实验五 类与对象

一．实验目的与要求

1、掌握声明类的方法，类和类的成员的概念以及定义对象的方法。

2、初步掌握用类和对象编制基于对象的程序。

3、学习检查和调试基于对象的程序。 【实验内容】

二．实验内容

1、有以下程序：

#include<iostream>

using namespace std；

class Time // 定义 Time 类

{

public： // 数据成员为公用的

int hour;

int minute;

int sec ;

};

int main()

{

Time tl; //定义 t1 为 Time 类对象

cin>>t1.hour; //输入设定的时间

cin>>t1.minute;

cin>>t1.sec;

cout<<t1.hourl<<”：”<<t1.minute<<”：”<<t1.sec<<endl ;

return 0;

}

改写程序，要求：

(1)将数据成员改为私有的；

(2)将输入和输出的功能改为由成员函数实现；

(3)在类体内定义成员函数。

然后编译和运行程序。请分析什么成员应指定为公用的?什么成员应指定为私有的? 什么函

数最好放在类中定义? 什么函数最好在类外定义?

2、分别给出如下的 3 个文件：

(1)含类定义的头文件 student.h，

//student.h (这是头文件，在此文件中进行类的声明)

class Student //类声明

{

public: //公用成员函数原型声明

void display();

private:

int num;

14

char name[20];

char sex;

};

(2)包含成员函数定义的源文件 student.cpp

//student.cpp 在此文件中进行函数的定义

#include <iostream>

#include”student.h” //不要漏写此行，否则编译通不过

void Student::display() //在类外定义 display 类函数

{

cout<<”num：”<<num<<endl;

cout<<”name：”<<name<<endl;

cout<<”sex：”<<sex<<endl;

}

(3)包含主函数的源文件 main.cpp。

为了组成一个完整的源程序，应当有包括主函数的源文件：

//main.cpp 主函数模块

#include <iostream> //将类声明头文件包含进来

#include “student.h”

int main()

{

Student stud; //定义对象

Student stud1(007,”tcg”,’m’);

stud.display(); //执行 stud 对象的 display 函数

return 0;

}

请完善该程序，在类中增加一个对数据成员赋初值的成员函数 set\_value。上机调试并运行。

3、需要求 3 个长方柱的体积，请编一个基于对象的程序。数据成员包括 length(长)、width(宽)、

height(高)。要求用成员函数实现以下功能：

(1)由键盘分别输入 3 个长方柱的长、宽、高；

(2)计算长方柱的体积；

(3)输出 3 个长方柱的体积。

请编程序，上机调试并运行。

4、编写程序：定义抽象基类 Shape，由它派生出五个派生类：Circle（圆形）、Square（正

方形）、Rectangle（ 长方形）、Trapezoid （梯形）和 Triangle （三角形），用虚函数分别计算

各种图形的面积，并求出它们的和。要求用基类指针数组。使它的每一个元素指向一个派生类的

对象。

注：主函数中定义如下对象

Circle circle(10.2);

Square square(3);

Rectangle rectangle(3,4);

Trapezoid trapezoid(2.0,4.5,3);

Triangle triangle(4,5,6);

15

5、设计一个函数，返回两个向量的交集，函数头如下：

template<typename T>

vector<T> intersection(const vector<T>& v1, const vector<T>& v2)

两个向量的交际为它们的共同包含的元素，编写一个测试程序，提示用户输入两个向量，每

个包含 5 个字符串，打印输出它们的交集。

三、算法分析、程序与结果

1.程序：

#include<iostream>

using namespace std;

class Time

{

private:

int hour;

int minute;

int sec;

public:

//Time(int a, int b, int c)

//{

// hour = a;

// minute = b;

// sec = c;

//}

//Time(){}

void set\_time()

{

cin >> hour >> minute >> sec;

}

void show\_time()

{

cout << hour << ": " << minute << ": " << sec << endl;

}

};

int main()

{

Time t1;

t1.set\_time();

t1.show\_time();

return 0;

}

回答：

用于改变类状态的成员应该定义为公有，用于保存类状态的成员应该定义为私有，简单得要死的函数最好在类中定义，有一定工作量的函数放在外面定义

2.程序：

Student.h文件：

#pragma once

class Student

{

public:

void display();

void set\_value()

{

std::cin >> num >> name >> sex;

}

private:

int num;

char name[20];

char sex;

