项目说明文档

数据结构课程设计

——排课系统

作 者 姓 名： 罗佳瑞

学 号： 1952528

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc12903)

[1.1 背景分析 1](#_Toc3048)

[1.2 功能分析 1](#_Toc5111)

[2 设计 2](#_Toc8260)

[2.1 数据结构设计 2](#_Toc10085)

[2.2 类结构设计 2](#_Toc405)

[2.3 成员与操作设计 2](#_Toc2360)

[2.4 系统设计 3](#_Toc16616)

[3 实现 4](#_Toc14974)

[3.1 插入功能的实现 4](#_Toc1655)

[3.1.1 插入功能流程图 4](#_Toc26699)

[3.1.2 插入功能核心代码 4](#_Toc25029)

[3.1.3 插入功能截屏示例 5](#_Toc2023)

[3.2 删除功能的实现 6](#_Toc20351)

[3.2.1 删除功能流程图 6](#_Toc13742)

[3.2.2 删除功能核心代码 7](#_Toc19312)

[3.2.3 删除功能截屏示例 7](#_Toc29616)

[3.3 查找功能的实现 8](#_Toc14091)

[3.3.1 查找功能流程图 8](#_Toc3732)

[3.3.2 查找功能核心代码 8](#_Toc24925)

[3.3.3 查找功能截图示例 9](#_Toc25161)

[3.4 修改功能的实现 10](#_Toc9053)

[3.4.1 修改功能流程图 10](#_Toc17802)

[3.4.2 修改功能核心代码 11](#_Toc23543)

[3.4.3 修改功能截屏示例 11](#_Toc23108)

[3.5 统计功能的实现 12](#_Toc12620)

[3.5.1 统计功能流程图 12](#_Toc12890)

[3.5.2 统计功能核心代码 12](#_Toc26864)

[3.5.3 统计功能截屏示例 13](#_Toc29532)

[3.6 总体系统的实现 14](#_Toc30521)

[3.6.1 总体系统流程图 14](#_Toc14639)

[3.6.2 总体系统核心代码 15](#_Toc9686)

[3.6.3 总体系统截屏示例 16](#_Toc14007)

[4 测试 17](#_Toc29765)

[4.1 功能测试 17](#_Toc19707)

[4.2 边界测试 17](#_Toc10472)

[4.2.1 初始化无输入数据 17](#_Toc21608)

[4.2.2 删除头结点 17](#_Toc32705)

[4.2.3 删除后链表为空 17](#_Toc28950)

[4.3 出错测试 18](#_Toc3111)

[4.3.1 考生人数错误 18](#_Toc21048)

[4.3.2 操作码错误 18](#_Toc19687)

[4.3.3 插入位置不存在 19](#_Toc32296)

[4.3.4 删除考号不存在 19](#_Toc11952)

[4.3.5 查找考号不存在 20](#_Toc6054)

[4.3.6 修改考号不存在 20](#_Toc22215)

# 1 分析

## 1.1 背景分析

排课系统是一个学校不可缺少的部分，对于学校开展教学任务有极其重要的作用。由于一所高校所开设的课程规模的庞大，课程安排显得极为复杂，传统的手工排课工作量极大且容易出错。

而使用计算机在排课方面，具有手工管理所无法比拟的优势。这些优点能够极大地提高学校的效率，也是学校走向信息化、科学化、国际化的重要条件之一。因此，开发一套排课系统具有十分重要的意义。

## 1.2 功能分析

作为一个最简易的排课系统，首先应该有的功能就是能够读取用户预设的课程信息。其次，还应该根据先修课程的要求安排每门课程的开设学期，根据课时数安排每周的上课次数。最后，排课系统软件还应该能够正确打印生成的各学期课表。

其中，读取数据功能中输入数据包括：各学期所开的课程数（必须使每学期所开的课程数之和与课程总数相等）、课程编号、课程名称、周学时数、指定开课学期和先决条件。（开课学期为0，表示有电脑自行指定开课学期）。如输入数据不合理，比如每学期所开的课程数值和与课程总数不相等，应显示适当的提示信息。

