**目 录**

[1 按照要求设计类(根据输出设计类) 1](#_Toc533525239)

[1.1 实验要求 1](#_Toc533525240)

[1.2 设计思路 2](#_Toc533525241)

[1.3 实现代码 2](#_Toc533525242)

[1.4 运行结果 4](#_Toc533525243)

[2 时间计算器 5](#_Toc533525244)

[2.1实验要求 5](#_Toc533525245)

[2.2设计思路 5](#_Toc533525246)

[2.3实现代码 5](#_Toc533525247)

[2.4运行结果 8](#_Toc533525248)

[3 判断线段是否与圆相交 9](#_Toc533525249)

[3.1实验要求 9](#_Toc533525250)

[3.2设计思路 9](#_Toc533525251)

[3.3实现代码 9](#_Toc533525252)

[3.4运行结果 16](#_Toc533525253)

[4 学生成绩管理 18](#_Toc533525254)

[4.1实验要求 18](#_Toc533525255)

[4.2设计思路 18](#_Toc533525256)

[4.3实现代码 18](#_Toc533525257)

[4.4运行结果 29](#_Toc533525258)

[5 总结或感悟 37](#_Toc533525259)

# 1 按照要求设计类(根据输出设计类)

## 1.1 实验要求

要求：设计类就是根据数据封装的要求，抽象出适合的类。

有如下情况的测试程序和测试程序的输出结果，要求设计类Smile。

（一）第1种情况：

（1）测试程序如下：

int main()

{

cout<<"科艺欢迎你！"<<endl;

return 0;

}

图1 任务一输出结果1

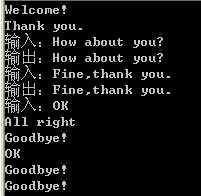
（2）测试程序的输出结果如图1；

（3）此题要求定义两个对象，用于完成图1的输出结果。

（二）第二种情况：

（1）测试程序如下：

int main()

{

Smile we;

Smile you(we);

you.Display();

you.Set(“Thank you.”);

cout<<you.Get()<<endl;

you.talk();

you.Display();

return 0;

}

图2 任务一输出结果2

（2）测试程序的输出结果如图2：

注：

后面5行为倒数第6行输出后接下去的输出内容；

倒数第1和2行调用析构函数。

倒数第3行调用Display函数

（3）设计类Smile 的原则:

1）在头文件Smile.h 中声明类

2）在Smile.cpp中实现类

3）使用内联函数定义Display()函数

4）演示文件名用学号作主名。

## 1.2 设计思路

1. 第一种情况：在不改变main函数的情况下，我在全局创建了一个对象，然后可以实现出现构造函数在程序结束时析构函数出现。
2. 第二种情况：写一个Smile类，在构造函数中将dis设为welcome，再以Display函数输出dis，在set函数中将in值设置为set中的内容，以get返回输出in中内容，后面调用talk函数输入任意字符都相同输出，直到输入OK，跳出循环，最后调用display输出，然后调用析构函数。

## 1.3 实现代码

(1).第一种情况：

main.cpp

#include <iostream>

using namespace std;

class Smile {

public:

Smile(string n) {

name = n;

cout << "Hi " << name << "!" << endl;

}

~Smile() {

cout << "Goodbye " << name << "!" << endl;

}

string name;

};

Smile s1("Rose"), s2("Jose");

int main() {

cout << "科艺欢迎你！" << endl;

return 0;

};

(2).第二种情况：

main.cpp

#include <iostream>

#include "Smile.h"

using namespace std;

int main() {

Smile we;

Smile you(we);

you.Display();

you.Set("Thank you.");

cout << you.Get() << endl;

you.talk();

you.Display();

return 0;

}

Smile.h

#ifndef INC\_1\_2WELCOME\_SMILE\_H

#define INC\_1\_2WELCOME\_SMILE\_H

#include <iostream>

#include <string>

using namespace std;

class Smile {

public:

Smile() {

dis = "Welcome!";

}

inline void Display() {

cout << dis << endl;

}

void Set(string n);

void talk();

string Get();

~Smile() {

cout << "Goodbye!" << endl;

}

string dis, in;

};

#endif //INC\_1\_2WELCOME\_SMILE\_H

Smile.cpp

#include "Smile.h"

void Smile::Set(string n) {

in = n;

}

string Smile::Get() {

return in;

}

void Smile::talk() {

while (true) {

cout << "输入:";

getline(cin, in);

if (in == "OK") {

cout << "All right" << endl;

cout << "Goodbye!" << endl;

dis = in;

break;

} else cout << "输出:" << Get() << endl;

}

}

## 1.4 运行结果



图 1‑1 第一种情况

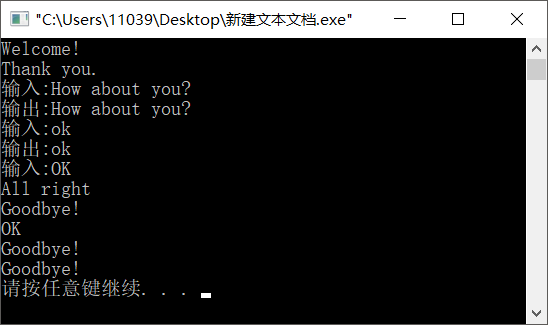


图 1‑2 第二种情况

# 2 时间计算器

## 2.1实验要求

要求：为时间定义一个类。属性为时、分、秒。要求重载“+”、“-”、“++”、“--”（自增和自减运算符要求分前置和后置）、“<<”、“>>”等运算符，实现时间的相加、相减、加1，减1、输入和输出。

## 2.2设计思路

(1).写一个time类对运算符+,-,<<,>>和前置++，后置++，前置--，后置--，进行重载，加减是考虑进位退位，实现24小时59分钟59秒的进位，0时0分0秒的退位。后置++和后置—返回的都是之前未++之前的内容，但实际上值是加一了。运算符重载是利用友元函数，将time类中的数据输出，前置++和前--实际上是将类中的值直接加一然后返回，+-重载是对time中的每个值进行加减，在每个操作之后都会判断时间是否正确，秒和分到59之后进位一。

## 2.3实现代码

main.cpp

#include <iostream>

#include "Time.h"

using namespace std;

int main() {

Time t1, t2, t3;

cout << "t1:";

cin >> t1;

cout << t1;

cout << "t2:";

cin >> t2;

t3 = t1 + t2;

cout << "t1+t2=" << t3;

t3 = t1 - t2;

cout << "t1-t2=" << t3;

t3 = ++t1;

cout << "++t1:" << t3;

t3 = --t1;

cout << "--t1:" << t3;

t3 = t2++;

cout << "t2++:" << t3;

t3 = t2--;

cout << "t2--:" << t3;

return 0;

}

Time.h

#ifndef INC\_2\_1TIMECOUNTER\_TIME\_H

#define INC\_2\_1TIMECOUNTER\_TIME\_H

#include <iostream>

using namespace std;

class Time {

public:

