第三次上机实验报告

**学号：**202030310081  **姓名：**付天锟  **班级：**电自2005

1. **实验目的**

1、理解类和对象的概念，掌握声明类和定义对象的方法

2、掌握构造函数和析构函数的实现方法

3、初步掌握使用类和对象编制C++程序

1. **实验内容**
2. **实验一**

**创建一个类，完成以下功能：**

连续输入一组二维坐标值；

二维坐标值的数目可以由用户自定义（默认为2个，最多为100组）；

显示用户输入的坐标值；

显示用户输入坐标值的均值；

**观察例程中的构造函数和析构函数的运行顺序；**

**在main()函数中加入如下代码，观察运行结果：**

Coordinate y(5);

y.InputCoord();

y.ShowCoord();

y.ShowAvgCoord();

1. **实验二**

**创建一个Score类，完成以下功能：**

连续输入多位学生的成绩（成绩=科目A成绩+科目B成绩+科目C成绩）；

学生数目可以由用户自定义（默认为2个，最多为100个）；

显示每位同学的每科成绩和平均分；

显示每门科目的平均成绩；

对每门成绩进行排序并由高到底显示；

对整个文件进行打包。

1. **实验程序结果**
2. **实验一**

#include<iostream>

using namespace std;

class Coordinate { // 定义Coordinate类

public:

Coordinate()

{

times = 2;

cout << "Coordinate construction1 called!" << endl;

}

Coordinate(int times1)

{

times = times1;

cout << "Coordinate construction2 called!" << endl;

}

~Coordinate()

{

cout << "Coordinate destruction called!" << endl;

}

void InputCoord()

{

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

{

cout << "Please Input x:" << endl;

cin >> Coord[i][1];

cout << "Please Input y:" << endl;

cin >> Coord[i][2];

}

}

void ShowCoord()

{

cout << "The coord is:" << endl;

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

{

cout << "(" << Coord[i][1] << "," << Coord[i][2] << ")" << endl;

}

}

void ShowAvgCoord()

{

float avgx = 0;

float avgy = 0;

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

{

avgx = avgx + Coord[i][1];

avgy = avgy + Coord[i][2];

}

avgx = avgx / times;

avgy = avgy / times;

cout << "The AVG coord is:" << endl;

cout << "(" << avgx << "," << avgy << ")" << endl;

}

private:

float Coord[100][100]; // 存放输入坐标的数组

int times; // 存放输入坐标数目

};

int main()

{

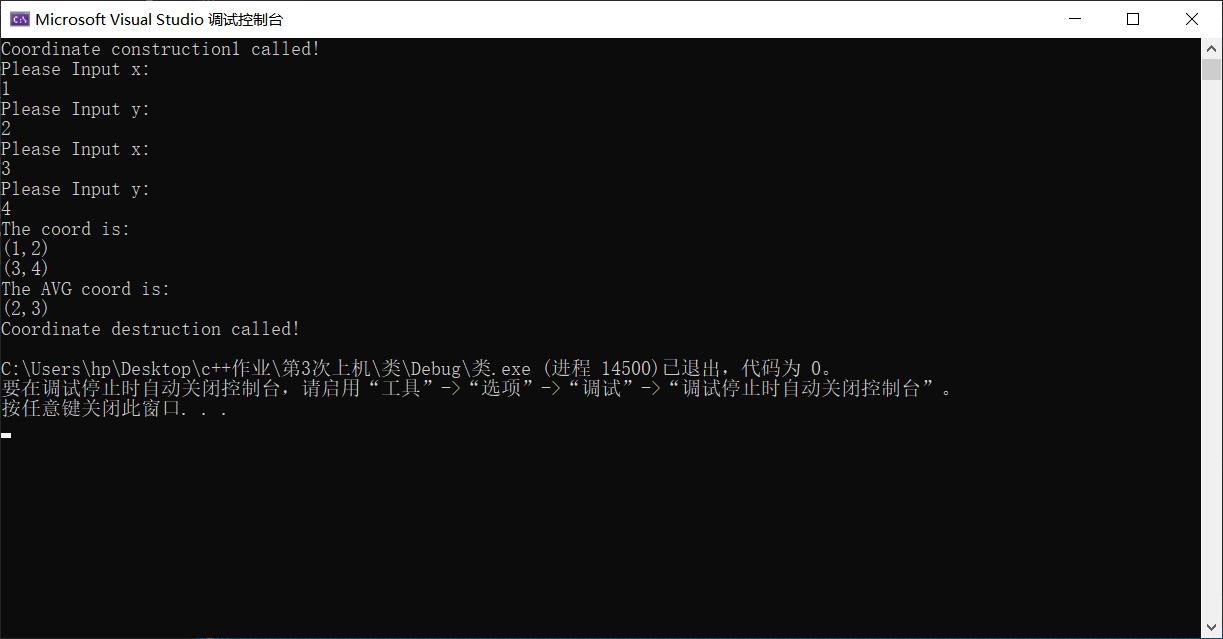
Coordinate y(5);

