# 实验七 运算符重载

### 题目1 重载<，>， ==

主函数:

#include <iostream>

#pragma warning(disable:4996)

#include<cstring>

#include"String.h"

using namespace std;

int main() {

String string1("china"), string2("China"), string3("National"), string4("Computer"), string5("Examination"), string6("Rank"), string7("swust"), string8("swust"), string9("Hello"), string10;

cout << (string1 > string2) << endl;

cout << (string3 < string4) << endl;

cout << (string5 < string6) << endl;

cout << (string7 == string8) << endl;

cout << string9 << endl;

cin >> string10;

cout << string10;

}

头文件:

#pragma once

#include<string>

#include<string.h>

#include <iostream>

using namespace std;

class String {

public:

String();

String(string b);

void show();

friend bool operator>(String& string1, String& string2);

friend bool operator<(String& string1, String& string2);

friend bool operator==(String& string1, String& string2);

friend ostream& operator<<(ostream& os, String& string2);

friend istream& operator>>(istream& is, String& string2);

friend void show();

private:

string a;

};

头文件.cpp

#include "String.h"

#include <iostream>

using namespace std;

String::String() {

a = "Hi";

}

String::String(string b) {

a = b;

}

void String::show() {

cout << a;

}

bool operator >(String& string1, String& string2) {

//声明运算符重载函数为友元函数,此函数并不在String里面而是用friend则可以访问该类里面的各种变量,不过此处是定义在String类里面的public函数，仍然可以将该几个重载函数弄出String 外定义且不加范围限定符号

if (string1.a > string2.a)

return true;

else

return false;

}

bool operator<(String& string1, String& string2) {

if (string1.a > string2.a)

return false;

else

return true;

}

bool operator==(String& string1, String& string2) {

if (string1.a == string2.a)

return true;

else

return false;

}

ostream& operator<<(ostream& os, String& string2) {

os << string2.a;

return os;

}

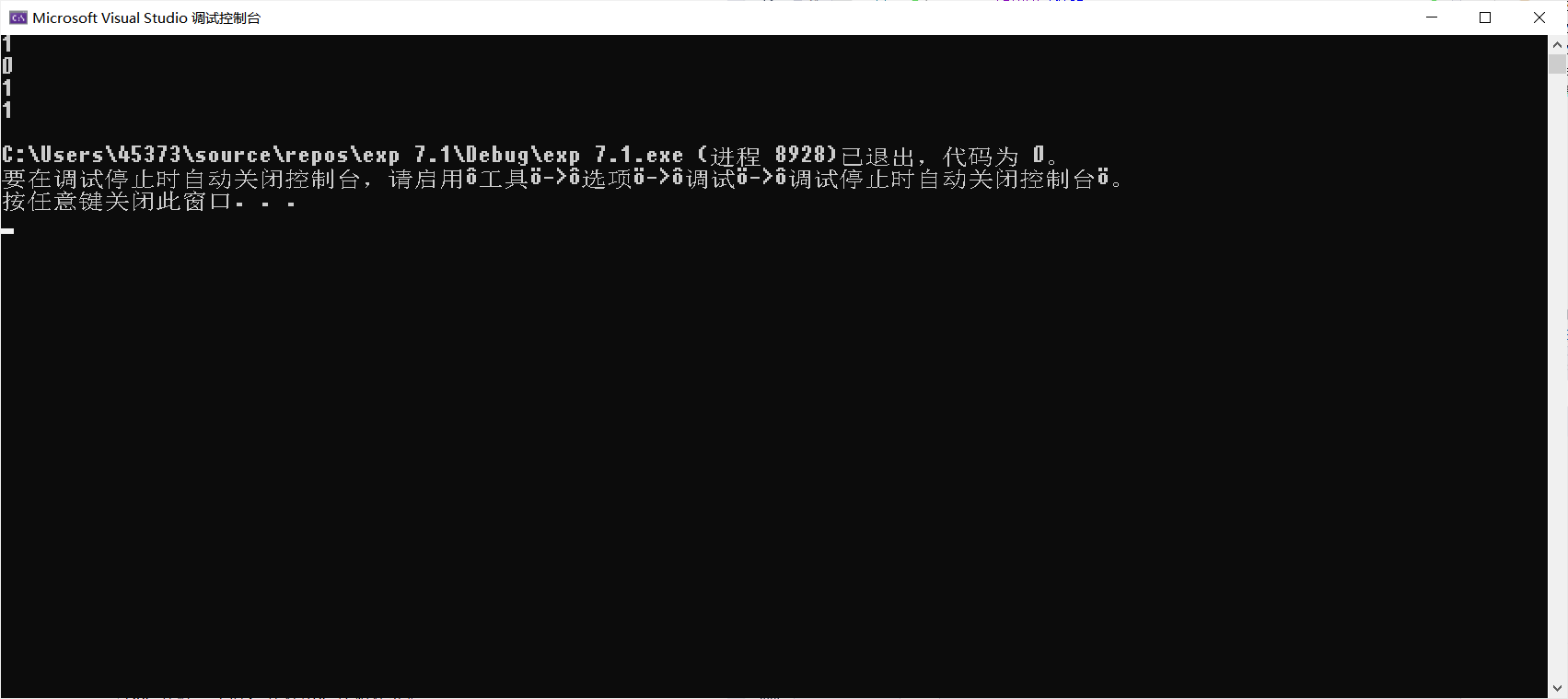
istream& operator>>(istream& is, String& string2) {

