b)\*(c-a);

return area;

}

void show()

{

cout<<"Rectangle Area:"<<mj(a,b,c,d);

}

};

int main()

{

rectangle mr;

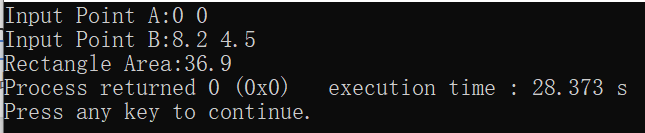
mr.get();

mr.show();

return 0;

}

运行截图：



2.【问题描述】

定义一个复数类Complex，使得下面的代码能够工作。（注：下列代码需放在主函数中。）

Complex c1(3,5);    //用复数3+5i初始化c1

Complex c2=4.5;     //用实数4.5初始化c2

c1.add(c2);   //将c1与c2相加，结果保存在c1中

c1.show(); //将c1输出（这时的结果应该是7.5+5i）  
【输入形式】

无  
【输出形式】

7.5+5i

【样例输入】  
【样例输出】

7.5+5i

【样例说明】  
【评分标准】

代码部分：

#include <iostream>

using namespace std;

class complex

{

private:

float a,b;

public:

complex(float a1,float b1);//构造函数

complex(float a1);

void add(complex &c);

void show();

};

complex::complex(float a1,float b1)

{

a=a1;

b=b1;

}

complex::complex(float a1)

{

a=a1;

b=0;

}

void complex::add(complex &c)

{

a+=c.a;

b+=c.b;

}

void complex::show()

{

cout<<a<<"+"<<b<<"i"<<endl;

}

int main()

{

complex c1(3,5);

complex c2=4.5;

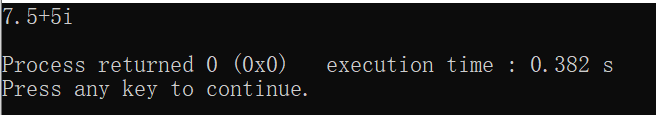
c1.add(c2);

c1.show();

return 0;

}

运行截图：



第三次作业

1.【问题描述】

下面程序的运行结果是什么？实际运行一下，看看与你的设想有何不同。

注：只需调试一下代码即可，原样输入。

#include <iostream>

using namespace std;

int x=5,y=7;

void myFunction(){

    int y=10;

    cout<<"x from myFunction:"<<x<<endl;

    cout<<"y from myFunction:"<<y<<endl;

}

int main()

{

    cout<<"x from main:"<<x<<endl;

    cout<<"y from main:"<<y<<endl;

    myFunction();

    cout<<"Back from myFunction!"<<endl;

    cout<<"x from main:"<<x<<endl;

    cout<<"y from main:"<<y<<endl;

    return 0;

}

【输入形式】  
【输出形式】  
【样例输入】  
【样例输出】  
【样例说明】  
【评分标准】

代码部分：

#include <iostream>

using namespace std;

int x=5,y=7;

void myFunction(){

int y=10;

cout<<"x from myFunction:"<<x<<endl;

cout<<"y from myFunction:"<<y<<endl;

}

int main()

{

cout<<"x from main:"<<x<<endl;

cout<<"y from main:"<<y<<endl;

myFunction();

cout<<"Back from myFunction!"<<endl;

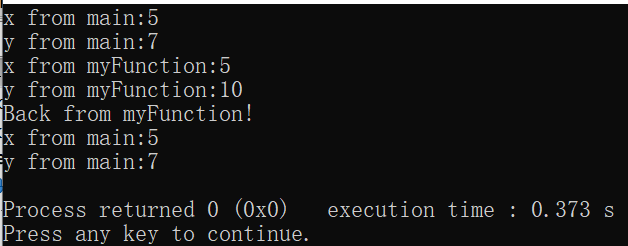
cout<<"x from main:"<<x<<endl;

cout<<"y from main:"<<y<<endl;

return 0;

}

运行截图：



2.【问题描述】

定义一个Cat类，拥有静态数据成员numOfCats，记录Cat的个体数目；静态成员函数getNumOfCats()，读取numOfCats。设计程序测试这个类，体会静态数据成员和静态成员函数的用法。

主函数如下：

int main()

{

    Cat c;

    cout<<"Num:"<<Cat::getNumOfCats()<<endl;

    Cat carr[4];

    cout<<"Num:"<<Cat::getNumOfCats()<<endl;

    return 0;

}

参考的输出结果为：

Num:1

Num:5  
【输入形式】

无  
【输出形式】

Num:1

Num:5

代码部分：

#include <iostream>

using namespace std;

class Cat

{

private:

static int numOfCats;

public:

Cat()

{

numOfCats++;

}

~Cat()

{

numOfCats--;

}

static int getNumOfCats()

{

return numOfCats;

}

};

int Cat::numOfCats = 0;

int main()

{

Cat c;

cout<<"Num:"<<Cat::getNumOfCats()<<endl;

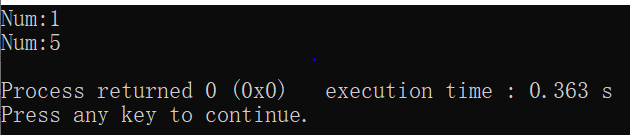
Cat carr[4];

cout<<"Num:"<<Cat::getNumOfCats()<<endl;

return 0;

}

运行截图：



3.【问题描述】

已知屏幕上的点可以构成三角形（不考虑一条线上的情况），定义点类Point，和三角形类Triangle。Point类的对象作为Triangle类的成员，补充完成下列代码。

（提示：可以用海伦公式计算面积，如三边长为a,b,c，那么s=(a+b+c)/2,area=sqrt(s\*(s-a)\*(s-b)\*(s-c))）

【输入形式】

程序的参考输入：

Input Point:0 0

Input Point:3 0

Input Point:0 4

【输出形式】

程序的参考输出：

Copy Point!

Copy Point!

Copy Point!

Triangle Area:6

其他参考输入输出：

Input Point:0 0

Input Point:0 3.3

Input Point:4.4 0

Copy Point!

Copy Point!

Copy Point!

Triangle Area:7.26

代码部分：

#include <iostream>

#include <cmath>

using namespace std;

class Point

{

public:

Point()

{

cout<<"Input Point:";

cin>>x>>y;

}

Point(const Point&p)

{

x=p.x;

y=p.y;

cout<<"Copy Point!"<<endl;

}

float getX(){return x;}

float getY(){return y;}

void setX(float a){x=a;}

void setY(float b){y=b;}

private:

float x,y;

};

class Triangle

{

public:

Triangle(Point &xp1,Point &xp2,Point &xp3);

float getArea();

private:

Point p1,p2,p3;

};

Triangle::Triangle(Point &xp1,Point &xp2,Point &xp3):

p1(xp1),p2(xp2),p3(xp3)

{}

float Triangle::getArea()

{

float s1,s2,s3;

s1=sqrt((p1.getX()-p2.getX())\*(p1.getX()-p2.getX())+(p1.getY()-p2.getY())\*(p1.getY()-p2.getY()));

s2= sqrt((p3.getX()-p2.getX())\*(p3.getX()-p2.getX())+(p3.getY()-p2.getY())\*(p3.getY()-p2.getY()));

s3= sqrt((p1.getX()-p3.getX())\*(p1.getX()-p3.getX())+(p1.getY()-p3.getY())\*(p1.getY()-p3.getY()));

float p,area;

p=(s1+s2+s3)/2.0;

area=sqrt(p\*(p-s1)\*(p-s2)\*(p-s3));

return area;

