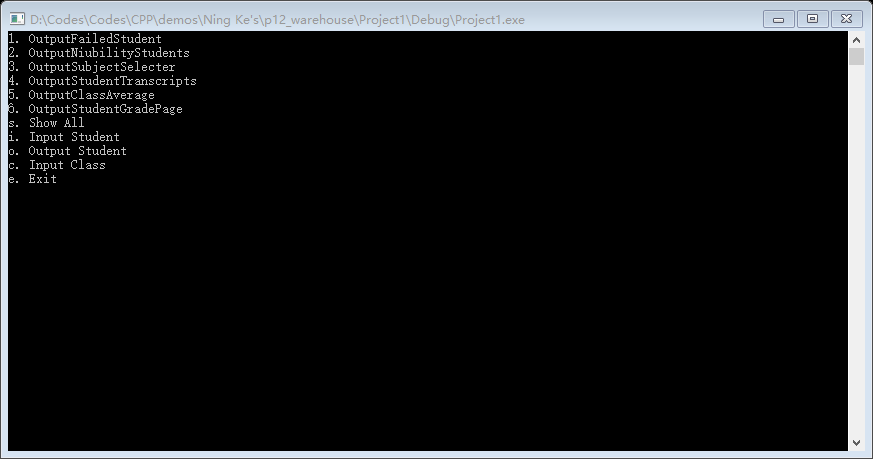
**思路：**

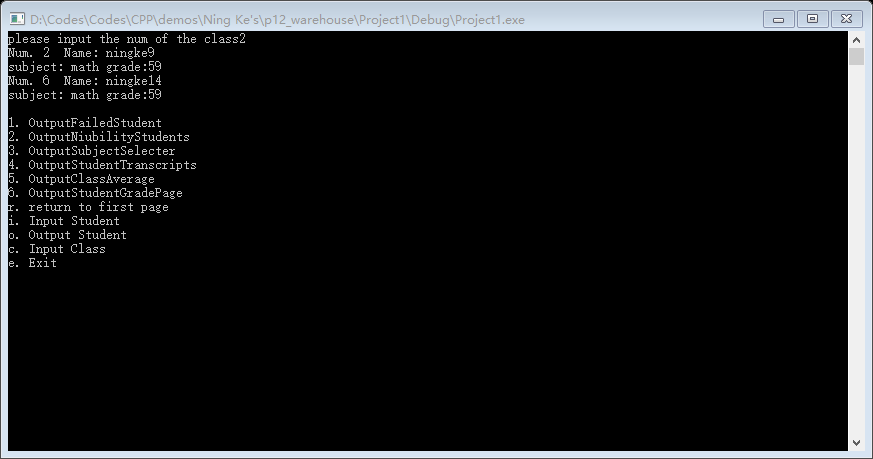
先构建好一个类似于学生管理系统的基础框架，实现学生信息的存取，添加，查看等基础功能；然后根据对于成品的要求进行针对性的方法添加，实现多样的功能；

数据处理时使用了链表和stl中的map，链表是为了对文件中的学生基础信息进行预读入，大大提升了运行时的效率与动态响应，且对程序启动效率几乎没有影响，内存占用也处于可以接受的范围内。在整体程序运行结束时，调用exitAndSave方法使链表中的数据根据既定规则进行文件的写入，以实现保存数据的目的。而在涉及到排序的方法中，为使代码更加简洁，因为stl中map类型的内部实现为红黑树，具有key值有序性的特征，所以使用map而不是数组来进行中介存储，不仅提升了代码的可读性，易于编写，更是提供了灵活的数据处理方式，提升了程序的可复用性与可修改性。