安排学期与周课表时，应满足如下要求：

1. 周一至周五上课，每天上10节课，第1大节为第1-2节课，第二大节为第3-5节课，第3大节为第6-7节课，第4大节为8-10节课
2. 在排课时，如一门课程有3节课，则优先安排3节课连续上；如3节课连续无法按排，再优先安排两节课连续上，最后再安排单节课上的情况；如果一门课程需要安排上两天，为教学效果较好，最好不安排在相邻的两天

综上所述，一个排课系统至少应该具有读取数据、安排课表、打印课表等功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的统计操作，因此使用三个层次（课程、周、总课表）的类，选用大量顺序表的数据结构进行存储。同时，为了实现简易，增加了一些辅助数组，使得程序更为简洁。

## 2.2 类结构设计

该类结构包括三个抽象数据类型——课程类（course）、周课表类（week）、课表类（schedule），而三个类之间的耦合关系采用嵌套关系。为方便处理，本系统采用class描述课程类（course）和周课表类（week），这样使得课表类（schedule）可以访问链表结点。

## 2.3 成员与操作设计

**课程类（course）**

**公有成员：**

string num, name;//序号、名称

char precourse[10];//先修课程

int period;//课时

int term;//开设时间

int weekday, time;//星期、时间

int auxpre[m];//先修课序号数组

**公有操作：**

course::course() ;//构造函数

bool course::have\_pre();//是否有先修课

void course::pre(string\* aux);//先修课处理

**周类（week）**

**公有成员：**

int day[6][11];//0！每天的课程序号数组

int day2 [6];//0！每天两节课连上的课

int day3 [6];//0！每天三节课连上的课

**公有操作：**

week::week() ;//构造函数

**课表类（schedule）**

**公有成员：**

int termNum[N];//每学期的课程数量

int termDegree[n];//每堂课的优先级

course classes[n];//所有课程按序号数组

Queue<int>termClass[N];//八个学期的课程队列

week termWeek[N];//八个学期的周课表

**公有操作：**

schedule::schedule() //构造函数

void schedule::putIn();//从文件中输入课程信息

void schedule::arrangeTerm();//安排课程所在学期

void schedule::tweekDegree(int t, int c, int sort);