}

int main()

{

Point p1,p2,p3;

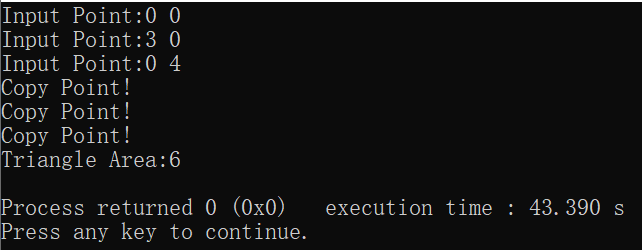
Triangle t(p1,p2,p3);

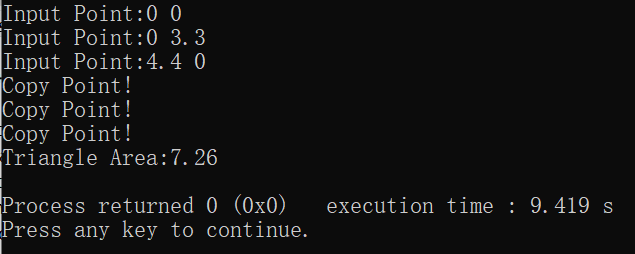
cout<<"Triangle Area:"<<t.getArea()<<endl;

return 0;

}

运行截图：





第四次作业

1.【问题描述】

定义一个共享信用卡类CCard，每张卡包含卡号id和密码pass，所有该信用卡类产生的对象共享信用卡额度money，有还款void deposit(double m)和取款二种方法void draw(double m)。显示额度方法void showMoney()定义成静态成员函数。【输入形式】

分别提示输入密码2张卡的密码，正确则取款，不正确则提示错误。

Input password:111111

【输出形式】

先输出信用额度，再提示输入密码，取款失败，输出提示。

Money:20000

完整的输入输出例子：

Money:20000

Input password:111111

Money:19200

Input password:111111

Money:18300

【样例输入与输出】

Money:20000

Input password:111111

Money:19200

Input password:222222

Password error!

Money:19200

代码部分：

#include <iostream>

#include <string>

using namespace std;

class CCard

{

private:

string id,pass;

static double money;

int flat=1;

public:

CCard(){};

CCard(string idname,string password)

{

cout<<"Input password:";

cin>>password;

if(password!="111111")

{

cout<<"Password error!";

flat=0;

}

}

void deposit(double m)

{

money=money-m;

}

void draw(double m)

{

if(flat!=0)

money=money-m;

else cout<<endl;

}

static void showMoney()

{

cout<<"Money:"<<money<<endl;

}

};

double CCard::money = 20000;

int main()

{

CCard::showMoney();

CCard card1("666666","111111");

card1.draw(800);

CCard::showMoney();

CCard card2("888888","111111");

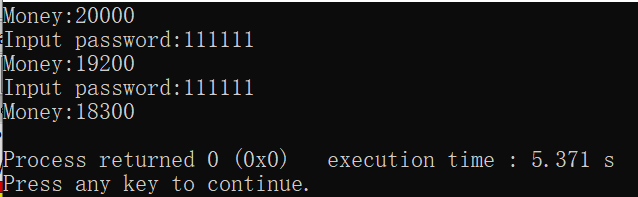
card2.draw(900);

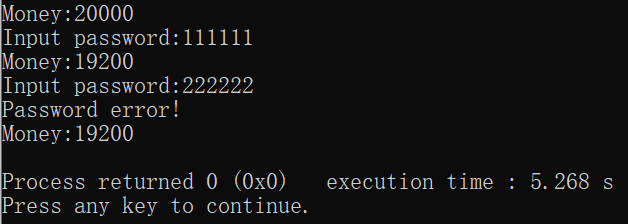
CCard::showMoney();

return 0;

}

运行截图：





2.【问题描述】

实现一个名为SimpleCircle的简单圆类。其数据成员int \*itsRadius为一个指向其半径值的指针，存放其半径值。设计对数据成员的各种操作并计算面积，给出这个类的完整实现并测试这个类。

注：简单计算半径为10的圆面积，不需要输入，圆周率取3.14。

参考的输出如下：

Area:314  
【输入形式】

无  
【输出形式】

Area:314

代码部分：

#include <iostream>

using namespace std;

const float pi=3.14;

class SimpleCircle

{

public:

SimpleCircle() {};

void setRadius(float r)

{

itsRadius=r;

}

float getRadius()

{

return itsRadius;

}

float getArea()

{

return (itsRadius)\*(itsRadius)\*pi;

}

private:

float itsRadius;

};

int main()

{

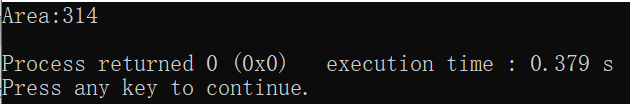
SimpleCircle c;

c.setRadius(10);

cout<<"Area:"<<c.getArea()<<endl;

}

运行截图：



第五次作业

1.问题描述】

定义一个基类Shape，在此基础上派生出Rectangle和Circle，二者都有getArea()函数计算对象面积。使用Rectangle类创建一个派生类Square。

注：圆周率取3.14

【输入形式】

参考的输入（数字前面的文字为提示）：

Input a,b:5 10

Input r:10

【输出形式】

参考的输出：

Rectangle Area:50,Circle Area:314

代码部分：

#include <iostream>

using namespace std;

#define PI 3.14

class Shape

{

public:

Shape(){}

~Shape(){}

virtual float getArea()=0;

};

class Rectangle:public Shape

{

public:

Rectangle(int a,int b):k(a),j(b){}

protected:

int k,j;

float getArea()

{

return k\*j;

}

};

class Circle:public Shape

{

public:

Circle(int r):radious(r){}

~Circle(){}

private:

int radious;

float getArea()

{

return PI\*radious\*radious;

}

};

class Square:public Rectangle

{

Square();

};

int main()

{

int a,b,r;

Shape \*m;

cout<<"Input a,b:";

cin>>a>>b;

cout<<"Input r:";

cin>>r;

m=new Rectangle(a,b);

cout<<"Rectangle Area:"<<m->getArea()<<",";

delete m;

m=new Circle(r);

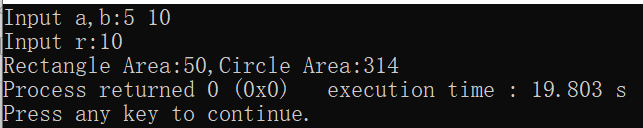
cout<<"Circle Area:"<<m->getArea();

delete m;

return 0;

}

运行截图：



2.【问题描述】

定义Boat与Car两个类，二者都有weight属性，定义二者的一个友元函数getTotalWeight()，计算二者的重量和。  
【输入形式】

参考的输入（数字前为提示文字）：

Input boat weight:3

Input car weight:5  
【输出形式】

参考的输出：

Total weight:8  
【样例输入】

Input boat weight:3

Input car weight:5

【样例输出】

Total weight:8

代码部分：

#include <iostream>

using namespace std;

class Car;

class Boat

{

private:

float boat;

public:

Boat(float \_boat = 0){ boat = \_boat; }

Boat(Boat &b);

float getB(){ return boat; }

friend int totalWeight(Boat b, Car c);

};

class Car

{

private:

float car;

public:

Car(float \_car = 0){ car = \_car; }

Car(Car &c);

float getC(){ return car; }

friend int totalWeight(Boat b, Car c);

};

Boat::Boat(Boat & b)

{

boat = b.boat;

}

Car::Car(Car & c)

{

car = c.car;

}

int totalWeight(Boat b, Car c)

{

float boat=b.boat ;

float car=c.car ;

return boat + car;

}

int main()

{

float boat;

float car;

cout << "Input boat weight:";

cin >> boat ;

cout<<"Input car weight:";

cin>>car;

Boat theb(boat);

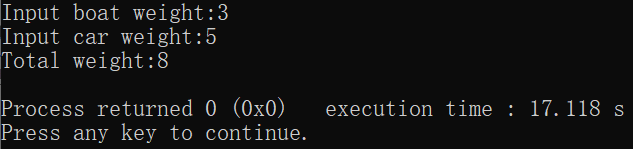
Car thec(car);

cout << "Total weight:"<< totalWeight(theb,thec) << endl;

return 0;

}

运行截图：



3.【问题描述】

定义类X,Y,Z,函数h(X\*),满足：类X有私有成员i，Y的成员函数g(X\*)是X的友元函数，实现对X的成员i加1；类Z是类X的友元类，其成员函数f(X\*)实现对X的成员i加5；函数h(X\*)是X的友元函数，实现对X的成员i加10。主函数实现测试。主函数需要定义如下：

int main()

{

    int a;

    cin>>a;

    X x(a);

    Y y;

    Z z;

    y.g(x);

    z.f(x);

    h(x);

    x.print();

}  
【输入形式】

参考的输入：

3  
【输出形式】

参考的输出：

19  
【样例输入】

3  
【样例输出】

19

代码部分：

#include <iostream>

using namespace std;

class X;

class Y

{

public:

void g(X &x);

};

class Z

{

public:

void f(X &x);

};

class X

{

public:

X(int i):i(i){}

friend void Y::g(X &x);

friend class Z;

friend void h(X &x);

void print()

{

cout<<i;

}

private:

int i;

};

void Y::g(X &x)

{

x.i+=1;

}

void Z::f(X &x)

{

x.i+=5;

}

void h(X &x)

{

x.i+=10;

}

int main()

{

int a;

cin>>a;

X x(a);

Y y;

Z z;

y.g(x);

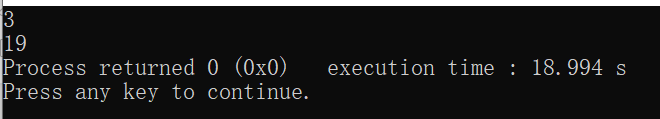
z.f(x);

h(x);

x.print();

}

运行截图：



第六次作业

1.【问题描述】

定义一个Document类，有数据成员name，从Document派生出Book类，增加数据成员pageCount。

【输入形式】

程序参考的输入（数据前面文字为提示）：

Input Name and Page:Math 280  
【输出形式】

程序参考的输出：

Name:Math

Page:280  
【样例输入】

Input Name and Page:Math 280

【样例输出】

Name:Math

Page:280

代码部分：

#include <iostream>

#include <string>

using namespace std;

class document

{

protected:

string name;

public:

document() {};

};

class book:public document

{

private:

int pagecount;

public:

book()

{

cout<<"Input Name and Page:";

cin>>name>>pagecount;

cout<<"Name:"<<name<<endl;

cout<<"Page:"<<pagecount;

}

};

int main(){

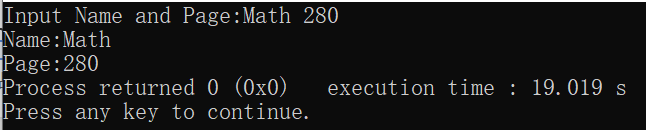
document a;

book b;

return 0;

}

运行截图：



2.【问题描述】

定义一个Object类，有数据成员weight及相应的操作函数，由此派生出Box类，增加数据成员height和width及相应的操作函数，声明一个Box对象，观察构造函数和析构函数的调用顺序。

【输入输出形式】

参考的输入及输出（其中Input开头的两段文字为输入提示和输入数据，其他均为输出）：

Constructing Object!

Constructing Box!

Input weight:50

Input height and width:20 30

Weight:50

Height and width:20,30

Destructing Box!

Destructing Object!

代码部分：

#include <iostream>

#include <string>

using namespace std;

class Object

{

public:

Object()

{

cout<<"Constructing Object!"<<endl;

}

~Object()

{

cout<<"Destructing Object!"<<endl;

}

void getInfo()

{

cout<<"Input weight:";

cin>>weight;

}

void showInfo()

{

cout<<"Weight:"<<weight<<endl;

}

private:

int weight;

};

class Box:public Object

{

private:

int height,width,weight;

public:

Box()

{

cout<<"Constructing Box!"<<endl;

}

~Box()

{

cout<<"Destructing Box!"<<endl;

}

void getInfo()

{

cout<<"Input weight:";

cin>>weight;

cout<<"Input height and width:";

cin>>height>>width;

}

void showInfo()

{

cout<<"Weight:"<<weight<<endl;

cout<<"Height and width:"<<height<<","<<width<<endl;

}

};

int main()

{

Box box;

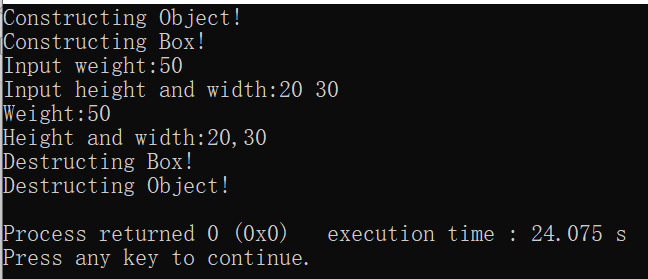
box.getInfo();

box.showInfo();

return 0;

}

运行截图：



第七次作业

1.【问题描述】

思考例7-6和例7-8中Derived类的各个数据成员在Derived对象中存放的位置，编写程序输出它们各自的地址来验证自己的判断。

录入下面的代码，自行分析输出结果。

//7\_8.cpp

#include <iostream>

using namespace std;

class Base0 {    //定义基类Base0

public:

    int var0;

    void fun0() { cout << "Member of Base0" << endl; }

};

class Base1: virtual public Base0 {    //定义派生类Base1

public:    //新增外部接口

    int var1;

};

class Base2: virtual public Base0 {    //定义派生类Base2

public:    //新增外部接口

    int var2;

};

class Derived: public Base1, public Base2 {    //定义派生类Derived

public:    //新增外部接口

    int var;

    void fun() { cout << "Member of Derived" << endl; }

};

int main() {    //程序主函数

    Derived d;    //定义Derived类对象d

    d.var0 = 2;    //直接访问虚基类的数据成员

    d.fun0();    //直接访问虚基类的函数成员

    if(&d.Base1::var0==&d.Base2::var0)

        cout<<"true"<<endl;

    else

        cout<<"false"<<endl;

    return 0;

