项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名： 谢宇翔

学 号： 1951708

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](#_Toc9545)

[1.1 背景分析 1](#_Toc11525)

[1.2 功能分析 1](#_Toc27979)

[2 设计 1](#_Toc8309)

[2.1 数据结构设计 1](#_Toc31666)

[2.2 类结构设计 2](#_Toc22972)

[2.3 成员与操作设计 2](#_Toc18062)

[2.4 系统设计 3](#_Toc10424)

[3 实现 4](#_Toc4111)

[3.1将课程从文档中读入 4](#_Toc22905)

[3.1.1 功能流程图 4](#_Toc17436)

[3.1.2 读入课程核心代码 5](#_Toc8191)

[3.1.3程序截屏示例 6](#_Toc10145)

[3.2 取课程操作 6](#_Toc13000)

[3.2.1 取课程功能流程图 6](#_Toc3938)

[3.2.2 取课程核心代码 7](#_Toc4824)

[3.2.3辅助函数 9](#_Toc16980)

[3.3 分配课程 10](#_Toc13621)

[3.3.1 分配课程功能流程图 10](#_Toc3016)

[3.3.2 分配课程核心代码 10](#_Toc8410)

[3.3.3 分配课程辅助函数 12](#_Toc25585)

[3.3.3 分配课程截屏示例 20](#_Toc28084)

# 1 分析

## 1.1 背景分析

大学的每个专业都要进行排课。假设任何专业都有固定的学习年限，每学年含两学期，每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的。每门课恰好占一个学期，假定每天上午与下午各有5节课。是在这样的前提下设计一个教学计划编制程序。

## 1.2 功能分析

1. 输入数据包括：个学期所开的课程数（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，先决条件。如指定开课学期为0，表示有电脑自行指定开课学期。
2. 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。
3. 用文本文件存储输入数据，并且读入计算机。
4. 用文本文件存储产生的各学期的课表。

假设周一至周五上课，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，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，比如优先安排相隔2天上课，设weekday表示当前安排上课的工作日期，下一次排课的工作日是：weekday=（weekday+2-5）？（weekday+2-5）：（weekday+2）；

2 设计

## 2.1 数据结构设计

本程序要先读取所有课程表，然后按照课程表里的要求，有些课程会有先修课程，所以本程序适合用AOV活动网络来实现课程的编排。

## 2.2 类结构设计

本程序用Course类来存储课程信息，用System来存储整个系统，包含所有课程的数组以及建立的课程表。

## 2.3 成员与操作设计

**课程类（Course）**

**公有成员**

string ClassCode;//课程编号

string ClassName;//课程名字

int Times;//每星期课时

int terms;//开课学期

int pCourseNum ;//先修课程数目

string pCourses[40];//后继课程数组

**共有操作**

Course(string ClassCode="0", string ClassName="0",int Times=0,int terms=0)//构造函数

**系统类(System)**

**公有成员**

string CourseTable[8][5][10];//所有学期课程表

int TERMS = 0;//打印课程表用的当前学期

Course Courses[38];//存储着所有课程信息

Course \*thisTermCourses=NULL;//当前学习挑选出来的课程

string workdays[5][10];//单学期课程表

# 

**公有函数**

System()//构造函数，初始化课程表为“自习”

{

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

{

for (int j = 0; j < 10; j++)

{

workdays[i][j] = "自习";