//学期数/课程序号/优先方式选择安排方式

void schedule::tWeek(int t);//安排t学期的周课程表

void schedule::arrangeWeek();//安排八个学期的周课表

void schedule::print();//输出课表到文件中

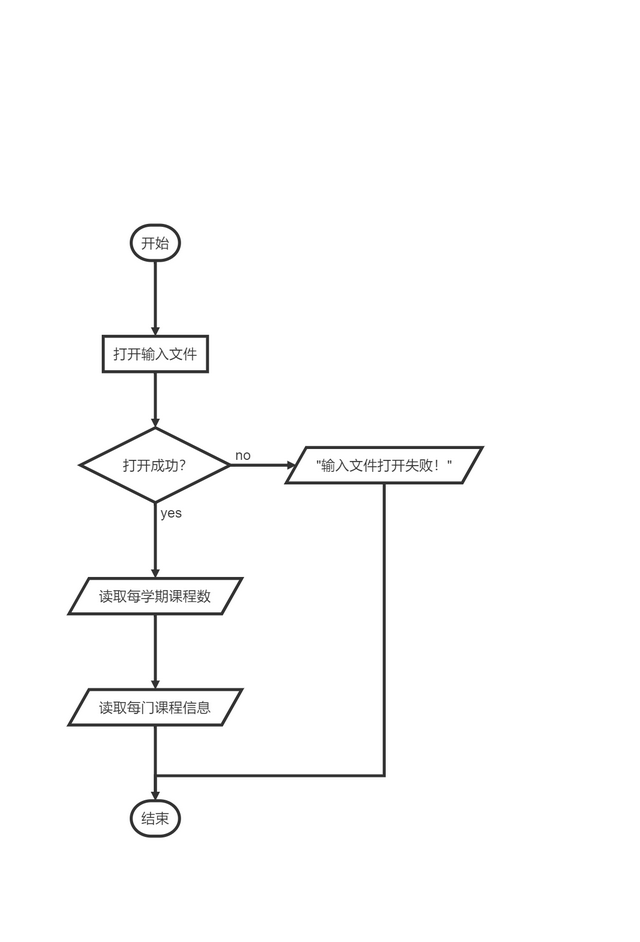
## 2.4 系统设计

系统首先从输入文件（IN.txt）中读取预设信息——每学期课程数量及每门课程的信息，完成对课程数组（classes）的创建和输入数据工作，然后根据课程安排要求安排课程开设学期与周课表，调用课表类（T）的成员函数完成课表构建，最后调用打印函数将课表输出到输出文件（OUT.txt）中。

# 3 实现

## 3.1 文件读取输入的实现

### 3.1.1 文件读取输入流程图



### 3.1.2 文件读取输入核心代码

void schedule::putIn() {//从文件中输入课程信息

fstream infile("D:\\vs\\TEST\_FIELD\\IN.txt", ios::in);

if (infile.is\_open() == 0) {

cerr << "输入文件打开失败！" << endl;

exit(1);

}

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

{

infile >> termNum[i];

}

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

{

infile >> classes[i].num >> classes[i].name >> classes[i].period >> classes[i].term >> classes[i].precourse;

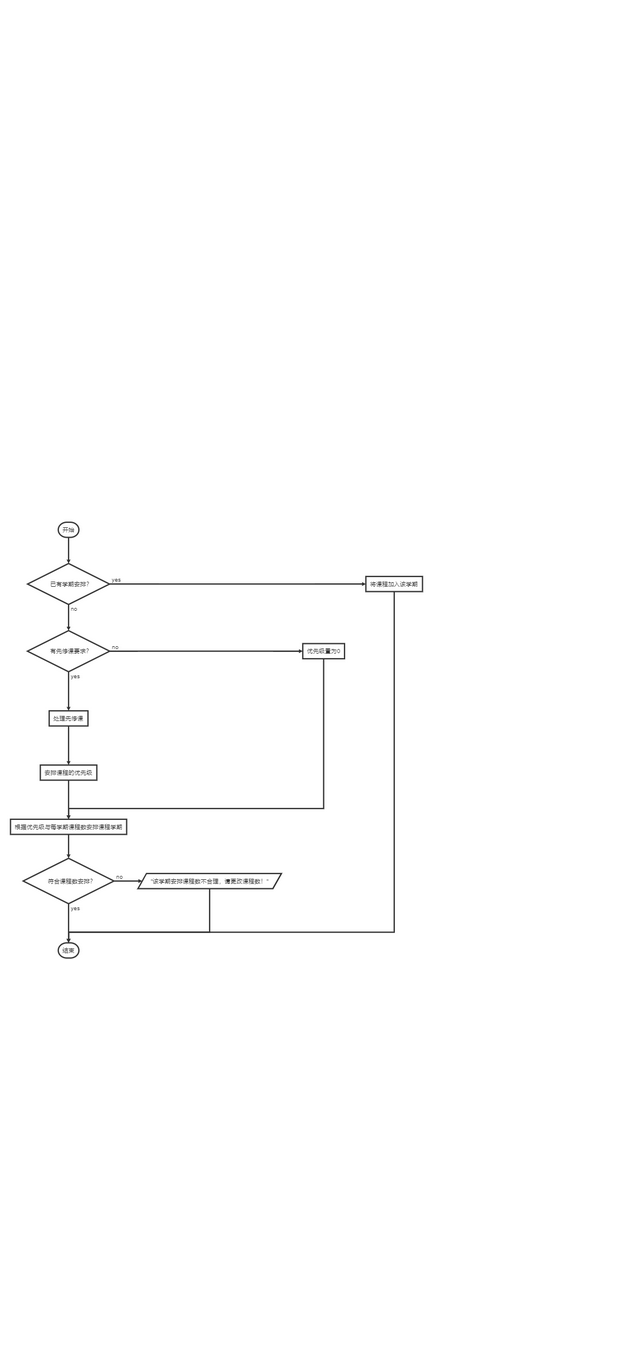
}

infile.close();

