项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名： 李圣

学 号： 1952723

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

目录

[1 分析 1](#_Toc8364)

[1.1 背景分析 1](#_Toc27666)

[1.2 功能分析 1](#_Toc4417)

[2 设计 1](#_Toc13052)

[2.1 文件结构设计 1](#_Toc25510)

[2.2 类结构设计 1](#_Toc29445)

[2.3 成员与函数设计 2](#_Toc30724)

[3 实现 11](#_Toc18980)

[3.1 读取文件功能的实现 11](#_Toc287)

[3.1.1 读取文件流程图 11](#_Toc8716)

[3.1.2 读取文件核心代码 12](#_Toc17365)

[3.1.3 读取文件截屏示例 13](#_Toc3641)

[3.2 读取课程数的实现 14](#_Toc10114)

[3.2.1 读取课程数流程图 14](#_Toc6968)

[3.2.2 读取课程数核心代码 15](#_Toc18072)

[3.2.3 读取课程数截屏示例 16](#_Toc8546)

[3.3 安排课程的实现 17](#_Toc26864)

[3.3.1 安排课程流程图 17](#_Toc32370)

[3.3.2 安排课程核心代码 18](#_Toc23961)

[3.4 安排课程表的实现 19](#_Toc29835)

[3.4.1 安排课程表流程图 19](#_Toc6197)

[3.4.2 安排课程表核心代码 20](#_Toc8874)

[3.4.3 安排课程表截屏示例 21](#_Toc22048)

[4 测试 21](#_Toc27050)

[4.1 鲁棒性测试 21](#_Toc29178)

[4.1.1 正常案例测试 21](#_Toc27186)

[4.1.2 课程输入测试 26](#_Toc70)

[4.1.3 课程输入测试 28](#_Toc27463)

# 分析

## 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。

## 功能分析

排课软件至少应该具有的基本功能：

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。

# 设计

## 文件结构设计

表达式求解的过程中需要使用Vector，因此该项目包含两个文件，如下：

main.cpp

MyVector.h

其中main.cpp是主程序，且main.cpp调用了MyVector.h

## 类结构设计

考虑到功能需要，除了Vector相关的类以外，拟定设计以下类：

class Exception;

class Course;

class Term;

class WorkData;

class System;

其中Exception是异常类，Course是课程类，Term是学期类，WorkData是工作信息暂存类，System是系统控制类

## 成员与函数设计

**MyVectorIterator类：**

template<typename val\_type>

using iterator = MyVectorIterator;

using pointer = val\_type\*;

using reference = val\_type&;

**私有成员：**

pointer \_ptr;

int \_pos;

**公有成员：**

MyVectorIterator(pointer ptr = nullptr, int pos = -1)；

~MyVectorIterator()；

int getPos()；

bool operator==(const iterator& it)const；

bool operator!=(const iterator& it)const；

iterator& operator++()；

iterator operator++(int)；

iterator& operator--()；

iterator operator--(int)；

reference operator\*()；

pointer operator->()；

**MyVector类：**

template<typename val\_type>

using vector = MyVector<val\_type>;

using reference = val\_type&;

using iterator = MyVectorIterator<val\_type>;

**私有成员：**

int \_size;

int \_capacity;

val\_type\* \_data;

void enlarge(int newcapacity);

**公有成员：**

MyVector();

MyVector(int size, val\_type val = val\_type());

MyVector(const vector& vec)；

~MyVector()；

bool empty()；

int size()；

int length()；

int capacity()；

void resize(int newsize, val\_type val = val\_type())；

void insert(iterator pos, const val\_type& val)；

val\_type erase(iterator pos)；

void push\_back(val\_type val)；

val\_type pop\_back()；

vector& operator=(vector& vec)；

vector& operator=(vector&& vec)；

iterator find(val\_type val)；

reference at(int pos)；

reference operator[](int pos)；

reference front()；

reference back()；

iterator begin()；

iterator end()；

void clear()；

**Exception类：**

enum EType

    {

        Fail,

        Bad

    };

**私有成员：**

EType \_type;

string \_data;

**公有成员：**

Exception(const string &str, EType type = Fail);

bool fail() const;

bool bad() const;

string data() const;

friend ostream &operator<<(ostream &os, const Exception exc);

**Course类：**

friend class System;

friend class WorkData;

friend class Term;

**私有成员：**

string \_id;               //课程编号

string \_name;             //课程名称

int \_hours;               //学时数

int \_term;                //开课学期

MyVector<string> \_preCrs; //先修课程

MyVector<int> \_indexOfPre;