};

Student.cpp文件：

#include<iostream>

#include"student.h"

using namespace std;

void Student::display()

{

cout << "num：" << num << endl;

cout << "name：" << name << endl;

cout << "sex： "<< sex << endl;

}

Main.cpp文件：

#include <iostream> //将类声明头文件包含进来

#include "student.h"

int main()

{

Student stud; //定义对象

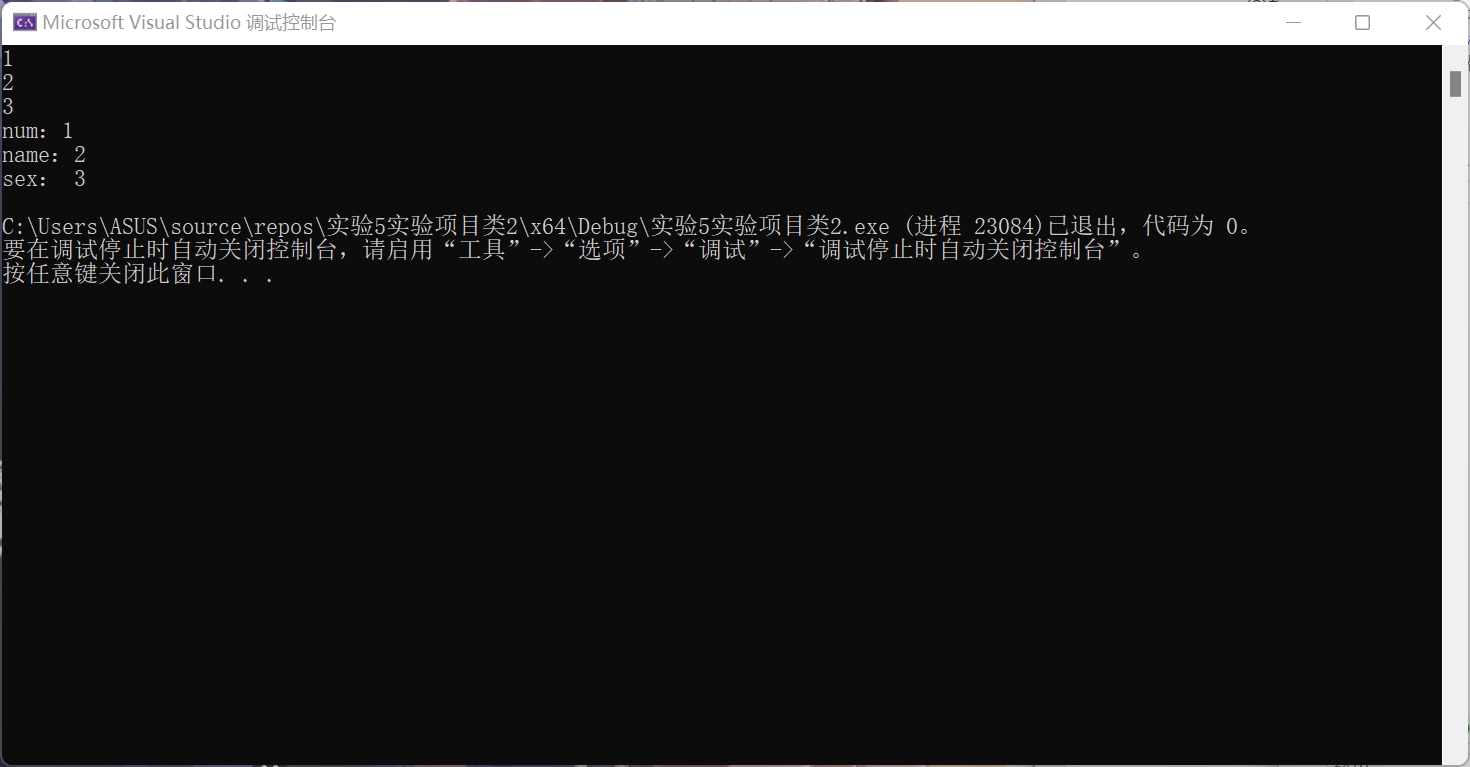
stud.set\_value();

stud.display(); //执行 stud 对象的 display 函数

return 0;

}

结果：



3.程序：

#include<iostream>

#include<string>

using namespace std;

class cft

{

private:

double length;

double width;

double height;

public:

cft()