}

## 3.2 安排课程开设学期的实现

### 3.2.1 安排课程开设学期流程图



### 3.2.2 安排课程开设学期核心代码

//是否有先修课

bool course::have\_pre() {

string temp = "x";

if (precourse == temp)

return false;

return true;

}

//先修课处理

void course::pre(string\* aux) {

int i = 0;

for (int j = 0;precourse[j] != '\0';j++)

{

if (precourse[j] == ',')//将先修课断成多个int型

{

i++;

continue;

}

aux[i] += precourse[j];

}

}

//安排课程所在学期

void schedule::arrangeTerm() {

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

if (classes[i].term != 0)//若已有规定学期

{

termDegree[i] = -1;

termClass[classes[i].term - 1].push(i);//加入后优先级置-1

}

else if (classes[i].have\_pre() == false) {//若无先修课程要求

termDegree[i] = 0;

}

else if (classes[i].have\_pre() == true)

{

string\* aux = new string[m];

classes[i].pre(aux);

for (int j = 0;j < m;j++) //将辅助数组中记录的先修课的序号录入

for (int t = 0;t < n;t++)

if (aux[j] == classes[t].num)

{

classes[i].auxpre[j] = t;

break;

}

}

}

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

if (classes[i].auxpre[0] != -1)//处理有先修课的课程

{

int temp[m] = { 1 };//有先修课的优先级目前为1

for (int j = 0;j < m && classes[i].auxpre[j] != -1;j++)

temp[j] += termDegree[classes[i].auxpre[j]];//计算每条先修课路径上有几节课

int max = 0;

for (int j = 1;j < m && classes[i].auxpre[j] != -1;j++)//取出大的那条

if (temp[j] > temp[max])

max = j;

termDegree[i] = temp[max];

}

}

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

int temp = 0;

while (termClass[i].size() < termNum[i]) {//每个学期的课程数不超过输入要求

for (int j = 0;j < n && termClass[i].size() < termNum[i];j++)

{

if (termDegree[j] == temp)//当优先级条件满足时，将该课序号加入该学期队列

{

termClass[i].push(j);

classes[j].term = i;

termDegree[j] = -1;//加入后优先级置-1

}

}

if (termClass[i].size() == termNum[i])//加满则退出循环

break;

temp++;

if (temp > i)//若加入优先级已大于当前学期，则该学期无法加满，应报错！

{

cerr << "第" << i + 1 << "学期安排课程数不合理，请更改课程数！" << endl;

exit(1);