}

}

}

void getEnglish(int thisTerm);//获取英语课

void getChinese();//获取语文课

void getCourses(int number,int thisTerm);//获取thisTerm课程

bool get1Course();//获取1个课程

void AdjustCourses(int i);//调整所有课程的先修课程的数量

int CodetoNo(string str);//将课程编码转化为数字

void ArrangeCourses(int number);//分配课程

int Arrange3(string name);//安排周学时为3的课

void Arrange4(Course item);//安排周学时为4的课

void Arrange5(Course item);//安排周学时为5的课

void Arrange6(Course item);//安排周学时为6的课

int FindFree2time(int i);//寻找空闲的2连，返回星期X

int FindFree3time(int i);//寻找空闲的3连，返回星期X

void saveToFile(ofstream& out);//输出到输出文档

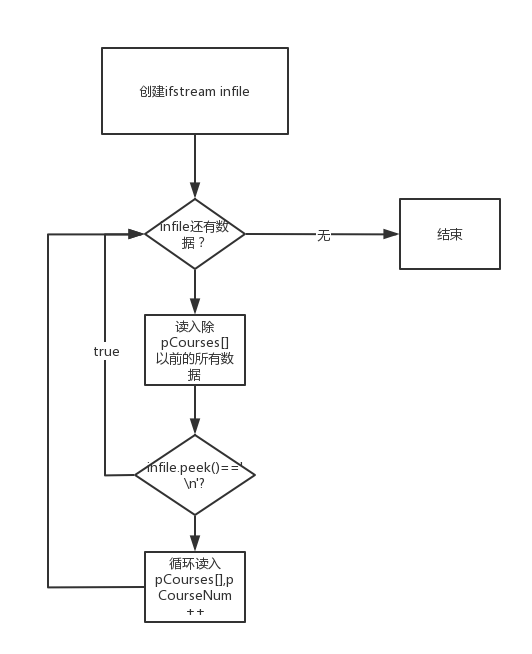
## 2.4 系统设计

首先从文件中读取所有课程信息进System课程数组里，然后按照每个学期开课数量，再根据AOV活动网络的判定条件，找到该学期所需要的课程的数量的课存储进thisTermCourses数组里，然后逐个取出分析放在课程表的哪个位置，最后将课程表存储进CourseTable里，将CourseTable输出到文档里即可。

# 3 实现

## 3.1将课程从文档中读入

### 3.1.1 功能流程图



### 3.1.2 读入课程核心代码

ifstream infile("courses.txt", ios::in);

if (!infile) {

cout << "No File" << endl;

}

for (int i = 0; i <8; i++)

infile >> ClassNum[i];//输入每学期开课数

if (!chargeCourseNum(ClassNum))//判断开课数是否正确

return -1;

System CourseSystem;

for (int i = 0; i < 38; i++)//写入数据

{

infile >> CourseSystem.Courses[i].ClassCode;

infile >> CourseSystem.Courses[i].ClassName;

infile >> CourseSystem.Courses[i].Times;

infile >> CourseSystem.Courses[i].terms;

int t = 0;

while(infile.peek() != '\n')

{

string pCourse;

infile >> pCourse;

CourseSystem.Courses[i].pCourses[t]=pCourse;