{

cout << "请依次输入长宽高" << endl;

cin >> length >> width >> height;

}

double show\_v()

{

double v = length \* width \* height;

return v;

}

};

int main()

{

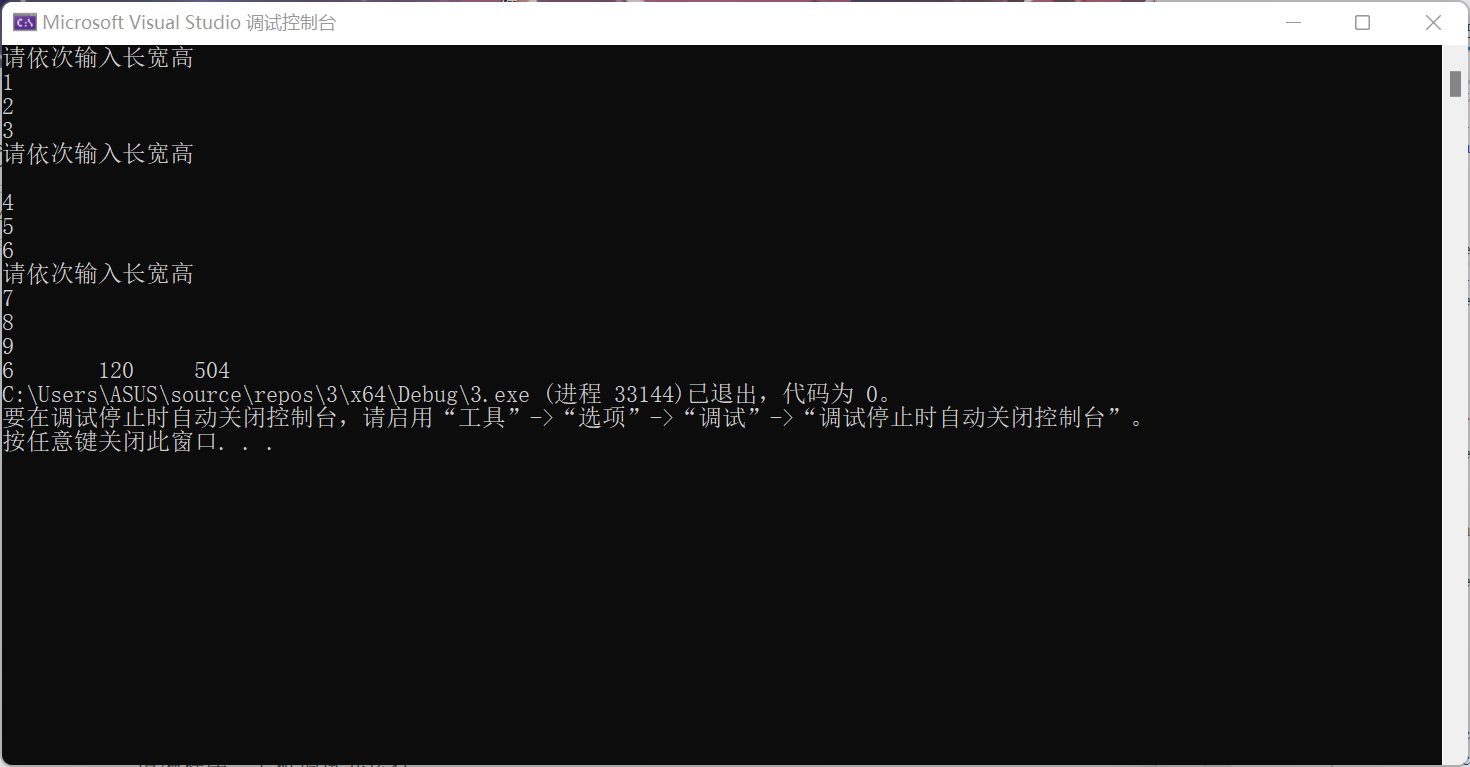
cft c1, c2, c3;

cout << c1.show\_v() <<'\t' << c2.show\_v()<<'\t' << c3.show\_v();

return 0;

}

结果：



4.程序：

#include<iostream>

#include<iomanip>

#include<math.h>

using namespace std;

class Shape

{

public:

Shape() {}

virtual ~Shape() {}

virtual void printArea() const = 0;

virtual double getArea() const = 0;