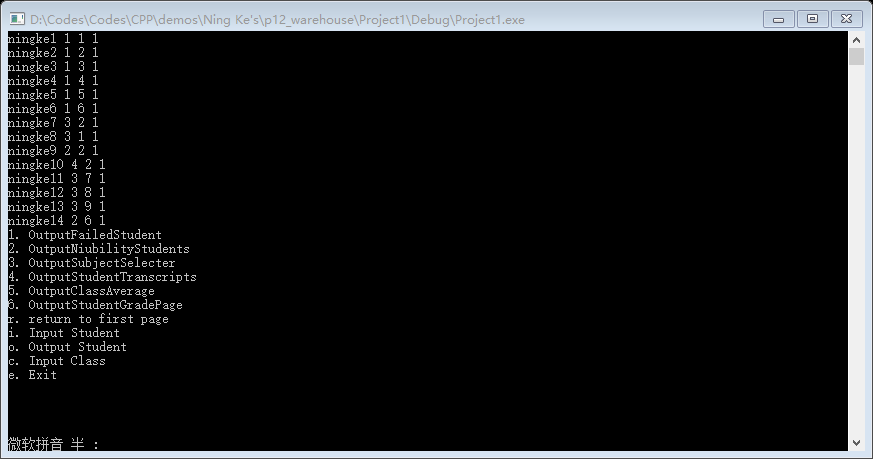
**运行结果样例：**



启动界面



某班不及格学生统计



显示所有学生信息

**使用说明：**

根据界面提示直接按键进行动态响应操作，具体功能为：

1. //分班的某门课程不及格的名单(含学号、姓名、成绩)；
2. //某门课程全年级前5名的学号、姓名、成绩；
3. //某门课程的选修者名单(含学号)；
4. //某个学生本学期的成绩单；
5. //某门课程每个班的总平均分(从高到低排列)；
6. //某门课程某班的成绩单(按学号排列)。

r. //返回主界面

i. //输入学生基本信息

o. //输出某个学生信息

s. //输出所有学生基本信息

c. //输入某个学生的课程信息

e. //结束并保存

**源代码：**

#include <iostream>

#include <conio.h>

#include <fstream>

#include <string>

#include <map>

#include <iomanip>

using namespace std;

const int FULL\_GRADE = 100;

struct Subject {

public:

string subjectName;

int subjectGradeFirst = 0, subjectGradeSecond = 0;

};

struct node

{

string name;

string classNum;

string studentNum; //value will be selected between 1 && 2

int term;

node \*next;

int subjectCount = 0;

Subject\* collectionOfSubjects[100] = { NULL };

};

struct classroom {

string classNum;

int scoreSum = 0;

int studentCount = 0;

};

bool isExit = false;

struct node \*head = NULL, \*p, \*q = NULL, \*t = NULL;

void FileInputAndDo();

void showAll();

void inStudent();

void exitAndSave();

void switchFunction(char dir);

void firstPage();

void showStudent();

void InSubject();

int max(int a, int b);

void OutputFailedStudent(); //分班的某门课程不及格的名单(含学号、姓名、成绩)；

void OutputNiubilityStudents(); //某门课程全年级前5名的学号、姓名、成绩；

void OutputSubjectSelecter(); //某门课程的选修者名单(含学号)；

void OutputStudentTranscripts(); //某个学生本学期的成绩单；

void OutputClassAverage(); //某门课程每个班的总平均分(从高到低排列)；

void OutputStudentGradePage(); //某门课程某班的成绩单(按学号排列)。 //use map

node\* findString(string tmp);

int main() {

FileInputAndDo();

return 0;

}

void OutputStudentGradePage() {

system("cls");

string classNum, subjectName;

cout << "please input the class :";

cin >> classNum;

cout << "please input the name of subject: ";

cin >> subjectName;

cout << endl;

map <string, int> collectionOfGrades;

node \*t = head;;

while (t != NULL) {

if (t->classNum == classNum) {

for (int tmpOut = 1; tmpOut <= t->subjectCount; tmpOut++) {

if (t->collectionOfSubjects[tmpOut]->subjectName == subjectName) {

int finalGradeTmp = max(t->collectionOfSubjects[tmpOut]->subjectGradeFirst, t->collectionOfSubjects[tmpOut]->subjectGradeSecond);

collectionOfGrades.insert(pair<string, int>(t->studentNum, finalGradeTmp));

}

}

}

t = t->next;

}

cout << "The Grade List Of " << subjectName << " In Class " << classNum << " :" << endl;

map<string, int>::iterator itr = collectionOfGrades.begin();

while (itr != collectionOfGrades.end()) {

cout << "Num. " << itr->first << " name: " << t->name << " grade: " << itr->second << endl;

itr++;

