项目说明文档

**数据结构课程设计**

**——排课软件**

作 者 姓 名： 吴英豪

学 号： 1953608

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

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。

1.3 其他要求及信息

1.3.1 排课要求

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）；

1.3.2 课程信息

课程编号 课程名称 学时数 指定开课学期 先修课程

c01 程序设计基础 5 0

c02 离散数学 6 0 c01

c03 数据结构算法 4 0 c01 c02

c04 汇编语言 5 0 c01

c05 算法设计 4 0 c03 c04

c06 计算机组成原理 6 0

c07 微机原理 4 0 c03

c08 单片机应用 3 0 c03

c09 编译原理 5 0 c03

c10 操作系统原理 4 0 c03

c11 数据库原理 5 0 c03

c12 高等数学 6 0

c13 线性代数 6 0

c14 数值分析 6 0 c12

c15 普通物理 4 0 c12

c16 计算机文化 3 0

c17 计算机系统结构 6 0 c06

c18 计算机网络 5 0 c03

c19 数据通信 6 0

c20 面向对象程序设计3 0 c01 c03

c21 Java 3 0 c01 c03

c22 C# .net 5 0 c01 c03

c23 PowerBuilder 5 0 c01 c03

c24 VC++ 3 0 c01 c03

c25 ASP程序设计 5 0 c01 c03

c26 JSP程序设计 5 0 c01 c03

c27 VB.net 5 0 c01 c03

c28 Delphi 5 0 c01 c03

c29 C++ Builder 5 0 c01 c03

c30 英语 5 1

c31 英语 5 2

c32 英语 5 3

c33 英语 5 4

c34 英语 5 5

c35 英语 5 6

c36 英语 5 7

c37 英语 5 8

c38 大学语文 3 1

2.设计

2.1数据结构与算法设计

学生在整个大学中修的部分课程是有先后顺序的,例如想修B课程需要先修读完A课程。那么上A课程的学期一定在B课程之前(这两个课不能在同一个学期修读)。所以采用拓扑排序的方式对每门课进行排序，同时确保课程的先后修读情况。

拓扑排序需要的数据结构为图(graph)。同时图需要成员顶点(vertex)。为了实现输出课表,此系统采用了队列(Queue)的数据结构暂存课程信息。图(graph)中的成员顶点集合采用数据结构链表实现。

2.2类结构设计

2.2.1 图类(Graph)

图的**成员**有链表形式的顶点类,由于拓扑排序整体上属于AOV，所以可以使用结点间的关系来表示边。

**成员函数--公有接口**:

(1)构造函数与析构函数

(2)插入顶点insertVertex(),

(3)插入边insertEdge()

(4)拓扑排序topologicalSort()

**-保护接口:**

(1)删除顶点deleteNode()

(2)调整结点的入度adjustInDegree();

2.2.2 顶点类(Vertex)

顶点类(Vertex)是图、队列、课程表类的重要组成部分,这三个类均为定点类Vertex的友元。

**成员变量：**

(1)课程编号\_number

(2)课程名称\_name,

(3)课时数\_classHour,

(4)开课学期\_semester.

(5)指针\_vnext;不同的结点通过\_vnext相连可以构成图中的顶点集合。

(6)指针\_enext，每一个顶点的\_enext指针所构成的链表中的元素是这个顶点的邻接顶点。

考虑到拓扑排序更加关注结点以及结点入度的变化(AOV)，所以图(graph)不额外设置边集合

**成员函数：**构造函数、析构函数

2.2.3 队列类(Queue)

队列主要用于拓扑排序时暂存拓扑序列，并且把拓扑序列传递给课程表类进行排课操作。使用链式队列的数据结构，其结点的类型为顶点类(因为需要存储一系列的拓扑排序的结果)。

**成员变量:**队首指针\_first,队尾指针\_last。

**成员函数:**

(1)构造、析构函数，

(2)队列清空函数makeEmpty()

(3)结点入队push()和出队pop(),

(4)打印结点printNode()