Time() {

hour = 0;

minute = 0;

second = 0;

}

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

hour = a;

minute = b;

second = c;

}

friend ostream &operator<<(ostream &in, Time &t);

friend istream &operator>>(istream &out, Time &t);

Time operator+(Time a);

Time operator-(Time a);

Time &operator++();

//前置

Time operator++(int);

//后置

Time &operator--();

//前置

Time operator--(int);

//后置

void deal();

protected:

int hour, minute, second;

};

#endif //INC\_2\_1TIMECOUNTER\_TIME\_H

Time.cpp

#include "Time.h"

ostream &operator<<(ostream &out, Time &t) {

out << t.hour << ":" << t.minute << ":" << t.second << endl;

}

istream &operator>>(istream &in, Time &t) {

in >> t.hour >> t.minute >> t.second;

t.deal();

}

Time Time::operator+(Time a) {

Time temp;

temp.hour = this->hour + a.hour;

temp.minute = this->minute + a.minute;

temp.second = this->second + a.second;

temp.deal();

return temp;

}

Time Time::operator-(Time a) {

Time temp;

temp.hour = this->hour - a.hour;

temp.minute = this->minute - a.minute;

temp.second = this->second - a.second;

temp.deal();

return temp;

}

Time &Time::operator++() {

this->second++;

this->deal();

return \*this;

}

Time Time::operator++(int) {

Time temp(\*this);

this->second++;

this->deal();

return temp;

}

Time &Time::operator--() {

this->second--;

this->deal();

return \*this;

}

Time Time::operator--(int) {

Time temp(\*this);

this->second--;

this->deal();

return temp;

}

void Time::deal() {

if (second >= 60) {

minute += second / 60;

second -= 60;

} else if (second < 0) {

minute--;

second += 60;

}

if (minute >= 60) {

hour += minute / 60;

minute -= 60;

} else if (minute < 0) {

hour--;

minute += 60;

}

if (hour >= 24) {

hour -= 24;

} else if (hour < 0) {

hour += 24;

}

}

## 2.4运行结果

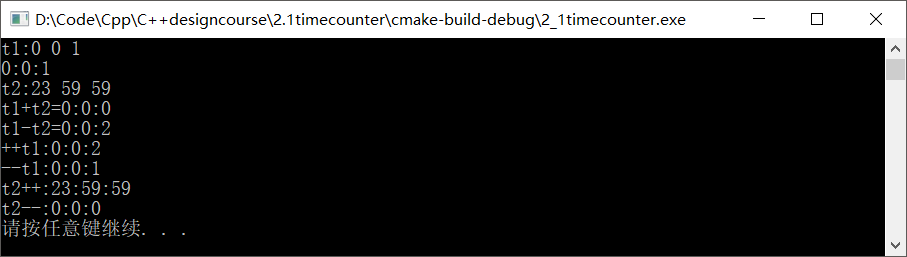


图 2‑1 时间计算器

# 3 判断线段是否与圆相交

## 3.1实验要求

要求：

1. 设计三个类（点、线和圆），每个类有良好的封装性，并合理设计它们之间的继承与派生的关系。
2. 要求程序中使用类对象做函数的参数传递数据。
3. 程序要求有菜单，能循环判断以下功能：
   1. 判断点是否在圆内；
   2. 判断线段是否与圆相交；
   3. 定义二个圆，判断两个圆是否相交。
4. （选做）定义一个椭圆类，判断点是否在椭圆内。
5. （选做）定义一个三角形类，判断三角形是否与圆相交。

## 3.2设计思路

我把Point类作为基类，派生出Round类、Oval类、Triangle类，

1.判断点与圆的位置关系时，计算点到点之间的距离，然后判断点与圆心的距离与圆半径之间的大小关系，

2.判断线段与圆的位置关系要考虑段点与圆的关系，利用直线到圆点的距离公式计算，然后判断，两个端点与圆的关系，如果是在圆一内一外是相交，如果两点在圆内是不相交，如果两点在圆外则要另外判断，

3.判断圆心到直线的距离，两个圆的位置关系要考虑圆心距与两圆半径和的关系，也是计算点到点的距离

4.判断点与椭圆的位置关系时，判断点与f1点和f2点的距离和与2a之间的大小关系，

5.判断三角形与圆的位置关系，我直接将三角形的三条边当作三条线段然后判断三次线段与圆的位置关系。

## 3.3实现代码

main.cpp

#include <iostream>

#include "Point.h"

#include "Line.h"

#include "Round.h"

#include "Oval.h"

#include "Triangle.h"

using namespace std;

int double\_round(Round r1, Round r2);

//两个圆的位置关系

int main() {

int choose;

Round round1, round2;

Point point1;

Line line1;

Oval oval1;

Triangle triangle1;

while (true) {

system("cls");

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl

<< "0.退出" << endl

<< "1.判断点是否在圆内" << endl

<< "2.判断线段是否与圆相交" << endl

<< "3.判断两个圆是否相交" << endl

<< "4.判断点是否在椭圆内" << endl

<< "5.判断三角形是否与圆相交" << endl

<< "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "请输入:";

cin >> choose;

if (choose == 0) break;

else if (choose == 1) {

round1.in();

point1.in();

if (point\_round(point1, round1) == 0)

cout << "点A在圆外" << endl;

else

cout << "点A在圆内" << endl;

system("pause");

} else if (choose == 2) {

round1.in();

line1.in();

int a = line\_round(line1, round1);

if (a == 0)

cout << "线与圆相交" << endl;

else if (a == 1)

cout << "线与圆不交" << endl;

system("pause");

} else if (choose == 3) {

round1.in();

round2.in();

int a = double\_round(round1, round2);

if (a == 0)

cout << "两圆相交" << endl;

else if (a == 1)

cout << "两圆不交" << endl;

system("pause");

} else if (choose == 4) {

//判断点是否在椭圆内

oval1.in();

if (oval1.point\_oval() == 1)

cout << "点在椭圆外" << endl;

else

cout << "点在椭圆内" << endl;

system("pause");

} else if (choose == 5) {

// 判断三角形是否与圆相交

triangle1.in();

if (triangle1.triangle\_round() == 1)

cout << "三角形与圆相交" << endl;

else

cout << "三角形与圆不相交" << endl;

system("pause");

}

}

}

int double\_round(Round r1, Round r2) {

int d = (r1.x - r2.x) \* (r1.x - r2.x) + (r1.y - r2.y) \* (r1.y - r2.y);

if (d >= (r1.r - r2.r) \* (r1.r - r2.r) && d <= (r1.r + r2.r) \* (r1.r + r2.r)) return 0;//相交

else return 1;//不交

}

Point.h

#ifndef INC\_3\_1ROUND\_POINT\_H

#define INC\_3\_1ROUND\_POINT\_H

#include <iostream>

#include <cmath>

using namespace std;

class Point {

public:

Point() {

x = 0;

y = 0;