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void OutputClassAverage() {

system("cls");

cout << "please input a subject: ";

string subjectName;

cin >> subjectName;

map<string, classroom\*> collectionOfClassrooms;

node \*t = head;

while (t != NULL) {

for (int tmp = 1; tmp <= t->subjectCount; tmp++) {

if (t->collectionOfSubjects[tmp]->subjectName == subjectName) {

map<string, classroom\*>::iterator results = collectionOfClassrooms.find(t->classNum);

if (results == collectionOfClassrooms.end()) {

classroom \*classroomPointer = new classroom;

collectionOfClassrooms.insert(pair<string, classroom\*>(t->classNum, classroomPointer));

}

results = collectionOfClassrooms.find(t->classNum);

int finalGradeTmp = max(t->collectionOfSubjects[tmp]->subjectGradeFirst, t->collectionOfSubjects[tmp]->subjectGradeSecond);

results->second->scoreSum += finalGradeTmp;

results->second->studentCount++;

break;

}

}

t = t->next;

}

map<string, classroom\*>::iterator itr = collectionOfClassrooms.begin();

map<float, string> onlyForSort;

while (itr != collectionOfClassrooms.end()) {

onlyForSort.insert(pair<float, string>((float)itr->second->scoreSum / itr->second->studentCount, itr->first));

itr++;

}

map<float, string>::iterator itrForOutput = onlyForSort.begin();

while (itrForOutput != onlyForSort.end()) {

cout << "class: " << itrForOutput->second << " Average Grade: ";

cout << setprecision(2) << setiosflags(ios::showpoint || ios::fixed) << itrForOutput->first << endl;

itrForOutput++;

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void OutputStudentTranscripts() {

system("cls");

cout << "please input the student number of the student" << endl;

string studentNum;

cin >> studentNum;

node \*t = head;

while (t != NULL) {

if (t->studentNum == studentNum) {

cout << "Num. " << t->studentNum << " name: " << t->name << endl;

for (int tmpOut = 1; tmpOut <= t->subjectCount; tmpOut++) {

cout << "subject: " << t->collectionOfSubjects[tmpOut]->subjectName << " gradeOne: " << t->collectionOfSubjects[tmpOut]->subjectGradeFirst << " gradeTwo: " << t->collectionOfSubjects[tmpOut]->subjectGradeSecond << endl;

}

}

t = t->next;

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void OutputSubjectSelecter() {

system("cls");

cout << "please input the subject" << endl;

string subjectName;

cin >> subjectName;

node \*t = head;

while (t != NULL) {

for (int tmpOut = 1; tmpOut <= t->subjectCount; tmpOut++) {

if (t->collectionOfSubjects[tmpOut]->subjectName == subjectName) {

cout << "Num. " << t->studentNum << " name: " << t->name << endl;

}

break;

}

t = t->next;

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

int max(int a, int b) {

if (a > b) {

return a;

}

else {

return b;

}

}

void OutputNiubilityStudents() {

system("cls");

cout << "please input the subject: ";

string subjectName;

cin >> subjectName;

node \*t = head;

struct node\* collectionOfNiubilityStudents[5] = { NULL };

int minGrade = 0;

while (t != NULL) {

for (int tmpOut = 1; tmpOut <= t->subjectCount; tmpOut++) {

if (t->collectionOfSubjects[tmpOut]->subjectName == subjectName) {

int finalGrade = max(t->collectionOfSubjects[tmpOut]->subjectGradeSecond, t->collectionOfSubjects[tmpOut]->subjectGradeFirst);

if (finalGrade > minGrade) {

for (int tmpIn = 0; tmpIn <= 4; tmpIn++) {

if (collectionOfNiubilityStudents[tmpIn] == NULL) {

collectionOfNiubilityStudents[tmpIn] = t;

}

else {

int subjectCount;

for (int tmp = 1; tmp <= collectionOfNiubilityStudents[tmpIn]->subjectCount; tmp++) {

if (collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[tmp]->subjectName == subjectName) {

subjectCount = tmp;

break;

}

}

int finalGradeInnerTmp = max(collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeFirst, collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeSecond);

if (finalGradeInnerTmp == minGrade) {

collectionOfNiubilityStudents[tmpIn] = t;