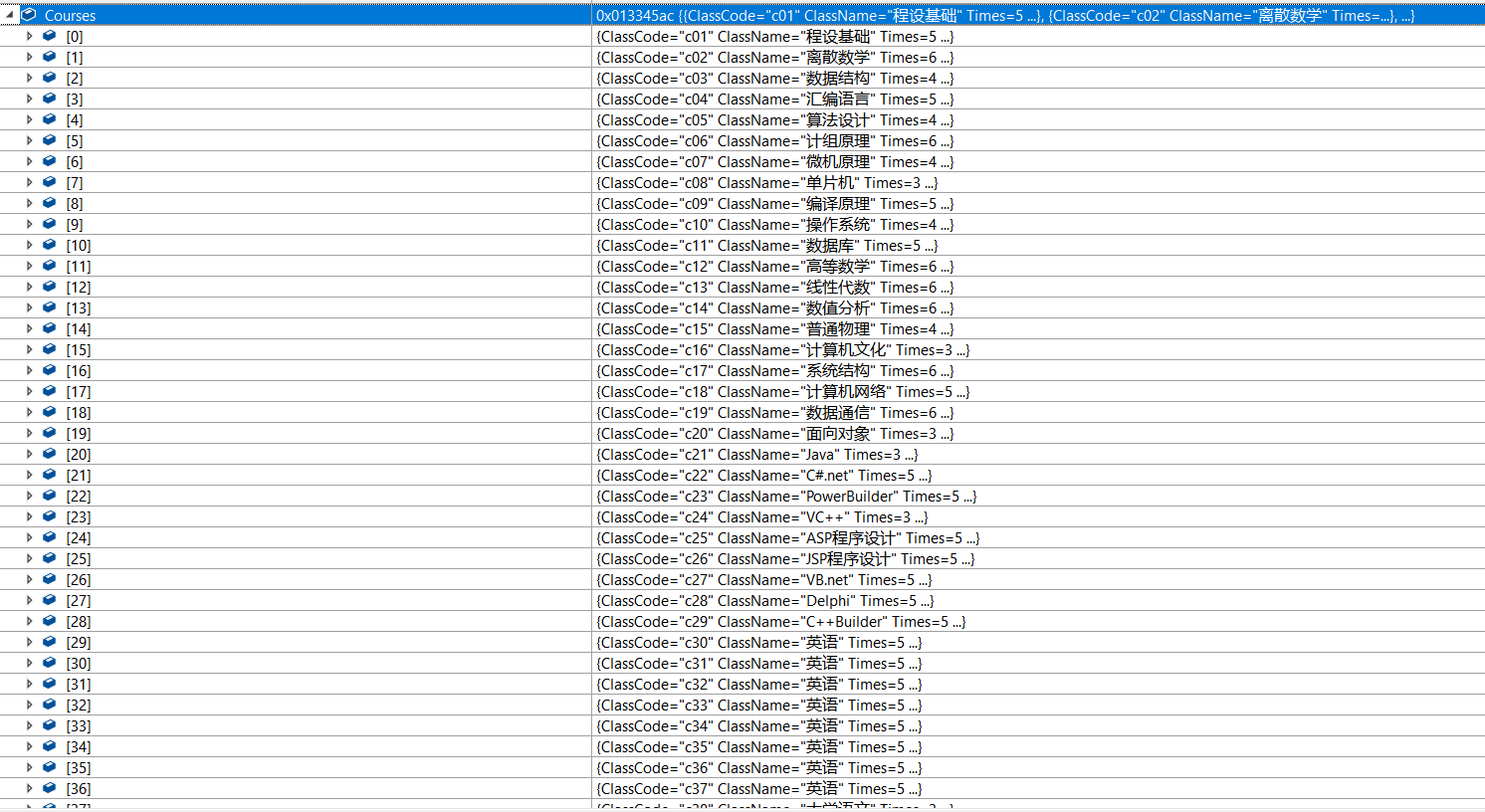
t++;

CourseSystem.Courses[i].pCourseNum++;

}

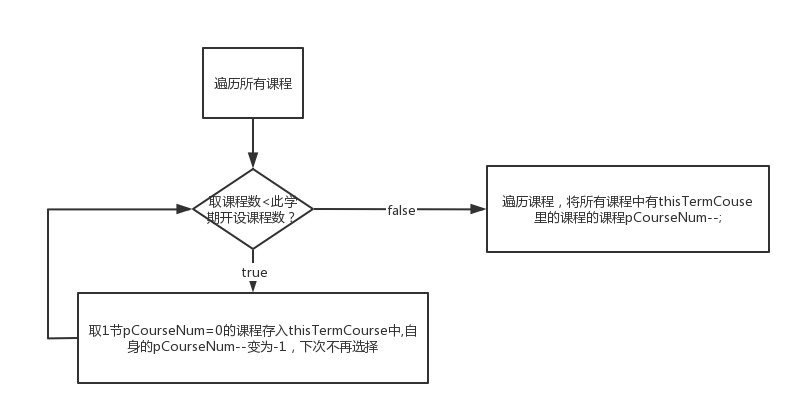
}

### 3.1.3程序截屏示例



## 3.2 取课程操作

### 3.2.1 取课程功能流程图



### 3.2.2 取课程核心代码

bool System::get1Course()

{

for (int i = 0; i < 29; i++)

{

if (Courses[i].pCourseNum >= 0)//先修课程数小于0说明已经选过，不再选

{

if (Courses[i].pCourseNum == 0)//先修课程数等于0，可以选择

{

int t = 0;

while (thisTermCourses[t].ClassCode != "0")

{

t++;

}//指向thisTermCourses中未分配的空间

thisTermCourses[t] = Courses[i];//存入，取1个课程操作结束

Courses[i].pCourseNum--;

return 1;

}

}

}

return 0;

}

void System::getCourses(int number,int thisTerm)

{

thisTermCourses = new Course[number];//记录当前学期的课程数

getEnglish(thisTerm);//先取英语

if (thisTerm == 0)//第一学期的话学语文

getChinese();

for (int t = (thisTerm == 0 ? (number - 2) : (number - 1)); t > 0; t--)