}

virtual void in();

int x, y;

};

#endif //INC\_3\_1ROUND\_POINT\_H

Point.cpp

#include "Point.h"

void Point::in() {

cout << "点的坐标:";

cin >> x >> y;

}

Line.h

#ifndef INC\_3\_1ROUND\_LINE\_H

#define INC\_3\_1ROUND\_LINE\_H

#include "Round.h"

class Line {

public:

Line() {

p1.x = 0;

p1.y = 0;

p2.x = 0;

p2.y = 0;

}

void in();

friend int line\_round(Line l1, Round r1);

Point p1, p2;

};

#endif //INC\_3\_1ROUND\_LINE\_H

Line.cpp

#include "Line.h"

void Line::in() {

cout << "p1的坐标:";

cin >> p1.x >> p1.y;

cout << "p2的坐标:";

cin >> p2.x >> p2.y;

}

int line\_round(Line l1, Round r1) {//线与圆的位置关系

double a, b, c, dist1, dist2, angle1, angle2;

if (point\_round(l1.p1, r1) == 2 || point\_round(l1.p2, r1) == 2)return 0;

if (point\_round(l1.p1, r1) \* point\_round(l1.p2, r1))return 1;//都在圆内

if (point\_round(l1.p1, r1) ^ point\_round(l1.p2, r1))return 0;//一内一外

if (point\_round(l1.p1, r1) + point\_round(l1.p2, r1) == 0) {//都在圆外

if (l1.p1.x == l1.p2.x)//ax2+bx+c

a = 1, b = 0, c = -l1.p1.x;

else if (l1.p1.y == l1.p2.y)

a = 0, b = 1, c = -l1.p1.y;

else {

a = l1.p1.y - l1.p2.y;

b = l1.p2.x - l1.p1.x;

c = l1.p1.x \* l1.p2.y - l1.p1.y \* l1.p2.x;

}

dist1 = a \* r1.x + b \* r1.y + c;

dist1 \*= dist1;

dist2 = (a \* a + b \* b) \* r1.r \* r1.r;

if (dist1 > dist2) return 1;//不交

angle1 = (r1.x - l1.p1.x) \* (l1.p2.x - l1.p1.x) + (r1.y - l1.p1.y) \* (l1.p2.y - l1.p1.y);

angle2 = (r1.x - l1.p2.x) \* (l1.p1.x - l1.p2.x) + (r1.y - l1.p2.y) \* (l1.p1.y - l1.p2.y);

if (angle1 > 0 && angle2 > 0) return 0;//相交

return 1;

}

}

Round.h

#ifndef INC\_3\_1ROUND\_ROUND\_H

#define INC\_3\_1ROUND\_ROUND\_H

#include "Point.h"

class Round : public Point {

public:

Round() {

x = 0;

y = 0;

r = 0;

}

void in();

friend int point\_round(Point p1, Round r1);//圆与点的位置关系

int r;

};

#endif //INC\_3\_1ROUND\_ROUND\_H

Round.cpp

#include "Round.h"

void Round::in() {

cout << "圆心坐标与半径:";

cin >> x >> y >> r;

}

int point\_round(Point p1, Round r1) {

int d = (p1.x - r1.x) \* (p1.x - r1.x) + (p1.y - r1.y) \* (p1.y - r1.y);

if (r1.r \* r1.r > d) return 1;//点A在圆内

else if (r1.r \* r1.r == d) return 2;//点在圆上

else return 0;//点A在圆外

}

Oval.h

#ifndef INC\_3\_1ROUND\_OVAL\_H

#define INC\_3\_1ROUND\_OVAL\_H

#include "Point.h"

class Oval : public Point {

public:

Oval() {

a = 0;

b = 0;

c = 0;

}

void in();

bool point\_oval();

int type = 0; //默认长轴比短轴长

double a, b, c;

};

#endif //INC\_3\_1ROUND\_OVAL\_H

Oval.cpp

#include "Oval.h"

void Oval::in() {

cout << "椭圆长轴长，短轴长:";

cin >> a >> b;

cout << "点坐标:";

cin >> x >> y;

if (b > a)type = 1; // 短轴比长轴长

a /= 2;

b /= 2;

c = sqrt(a \* a - b \* b);

}

bool Oval::point\_oval() {

double pf1, pf2;

if (type == 0) {

pf1 = y \* y + (x - c) \* (x - c);

pf2 = y \* y + (x + c) \* (x + c);

} else {

pf1 = x \* x + (y - c) \* (y - c);

pf2 = x \* x + (y + c) \* (y + c);

}

if (sqrt(pf1) + sqrt(pf2) > 2 \* a)return 1;//椭圆外

else return 0;//椭圆内

}

Triangle.h

#ifndef INC\_3\_1ROUND\_TRIANGLE\_H

#define INC\_3\_1ROUND\_TRIANGLE\_H

#include "Line.h"

#include "Round.h"

class Triangle {

public:

void in();

Line line1, line2, line3;

Round round1;

bool triangle\_round();

};

#endif //INC\_3\_1ROUND\_TRIANGLE\_H

Triangle.cpp

#include "Triangle.h"

void Triangle::in() {

int a[6];

for (int i = 0; i < 6; i += 2) {

cout << "请输入第" << i / 2 + 1 << "个坐标:";

cin >> a[i] >> a[i + 1];

}

round1.in();

line1.p1.x = a[0];

line1.p1.y = a[1];

line1.p2.x = a[2];

line1.p1.y = a[3];

line2.p1.x = a[0];

line2.p1.y = a[1];

line2.p1.x = a[4];

line2.p1.y = a[5];

line3.p2.x = a[2];

line3.p1.y = a[3];

line3.p1.x = a[4];

line3.p1.y = a[5];

}

bool Triangle::triangle\_round() {

if (line\_round(line1, round1) && line\_round(line2, round1) && line\_round(line3, round1))return 0;//相交

return 1;