}

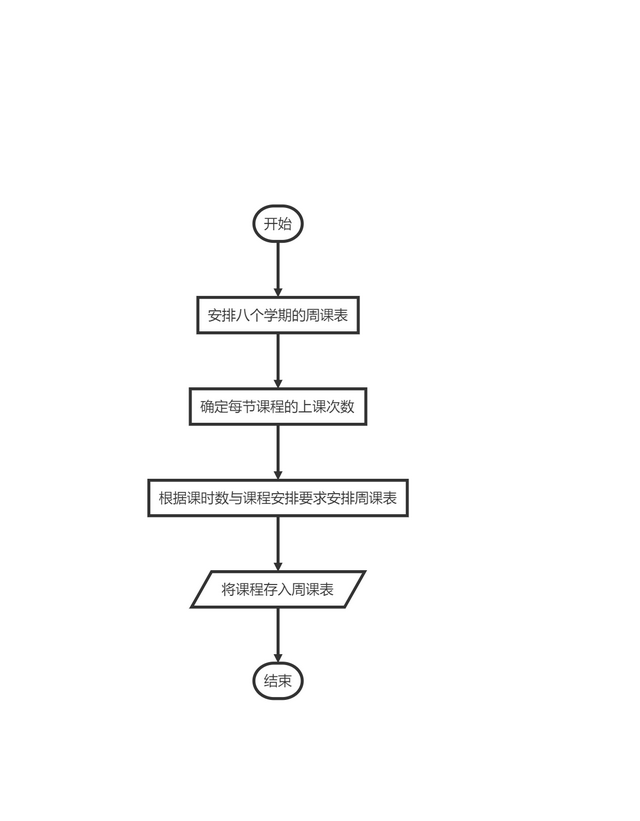
}

}

}

## 3.3 安排周课程表的实现

### 3.3.1 安排周课程表流程图



### 3.3.2 安排周课程表核心代码

//传入学期数，课程序号，根据课时与传入优先方式选择该课程的安排方式

void schedule::tweekDegree(int t, int c, int sort) {

if (sort == 3 && classes[c].period == 3) {//优先三课时安排

const int dayDegree[5] = { 3,2,4,5,1 };//每天安排该课的优先级序列

for (int i = 0; i < 5; i++) {

if (termWeek[t].day3[dayDegree[i]] < 2) {//将该课放入周[dayDegree[i]]课表，并将其上课时间填好

classes[c].weekday = dayDegree[i];

switch (termWeek[t].day3[dayDegree[i]]) {

case 0:classes[c].time = 2;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][3] = c;

termWeek[t].day[dayDegree[i]][4] = c;

termWeek[t].day[dayDegree[i]][5] = c;break;

case 1:classes[c].time = 4;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][8] = c;

termWeek[t].day[dayDegree[i]][9] = c;

termWeek[t].day[dayDegree[i]][10] = c;break;

}

return;

}

}

}

else if (sort == 3) {//三节优先，分5/6课时两种情况

const int dayDegree[5] = { 1,5,4,2,3 };

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

if (termWeek[t].day3[dayDegree[i]] < 2) {//若隔两天允许，将该课放入周[dayDegree[i]]课表，并将其上课时间填好

switch (dayDegree[i]) {

case 1:classes[c].weekday = 4;break;

case 4:classes[c].weekday = 2;break;

case 2:classes[c].weekday = 5;break;

case 5:classes[c].weekday = 1;break;

}

if ((classes[c].period == 6 && termWeek[t].day3[classes[c].weekday] < 2) || (classes[c].period == 5 && termWeek[t].day2[classes[c].weekday] < 2)) {

switch (termWeek[t].day3[dayDegree[i]]) {

case 0:classes[c].time = 2;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][3] = c;

termWeek[t].day[dayDegree[i]][4] = c;

termWeek[t].day[dayDegree[i]][5] = c;break;

case 1:classes[c].time = 4;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][8] = c;

termWeek[t].day[dayDegree[i]][9] = c;

termWeek[t].day[dayDegree[i]][10] = c;break;

}

if (classes[c].period == 6)

switch (termWeek[t].day3[classes[c].weekday]) {

case 0:classes[c].time = 2;

termWeek[t].day3[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][3] = c;

termWeek[t].day[classes[c].weekday][4] = c;

termWeek[t].day[classes[c].weekday][5] = c;break;

case 1:classes[c].time = 4;

termWeek[t].day3[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][8] = c;

termWeek[t].day[classes[c].weekday][9] = c;

termWeek[t].day[classes[c].weekday][10] = c;break;

}

else

switch (termWeek[t].day2[classes[c].weekday]) {

case 0:classes[c].time = 1;

termWeek[t].day2[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][1] = c;

termWeek[t].day[classes[c].weekday][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][6] = c;

termWeek[t].day[classes[c].weekday][7] = c;break;

}

return;