{

get1Course();

}

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

{

AdjustCourses( CodetoNo(thisTermCourses[t].ClassCode));

}

}

void System::AdjustCourses(int i)

{

string Code = Courses[i].ClassCode;

for (int t = 0; t < 29; t++)

{

if (Courses[t].pCourseNum <= 0)

continue;

else

{

for (int j = 0; Courses[t].pCourses[j]!="\0"; j++)//循环pCourses;

{

if (Courses[t].pCourses[j] == Code)//Courses[t]的先修课程里有Courses[i]

{

Courses[t].pCourseNum--;

break;

}

}

}

}

}

### 3.2.3辅助函数

int System::CodetoNo(string str)

{

for (int i = 0; i < 38; i++)

{

if (Courses[i].ClassCode == str)

return i;

}

return -1;

}

void System::getEnglish(int thisTerm)

{

thisTermCourses[0] = Courses[29 + thisTerm];

Courses[29 + thisTerm].pCourseNum--;

}

void System::getChinese()

{

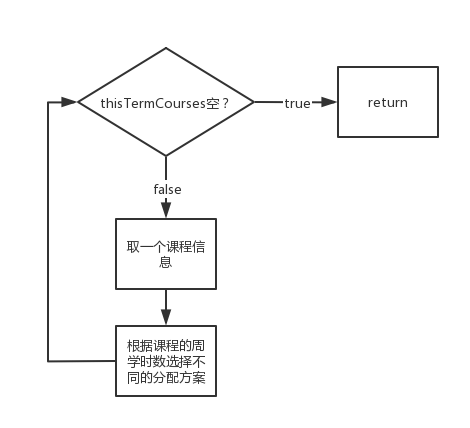
thisTermCourses[1] = Courses[37];

(Courses[37].pCourseNum)--;

}

## 3.3 分配课程

### 3.3.1 分配课程功能流程图



### 3.3.2 分配课程核心代码

void System::ArrangeCourses(int number)

{

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

{

int times = thisTermCourses[i].Times;//获取周学时

string name = thisTermCourses[i].ClassName;

switch (times)//根据周学时进行不同的分配模式

{

case 3:

Arrange3(name);

break;

case 4:

Arrange4(thisTermCourses[i]);

break;

case 5:

Arrange5(thisTermCourses[i]);

break;

case 6:

Arrange6(thisTermCourses[i]);

break;

}

}

for (int t = 0; t < 10; t++)//写入总课程表

{

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

{

CourseTable[TERMS][u][t] = workdays[u][t];

cout << workdays[u][t] << " ";

}

cout << endl;

}

TERMS++;//学期加1

for (int t = 0; t < 10; t++)//重置每学期的课表

{

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

{

workdays[u][t]="自习";

}

}

cout << endl;

}

### 3.3.3 分配课程辅助函数

int System::FindFree2time(int i)//从i日往后找2个空的时间

{

int j = i;

i = i + 1 < 5 ? i + 1 : i - 4;

while (j != i)

{

if (workdays[i][0] == "自习" || workdays[i][5] == "自习")//找到，退出

return i;

i = i + 1 < 5 ? i + 1 : i - 4;

}

i = i + 1 < 5 ? i + 1 : i - 4;

if (workdays[i][0] == "自习" || workdays[i][5] == "自习")

return i;

return i-1;

}

int System::FindFree3time(int i)//从i日往后找3个空闲的时间

{

int j = i;

i = i + 1 < 5 ? i + 1 : i - 4;

while (j != i)

{

if (workdays[i][2] == "自习" || workdays[i][7] == "自习")

return i;

i = i + 1 < 5 ? i + 1 : i - 4;

}

i = i + 1 < 5 ? i + 1 : i - 4;

if (workdays[i][2] == "自习" || workdays[i][7] == "自习")

return i;

return i-1;

}

void System::Arrange6(Course item)//分配周学时为6的课程

{

bool isdone = 0;

int i = Arrange3(item.ClassName);//先分匹配3个

i = (i + 2 - 5)>0 ? (i + 2 - 5) : i + 2;

if (workdays[i][2] == "自习")