2.2.4 课程表类(Timetable)

课程表类为自定义类，主要实现的功能为根据拓扑排序得到的拓扑序列进行排课，并输入排课结果到输出流和文件。

**成员变量:**

(1)顶点数组\_semesters,此数组用于存储每一个工作日的课程；

(2)二维数组\_schedule，用于排课和课程表的打印;

(3)二维顶点数组\_thereClass，用于存储每一个工作日的连续三节课的大课;

(4)二维顶点数组\_twoClasses，用于存储每一个工作日的连续两节课的小课。

成员函数:

(1)构造与析构函数

(2)插入课程insertClasses

(3)调整课程arrangeCourses()

(4)输出outputToWindow()、outputToFile()

(5)清空课程 cleanClasses()

2.3成员与操作设计

2.2.1 图类(Graph)

//图

template<class T,class E>

class Graph

{

private:

Vertex<T, E>\* \_first;

protected:

T deleteNode(Vertex<T, E>\*& pre); //删除pre的下一个\_enext结点

void adjustInDegree(T number); //调整number的入度(--)

public:

Graph();

~Graph();

void topologicalSort(Queue<T,E>& queue); //拓扑排序,将最终的结果存在queue中

bool insertVertex(T number,T name,E classHour,E semester); //插入顶点

bool insertEdge(T number1, T number2); //通过课号来插入边

};

2.2.2 顶点类(Vertex)

//顶点类

template<class T,class E>

class Vertex

{

private:

T \_number; //课程编号

T \_name; //课程名称

E \_classHour; //课时数

E \_semester; //开课学期

Vertex<T, E>\* \_vnext; //指向下一个顶点

Vertex<T, E>\* \_enext; //指向下一个临接顶点

int \_inDegree; //入度数

public:

Vertex(T number = "", T name = "", E classHour = 0, E semester = 0);

Vertex(Vertex<T, E>\* vertex); //复制构造函数

~Vertex();

friend class Graph<T, E>;

friend class Queue<T, E>;

friend class Timetable<T, E>;

};

2.2.3 队列类(Queue)

//队列

template<class T,class E>

class Queue

{

private:

Vertex<T, E>\* \_first;

Vertex<T, E>\* \_last;

//队列使用顶点Vertex<T,E>的\_vnext指向下一个队列结点

public:

Queue();

~Queue();

void makeEmpty();

void push(Vertex<T,E>\* vertex);

bool pop(Vertex<T, E>\*& vertex);

void printNode();

};

2.2.4 课程表类(Timetable)

//课程表类

template<class T,class E>

class Timetable

{

private:

static const int \_halfDay = 2;

static const int \_semesterNumber= 8; //学期个数

static const int \_workdays = 5; //工作日个数

static const int \_x = 11;

static const int \_y = 6; //画一个11\*6的数组

Vertex<T, E>\*\* \_semesters; //工作日链表数组

T\*\* \_schedule; //每一天共用一个课程表

Vertex<T, E>\*\* \_threeClasses; //三节的大课

Vertex<T, E>\*\* \_twoClasses; //两节的小课

protected:

void initSchedule();

//安排并输出课程到文件

void outputToWindow(E semester);

void outputToFile(E semester);

void cleanClasses();

void adjustClasses();

bool arrangeClass(Vertex<T,E>\*& vertex,E num); //插入两节课

bool isFull()const; //判断某一天的课表是否被排满

public:

Timetable();

~Timetable();

void insertClasses(Queue<T, E>& queue); //插入拓扑序列对应的课表

void insertClass(T number, T name, E classHour, E semester); //插入已经安排好的课表

void arrangeCourses();

};

2.4文件设计

本系统支持windows系统和Linux系统。

(1)Windows系统

windows系统的相关文件在文件夹04\_1953608\_WuYinghao\_Windows中，

其中类的定义以及其成员函数的定义在04\_1953608\_WuYinghao.h中;main函数以及相关的函数定义在04\_1953608\_WuYinghao.cpp中.04\_1953608\_WuYinghao.exe为Windows平台下的可执行文件。