}

}

}

for (int i = 0; i < 5; i++) {//若所有情况都无法满足间隔两天的条件

if (termWeek[t].day3[dayDegree[i]] < 2) {//若允许，将该课放入周[dayDegree[i]]课表，并将其上课时间填好

for (int j = 1;j <= 5;j++) {

if (j == dayDegree[i])

continue;

if ((classes[c].period == 6 && termWeek[t].day3[j] < 2) || (classes[c].period == 5 && termWeek[t].day2[j] < 2)) {

switch (termWeek[t].day3[dayDegree[i]]) {

case 0:classes[c].time = 2;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][3] = c;

termWeek[t].day[dayDegree[i]][4] = c;

termWeek[t].day[dayDegree[i]][5] = c;break;

case 1:classes[c].time = 4;

termWeek[t].day3[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][8] = c;

termWeek[t].day[dayDegree[i]][9] = c;

termWeek[t].day[dayDegree[i]][10] = c;break;

}

if (classes[c].period == 6)

switch (termWeek[t].day3[j]) {

case 0:classes[c].time = 2;

termWeek[t].day3[j]++;

termWeek[t].day[j][3] = c;

termWeek[t].day[j][4] = c;

termWeek[t].day[j][5] = c;break;

case 1:classes[c].time = 4;

termWeek[t].day3[j]++;

termWeek[t].day[j][8] = c;

termWeek[t].day[j][9] = c;

termWeek[t].day[j][10] = c;break;

}

else

switch (termWeek[t].day2[j]) {

case 0:classes[c].time = 1;

termWeek[t].day2[j]++;

termWeek[t].day[j][1] = c;

termWeek[t].day[j][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[j]++;

termWeek[t].day[j][6] = c;

termWeek[t].day[j][7] = c;break;

}

return;

}

}

}

}

}

else {//若课时为4

const int dayDegree[5] = { 5,1,2,4,3 };

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

if (termWeek[t].day2[dayDegree[i]] < 2) {

switch (dayDegree[i]) {

case 1:classes[c].weekday = 4;break;

case 4:classes[c].weekday = 2;break;

case 2:classes[c].weekday = 5;break;

case 5:classes[c].weekday = 1;break;

}

if (termWeek[t].day2[dayDegree[i]] < 2 && termWeek[t].day2[classes[c].weekday] < 2) {//若隔两天允许，将该课放入周[dayDegree[i]]课表，并将其上课时间填好

switch (termWeek[t].day2[dayDegree[i]]) {

case 0:classes[c].time = 1;

termWeek[t].day2[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][1] = c;

termWeek[t].day[dayDegree[i]][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][6] = c;

termWeek[t].day[dayDegree[i]][7] = c;break;

}

switch (termWeek[t].day2[classes[c].weekday]) {

case 0:classes[c].time = 1;

termWeek[t].day2[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][1] = c;

termWeek[t].day[classes[c].weekday][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[classes[c].weekday]++;

termWeek[t].day[classes[c].weekday][6] = c;

termWeek[t].day[classes[c].weekday][7] = c;break;

}

return;

}

}

}

for (int i = 0; i < 5; i++) {//若无法满足间隔两天的条件

if (termWeek[t].day2[dayDegree[i]] < 2) {

for (int j = 1;j <= 5;j++) {

if (j == dayDegree[i])

continue;

if (termWeek[t].day2[j] < 2) {//若允许，将该课放入周[dayDegree[i]]课表，并将其上课时间填好

switch (termWeek[t].day2[dayDegree[i]]) {

case 0:classes[c].time = 1;

termWeek[t].day2[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][1] = c;

termWeek[t].day[dayDegree[i]][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[dayDegree[i]]++;

termWeek[t].day[dayDegree[i]][6] = c;

termWeek[t].day[dayDegree[i]][7] = c;break;

}

switch (termWeek[t].day2[j]) {

case 0:classes[c].time = 1;

termWeek[t].day2[j]++;

termWeek[t].day[j][1] = c;

termWeek[t].day[j][2] = c;break;

case 1:classes[c].time = 3;

termWeek[t].day2[j]++;

termWeek[t].day[j][6] = c;

termWeek[t].day[j][7] = c;break;

}

return;

}

}

}

}

}

}

//安排t学期的周课程表

void schedule::tWeek(int t) { int size = termNum[t];

int\* temp = new int[size];

for (int i = 0;i < size;i++)//取出本学期课程，并确定每个课程所需上课天数

{

temp[i] = termClass[t].front();

termClass[t].pop();