}

代码部分：

#include <iostream>

using namespace std;

class Base0 { //定义基类Base0

public:

int var0;

void fun0() { cout << "Member of Base0" << endl; }

};

class Base1 : virtual public Base0 { //定义派生类Base1

public: //新增外部接口

int var1;

};

class Base2 : virtual public Base0 { //定义派生类Base2

public: //新增外部接口

int var2;

};

class Derived : public Base1, public Base2 { //定义派生类Derived

public: //新增外部接口

int var;

void fun() { cout << "Member of Derived" << endl; }

};

int main() { //程序主函数

Derived d; //定义Derived类对象d

d.var0 = 2; //直接访问虚基类的数据成员

d.fun0(); //直接访问虚基类的函数成员

if (&d.Base1::var0 == &d.Base2::var0)

cout << "true" << endl;

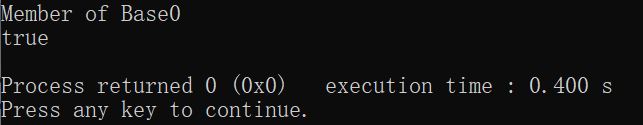
else

cout << "false" << endl;

return 0;

}

运行截图：



2.【问题描述】

下面的程序能得到预期的结果吗？如何避免类似问题的发生？

请修改程序，使它得到我们想要的结果。即希望两个输出的值相同。

提示：struct在C++中也可以用来定义类，它与class的不同在于struct定义的类成员的默认访问权限是public。

#include <iostream>

using namespace std;

struct Base1 {int x;};

struct Base2 {float y;};

struct Derived : Base1,Base2{};

int main()

{

    Derived \*pd=new Derived;

    pd->x=1;pd->y=2.0f;

    void \*pv = pd;

    Base2 \*pb=static\_cast<Base2\*>(pv);

    cout<<pd->y<<" "<<pb->y<<endl;

    delete pd;

    return 0;

}

【输入形式】

无  
【输出形式】

2 2

代码部分：

#include <iostream>

using namespace std;

struct Base1 { int x; };

struct Base2 { float y; };

struct Derived : Base1, Base2 {};

int main()

{

Derived\* pd = new Derived;

pd->x = 1; pd->y = 2.0f;

void\* pv = pd;

Base2\* pb = static\_cast<Base2\*>(pv);

pb = pd;

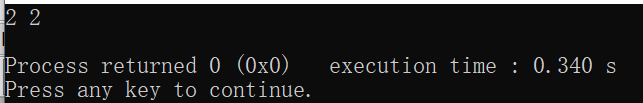
cout << pd->y << " " << pb->y << endl;

delete pd;

return 0;

}

运行截图：



3.【问题描述】

定义一个基类BaseClass，从它派生出类DerivedClass。BaseClass有成员函数fn1()，fn2()，DerivedClass也有成员函数fn1()，fn2()。在主函数中声明一个DerivedClass的对象，分别用DerivedClass的对象以及BaseClass和DerivedClass的指针来调用fn1()，fn2()，观察运行结果。

主函数如下：

int main()

{

    DerivedClass dc;

    BaseClass \*pb = &dc;

    DerivedClass \*pd = &dc;

    dc.fn1();

    dc.fn2();

    pb->fn1();

    pb->fn2();

    pd->fn1();

    pd->fn2();

}

【输入形式】

无  
【输出形式】

程序的输出结果为：

DerivedClass:fn1()

DerivedClass:fn2()

BaseClass:fn1()

BaseClass:fn2()

DerivedClass:fn1()

DerivedClass:fn2()

代码部分：

#include <iostream>

using namespace std;

class BaseClass

{

public:

BaseClass(){}

void fn1()

{

cout<<"BaseClass:fn1()"<<endl;

}

void fn2()

{

cout << "BaseClass:fn2()"<<endl;

}

};

class DerivedClass :public BaseClass

{

public:

void fn1()

{

cout<<"DerivedClass:fn1()"<<endl;

}

void fn2()

{

cout<<"DerivedClass:fn2()"<<endl;

}

};

int main()

{

DerivedClass dc;

BaseClass \*pb = &dc;

DerivedClass \*pd = &dc;

dc.fn1();

dc.fn2();

pb->fn1();

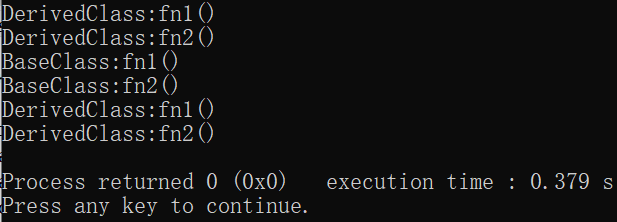
pb->fn2();

pd->fn1();

pd->fn2();

}

运行截图：



第八次作业

1.【问题描述】

设计一个类，用自己的成员函数重载运算符，使对整型的运算符=、+、-、\*、/ 适用于分数运算。要求：

（1）输出结果是最简分数（可以是带分数）；

（2）分母为1，只输出分子。

参考的输入输出：

Input x: 1/6

Input y: 2/9

x+y=7/18

x-y=-1/18

x\*y=1/27

x/y=3/4

【输入形式】

提示“Input x: ”，输入第一个分数。如：1/6

提示“Input y: ”，输入第二个分数。如：2/9

【输出形式】

提示并输出+、-、\*、/的结果，如

x+y=7/18

x-y=-1/18

x\*y=1/27

x/y=3/4

【样例输入】

Input x: 1/6

Input y: 2/9

【样例输出】

x+y=7/18

x-y=-1/18

x\*y=1/27

x/y=3/4

代码部分：

#include <iostream>

#include <math.h>

using namespace std;

class CFraction

{

private:

int nume; // 分子

int deno; // 分母

public:

CFraction(int nu = 0, int de = 1) :nume(nu), deno(de) {}

void simplify();

//输入输出的重载

friend istream& operator>>(istream& in, CFraction& x);

friend ostream& operator<<(ostream& out, CFraction x);

CFraction operator+(const CFraction& c); //两个分数相加，结果要化简

CFraction operator-(const CFraction& c); //两个分数相减，结果要化简

CFraction operator\*(const CFraction& c); //两个分数相乘，结果要化简

CFraction operator/(const CFraction& c); //两个分数相除，结果要化简

CFraction operator+(); //取正一目运算

CFraction operator-(); //取反一目运算

CFraction operator~(); //取倒数一目运算

bool operator>(const CFraction& c);

bool operator<(const CFraction& c);

bool operator==(const CFraction& c);

bool operator!=(const CFraction& c);

bool operator>=(const CFraction& c);

bool operator<=(const CFraction& c);

};

// 分数化简

void CFraction::simplify()

{

int m, n, r;

n = fabs(deno);

m = fabs(nume);

while (r = m % n) // 求m，n的最大公约数

{

m = n;

n = r;

}

deno /= n; // 化简

nume /= n;

if (deno < 0) // 将分母转化为正数

{

deno = -deno;

nume = -nume;

}

}

// 重载输入运算符>>

istream& operator>>(istream& in, CFraction& x)

{

char ch;

while (1)