is >> string2.a;

return is;

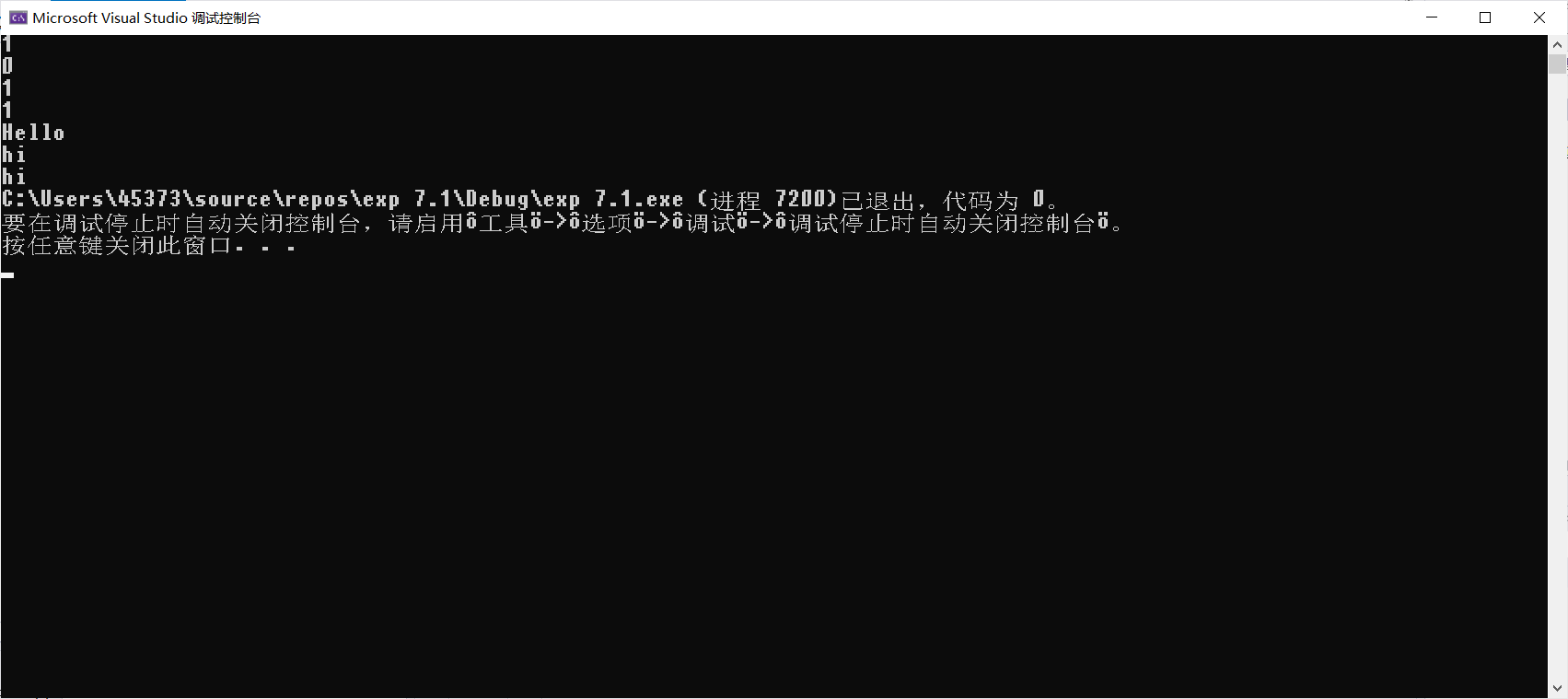
}

输出分别对应1组是否大于，2组是否小于，3组是否小于，4组是否相等



思考与扩展

调试结果如图：



### 题目2 重载　　时间加减法

主函数：

#include<iostream>

#include"Time.h"

using namespace std;

int main() {

Ctime t;

t.SetHour(13);

t.SetMinute(45);

t.SetSecond(39);

for (int i = 0; i < 120; i++) {

++t;

}

t.Display();

t.SetHour(23);

t.SetMinute(58);

t.SetSecond(5);

for (int i = 0; i < 120; i++) {

++t;

}

t.Display();

t.SetHour(23);

t.SetMinute(58);

t.SetSecond(5);

Ctime b;

b = t.operator++(1);

b.Display();

t.Display();

return 0;

}

头文件

#pragma once

#include<string>

#include<string.h>

#include <iostream>

using namespace std;

class Ctime {

public:

Ctime();

Ctime(int h, int m, int s);

Ctime pp();

//重载前自增运算符，如果后自增要变成这样： operator++(int)

Ctime operator ++() //前置型自增

{

return pp();

}

Ctime operator ++(int)//后置型自增

{

Ctime p = \*this;

pp();

return p;

}

void Display();

void SetHour(int);

void SetMinute(int);

void SetSecond(int);

private:

int hour, minute, second;

};

头文件.cpp

#include "Time.h"

#include <iostream>

using namespace std;

Ctime::Ctime()

{

hour = minute = second = 0;

}

Ctime::Ctime(int h, int m, int s) {

hour = h;

minute = m;

second = s;

}

Ctime Ctime::pp() {

second++;

if (second >= 60) {

second = 0;

minute++;

}

if (minute >= 60) {

minute = 0;

hour++;

}

if (hour >= 24)hour = 0;

return \*this;

}

//重载前自增运算符，如果后自增要变成这样： operator++(int)

void Ctime::Display() {

cout << hour << ':' << minute << ':' << second << endl;

}

void Ctime::SetHour(int h) {

hour = h;

};