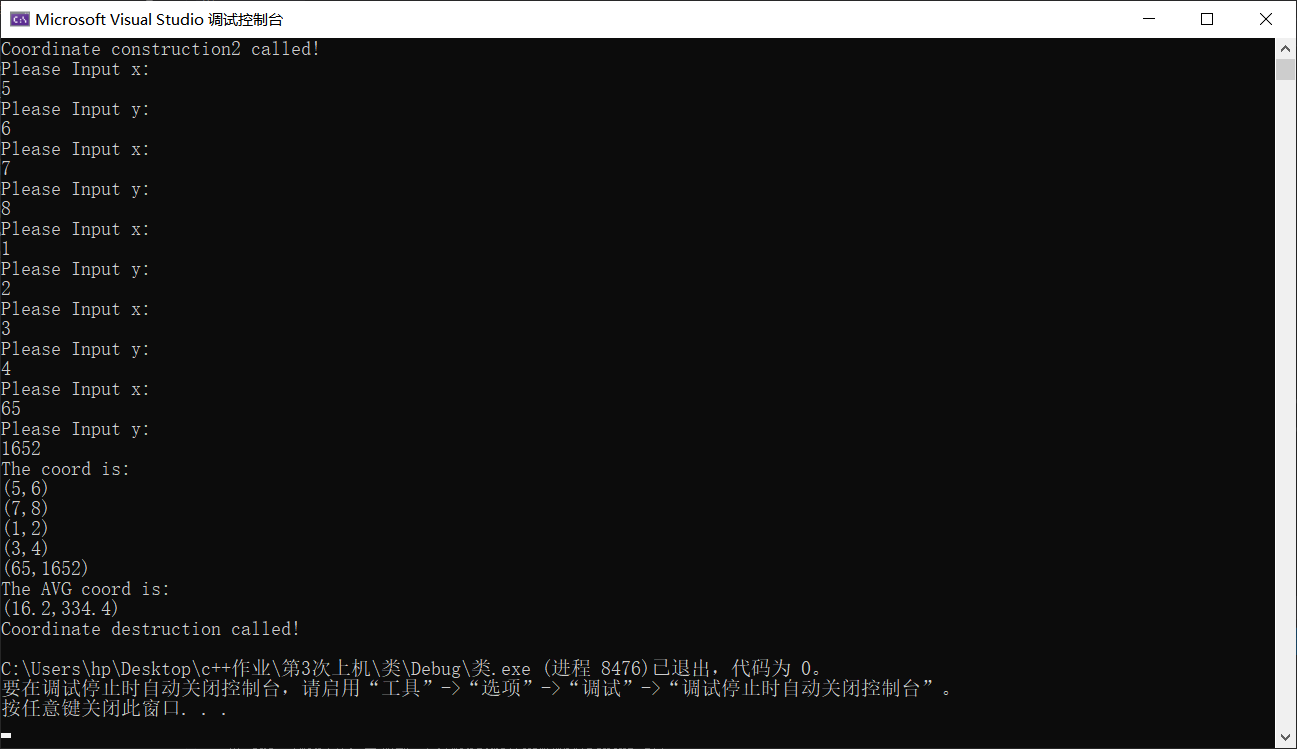
y.InputCoord();

y.ShowCoord();

y.ShowAvgCoord();

}





1. **实验二**

#include<iostream>

#include<string>

using namespace std;

class Score {

public:

Score()

{

times = 2;

}

Score(int times1)

{

times = times1;

}

void InputNameAndScore()

{

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

{

cout << "请输入学生姓名:" << endl;

cin >> Name[i];

cout << "请输入科目A成绩:" << endl;

cin >> SScore[i][1];

cout << "请输入科目B成绩:" << endl;

cin >> SScore[i][2];

cout << "请输入科目C成绩:" << endl;

cin >> SScore[i][3];

}

}

void ShowNameAndScore()

{

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

{

cout << "姓名: " << Name[i] << " 科目A成绩: " << SScore[i][1] << " 科目B成绩 " << SScore[i][2] << " 科目C成绩: " << SScore[i][3] << endl;

}

}

void ShowStdentAvgScore(int Sid)

{

float avg = 0;

avg = (SScore[Sid][1] + SScore[Sid][2] + SScore[Sid][3]) / 3;

cout << "姓名: " << Name[Sid] << " 平均成绩: " << avg << endl;

}

void ShowClassAvgScore(string ClassName)

{

int Cid;

float avg = 0;

if (ClassName == "A") Cid = 1;

if (ClassName == "B") Cid = 2;

if (ClassName == "C") Cid = 3;

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

{

avg = avg + SScore[i][Cid];