因为此系统涉及文件的读写，所以文件夹09\_1953608\_WuYinghao\_Windows中含有输入文本文件(inputFile.txt)和输出文本文件(outputFile.txt)

**04\_1953608\_WuYinghao.cpp和04\_1953608\_WuYinghao.h的编码格式均为**

**简体中文(GB2312)-代码页 936**

(2)Linux 系统

Linux系统的相关文件在文件夹04\_1953608\_WuYinghao\_Linux中,

其中类的定义以及其成员函数的定义在04\_1953608\_WuYinghao\_Linux.h中;main函数以及相关的函数定义在04\_1953608\_WuYinghao\_Linux.cpp中.04\_1953608\_WuYinghao.out为Linux平台下的可执行文件。

因为此系统涉及文件的读写，所以文件夹09\_1953608\_WuYinghao\_Linux中含有输入文本文件(inputFile.txt)和输出文本文件(outputFile.txt)

**04\_1953608\_WuYinghao\_Linux.cpp和04\_1953608\_WuYinghao\_Linux.h的编码格式均为Unicode(UTF-8 无签名)-代码页65001**

3.实现

3.1 整体功能

3.1.1 功能说明

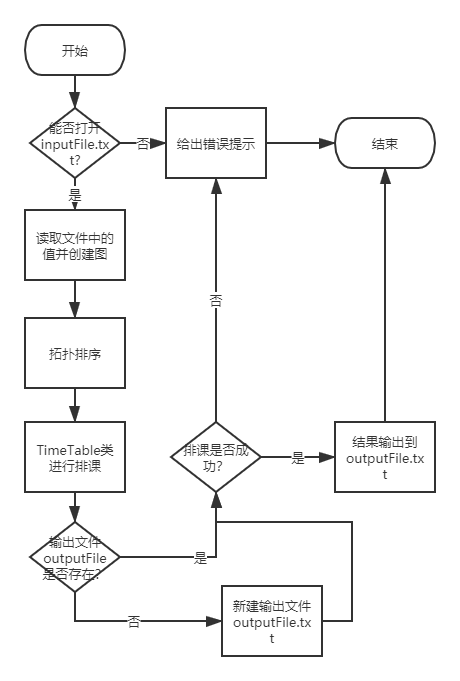
首先按行读取文件，将读到的信息进行整理之后通过插入顶点和插入边的形式构造图。然后对图中的各个结点进行拓扑排序，拓扑排序的同时按照上课的先后顺序给课程分配学期，将得到的拓扑序列暂存在队列当中。其次利用课程表类对队列中的拓扑序列按照题目要求的顺序进行排课，如果一个学期的课排满了，则将多余出来的课安排到之后的学期。