{

cin >> x.nume >> ch >> x.deno;

if (x.deno == 0)

cerr << "分母为0, 请重新输入\n";

else if (ch != '/')

cerr << "格式错误（形如m/n）! 请重新输入\n";

else

break;

}

return cin;

}

// 重载输出运算符<<

ostream& operator<<(ostream& out, CFraction x)

{

if (x.deno == 1)

cout << x.nume;

else

cout << x.nume << '/' << x.deno;

return cout;

}

// 分数相加

CFraction CFraction::operator+(const CFraction& c)

{

CFraction t;

t.nume = nume \* c.deno + c.nume \* deno;

t.deno = deno \* c.deno;

t.simplify();

return t;

}

// 分数相减

CFraction CFraction:: operator-(const CFraction& c)

{

CFraction t;

t.nume = nume \* c.deno - c.nume \* deno;

t.deno = deno \* c.deno;

t.simplify();

return t;

}

// 分数相乘

CFraction CFraction:: operator\*(const CFraction& c)

{

CFraction t;

t.nume = nume \* c.nume;

t.deno = deno \* c.deno;

t.simplify();

return t;

}

// 分数相除

CFraction CFraction:: operator/(const CFraction& c)

{

CFraction t;

if (!c.nume) return \*this; //除法无效（除数为）时，这种情况需要考虑，但这种处理仍不算合理

t.nume = nume \* c.deno;

t.deno = deno \* c.nume;

t.simplify();

return t;

}

// 分数取正号

CFraction CFraction:: operator+()

{

return \*this;

}

// 分数取负号

CFraction CFraction:: operator-()

{

CFraction x;

x.nume = -nume;

x.deno = deno;

return x;

}

// 分数取倒数

CFraction CFraction:: operator~()

{

CFraction x;

x.nume = deno;

x.deno = nume; //未对原分子为0的情况进行处理

if (x.deno < 0) //保证负分数的负号在分子上

{

x.deno = -x.deno;

x.nume = -x.nume;

}

return x;

}

// 分数比较大小

bool CFraction::operator>(const CFraction& c)

{

int this\_nume, c\_nume, common\_deno;

this\_nume = nume \* c.deno; // 计算分数通分后的分子，同分母为deno\*c.deno

c\_nume = c.nume \* deno;

common\_deno = deno \* c.deno;

if ((this\_nume - c\_nume) \* common\_deno > 0) return true;

return false;

}

// 分数比较大小

bool CFraction::operator<(const CFraction& c)

{

int this\_nume, c\_nume, common\_deno;

this\_nume = nume \* c.deno;

c\_nume = c.nume \* deno;

common\_deno = deno \* c.deno;

if ((this\_nume - c\_nume) \* common\_deno < 0) return true;

return false;

}

// 分数比较大小

bool CFraction::operator==(const CFraction& c)

{

if (\*this != c) return false;

return true;

}

// 分数比较大小

bool CFraction::operator!=(const CFraction& c)

{

if (\*this > c || \*this < c) return true;

return false;

}

// 分数比较大小

bool CFraction::operator>=(const CFraction& c)

{

if (\*this < c) return false;

return true;

}

// 分数比较大小

bool CFraction::operator<=(const CFraction& c)

{

if (\*this > c) return false;

return true;

}

int main()

{

CFraction x, y, s;

cout << "Input x: ";

cin >> x;

cout << "Input y: ";

cin >> y;

s = +x + y;

cout << "x+y=" << s << endl;

s = x - y;

cout << "x-y=" << s << endl;

s = x \* y;

cout << "x\*y=" << s << endl;

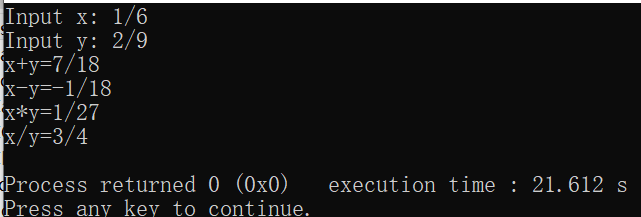
s = x / y;

cout << "x/y=" << s << endl;

return 0;

}

运行截图：



2.【问题描述】

对类Point重载“++”（自增）、“--”（自减）运算符，要求同时重载前缀和后缀。

使得下列主函数能够正确运行：

int main()

{

    Point a,b(5,5);

    a=b++;

    a.display();

    a=++b;

    a.display();

    a=--b;

    a.display();

    a=b--;

    a.display();

}

【输入形式】

无  
【输出形式】

参考的输出结果如下（没有输入）：

(5,5)

(7,7)

(6,6)

(6,6)

代码部分：

#include <iostream>

using namespace std;

class Point

{

public:

Point(int x = 0, int y = 0)

{

this->x = x;

this->y = y;

}

Point operator++ (int);

Point operator-- (int);

Point& operator++ ();

Point& operator-- ();

void display();

private:

int x;

int y;

};

Point Point::operator++ (int)

{

Point t = \*this;

++\*this;

return t;

}

Point Point::operator-- (int)

{

Point t = \*this;

--\*this;

return t;

}

Point& Point::operator++ ()

{

x++;

y++;

return \*this;

}

Point& Point::operator-- ()

{

x--;

y--;

return \*this;

}

void Point::display()

{

cout << "(" << x<< "," << y << ")" << endl;

}

int main()

{

Point a,b(5,5);

a=b++;

a.display();

a=++b;

a.display();

a=--b;

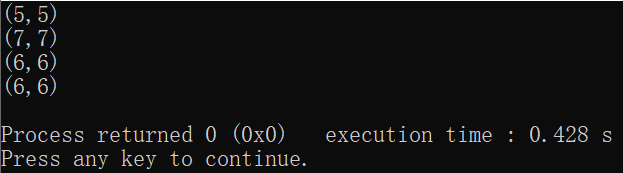
a.display();

a=b--;

a.display();

}

运行截图：



3.【问题描述】

编写一个哺乳动物类Mammal，再由此派生出狗类Dog，二者都声明speak()成员函数，该函数在基类中被声明为虚函数。 声明一个Dog类的对象，通过此对象调用speak函数，声明一个Mammal类的指针指向Dog类的对象，通过指针调用speak函数。

提示：下列主函数代码已经有了。

#include <iostream>

using namespace std;

/\* 你需要补充代码 \*/

int main()

{

    Dog dog;

    Mammal \*p;

    dog.speak();

    p=&dog;

    p->speak();

    return 0;

}

【输入形式】

程序没有输入：

【输出形式】

程序的输出结果如下：

Dog is speaking!

Dog is speaking!