}

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

if (classes[temp[i]].period == 3) //该课程三节课连上优先

tweekDegree(t, temp[i], 3);

}

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

if (classes[temp[i]].period == 5 || classes[temp[i]].period == 6)//该课程课时数为5或6，则优先3节

tweekDegree(t, temp[i], 3);

}

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

if (classes[temp[i]].period == 4)//该课程课时数为5或6，则优先2节

tweekDegree(t, temp[i], 2);

}

}

//安排八个学期的周课表

void schedule::arrangeWeek() {

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

tWeek(i);

}

## 3.4 打印课表的实现

### 3.4.1 打印功能核心代码

void schedule::print() {//输出课表到文件中

fstream outfile("D:\\vs\\TEST\_FIELD\\OUT.txt", ios::out);

if (outfile.is\_open() == 0) {

cerr << "输出文件打开失败！" << endl;

exit(1);

}

outfile << "打印课表如下：" << endl;

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

outfile << "第" << i + 1 << "学期课表如下：" << endl;

outfile << "\t ";

for (int j = 0; j < 5; j++)

{

outfile << setiosflags(ios::fixed) << setiosflags(ios::left);

outfile << "星期" << setw(12) << j + 1;

}

outfile << endl;

for (int j = 1;j < 11;j++) {

if (j == 10)

outfile << "第" << j << "节课：";

else

outfile << "第 " << j << "节课：";

for (int k = 1;k < 6;k++) {

if (termWeek[i].day[k][j] != -1)

{

outfile << setiosflags(ios::fixed) << setiosflags(ios::left);

outfile << setw(16) << classes[termWeek[i].day[k][j]].name;

}

else

outfile << setw(16) << " ";

}

outfile << endl;

}

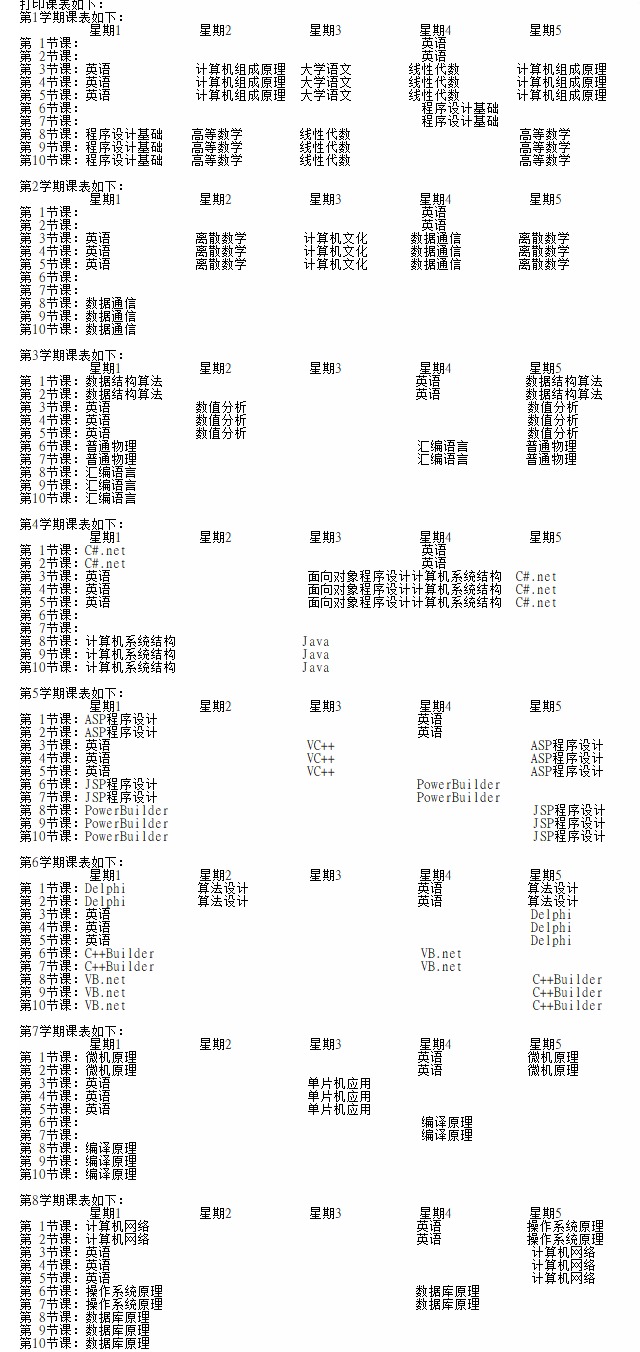
outfile << endl;