{

workdays[i][2] = item.ClassName;

workdays[i][3] = item.ClassName;

workdays[i][4] = item.ClassName;

}

else if (workdays[i][7] == "自习")

{

workdays[i][7] = item.ClassName;

workdays[i][8] = item.ClassName;

workdays[i][9] = item.ClassName;

}

else

{

int date3 = FindFree3time(i);

if (date3 != i)//找到了

{

if (workdays[i][2] == "自习")

{

workdays[i][2] = item.ClassName;

workdays[i][3] = item.ClassName;

workdays[i][4] = item.ClassName;

isdone = 1;

}

else

{

workdays[i][7] = item.ClassName;

workdays[i][8] = item.ClassName;

workdays[i][9] = item.ClassName;

isdone = 1;

}

}

else

{

cout << "排课失败！" << endl;

return;

}

}

}

void System::Arrange5(Course item)//分配周学时为5的课程

{

bool isdone = 0;

int i = Arrange3(item.ClassName);//先分配3个

i = (i + 2 - 5)>0 ? (i + 2 - 5) : i + 2;

if (workdays[i][0] == "自习")

{

workdays[i][0] = item.ClassName;

workdays[i][1] = item.ClassName;

isdone = 1;

}

else if (workdays[i][5] == "自习")

{

workdays[i][5] = item.ClassName;

workdays[i][6] = item.ClassName;

isdone = 1;

}

else

{

int date2 = FindFree2time(i);

if (date2 == i)//没有空闲的2个空间，那就找空闲的3个空间但只分配2节课

{

int date3 = FindFree3time(i);

if (date3 != i)//找到了

{

if (workdays[i][2] == "自习")

{

workdays[i][2] = item.ClassName;

workdays[i][3] = item.ClassName;

isdone = 1;

}

else

{

workdays[i][7] = item.ClassName;

workdays[i][8] = item.ClassName;

isdone = 1;

}

}

else

{

cout << "排课失败！" << endl;

return;

}

}

else

{

if (workdays[i][0] == "自习")

{

workdays[i][0] = item.ClassName;

workdays[i][1] = item.ClassName;

isdone = 1;

}

else

{

workdays[i][5] = item.ClassName;

workdays[i][6] = item.ClassName;

isdone = 1;

}

}

}

}

void System::Arrange4(Course item)//安排2次2节课

{

bool isdone = 0;

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

{

if (isdone == 1)

return;

if (workdays[i][0] == "自习")

{

workdays[i][0] = item.ClassName;

workdays[i][1] = item.ClassName;

i = (i + 2 - 5)>0 ? (i + 2 - 5) : i + 2;

if (workdays[i][0] == "自习")

{

workdays[i][0] = item.ClassName;

workdays[i][1] = item.ClassName;

isdone = 1;

}

else if (workdays[i][5] == "自习")

{

workdays[i][5] = item.ClassName;

workdays[i][6] = item.ClassName;

isdone = 1;

}

else

{

int date2 = FindFree2time(i);

if (date2 == i)

{

int date3 = FindFree3time(i);

if (date3 != i)//找到了

{

if (workdays[i][2] == "自习")

{

workdays[i][2] = item.ClassName;

workdays[i][3] = item.ClassName;

isdone = 1;

}

else

{

workdays[i][7] = item.ClassName;

workdays[i][8] = item.ClassName;

isdone = 1;

}

}

else

{

cout << "排课失败！" << endl;

return;

}

}

else

{

if (workdays[i][0] == "自习")

{

workdays[i][0] = item.ClassName;

workdays[i][1] = item.ClassName;

isdone = 1;

}

else

{

workdays[i][5] = item.ClassName;

workdays[i][6] = item.ClassName;

isdone = 1;

}

}

}

}

}

}

int System::Arrange3(string name)

{

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

{

if (workdays[i][2] == "自习")

{

workdays[i][2] = name;

workdays[i][3] = name;

workdays[i][4] = name;

return i;

}

else if (workdays[i][7] == "自习")

{

workdays[i][7] = name;

workdays[i][8] = name;

workdays[i][9] = name;

return i;

}

}

return -1;

}

### 3.3.3 分配课程截屏示例