}

## 3.4运行结果

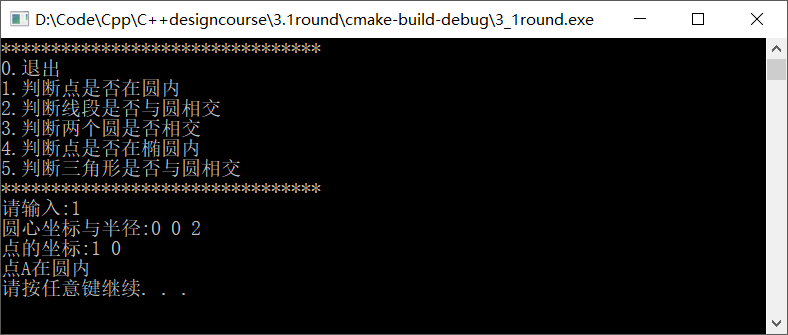


图 3‑1 判断点是否在圆内

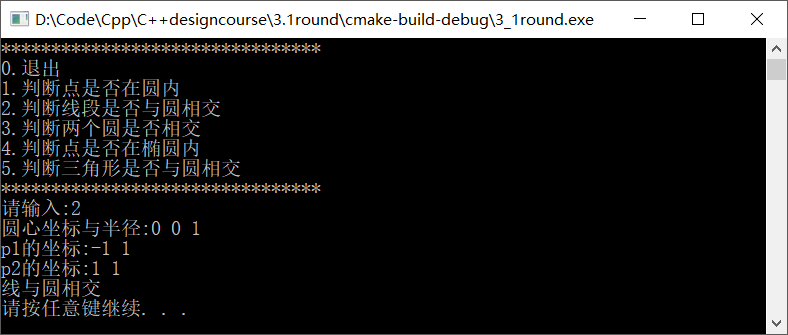


图 3‑2 判断线段是否与圆相交

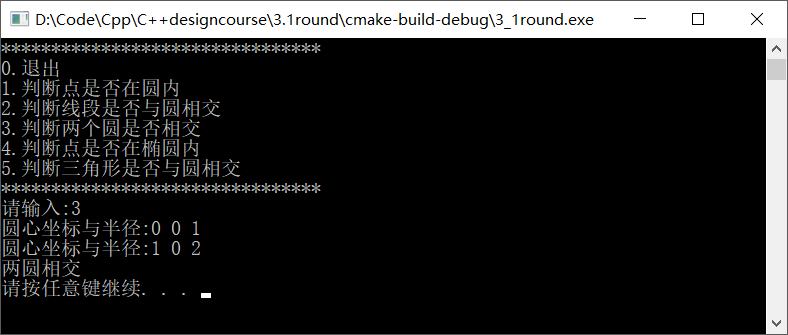


图 3‑3 判断两个圆是否相交

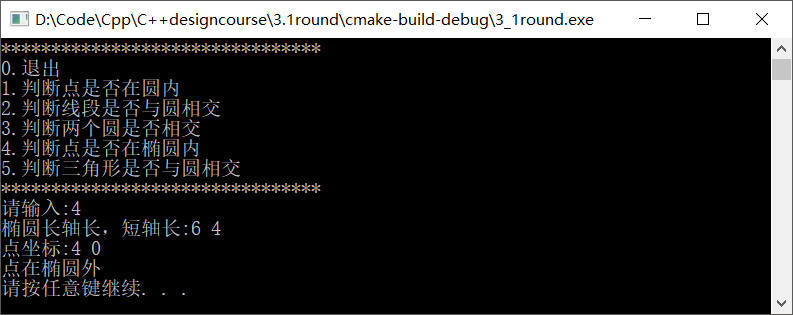


图 3‑4 判断点是否在椭圆内

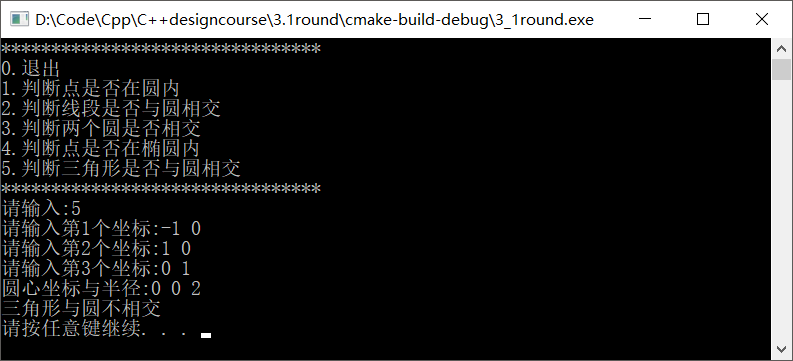


图 3‑5 判断三角形是否与圆相交

# 4 学生成绩管理

## 4.1实验要求

要求：能够对文件进行存储和读取。要求用一个结构记载学生属性，编写一个学生类以完成各种操作。具体要求如下：

1. 有10个小学生，每个学生有数学、语文两门课，从键盘输入学生的学号、姓名及成绩，计算每个学生的平均成绩，将数据存入stud.dat中。
2. 学生的所有属性用一个结构进行描述。
3. 可将两门课都不及格的学生信息存入bad.dat中；将学生姓名和平均成绩抽取出来存入到studmean.dat文件中。
4. 读取文件中的数据存入链表或数组中。
5. 可按学号或姓名查找文件中的记录。

## 4.2设计思路

设计了一个Student类,类中name，id，和语文，数学成绩，而且重载输入输出函数来对Student进行输入输出，再用Function类来封装其他的函数，包括菜单，存入，输出，删除文件，查询文件等等，

1. 菜单函数，我将所有出现菜单的地方调用函数，还利用了函数重载的知识，
2. 存入函数，在判断id为0时，我在类中加了一个值ifvoid，判断是否空，当ifvoid=1时类为空。
3. 输出函数，传入链表使用迭代器遍历输出
4. 删除文件函数，传入链表和删除类型是按学号还是按姓名，并使用c++11标准中的方法遍历容器，找到然后用erase函数删除
5. 查找函数，使用迭代器遍历容器，分别按学号或姓名查找
6. 不及格人数，通过人数计算，返回容器类型，将判断后的数据存入另一个容器然后返回
7. 保存文件函数，利用c++11标准中遍历容器的方法，将string类型，计算其长度，用write写入文件，其他的类型的数据直接用write函数写入文件中。
8. 读取文件函数，当文件不存在时直接返回，这里的循环直到文件末尾，在处理其他类型的文件时都可以用read函数来读取，在读取string文件时则需要一个字符一个字符的读，存入临时类中，最后再压入容器中。

## 4.3实现代码

Main.cpp

#include "Function.h"

int main() {

Function fun;

fun.major();

}

Function.h

#ifndef STUSYSTEM\_FUNCTION\_H

#define STUSYSTEM\_FUNCTION\_H

#include <iostream>

#include "Student.h"

#include <vector>

#include <fstream>

using namespace std;

class Function {

public:

void major();

int menu();

int menu(int choose);

void add(vector<Student> &vs);

void out(vector<Student> &vs);

void deletefile(vector<Student> &vs, int n);

void seek(vector<Student> &vs, int n);

vector<Student> fail(vector<Student> &vs);

vector<Student> pass(vector<Student> &vs);

void outmean(vector<Student> &vs);

};

#endif //STUSYSTEM\_FUNCTION\_H

Function.cpp

#include "Function.h"

void Function::major() {

int cho;

vector<Student> vStuAll;

vector<Student> vStuBad;

vector<Student> vStuMean(vStuAll);

vector<Student> vStuPass;

readfile(vStuAll, "stud.dat");

while (true) {

cho = menu();

if (cho == 9)exit(0);

else if (cho == 1) {

//1.增加原始记录

fstream ifempty("stud.dat", ios::in | ios::binary);

if (!ifempty)

cout << "还没有建立文件，退出时别忘记保存！" << endl;

ifempty.close();

add(vStuAll);