}

}

}

//重新获取minGrade

minGrade = FULL\_GRADE;

for (int tmpIn = 0; tmpIn <= 4; tmpIn++) {

int subjectCount;

for (int tmp = 1; tmp <= collectionOfNiubilityStudents[tmpIn]->subjectCount; tmp++) {

if (collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[tmp]->subjectName == subjectName) {

subjectCount = tmp;

break;

}

}

int finalGradeInnerTmp = max(collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeFirst, collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeSecond);

if (finalGradeInnerTmp < minGrade) {

minGrade = finalGradeInnerTmp;

}

}

}

}

break;

}

t = t->next;

}

for (int tmpIn = 0; tmpIn <= 4; tmpIn++) {

if (collectionOfNiubilityStudents[tmpIn] != NULL) {

int subjectCount;

for (int tmp = 1; tmp <= collectionOfNiubilityStudents[tmpIn]->subjectCount; tmp++) {

if (collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[tmp]->subjectName == subjectName) {

subjectCount = tmp;

break;

}

}

int finalGradeInnerTmp = max(collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeFirst, collectionOfNiubilityStudents[tmpIn]->collectionOfSubjects[subjectCount]->subjectGradeSecond);

cout << "class : " << collectionOfNiubilityStudents[tmpIn]->classNum << " Num. " << collectionOfNiubilityStudents[tmpIn]->studentNum << " name: " << collectionOfNiubilityStudents[tmpIn]->name << " grade: " << finalGradeInnerTmp << endl;

}

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void InSubject() {

system("cls");

cout << "please input the name,subject,first grade and second grade" << endl;

Subject\* subjectPointer = NULL;

string name, subjectName;

int gradeOne, gradeTwo;

cin >> name;

while (name != "exit") {

t = findString(name);

if (t != NULL) {

cin >> subjectName >> gradeOne >> gradeTwo;

subjectPointer = new Subject;

subjectPointer->subjectName = subjectName;

subjectPointer->subjectGradeFirst = gradeOne;

subjectPointer->subjectGradeSecond = gradeTwo;

t->collectionOfSubjects[t->subjectCount + 1] = subjectPointer;

t->subjectCount++;

break;

}

else {

cout << "can't find this people!" << endl;

}

cin >> name;

}

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "s. showAll" << endl;

cout << "o. Output A Student" << endl;

cout << "e. Exit" << endl;

}

node\* findString(string tmp) {

node \*t = head;

while (t != NULL) {

if (t->name == tmp) {

return t;

}

t = t->next;

}

return NULL;

}

void firstPage() {

system("cls");

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "s. Show All" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void showStudent() {

system("cls");

cout << "please input the name of the student who you want to inquire" << endl;

string name;

node\* innerTmp;

cin >> name;

innerTmp = findString(name);

if (innerTmp != NULL) {

cout << "name: " << innerTmp->name << endl;

cout << "class: " << innerTmp->classNum << endl;

cout << "Num. " << innerTmp->studentNum << endl;

if (innerTmp->term == 1) {

cout << "Term: UP";

}

else {

cout << "Term: DOWN";

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "s. Show All" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "e. Exit" << endl;

}

else {

cout << "The Student Is Not Included In The Database,Please Check Your Input" << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "s. Show All" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

}

void FileInputAndDo() {

fstream fin;

fin.open("Student.sto", ios\_base::in);

string name, classNum, studentNum;

int term;

Subject\* subjectPointer = NULL;

while (1) {

fin >> name >> classNum >> studentNum >> term;

if (fin.eof()) {

break;

}

p = new node;

p->name = name;

p->classNum = classNum;

p->studentNum = studentNum;

p->term = term;

string subjectName;

int gradeOne, gradeTwo;

fin >> subjectName;

while (subjectName != "endOfLine") {

fin >> gradeOne >> gradeTwo;

subjectPointer = new Subject;

subjectPointer->subjectName = subjectName;

subjectPointer->subjectGradeFirst = gradeOne;

subjectPointer->subjectGradeSecond = gradeTwo;

p->collectionOfSubjects[p->subjectCount + 1] = subjectPointer;

p->subjectCount++;

fin >> subjectName;