}

avg = avg / times;

cout << "课程名称: " << ClassName << "平均成绩: " << avg << endl;

}

void OrderScore(string ClassName)

{

int Cid;

if (ClassName == "A") Cid = 1;

if (ClassName == "B") Cid = 2;

if (ClassName == "C") Cid = 3;

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

{

SScore1[i] = SScore[i][Cid];

}

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

{

Name1[i] = Name[i];

}

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

{

if (SScore1[i] > SScore1[i - 1])

{

float temp = SScore1[i - 1];

SScore1[i - 1] = SScore1[i];

SScore1[i] = temp;

string temp1;

temp1 = Name1[i - 1];

Name1[i - 1] = Name1[i];

Name1[i] = temp1;

}

}

cout << "课程名称: " << ClassName << endl;

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

{

cout << "姓名: " << Name1[i] << " 成绩: " << SScore1[i] << endl;

}

}

private:

float SScore[100][3], SScore1[100];

string Name[100], Name1[100];

int times;

};

int main()

{

Score x;

x.InputNameAndScore();

x.ShowNameAndScore();

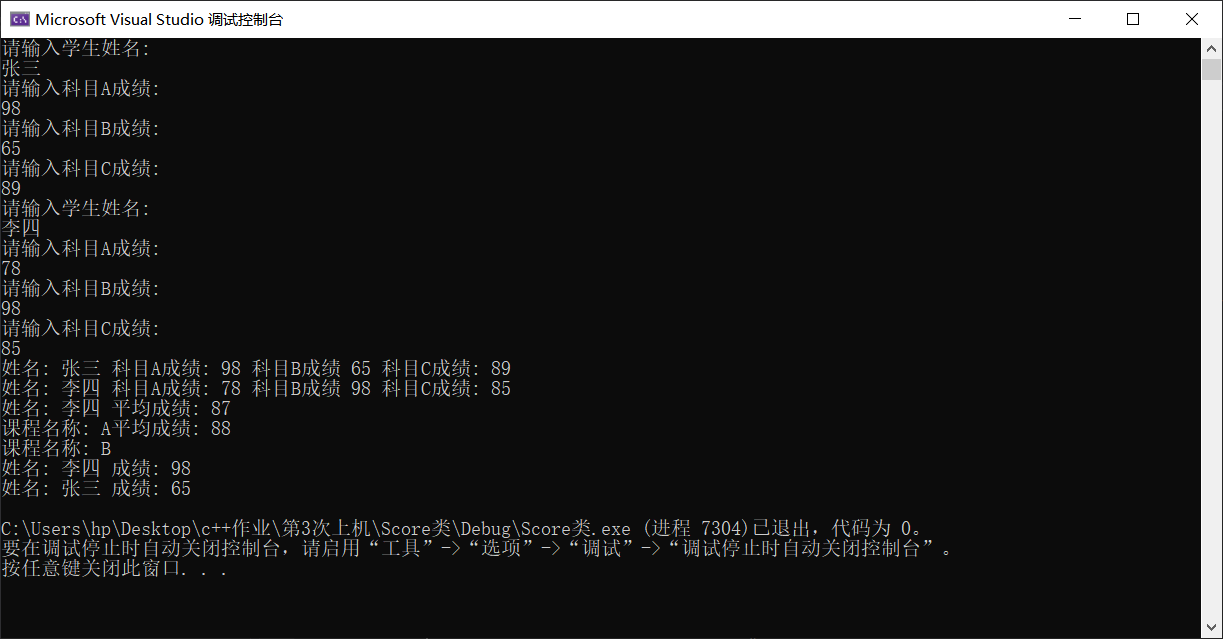
x.ShowStdentAvgScore(1);

x.ShowClassAvgScore("A");

x.OrderScore("B");

return 0;

}



1. **实验心得**

学习C++类的过程中，我体会到了类面向对象的方便之处，与面向过程的C语言不同的是，C++能系统的、结构更清晰的去设计一个程序，可以让一个程序模块化，每一个功能都分配给不同的类去完成，这样哪里出问题了就去找相应的类去调试，避免了大面积出错，节约了调试时间。

类的知识点是抽象的，但是仔细琢磨后，就会发现，这其实和我们刚开始学习基本的数据类型是一样的，并没有很费脑的地方，说的通俗一点就是学会定义类的基本语法，掌握“套路”，去“套模板”，只要是细心一点，把每个功能都逐一调试，就不会大面积的出错。

在使用类设计程序时，要提前设计好方案，明确哪一个部分实现什么功能，需要什么数据，分别设计数据类和操作类，恰当的把数据私有化，在有了一个整体的思路之后，在去写代码，切记不能把全部的代码写完再去调试，这样出错以后会很难改正，而且非常浪费时间，应当做到写一个功能就去调试一个功能。

Copyright ©2021-2099 TiankunFu. All rights reserved