} else if (cho == 2) {

//2.显示各种文件

cho = menu(cho);

if (cho == 1) {

//源文件

//判断记录数量

cout << "现在共有如下" << vStuAll.size() << "条记录" << endl;

out(vStuAll);

} else if (cho == 2) {

//简明文件

vStuMean.clear();

cout << "取文件...." << endl;

readfile(vStuMean, "studmean.dat");

outmean(vStuMean);

} else if (cho == 3) {

//均不及格文件

vStuBad.clear();

cout << "取文件...." << endl;

readfile(vStuBad, "bad.dat");

out(vStuBad);

} else if (cho == 4) {

//输出高于或等于平均成绩的学生名单

vStuPass = pass(vStuAll);

out(vStuPass);

}

} else if (cho == 3) {

//3.保存原始文件

cout << "存文件" << endl;

cout << savefile(vStuAll, "stud.dat") << "条记录已经存入文件，请继续操作。" << endl;

} else if (cho == 4) {

//4.读取原始文件

cout << "取文件...." << endl;

cout << "取入" << readfile(vStuAll, "stud.dat") << "条记录。" << endl;

} else if (cho == 5) {

//5.删除原始记录

cho = menu(cho);

deletefile(vStuAll, cho);

} else if (cho == 6) {

//6.建立简明文件

cout << "存文件" << endl;

vStuMean = vStuAll;

cout << savefile(vStuMean, "studmean.dat") << "条记录已经存入文件，内容如下：" << endl;

outmean(vStuMean);

} else if (cho == 7) {

cout << "两科均不及格的学生信息如下：" << endl;

vStuBad = fail(vStuAll);

out(vStuBad);

savefile(vStuBad, "bad.dat");

cout << "文件存储成功" << endl;

//7.均不及格文件

} else if (cho == 8) {

//8.寻找链表记录

cho = menu(cho);

seek(vStuAll, cho);

}

}

}

int Function::menu() {

int choose;

while (true) {

cout << "1.增加原始记录" << endl

<< "2.显示各种文件" << endl

<< "3.保存原始文件" << endl

<< "4.读取原始文件" << endl

<< "5.删除原始记录" << endl

<< "6.建立简明文件" << endl

<< "7.均不及格文件" << endl

<< "8.寻找链表记录" << endl

<< "9.结束程序运行" << endl;

cout << "左边数字对应相应功能选择，请选1-9：";

cin >> choose;

cin.clear();

cin.sync();

if (choose >= 1 && choose <= 9)break;

else cout << "输入错误！" << endl;

}

return choose;

}

int Function::menu(int choose) {

if (choose == 2) {//第二项的二级菜单

cout << "选择要看的文件类别" << endl

<< "1.原文件" << endl

<< "2.简明文件" << endl

<< "3.均不及格文件" << endl

<< "4.输出高于或等于平均成绩的学生名单" << endl;

cout << "请选1-4：";

cin >> choose;

cin.clear();

cin.sync();

return choose;

} else if (choose == 8 || choose == 5) {//第八,五项的二级菜单

cout << "1.通过学号寻找" << endl

<< "2.通过名字寻找" << endl;

cout << "请选1-2：";

cin >> choose;

cin.clear();

cin.sync();

return choose;

}

}

void Function::add(vector<Student> &vs) {

Student temp;

cout << "输入数据，输入0时结束。" << endl;

while (true) {

cin >> temp;

if (temp.ifvoid == 1)

break;

vs.push\_back(temp);//添加到容器中

}

}

void Function::out(vector<Student> &vs) {

cout << "学号\t姓名\t数学\t语文\t平均" << endl;

vector<Student>::iterator iter;//迭代器使用

for (iter = vs.begin(); iter != vs.end(); iter++) {

cout << \*iter << endl;

}

}

void Function::deletefile(vector<Student> &vs, int n) {

//fstream file("stud.dat", ios::trunc);

string info;

int num = 0, count = 0;

if (n == 1)cout << "请输入学号：";

else if (n == 2)cout << "请输入名字：";

cin >> info;

cout << "被删除记录信息如下：" << endl;

cout << "学号\t姓名\t数学\t语文\t平均" << endl;

for (auto i :vs) {

if (n == 1) {

if (info == i.outputid()) {

cout << i << endl;

num++;

vs.erase(vs.begin() + count);

}

} else if (n == 2) {

if (info == i.outputname()) {

cout << i << endl;

num++;

vs.erase(vs.begin() + count);

}

}

count++;

}

if (num == 0) {

cout << "没有找到此人" << endl;

return;

}

cout << "当前选择保存记录之后，才能确定删除。" << endl;

}

void Function::seek(vector<Student> &vs, int n) {

string info;

int count = 0;

vector<Student>::iterator iter;

if (n == 1)cout << "请输入学号：";

else if (n == 2)cout << "请输入名字：";

cin >> info;

cout << "找到信息如下：" << endl;

cout << "学号\t姓名\t数学\t语文\t平均" << endl;

for (iter = vs.begin(); iter != vs.end(); iter++) {

if (n == 1) {

if (info == iter->outputid()) {

cout << \*iter << endl;

count++;

}

} else if (n == 2) {

if (info == iter->outputname()) {

cout << \*iter << endl;

count++;

}

}

}

if (count == 0)

cout << "没有找到此人" << endl;

}

vector<Student> Function::fail(vector<Student> &vs) {

vector<Student> temp;

for (auto i : vs) {

if (i.outputchinese() < 60 && i.outputmath() < 60)

temp.push\_back(i);

}

return temp;

}

vector<Student> Function::pass(vector<Student> &vs) {

vector<Student> temp;

double average = 0;

for (auto i : vs)

average += i.outputaver();

average /= vs.size();

for (auto i : vs) {

if (i.outputaver() >= average)

temp.push\_back(i);

}

return temp;

}

void Function::outmean(vector<Student> &vs) {

cout << "姓名\t平均成绩" << endl;

vector<Student>::iterator iter;//迭代器使用

for (iter = vs.begin(); iter != vs.end(); iter++) {

cout << iter->outputname() << "\t" << iter->outputaver() << endl;

}

}

Student.h

#ifndef STUSYSTEM\_STUDENT\_H

#define STUSYSTEM\_STUDENT\_H

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

class Student {

public:

friend istream &operator>>(istream &in, Student &s);//重载输入输出流

friend ostream &operator<<(ostream &in, Student &s);

int ifvoid = 0;

string outputid();

string outputname();

int outputmath();

int outputchinese();

double outputaver();

friend int savefile(vector<Student> &vs, const string &address);

friend int readfile(vector<Student> &vs, const string &address);

~Student() = default;

private:

string id, name;

int score\_math, score\_chinese;

double aver;

};

#endif //STUSYSTEM\_STUDENT\_H

Student.cpp

#include "Student.h"

istream &operator>>(istream &in, Student &s) {

cout << "学号：";

in >> s.id;

if (s.id == "0") {

s.ifvoid = 1;

return in;

