实验内容一

一、程序代码

#include<iostream>

using namespace std;

class Coordinate {

private:

float Coord[100][100];

int times;

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;

}

};

int main()

{

Coordinate x;

x.InputCoord();

x.ShowCoord();

x.ShowAvgCoord();

Coordinate y(5);

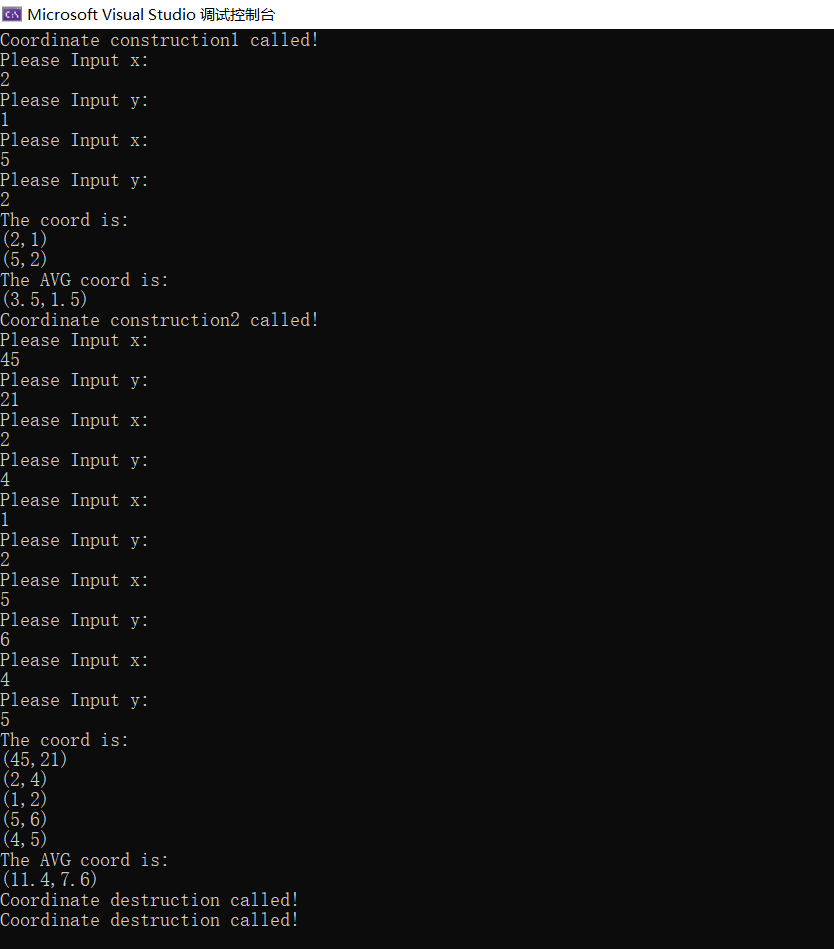
y.InputCoord();

y.ShowCoord();

y.ShowAvgCoord();

return 0;

}

二、程序结果

三、感想心得

可以通过重载构造函数实现用默认值对构造函数进行初始化的操作。

构造函数的执行顺序为先基类后逐级派生类，析构函数为先最外层派生类后逐步执行基类的。

实验内容二

一、程序代码

#include<iostream>

#include<string.h>

using namespace std;

class Coordinate {

private:

float Coord[100][300];

int times;

char name[100][20];

float avg[100];

public:

Coordinate(int times1 = 2)

{

times = times1;

}

void InputCoord()

{

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 >> Coord[i][0];

cout << "请输入科目B成绩：" << endl;

cin >> Coord[i][1];

cout << "请输入科目C成绩：" << endl;

cin >> Coord[i][2];

}

}

void ShowCoord()

{

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

{

cout << "姓名：" << name[i] << " 科目A成绩：" << Coord[i][0] << " 科目B成绩" << Coord[i][1] << " 科目C成绩" << Coord[i][2] << endl;

}

}

void ShowAvgCoord()

{

float avgA = 0;

float avgB = 0;

float avgC = 0;