};

class Circle : public Shape //Circle

{

public:

Circle() {}

Circle(double r) :radius(r) {}

virtual void printArea() const;

virtual double getArea() const

{

return 3.14159 \* radius \* radius;

}

private:

double radius;

};

void Circle::printArea() const

{

cout << "Circle's area : " << setiosflags(ios::fixed) << setprecision(3) << (3.14159 \* radius \* radius) << endl;

}

class Rectangle : public Shape //Rectangle

{

public:

Rectangle() {}

Rectangle(double h, double w) :height(h), width(w) {}

virtual void printArea() const;

virtual double getArea() const

{

return height \* width;

}

private:

double height;

double width;

};

void Rectangle::printArea() const

{

cout << "Rectangle's area : " << setiosflags(ios::fixed) << setprecision(3) << height \* width << endl;

}

class Triangle : public Shape //Triangle

{

public:

Triangle() {}

Triangle(double a, double b,double c) :bian1(a), bian2(b),bian3(c) {}

virtual void printArea() const;

virtual double getArea() const

{

double result = sqrt(p \* (p - bian1) \* (p - bian2) \* (p - bian3));

return result;

}

private:

double bian1;

double bian2;

double bian3;

double p = bian1 + bian2 + bian3;

};

void Triangle::printArea() const

{

cout << "Triangle's area : " << setiosflags(ios::fixed) << setprecision(3) << sqrt(p \* (p - bian1) \* (p - bian2) \* (p - bian3)) << endl;

}

class Trapezoid : public Shape //Trapezoid

{

public:

Trapezoid() {}

Trapezoid(double t, double b, double h) :top(t), bottom(b), height(h) {}

virtual void printArea() const;

virtual double getArea() const

{

return (top + bottom) \* height / 2;

}

private:

double top;

double bottom;

double height;

};

void Trapezoid::printArea() const

{

cout << "Trapezoid's area : " << setiosflags(ios::fixed) << setprecision(3) << (top + bottom) \* height / 2 << endl;

}

class Square : public Shape //Square

{

public:

Square() {}

Square(double s) :s\_length(s) {}

virtual void printArea() const;

virtual double getArea() const

{

return s\_length \* s\_length;

}

private:

double s\_length;

};

void Square::printArea() const

{

cout << "Square 's area : " << setiosflags(ios::fixed) << setprecision(3) << s\_length \* s\_length << endl;

}

int main()

{

Circle circle(10.2);

Rectangle rectangle(3,4);

Triangle triangle(4,5,6);

Trapezoid trapezoid(2.0, 4.5, 3);

Square square(3);

Shape\* p[5] = { &circle,&rectangle,&triangle,&trapezoid,&square };

int i;

double area;

for (i = 0; i != 5; ++i)

{

p[i]->printArea();

}

for (i = 0, area = 0; i != 5; ++i)

{

area += p[i]->getArea();