3.1.2 核心代码展示





3.1.3 流程图展示



3.2 拓扑排序实现

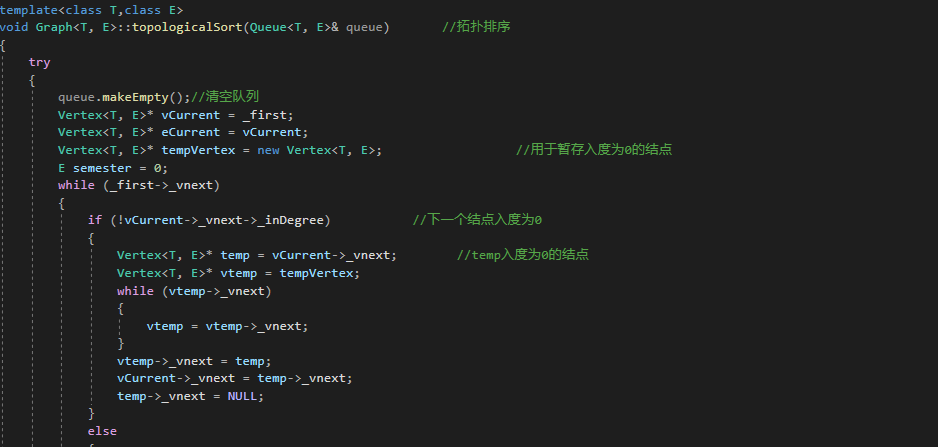
3.2.1 功能说明

对于图来说，其中的顶点集合通过每一个顶点的\_vnext指针相连，每一个顶点的邻接顶点通过指针\_enext来表示，首先在图中寻找所有入度为0的顶点，将它们的排课学期设置为1，并删除这些结点并把它们保存在队列当中;

删除结点之后调整这个结点邻接顶点的入度(入度减1)。

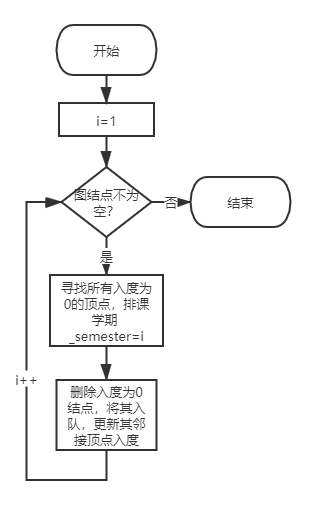
重复以上的操作，没进行一次循环就将这个课的上课学期加一，直到图中结点数为0.最终得到的拓扑序列存在队列中，且每一个队列结点的成员变量E \_semester表示着其初步的上课学期。

3.2.2 核心代码展示





3.2.3 流程图展示



3.3 排课功能的实现

3.3.1 功能说明

TimeTable类借助队列中的拓扑序列对课表进行排序。由于二维数组\_schedule一次只能保存一个学期的课表，所以在排课的时候（arrangeClass()函数）需要进行8次排课。

由于在进行拓扑排序的时候已经基本上确定了每一门课的排课学期数，所以利用拓扑序列，先将第一学期的课按照课时数分配到两个数组\_threeClasses(三节课)和\_twoClasses(两节课)中。按照题目中优先排三节课的顺序从周一到周五的顺序进行排课。每当插入一门课程但一天安排不下这门课时，按照同一门课不能连续两天上的顺序考虑其他的工作日，进行排课。

每当一个学期安排好课程之后，将课表输出到文件，并采用cleanClasses()清空辅助的数组，为下一次排课作准备

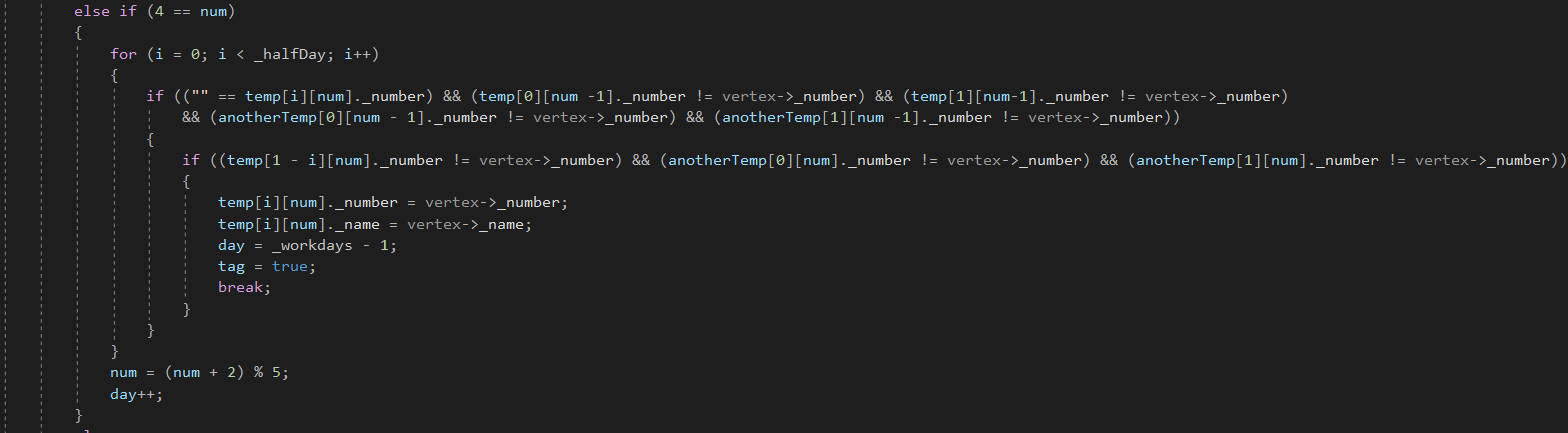
若某一个学期的已经安排满但本应排在这个学期的课还有很多，则将这些课安排在下一个学期。重复以上操作指导遍历了队列中的每一个结点。