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

{

avgA = avgA + Coord[i][0];

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

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

avg[i] = Coord[i][0] + Coord[i][1] + Coord[i][2];

}

avgA = avgA / times;

avgB = avgB / times;

avgC = avgC / times;

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

{

cout << "姓名：" << name[i] << "平均成绩：" << avg[i] << endl;

}

cout << "课程名称：A 平均成绩：" << avgA << endl;

cout << "课程名称：B 平均成绩：" << avgB << endl;

cout << "课程名称：C 平均成绩：" << avgC << endl;

}

void sort()

{

for (int i = 0; i < times - 1; i++) {

for (int j = 0; j < times - 1 - i; j++) {

if (avg[j] > avg[j + 1])

{

float temp = avg[j + 1];

avg[j + 1] = avg[j];

avg[j] = temp;

char temp1[100][20];

strcpy\_s(temp1[0], name[j + 1]);

strcpy\_s(name[j + 1] , name[j]);

strcpy\_s(name[j] , temp1[0]);

}

}

}

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

{

cout << "姓名：" << name[i] << " 平均成绩：" << avg[i] << endl;

}

}

};

int main()

{

Coordinate x(2);

x.InputCoord();

x.ShowCoord();

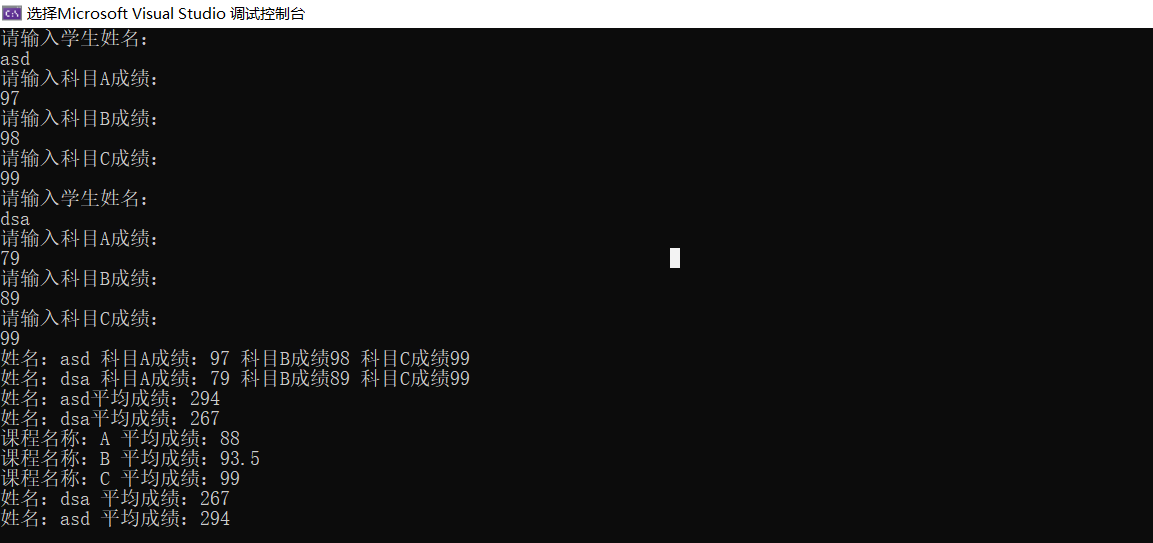
x.ShowAvgCoord();

x.sort();

return 0;

}

二、程序结果



三、感想心得



出现这个是因为strcpy的问题。

strcpy\_s和strcpy()函数的功能几乎是一样的。strcpy函数，就像gets函数一样，它没有方法来保证有效的缓冲区尺寸，所以它只能假定缓冲足够大来容纳要拷贝的字符串。在程序运行时，这将导致不可预料的行为。用strcpy\_s就可以避免这些不可预料的行为。

此外还以在main前面加上#pragma warning(disable:4996)。

使用循环可以有效减少代码长度。

Copyright ©2021-2099 ZhongkaiCui. All rights reserved