}

outfile.close();

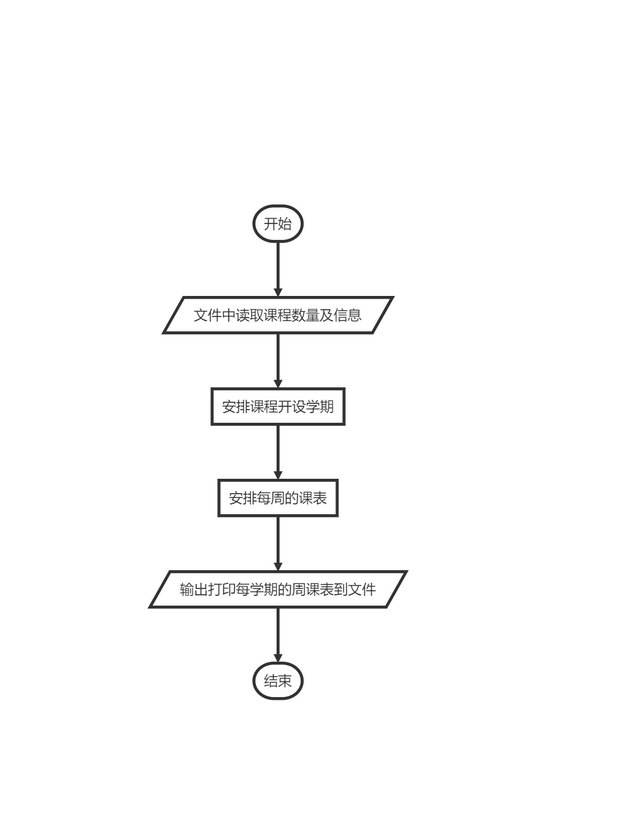
}

### 3.4.2 打印功能截屏示例



## 3.5 总体系统的实现

### 3.5.1 总体系统流程图



### 3.5.2 总体系统核心代码

int main() {

schedule T;

T.putIn();

T.arrangeTerm();

T.arrangeWeek();

T.print();

return 0;

}

# 4 测试

## 4.0 功能测试

**测试用例：**见IN.txt

**预期结果：**程序正常运行，输出所生成的

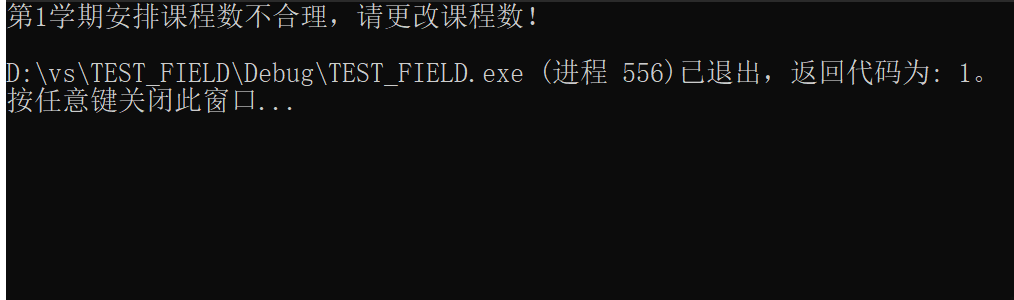
**实验结果：**见OUT.txt

## 4.1 出错测试

**测试用例：**输入每学期课程数不合适

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

****