}

cout << "姓名：";

in >> s.name;

cout << "数学：";

in >> s.score\_math;

cout << "语文：";

in >> s.score\_chinese;

s.aver = ((double) s.score\_chinese + (double) s.score\_math) / 2;

return in;

}

ostream &operator<<(ostream &out, Student &s) {

out << s.id << "\t" << s.name << "\t"

<< s.score\_math << "\t"

<< s.score\_chinese << "\t"

<< s.aver;

return out;

}

string Student::outputid() {

return id;

}

string Student::outputname() {

return name;

}

int Student::outputmath() {

return score\_math;

}

int Student::outputchinese() {

return score\_chinese;

}

double Student::outputaver() {

return aver;

}

int savefile(vector<Student> &vs, const string &address) {

ofstream tofile(address, ios::out | ios::trunc | ios::binary);

int Strlen;

int count = 0;

for (auto i:vs) {

count++;

Strlen = (int) i.id.length();

tofile.write((char \*) &Strlen, sizeof(int));

tofile.write(i.id.c\_str(), Strlen);

Strlen = (int) i.name.length();

tofile.write((char \*) &Strlen, sizeof(int));

tofile.write(i.name.c\_str(), Strlen);

tofile.write((char \*) &i.score\_math, sizeof(int));

tofile.write((char \*) &i.score\_chinese, sizeof(int));

tofile.write((char \*) &i.aver, sizeof(double));

}

tofile.close();

return count;

}

int readfile(vector<Student> &vs, const string &address) {

vs.clear();

fstream fromfile(address, ios::in | ios::binary);

if (!fromfile)

return 0;

Student temp;

int count = 0, StrLen;

char tChar;

while (fromfile.read((char \*) &StrLen, sizeof(int))) {

count++;

while (StrLen--) fromfile.read(&tChar, sizeof(char)), temp.id.push\_back(tChar);

fromfile.read((char \*) &StrLen, sizeof(int));

while (StrLen--) fromfile.read(&tChar, sizeof(char)), temp.name.push\_back(tChar);

fromfile.read((char \*) &temp.score\_math, sizeof(int));

fromfile.read((char \*) &temp.score\_chinese, sizeof(int));

fromfile.read((char \*) &temp.aver, sizeof(double));

vs.push\_back(temp);

temp.id.clear();

temp.name.clear();

}

fromfile.close();

return count;