}

p->next = NULL;

if (head == NULL) {

head = p;

}

else {

q->next = p;

}

q = p;

}

firstPage();

char dir;

while (!isExit) {

if (\_kbhit() != 0) {

dir = \_getch();

if (dir == 's' || dir == 'i' || dir == 'o' || dir == 'e' || dir == 'r' || dir == 'c' || dir == '1' || dir == '2' || dir == '3' || dir == '4' || dir == '5' || dir == '6') {

switchFunction(dir);

}

}

}

}

void switchFunction(char dir) {

switch (dir) {

case 's': {

showAll();

break;

}

case 'i': {

inStudent();

break;

}

case 'o': {

showStudent();

break;

}

case 'e': {

exitAndSave();

break;

}

case 'r': {

firstPage();

break;

}

case 'c': {

InSubject();

break;

}

case '1': {

OutputFailedStudent();

break;

}

case '2': {

OutputNiubilityStudents();

break;

}

case '3': {

OutputSubjectSelecter();

break;

}

case '4': {

OutputStudentTranscripts();

break;

}

case '5': {

OutputClassAverage();

break;

}

case '6': {

OutputStudentGradePage();

break;

}

default: {

break;

}

}

}

void showAll() {

system("cls");

string Name;

node \*t = head;

while (t != NULL) {

cout << t->name << ' ' << t->classNum << ' ' << t->studentNum << ' ' << t->term << ' ' << endl;

// what do you want to do to all the things

t = t->next;

}

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void inStudent() {

system("cls");

cout << "please input the name and the add num of your items(exit to more function)" << endl;

string name, classNum, studentNum;

int term;

cin >> name;

while (name != "exit") {

cin >> classNum >> studentNum >> term;

node \*t = head;

while (t != NULL) {

if (t->next == NULL) {

p = new node;

p->name = name;

p->classNum = classNum;

p->studentNum = studentNum;

p->term = term;

p->next = NULL;

t->next = p;

break;

}

t = t->next;

}

cin >> name;

}

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "s. showAll" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}

void exitAndSave() {

system("cls");

cout << "writing files,please don't close the window" << endl;

ofstream fout;

fout.open("Student.sto", ios\_base::out);

fout.clear();

t = head;

while (t != NULL) {

fout << t->name << ' ' << t->classNum << ' ' << t->studentNum << ' ' << t->term << ' ';

for (int tmp = 1; tmp <= t->subjectCount; tmp++) {

fout << t->collectionOfSubjects[tmp]->subjectName << ' ' << t->collectionOfSubjects[tmp]->subjectGradeFirst << ' ' << t->collectionOfSubjects[tmp]->subjectGradeSecond << ' ';

}

fout << "endOfLine" << endl;

t = t->next;

}

fout.close();

cout << "save finished!";

isExit = true;

}

void OutputFailedStudent() {

system("cls");

string classNum;

cout << "please input the num of the class";

cin >> classNum;

node \*t = head;

bool firstOutput = true;

while (t != NULL) {

if (t->classNum == classNum) {

for (int tmp = 1; tmp <= t->subjectCount; tmp++) {

int finalGrade = max(t->collectionOfSubjects[tmp]->subjectGradeSecond, t->collectionOfSubjects[tmp]->subjectGradeFirst);

if (finalGrade < (0.6\*FULL\_GRADE)) {

if (firstOutput) {

cout << "Num. " << t->studentNum << " Name: " << t->name << ' ' << endl;

}

cout << "subject: " << t->collectionOfSubjects[tmp]->subjectName << " grade:" << finalGrade << endl;

}

}

}

t = t->next;

}

cout << endl;

cout << "1. OutputFailedStudent" << endl;

cout << "2. OutputNiubilityStudents" << endl;

cout << "3. OutputSubjectSelecter" << endl;

cout << "4. OutputStudentTranscripts" << endl;

cout << "5. OutputClassAverage" << endl;

cout << "6. OutputStudentGradePage" << endl;

cout << "r. return to first page" << endl;

cout << "i. Input Student" << endl;

cout << "o. Output A Student" << endl;

cout << "c. Input Class" << endl;

cout << "e. Exit" << endl;

}