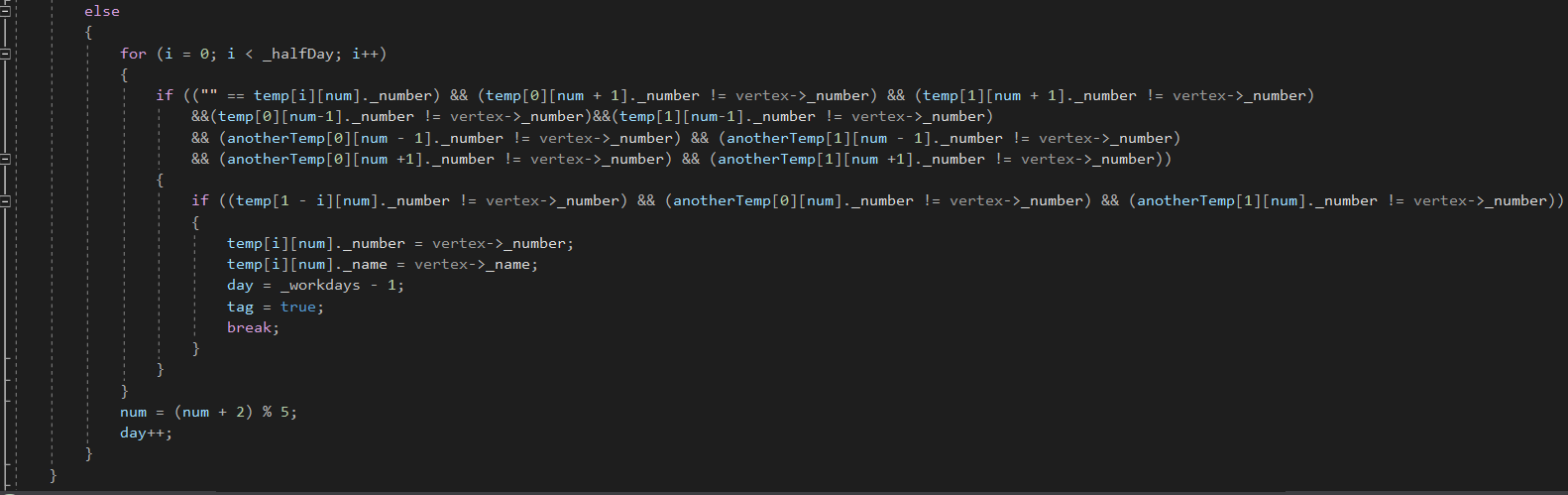
在进行一次遍历之后，若队列中还有剩余元素(在保证拓扑序列的前提下无法进行排课)，或一个学期安排结束之后还有课不能排到下一个学期，或者这些课被安排到了周六或是第九个学期，则说明inputFile中给出的课程信息有误，这时需要显示输入错误。