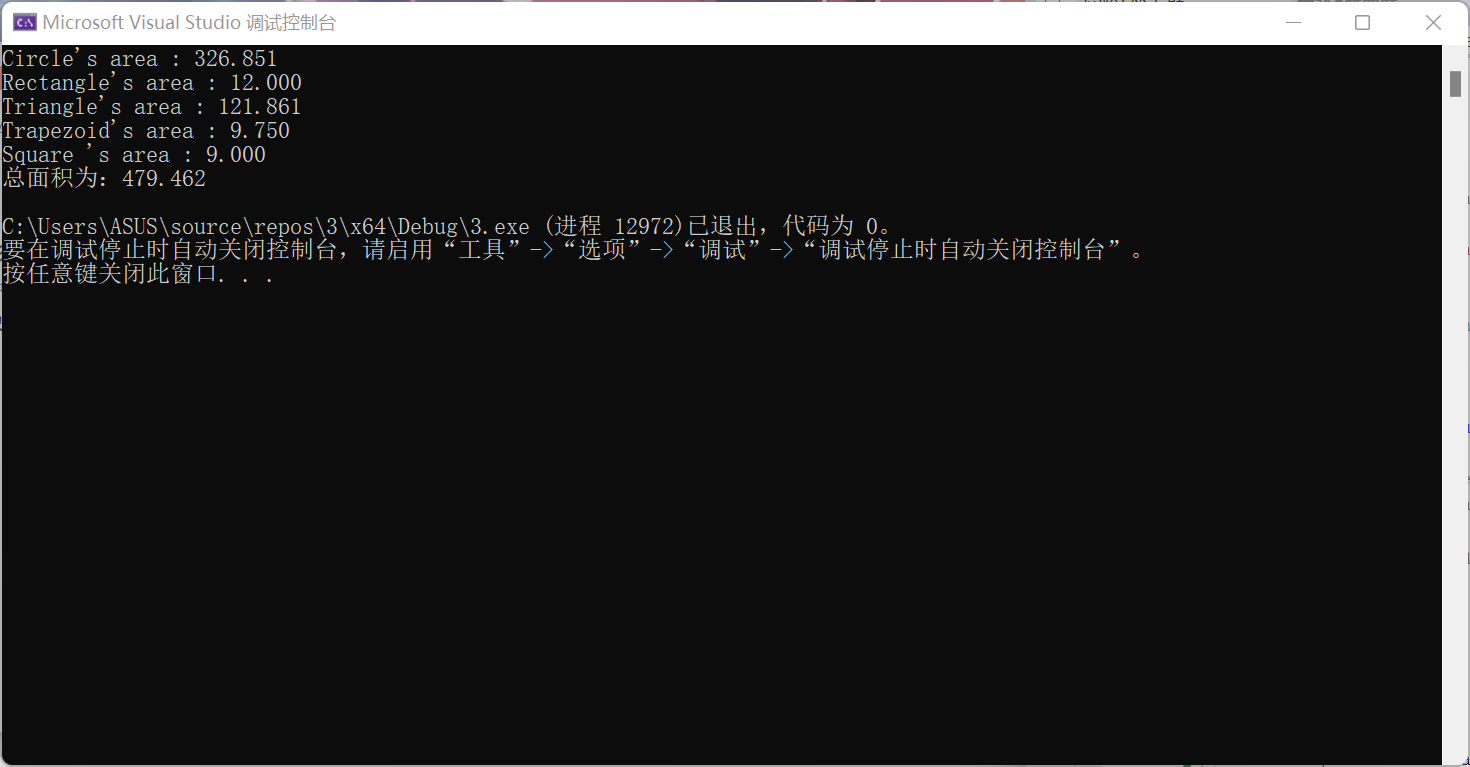
}

cout << "总面积为：" << area << endl;

return 0;

}

结果：



5.

程序：

#include "iostream"

#include "vector"

#include "algorithm"

using namespace std;

void print\_vector(vector<int> v) {

if (v.size() > 0) {

cout << "{";

for (int i = 0; i<int(v.size()); i++) {

cout << v[i] << ",";

}

cout << "\b}";

}

else {

cout << "{}";

}

}

template<typename T>

vector<T> intersection(vector<T>& v1, vector<T>& v2) {

vector<T> v;

sort(v1.begin(), v1.end());

sort(v2.begin(), v2.end());

set\_intersection(v1.begin(), v1.end(), v2.begin(), v2.end(), back\_inserter(v));//求交集

return v;

}

int main() {

vector<int> v1, v2, v;

cout << "输入第一个vector" << endl;

for (int i = 0; i < 5; i++) {

int a;

cin >> a;

v1.push\_back(a);

}

cout << "输入第二个vector" << endl;

for (int i = 0; i < 5; i++) {

int a;

cin >> a;

v2.push\_back(a);

}

cout << "求v1与v2的交集：";

v = intersection(v1, v2);