代码部分：

#include <iostream>

using namespace std;

class Mammal

{

public:

Mammal(){}

void virtual speak(){}

};

class Dog:public Mammal

{

public:

Dog(){}

void virtual speak()

{

cout<<"Dog is speaking!"<<endl;;

}

};

int main()

{

Dog dog;

Mammal \*p;

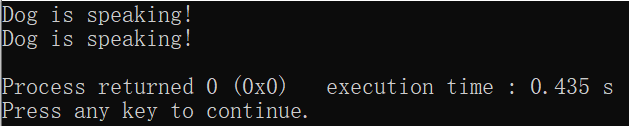
dog.speak();

p=&dog;

p->speak();

}

运行截图：



4.【问题描述】

定义一个基类BaseClass，从它派生出类DerivedClass。BaseClass有成员函数fn1()，fn2()。fn1()是虚函数；DerivedClass也有成员函数fn1()，fn2()。在主函数中声明一个DerivedClass类的对象，分别用BaseClass和DerivedClass的指针指向DerivedClass的对象，并通过指针调用fn1()，fn2()，观察运行结果。

提示：主函数如下：

int main()

{

    DerivedClass dc;

    BaseClass \*pb = &dc;

    DerivedClass \*pd = &dc;

    pb->fn1();

    pb->fn2();

    pd->fn1();

    pd->fn2();

    return 0;

}

【输入形式】

无  
【输出形式】

程序运行结果如下：

DerivedClass:fn1()

BaseClass:fn2()

DerivedClass:fn1()

DerivedClass:fn2()

代码部分：

#include <iostream>

using namespace std;

class BaseClass

{

public:

BaseClass(){}

void virtual fn1()

{

cout<<"BaseClass:fn1()"<<endl;

}

void fn2()

{

cout<<"BaseClass:fn2()"<<endl;

}

};

class DerivedClass:public BaseClass

{

public:

DerivedClass(){}

void virtual fn1()

{

cout<<"DerivedClass:fn1()"<<endl;

}

void fn2()

{

cout<<"DerivedClass:fn2()"<<endl;

}

};

int main()

{

DerivedClass dc;

BaseClass \*pb = &dc;

DerivedClass \*pd = &dc;

pb->fn1();

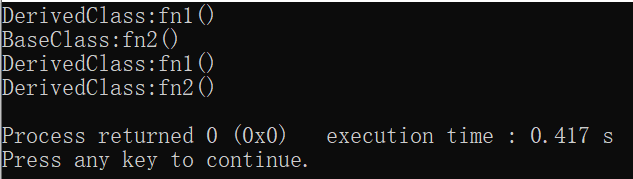
pb->fn2();

pd->fn1();

pd->fn2();

}

运行截图：



第九次作业

1.【问题描述】

定义一个时间类CTime，分钟和秒钟是其两个私有成员数据。输入一个起始时间和一个结束时间(起始时间早于结束时间)，通过运算符重载-（减号），计算这两个时间相隔多少秒钟。说明：这两个时间在同一小时之内，且采用60分钟60秒钟的计时分式，即从00:00-59:59。

【输入形式】

测试输入包含若干测试用例，每个测试用例占一行。每个测试用例包括四个数，每个数之间用空格间隔，每个数都由两位数字组成，第一个数和第二个数分别表示起始时间的分钟和秒钟，第三个数和第四个数分别表示结束时间的分钟和秒钟。当读入一个测试用例是00 00 00 00时输入结束，相应的结果不要输出。

【输出形式】

对每个测试用例输出一行。输出一个数即可，表示两者之间间隔的秒钟数。

【样例输入】

12 11 12 58

00 13 16 00

09 07 23 59

00 00 00 00

【样例输出】

47

947

892

代码部分：

#include <iostream>

using namespace std;

#define N 100

class CTime

{

public:

CTime(){}

friend int operator-(CTime, CTime);

void input()

{

cin >> minute >> second ;

}

int beZero()

{

if (minute == 00 && second == 00)

return 1;

else return 0;

}

private:

int minute;

int second;

};

int operator-(CTime t2, CTime t1)

{

return (t2.minute - t1.minute) \* 60 + t2.second - t1.second;

}

int main()

{

CTime time[N];

int count=-1;

do

{

count++;

time[2\*count].input();

time[2\*count+1].input();

}while(!(time[2\*count].beZero()&&time[2\*count+1].beZero()));

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

{

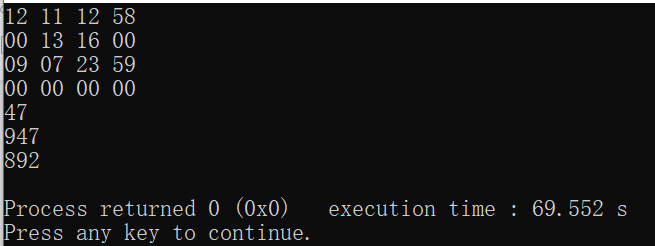
cout<<time[2\*i+1]-time[2\*i]<<endl;

}

return 0;

}

运行截图：



第十次作业

1.【问题描述】

录入下面程序，分析编译错误信息。并提交改正后的程序。

#include <iostream>

#include <new>

#include <assert.h>

using namespace std;

class Abstract

{

public:

    Abstract()

    {

        cout << "in Abstract()"<<endl;

    }

    virtual void f() = 0;

};

int main()

{

    Abstract \*p = new Abstract;

    p->f();

    return 0;

}

【输入形式】

无  
【输出形式】

程序的参考输出为：

in Abstract()

in Subclass()

Subclass::f()

代码部分：

#include <iostream>

#include <new>

#include <assert.h>

using namespace std;

class Abstract//抽象类:不能定义对象

{

public:

Abstract()

{

cout << "in Abstract()" << endl;

}

virtual void f() = 0;//纯虚函数

};

class Subclass :public Abstract

{

public:

Subclass()

{

cout << "in Subclass()" << endl;

}

virtual void f()

{

cout << "Subclass::f()";

}

};

int main()

{

Abstract \*p=new Subclass;

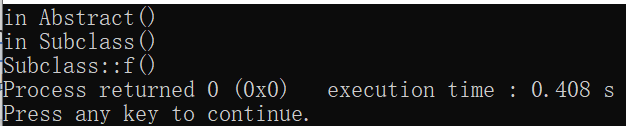
p->f();

delete p;

return 0;

}

运行截图：



2.【问题描述】

定义一个抽象类Shape类表示图形类，有输入数据函数input()，计算面积float getArea()和计算周长float getPerim()函数，在此基础上派生矩形类Rectangle和圆类Circle。圆周率取3.14。

【输入形式】

输入圆半径和矩形的长宽。  
【输出形式】

输出面积和周长。  
【样例输入】

input r:10

input l and w:20 10  
【样例输出】

Area:314 Perim:62.8

Area:200 Perim:60

代码部分：

#include <iostream>

using namespace std;

#define PI 3.14;

class Shape

{

public:

virtual void input() = 0;

virtual double getArea() = 0;

virtual double getPerim()= 0;

protected:

double r;

double l, w;

};

class Circle :public Shape

{

public:

Circle(){}

void input()

{

cout << "input r:";

cin >> r;

}

double getArea()

{

return r \* r \* PI;

}

double getPerim() {

return 2 \* 3.14 \* r;

}

};

class Rectangle :public Shape

{

public:

void input()

{

cout << "input l and w:";

cin >> l >> w;

}

Rectangle(){}

double getArea()

{

return l \* w;

}

double getPerim()

{

return 2 \*( l +w);

}

};

int main()

{

Circle a;

Rectangle b;

Shape \*ptr[2]={&a,&b};

int i;

for(i=0;i<2;i++)

ptr[i]->input();

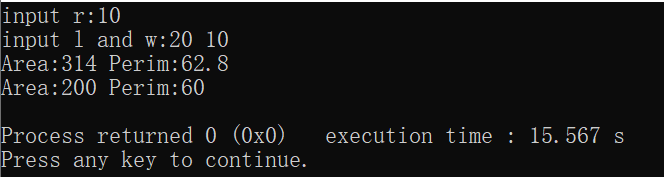
for(i=0;i<2;i++)

cout<<"Area:"<<ptr[i]->getArea()<<" Perim:"<<ptr[i]->getPerim()<<endl;

return 0;