3.3.2 核心代码展示

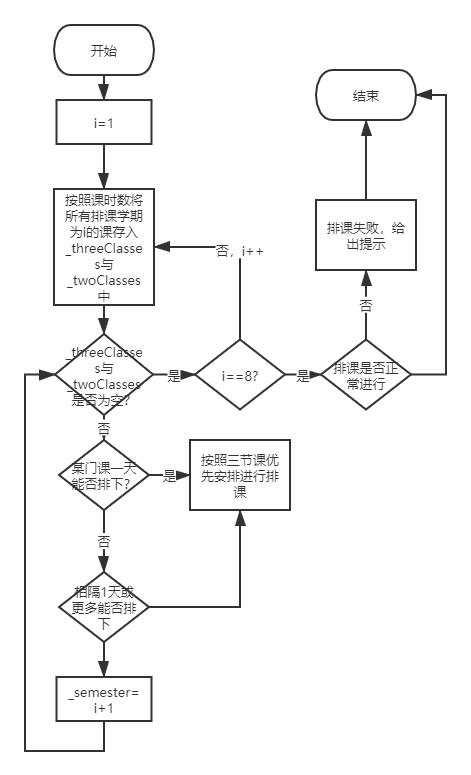








3.3.3 流程图展示



3.4 文件读写功能实现

3.4.1 功能说明

首先需要引入头文件<fstream>和<instream>并定义读取对象fin和写入对象fout。利用fin按行读取文件inputFile，将读取到的结果输出到屏幕中。并且按照相应的规则构建顶点与边，实现图的构造。

在进行拓扑排序之后，利用课程表类TimeTable可以进行排课，将排好的课存在二维数组\_schedule，随后遍历二维数组将\_schedule输出在屏幕上(cout)和文件中(fout).

**特别注意:由于此系统需要从文件中读取内容，所以系统对文件内容的依赖性非常的高，所以在输入inputFile.txt时，请务必按照系统已经给出的inputFileTemplate.txt格式进行输入，不规范的输入将无法得到正确的排课结果。**