void Ctime::SetMinute(int m) {

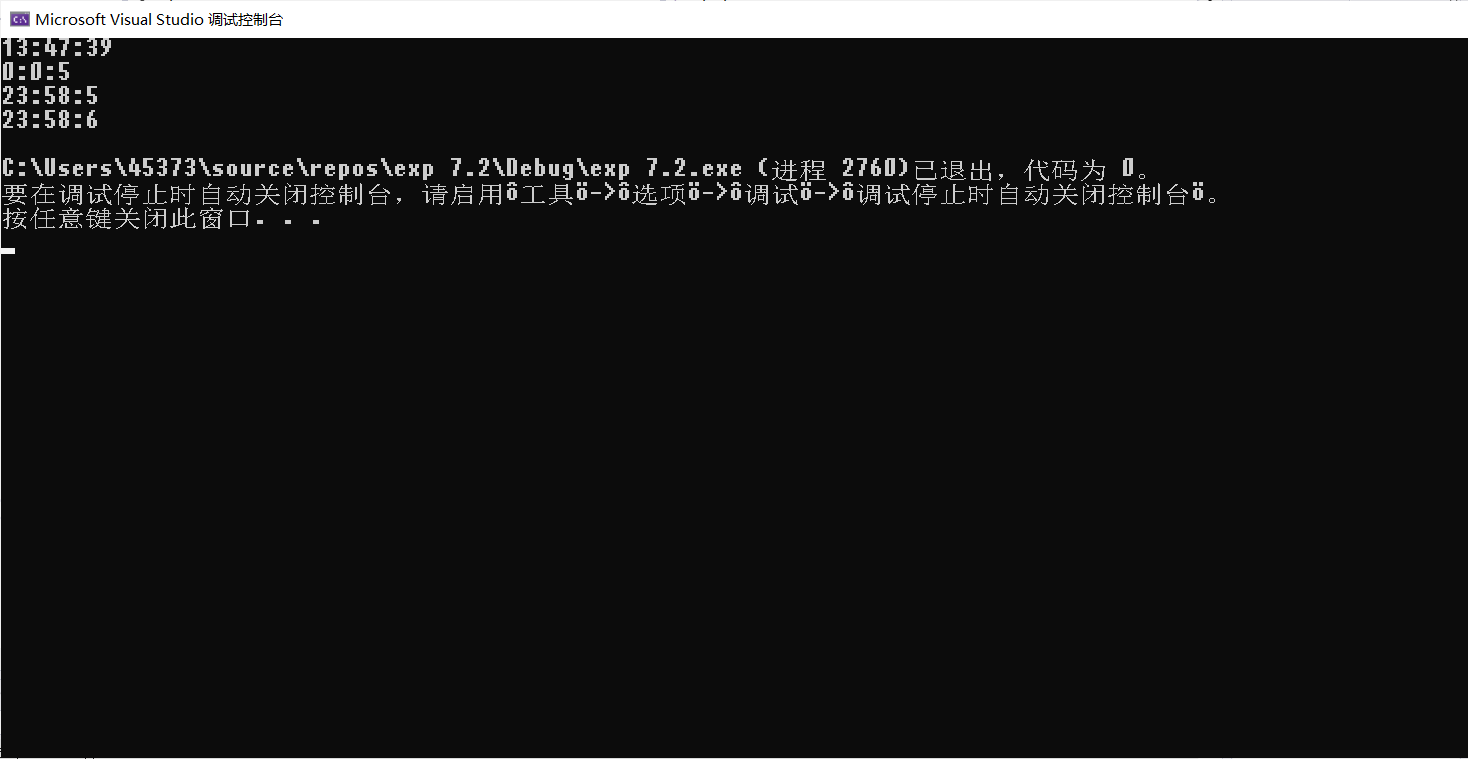
minute = m;

};

void Ctime::SetSecond(int s) {

second = s;

};



如图所示，经过120s后根据测试结果的输出，以及有区分前置++和后置++

### 题目3：重载 类对象作参数

主函数:

#include<iostream>

#pragma warning(disable:4996)

using namespace std;

#include"Student.h"

int main() {

Student s1("Jim", 70, 80, 90);

Student s2("Ftiz", 85, 90, 95);

Student s3("Jiakang", 70, 80, 76);

Student s4("Jessica", 87, 89, 95);

Student s5("Abbolo", 86, 54, 97);

Student s6 = s1 + s2 + s3 + s4 + s5;

s6.cal\_for\_ave();

s1.set(s6.average1, s6.average2, s6.average3);

s2.set(s6.average1, s6.average2, s6.average3);

s3.set(s6.average1, s6.average2, s6.average3);

s4.set(s6.average1, s6.average2, s6.average3);

s5.set(s6.average1, s6.average2, s6.average3);

avg(s1);

avg(s2);

avg(s3);

avg(s4);

avg(s5);

}

头文件:

#pragma once

#include<iostream>

#pragma warning(disable:4996)

#include"Student.h"

using namespace std;

class Student {

public:

Student();

Student(const char\* na, int d1, int d2, int d3);

friend Student operator+(Student s1, Student s2);

friend void avg(Student& s);

void set(float average11, float average22, float average33);

void cal\_for\_ave();

private:

char name[10];

int deg1, deg2, deg3;

public:

float average1;

float average2;

float average3;

};

头文件.cpp

#include<iostream>

#pragma warning(disable:4996)

#include"Student.h"

using namespace std;

Student::Student() {

deg1 = 0, deg2 = 0, deg3 = 0;

}

Student::Student(const char\* na, int d1, int d2, int d3) {

strcpy(name, na);

deg1 = d1;//高数

deg2 = d2;//英语

deg3 = d3;//计算机

}

Student operator+(Student s1, Student s2) {

char name1[100] = "combination:";

return Student(strcat(strcat(name1, s1.name), s2.name), s1.deg1 + s2.deg1, s1.deg2 + s2.deg2, s1.deg3 + s2.deg3);