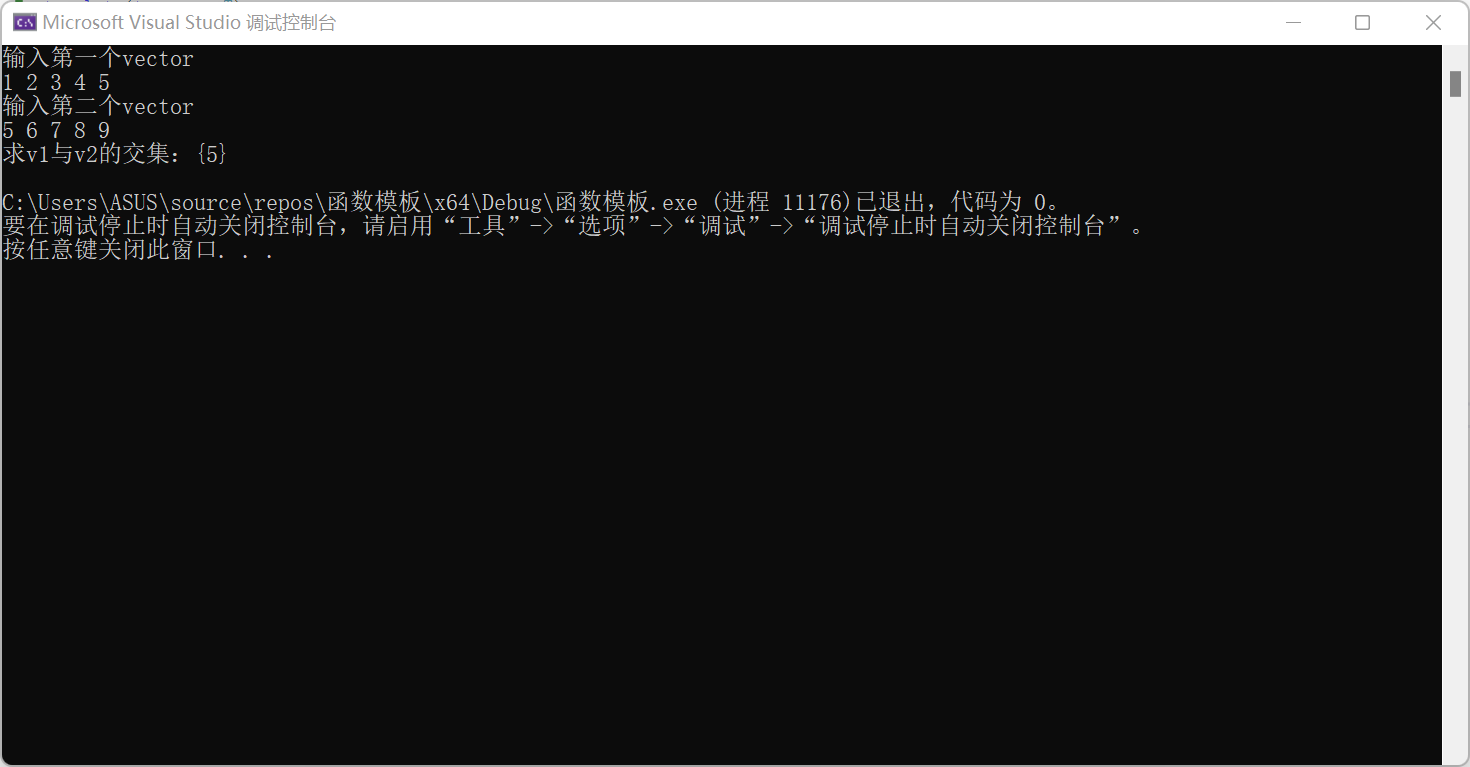
print\_vector(v);

cout << endl;

return 0;

}

结果：



四．遇到的问题与解决办法

下面是我在这次出现比较多的错误，当然还有好多没有列举出来的。1.在源码中遗失“;”

调试器错误信息:syntax error : missing ';'

1.在程序中使用中文标示符，如将英文”;”错误输入成了”;”调试器错误信息: error C2018: unknown character '0xa3'

在C++中，除程序注释可以采用中文外，其余字符要求使用英文。不少同学在建立工程或程序名称时也使用中文名称，建议改掉这种习惯。

2．在使用输入输出流的时候错误使用了标示符“>>”“<<”,例cout>>a;

调试器错误信息:例:error C2676: binary '>>': 'classstd:: basic\_ostream<char, struct std::char\_traits<char>>' does not define thisoperator or a conversion to a type acceptable to the \_predefined operator

对于流操作的方向搞错是一个普遍错误，问题本来并不复杂，可能是由于没有认真看书的原因。

五．体会

通过这次课程设计，我感觉到要真正做出一个程序并不很容易，但只要用心去做，总会有收获，特别是当我遇到一个问题，想办法去解决，最后终于找到方法时，心里的那份喜悦之情真是难以形容。编写程序中遇到问题再所难免，应耐心探究其中的原因，从出现问题的地方起，并联系前后程序，仔细推敲，逐个排查。直到最终搞清为止。

对于循环结构有了更深层次的理解，尤其是循环嵌套的使用，还有随机数生成函数rand的使用。