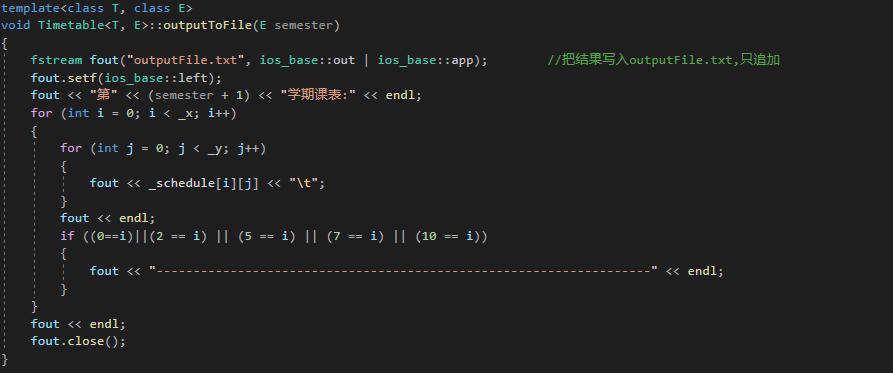
3.4.2 核心代码展示

（1）读取文件

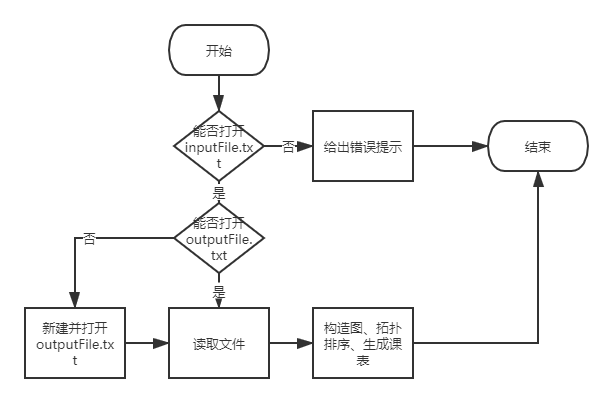




（2）写入文件



3.4.3 流程图展示

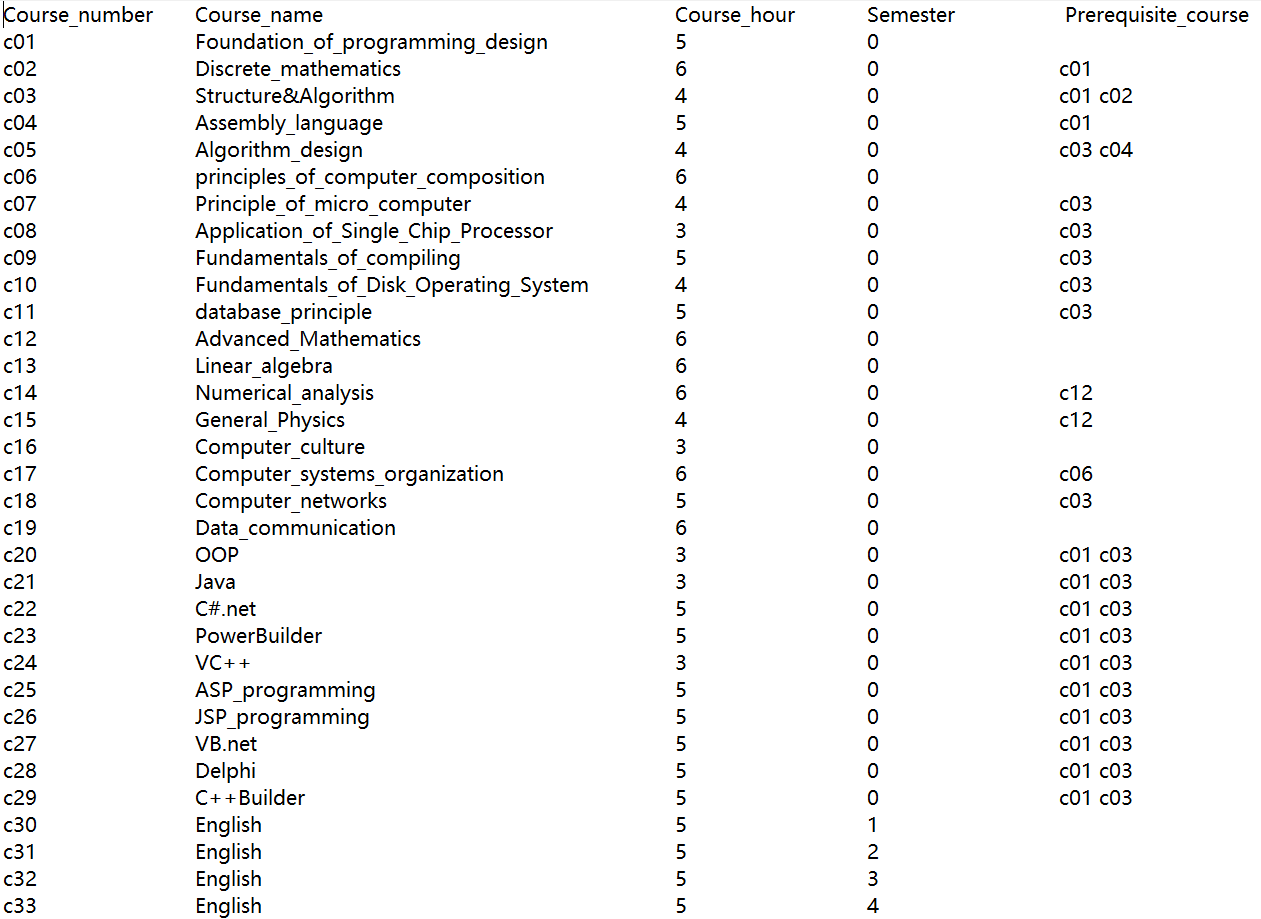


**4.测试**

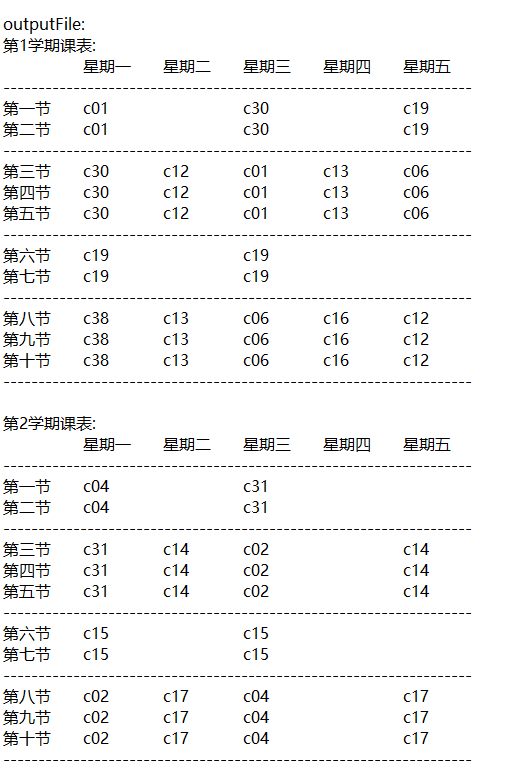
4.1 样例测试

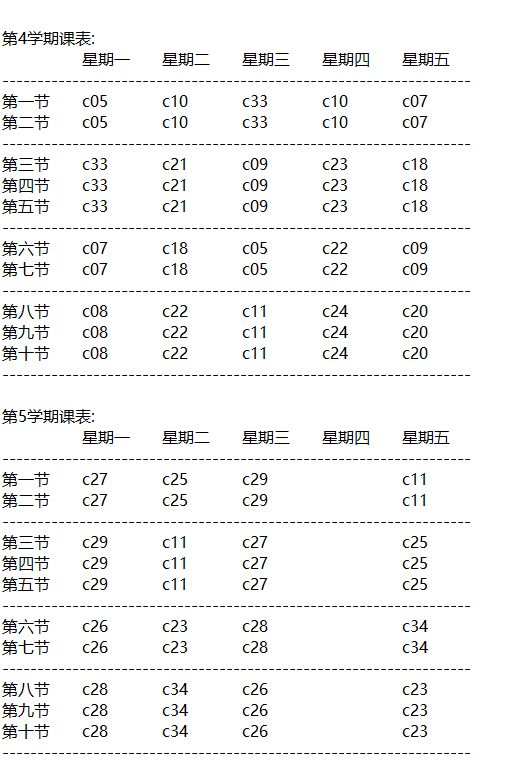
测试结果如下:

**（1）输入文件**



**(2)输出结果**

****

****

4.2 边界测试与错误测试

4.2.1 输入文件不存在