}

void avg(Student& s) {

cout << "regarding info is as below:" << endl;

cout << "name:" << s.name << "高数:" << s.deg1 << "英语" << s.deg2 << "计算机" << s.deg3;

cout << "高数平均分:" << s.average1 << "英语平均分:" << s.average2 << "计算机平均分:" << s.average3 << endl;

}

void Student::set(float average11, float average22, float average33) {

average1 = average11;

average2 = average22;

average3 = average33;

}

void Student::cal\_for\_ave() {

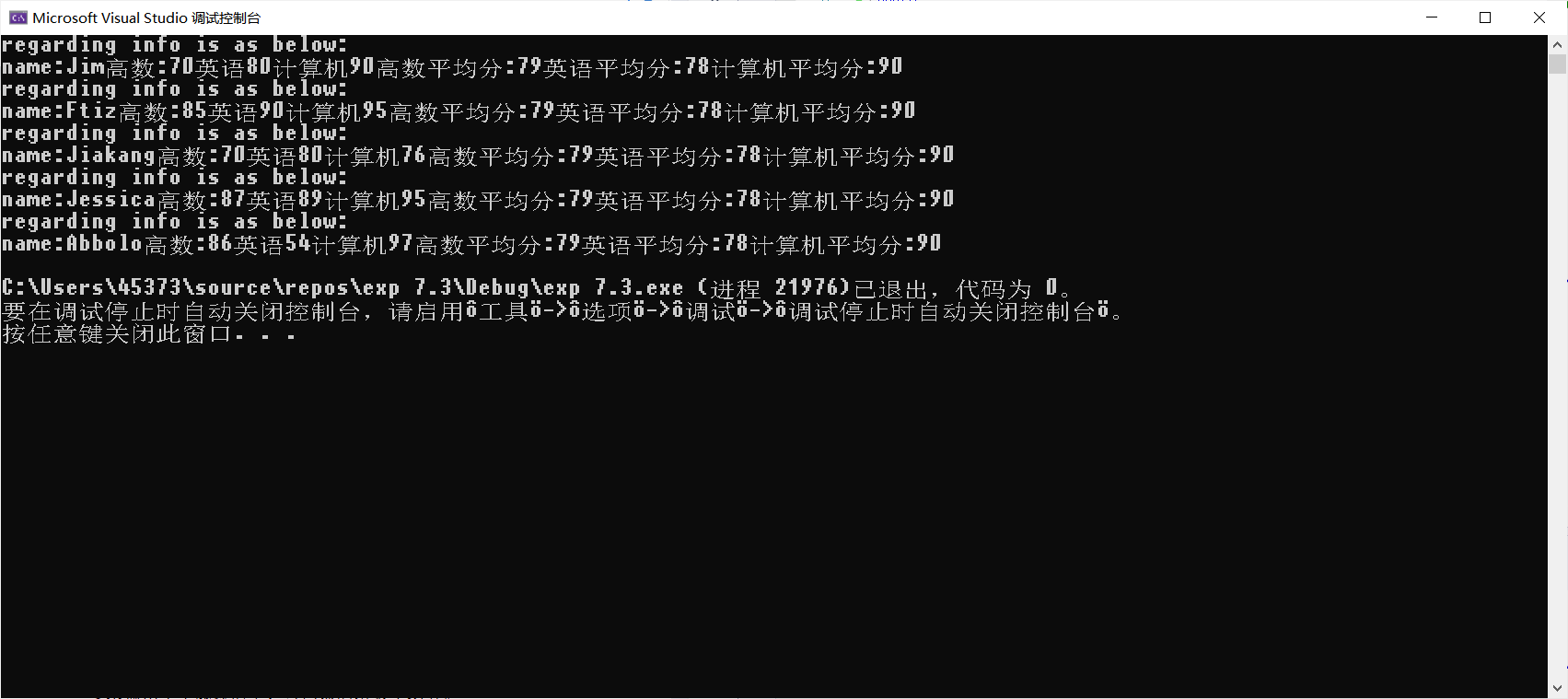
average1 = deg1 / 5;

average2 = deg2 / 5;

average3 = deg3 / 5;

}

如图所示：



测试如代码里