}

## 4.4运行结果

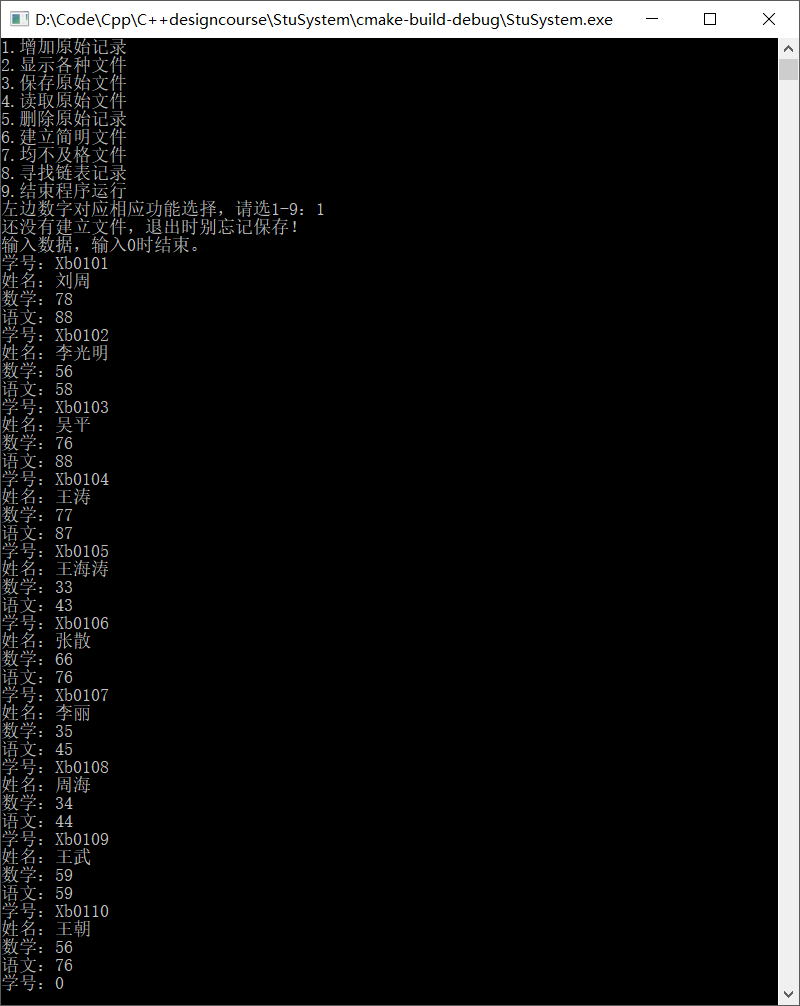


图 4‑1 增加原始记录

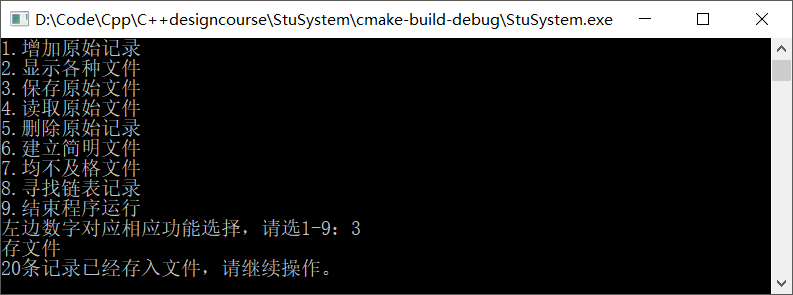


图 4‑2 保存原始文件

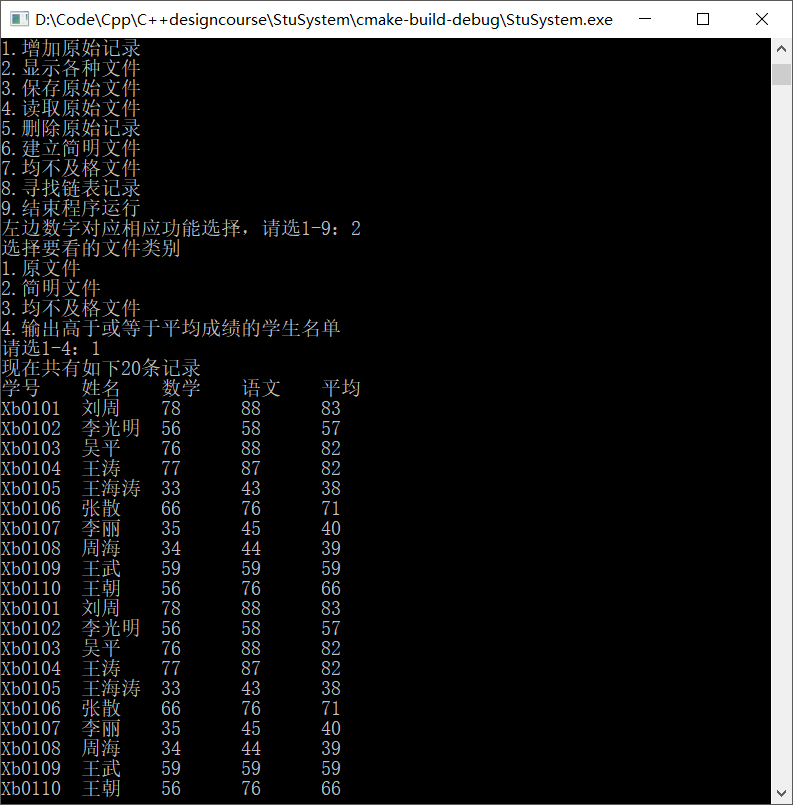


图 4‑3 显示原文件

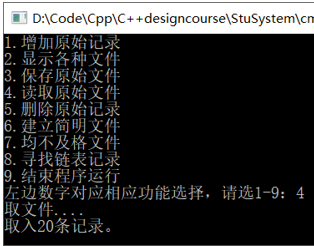
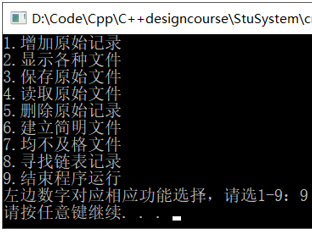
 

图 4‑4 读取原始文件 图 4‑5 结束程序运行

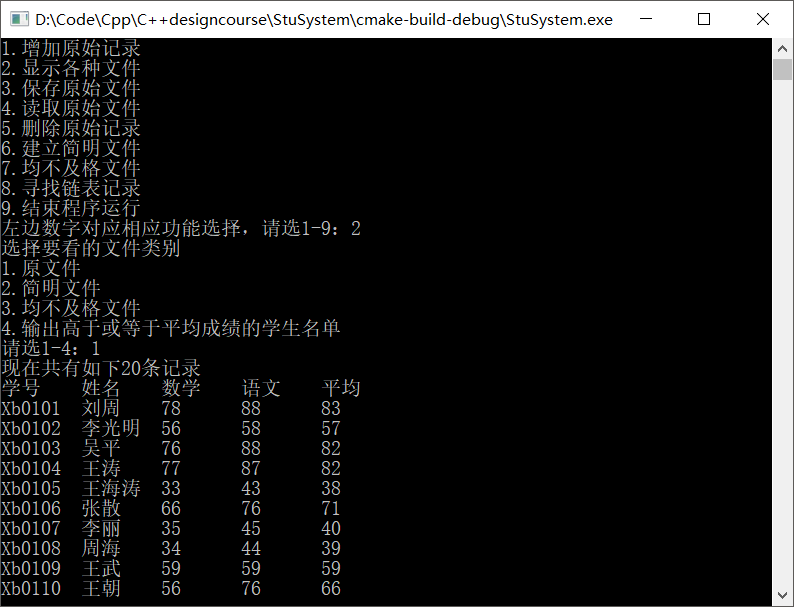


图 4‑6 显示原文件

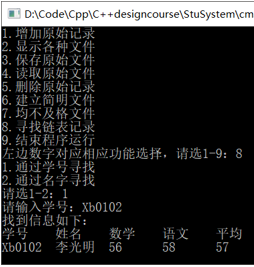
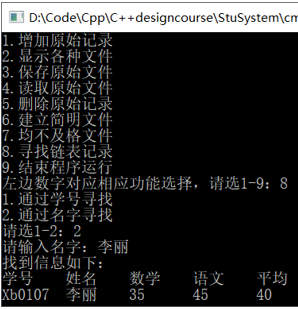
 

图 4‑7 按学号寻找链表记录 图 4-8 按姓名寻找链表记录

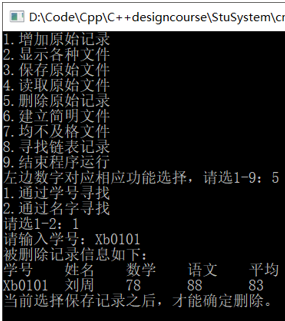
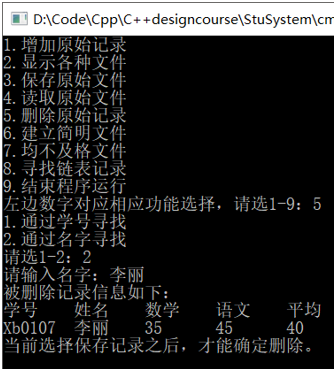
 