如果输入文件inputFile.txt不存在，则无法进行排课,给出相应的错误提示



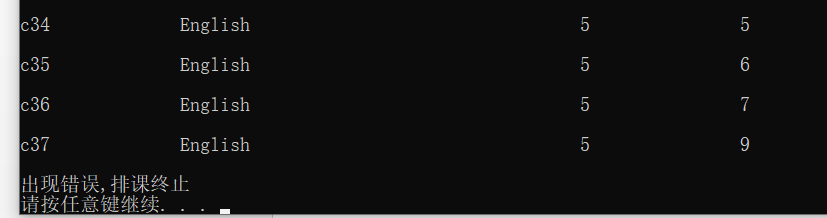
4.2.2 排课超出学期数或者不合法

在正常情况下，大学一共有四年八个学期，若输入的学期数大于8或小于0，则会给出错误提示，错误提示将同时输出到控制台和输出文件outputFile.txt中

(1)错误输入:



(2)错误提示





4.2.3 课时安排不合理或者输入非法

此题目有一定的实际背景，所以笔者在设置学时数时，将可选的学时规定为了大于1小于7的正整数(2、3、4、5、6),若输入其他的值，将会给出一定的错误提示，例如将某些课的学时数设置为负数，会给出一定的提示.

（1）错误输入



（2）出错提示