}

运行截图：



第十一次作业

1.【问题描述】

初始化int类型数组data[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20}，应用本章的直接插入排序函数模板进行排序。对此函数模板稍做修改，加入输出语句，在每插入一个待排序元素后显示整个数组，观察排序过程中数据的变化，加深对插入排序算法的理解。

【输入形式】

无  
【输出形式】

参考的输出结果（注：只有19行，第1个数不用插入）：

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20

1 2 3 5 7 9 11 13 15 17 19 4 6 8 10 12 14 16 18 20

1 2 3 4 5 7 9 11 13 15 17 19 6 8 10 12 14 16 18 20

1 2 3 4 5 6 7 9 11 13 15 17 19 8 10 12 14 16 18 20

1 2 3 4 5 6 7 8 9 11 13 15 17 19 10 12 14 16 18 20

1 2 3 4 5 6 7 8 9 10 11 13 15 17 19 12 14 16 18 20

1 2 3 4 5 6 7 8 9 10 11 12 13 15 17 19 14 16 18 20

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 19 16 18 20

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 18 20

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

代码部分：

#include <iostream>

using namespace std;

template <typename T>

void insertionSort(T a[], int n) {

for (int i = 1; i < n; i++) {

int j = i;

T temp = a[i];

for (; j != 0 && a[j] < a[j - 1]; j--) {

T swap = a[j];

a[j] = a[j - 1];

a[j - 1] = swap;

}

a[j] = temp;

for (int m = 0; m < n; m++) {

cout << a[m] << " ";

}

cout << endl;

}

}

int main() {

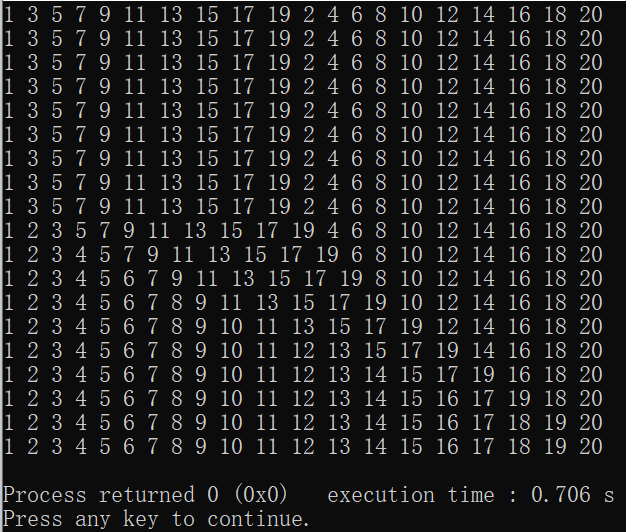
int data[] = { 1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20 };

insertionSort(data, 20);

return 0;

}

运行截图：



2.【问题描述】

编写程序提示用户输入一个班级中的学生人数n，再依次提示用户输入n个人在课程A中的考试成绩，然后计算出平均成绩，显示出来。

请使用本书第9章中的数组类模板Array定义浮点型数组存储考试成绩。

【输入形式】

参考的输入（数据前文字为提示信息）：

Input n:5

Input scores:60 70 80 90 100

【输出形式】

参考的输出：

Average:80

【样例输入】

Input n:5

Input scores:60 70 80 90 100  
【样例输出】

Average:80

代码部分：

#include <iostream>

using namespace std;

int main()

{

int i,n;

float sum=0,avg;

cout<<"Input n:";

cin>>n;

float \*score;

score=new float[n];

cout<<"Input scores:";

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

{

cin>>score[i];

sum+=score[i];

}

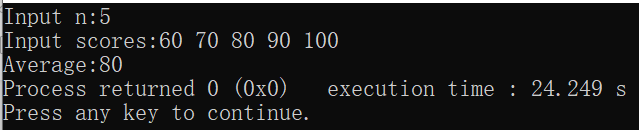
avg=sum/n;

cout<<"Average:"<<avg;

return 0;

}

运行截图：



第十二次作业：

[教材习题10\_8：统计单词曾经出现的次数](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10361');)

【问题描述】

编写一个程序，从键盘上输入一个个单词，每接收到一个单词后，输出该单词是否曾经出现过以及出现次数。可以尝试分别用多重集合（multiset）或映射（map）两种途径实现，将二者进行比较。本题答案你可以任意提交一种方法的代码。

注：输入一个单词并回车，立即输出出现的次数（用times:次数）表示。输入#结束。

参考的输入输出如下（下面单数行为输入，双数行为输出）：

hello

times:0

world

times:0

hello

times:1

hello

times:2

#

代码部分：

#include <iostream>

#include <string>

using namespace std;

#include <set>

int main()

{

multiset<string> arr;

string s;

cin>>s;

while(s!="#")

{

int times=0;

multiset<string>::iterator iter;

for (iter = arr.begin(); iter != arr.end(); iter++)

{

if (\*iter == s)

times++;

}

cout<<"times:"<<times<<endl;

arr.insert(s);

cin>>s;

}

}

（2）

|  |
| --- |
| [教材习题10\_5：约瑟夫问题](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10360');)  【问题描述】  约瑟夫问题：n个骑士编号1，2，...，n，围坐在圆桌旁。编号为1的骑士从1开始报数，报到m的骑士出列，然后下一个位置再从1开始报数，找出最后留在圆桌旁的骑士编号。  （1）编写一个函数模板。以一种顺序容器的类型作为模板参数，在模板中使用指定类型的顺序容器求解约瑟夫问题。m,n是该函数模板的形参。  （2）分别以vector<int>,deque<int>,list<int>作为类型参数调用该函数模板，调用时将n设为较大的数，将m设为较小的数（例如令n=100000,n=5）。观察3种情况下调用该函数模板所需花费的时间。  注：本题答案的提交只需选择一种顺序容器类型作为模板参数。  【输入形式】  程序参考的输入（数字前为提示文字）：  Input n and m:7 3  【输出形式】  程序参考的输出：  Result:4  Input n and m:7 3  Result:4 |

代码部分：

#include<iostream>

#include<vector>

using namespace std;

int main()

{

vector<int> a;

int n,m,x=0;

cout<<"Input n and m:";

cin>>n>>m;

a.resize(n);

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

{

a[i]=i+1;

}

int start=0;

while (a.size() != 1)

{

start = (start + m - 1) % a.size();

a.erase(a.begin() + start);

}

cout<<"Result:"<<a[0]<<endl;

return 0;

}

第十三次作业：

|  |
| --- |
| [教材习题12\_6：修改例9-3的Array类模板并增加异常处理](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10367');)  【问题描述】  修改例9-3的Array类模板，在执行“[]”运算符时，若输入的索引i在有效范围之外，抛出out\_of\_range异常。  编写主函数测试Array类模板：  主函数参考如下：  int main()  {      Array<int> a(10);      try      {          a[10]=5;          cout<<a[10]<<endl;      }      catch(out\_of\_range& e)      {          cout<<e.what()<<endl;      }  }  【输入形式】  无 【输出形式】  程序的输出结果：  Exception:out of range! |

代码：

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

int a;

cout << "Input n:";

cin >> a;

cout << "Dec:" << dec << a;

cout << " Oct:" << oct << a;

cout << " Hex:" << hex << a;

return 0;

}

[教材习题12\_4：设计一个异常类Exception测试范围异常](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10366');)

【问题描述】

设计一个异常类Exception，在此基础上派生一个OutOfMemory类响应内存不住，一个RangeError类响应输入的数不在制定范围内，实现并测试这几个类。

注意：因内存不足服务器不便于测试，故本题提交答案只测试RangeError类。

【输入形式】

正常的输入输出（数字前为提示文字）：

Input n(1<=n<=100):88

【输出形式】

n:88

有异常的输入输出（数字前为提示文字）：

Input n(1<=n<=100):-5

Exception:Range Error!