图 4‑9 按学号删除原始记录 图 4‑10 按姓名删除原始记录

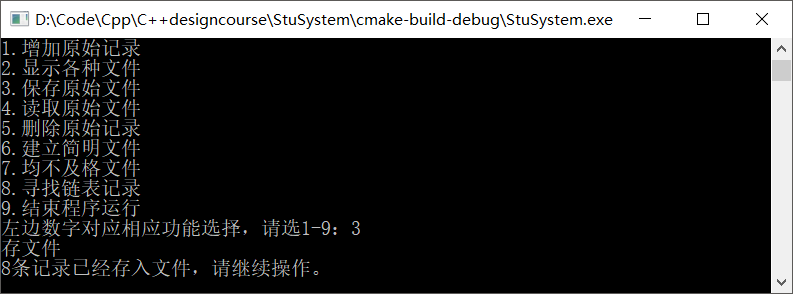


图 4-11 保存原始文件

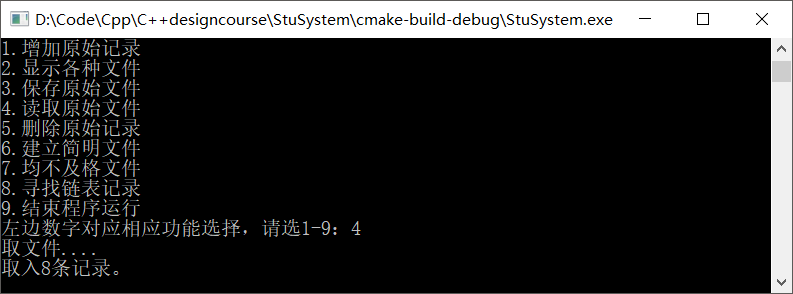


图 4-12 读取原始文件

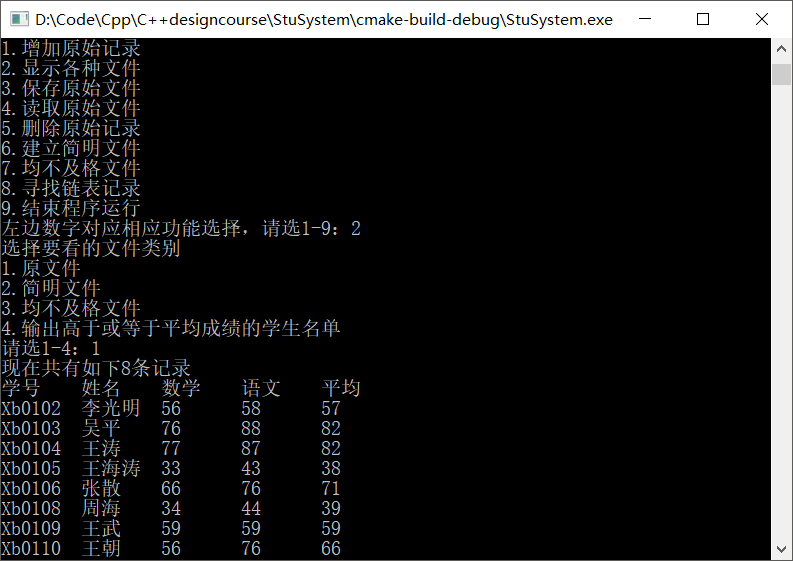


图 4-13 显示原文件

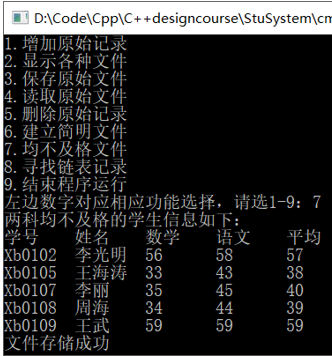
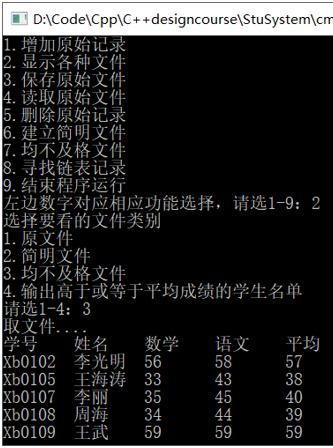
 

图 4-14 均不及格文件 图 4-15 显示均不及格文件



图 4-16 建立简明文件

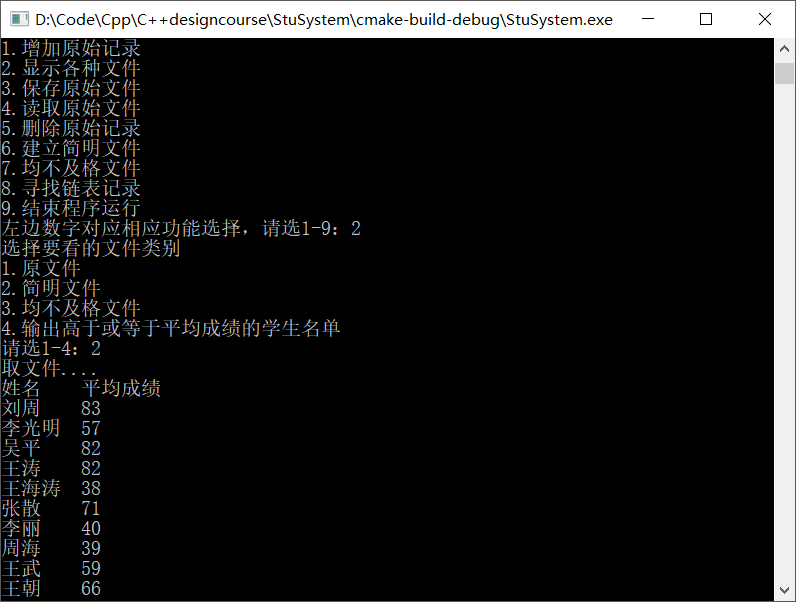


图 4-17 显示简明文件

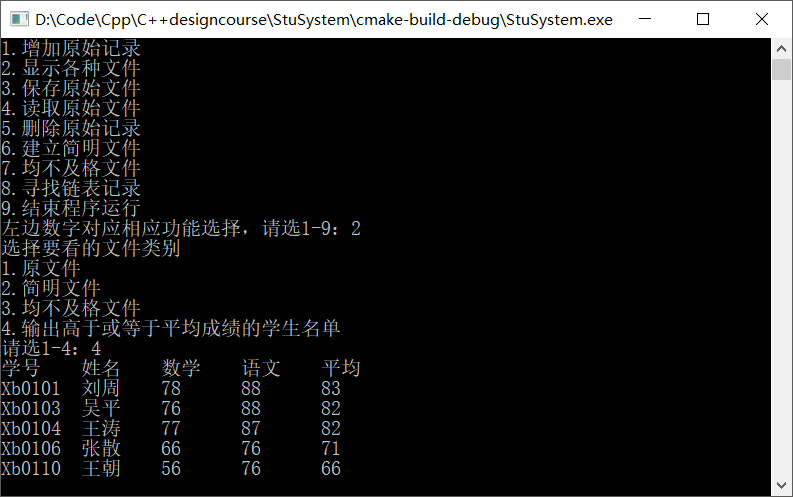


图 4-18 输出高于或等于平均成绩的学生名单

# 5 总结或感悟

通过本次课程设计，我学习到了很多关于C++的知识，并且对C++类的使用，文件读写，有了更深的理解，要学好这们语言，仅仅学习课本上的知识是不够的，还要经常自己动手，自己查找相关资料，要多动手实践。只有多动手，多写代码，才能发现自己在学习上的漏洞和自己的不足，并在实践中解决这些问题，只有这样才能不断提高自己的能力。

在这次课程设计中前几个题目是比较简单的，不复杂，要求明白类的使用，构造，析构的使用，之前还好我有较扎实的基础所以不难解决，后面两个任务难度加大了一些，我会主动查阅相关的资料，向其他同学询问，解决了这些难题，例如，在对string类型的数据进行储存到文件之中，我遇到了一些问题，发现string类型只是存了一个指针，而不是像其他类型一样将所以内容存入，因此在存入文件时，发生了错误，读入文件时也发生了错误，为了解决这个问题，我先是到网络上查找了相关的问题，明白了问题产生的原因，但是网络上的解决方式不太能理解，我于是问了同专业的同学，经过我们的不懈努力，我们终于有了突破，我成功的完成了这个功能，读写文件正常。

在这次课程设计中的后两题，我首先对系统的整体功能进行了构思，然后对题目进行分析，将整个程序清楚的划分为几个模块，再根据每个模块的功能编写代码。而且尽可能的将模块细分，最后再进行函数的调用。在调试的过程中也遇到了困难，但经过我耐心的修改，终于功夫不负有心人，我完成了这次的课程设计。

这次课程设计即将结束，时间虽短，但是我收获了很多。最后，谢谢老师和同学们的指导和帮助，谢谢大家。