【样例输入】

Input n(1<=n<=100):-5

【样例输出】

Exception:Range Error!

#include <iostream>

#include <fstream>

using namespace std;

class dog {

public:

dog(int weight, int age) :m\_Weight(weight), m\_Age(age) {};

~dog() {};

int getW() { return m\_Weight; }

int getAge() { return m\_Age; }

private:

int m\_Weight,m\_Age;

};

int main()

{

ofstream out("./dog.txt", ios\_base::out|ios::binary);

dog d1(5, 10);

out << d1.getW() << endl << d1.getAge() << endl;

out.close();

ifstream in("./dog.txt", ios\_base::in|ios::binary);

int w, a;

in >> w >> a;

dog d2(w, a);

in.close();

cout << "Weight:" << d2.getW() << " Age:" << d2.getAge();

return 0;

}

[教材习题12\_6：修改例9-3的Array类模板并增加异常处理](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10367');)

【问题描述】

修改例9-3的Array类模板，在执行“[]”运算符时，若输入的索引i在有效范围之外，抛出out\_of\_range异常。

编写主函数测试Array类模板：

主函数参考如下：

int main()

{

    Array<int> a(10);

    try

    {

        a[10]=5;

        cout<<a[10]<<endl;

    }

    catch(out\_of\_range& e)

    {

        cout<<e.what()<<endl;

    }

}

【输入形式】

无  
【输出形式】

程序的输出结果：

Exception:out of range!

代码：

#include <cassert>

#include <iostream>

#include <stdexcept>

using namespace std;

//数组类模板定义

template <class T>

class Array {

private:

T\* list; //T类型指针，用于存放动态分配的数组内存首地址

int size; //数组大小（元素个数）

public:

Array(int sz = 50); //构造函数

Array(const Array<T> &a); //拷贝构造函数

~Array(); //析构函数

Array<T> & operator = (const Array<T> &rhs); //重载"="使数组对象可以整体赋值

T & operator [] (int i); //重载"[]"，使Array对象可以起到C++普通数组的作用

const T & operator [] (int i) const; //"[]"运算符的const版本

operator T \* (); //重载到T\*类型的转换，使Array对象可以起到C++普通数组的作用

operator const T \* () const; //到T\*类型转换操作符的const版本

int getSize() const; //取数组的大小

void resize(int sz); //修改数组的大小

};

//构造函数

template <class T>

Array<T>::Array(int sz) {

assert(sz >= 0); //sz为数组大小（元素个数），应当非负

size = sz; // 将元素个数赋值给变量size

list = new T [size]; //动态分配size个T类型的元素空间

}

//析构函数

template <class T>

Array<T>::~Array() {

delete [] list;

}

//拷贝构造函数

template <class T>

Array<T>::Array(const Array<T> &a) {

//从对象x取得数组大小，并赋值给当前对象的成员

size = a.size;

//为对象申请内存并进行出错检查

list = new T[size]; // 动态分配n个T类型的元素空间

//从对象X复制数组元素到本对象

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

list[i] = a.list[i];

}

//重载"="运算符，将对象rhs赋值给本对象。实现对象之间的整体赋值

template <class T>

Array<T> &Array<T>::operator = (const Array<T>& rhs) {

if (&rhs != this) {

//如果本对象中数组大小与rhs不同，则删除数组原有内存，然后重新分配

if (size != rhs.size) {

delete [] list; //删除数组原有内存

size = rhs.size; //设置本对象的数组大小

list = new T[size]; //重新分配n个元素的内存

}

//从对象X复制数组元素到本对象

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

list[i] = rhs.list[i];

}

return \*this; //返回当前对象的引用

}

//重载下标运算符，实现与普通数组一样通过下标访问元素，并且具有越界检查功能

//插入你的代码

template <typename T>

T& Array<T>::operator[](int n)

{

if (n < 0 || n >= size)

throw out\_of\_range("Exception:out of range!");

else

return list[n];

}

//重载指针转换运算符，将Array类的对象名转换为T类型的指针，

//指向当前对象中的私有数组。

//因而可以象使用普通数组首地址一样使用Array类的对象名

template <class T>

Array<T>::operator T \* () {

return list; //返回当前对象中私有数组的首地址

}

template <class T>

Array<T>::operator const T \* () const {

return list; //返回当前对象中私有数组的首地址

}

//取当前数组的大小

template <class T>

int Array<T>::getSize() const {

return size;

}

// 将数组大小修改为sz

template <class T>

void Array<T>::resize(int sz) {

assert(sz >= 0); //检查sz是否非负

if (sz == size) //如果指定的大小与原有大小一样，什么也不做

return;

T\* newList = new T [sz]; //申请新的数组内存

int n = (sz < size) ? sz : size; //将sz与size中较小的一个赋值给n

//将原有数组中前n个元素复制到新数组中

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

newList[i] = list[i];

delete[] list; //删除原数组

list = newList; // 使list指向新数组

size = sz; //更新size

}

int main()

{

Array<int> a(10);

try

{

a[10]=5;

cout<<a[10]<<endl;

}

catch(out\_of\_range& e)

{

cout<<e.what()<<endl;

}

}

|  |
| --- |
| [教材习题12\_4：设计一个异常类Exception测试范围异常](http://prg.cqupt.edu.cn/assignment/javascript:displayObject('problemDesc10366');)  【问题描述】  设计一个异常类Exception，在此基础上派生一个OutOfMemory类响应内存不住，一个RangeError类响应输入的数不在制定范围内，实现并测试这几个类。  注意：因内存不足服务器不便于测试，故本题提交答案只测试RangeError类。  【输入形式】  正常的输入输出（数字前为提示文字）：  Input n(1<=n<=100):88  【输出形式】  n:88  有异常的输入输出（数字前为提示文字）：  Input n(1<=n<=100):-5  Exception:Range Error!  【样例输入】  Input n(1<=n<=100):-5  【样例输出】  Exception:Range Error! |

代码：

#include <iostream>

using namespace std;

class Exception

{

public:

virtual void showMessage()=0;

};

class RangeError : public Exception

{

public:

RangeError() {

}

void showMessage()

{

cout << "Exception:Range Error!" << endl;

}

};

int main()

{

try

{

int n;

cout<<"Input n(1<=n<=100):";

cin>>n;

if(n<1||n>100)

throw RangeError();

cout<<"n:"<<n<<endl;

}

catch(Exception& e)

{

e.showMessage();

}

}