**公有成员：**

Course(){} //默认构造

Course(const Course &crs) ;//复制构造

~Course(){} //折构

bool empty() const;

void clear();

bool operator==(string str) const;

bool havePre() const;

friend ifstream &operator>>(ifstream &ifs, Course &crs); //从文件读取输入

friend ostream &operator<<(ostream &os, Course &crs);

**Term类：**

friend class WorkData;

**私有成员：**

bool \_done = false;                   //安排妥当

MyVector<Course> \_courses;            //课程

MyVector<MyVector<Course>> \_schedule; //课程表,存的是序号

**公有成员：**

Term() {}

Term(MyVector<Course> &crss);

bool done();

void push\_back(Course &crs);

void clear();

void showCrs();

int available\_3(int day);

int attributeCourse\_3(Course crs, int &hours, int& prefer);

int available\_2(int day);

int attributeCourse\_2(Course crs, int &hours, int& prefer);

int available\_1(int day);

int attributeCourse\_1(Course crs, int &hours, int &prefer);

void attribute();

void showSchedule(ostream& os=cout);

**WorkData类：**

friend class System;

**私有成员：**

int \_cantPre = 0; //havePre中可安排的数量

bool \_allDone = false;

//定长状态量

MyVector<bool> \_isSet;     //已经安排

MyVector<int> \_needed;     //优先级

MyVector<bool> \_available; //先修课程已上完

int \_ncrs[8];              //已经安排的课程数

//变长状态量

MyVector<int> \_indexLast;   //不与先修相关的课程

MyVector<int> \_indexHIsPre; //与先修相关的课程的课的序号表

Term \_term[8];              //每学期的课程

**公有成员：**

WorkData(); //默认构造

WorkData(WorkData &wd) ;//复制构造

WorkData(WorkData &&wd); //复制构造

WorkData &operator=(WorkData &wd);

WorkData &operator=(WorkData &&wd);

void clearTerm();

void initialize(); //初始化

void updateNeeded(int index); //给index的所有先修课程++

bool isAvailable(int index);

void updateAvailable(); //更新available

void updateUnset(); //设置needed并给unset排序

void update();

bool allDone();

void attribute(int index, int term); //给学期安排课程

**System类：**

**私有成员：**

WorkData \_wd;

Term \_term[8];                  //结果

void readFile(string fileName); //从文件读取所有课程

void checkPre(int index); //确认先修课程是否存在并更新index

void initializeIndex(); //对序号表进行初始化

void resetNCrs();

bool findBefore(string id, int index); //id可以在index或其先修里找到

void simplifyPre(); //简化先修链条：我的先修课的先修课不必写在我的先修课中

void showCrs(); //输出到n的各学期课程数

**公有成员：**

static MyVector<Course> \_courses;      //所有课程

static MyVector<int> \_indexHavePre;    //有先修课程的课的序号表，方便处理

static MyVector<int> \_indexDueTerm[8]; //固定学期的课的序号表

static int \_ndue[8];                   //固定课程数

static int \_ncrs[8];                   //所需课程数

System();

void initialize(string fileName); //总体读文件初始化

void inputCrs(); //输入各学期的课程数

void clearTerm();

WorkData distributePre(WorkData wd, int term, int it) ;//递归寻找Pre解决方案

void distributeLst(WorkData &wd); //递归寻找Pre解决方案

void distribute() ;//总体寻找

bool showTermCrs();

bool allDone();

void setSchedule();

void showSchedule();

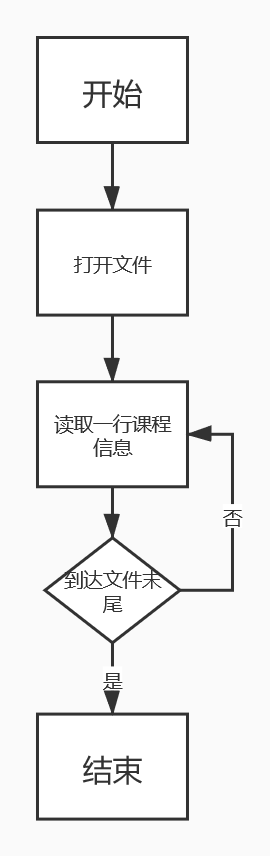
void writeSchedule(string fileName);

void showCourses(); //输出所有课程

# 实现

## 读取文件功能的实现

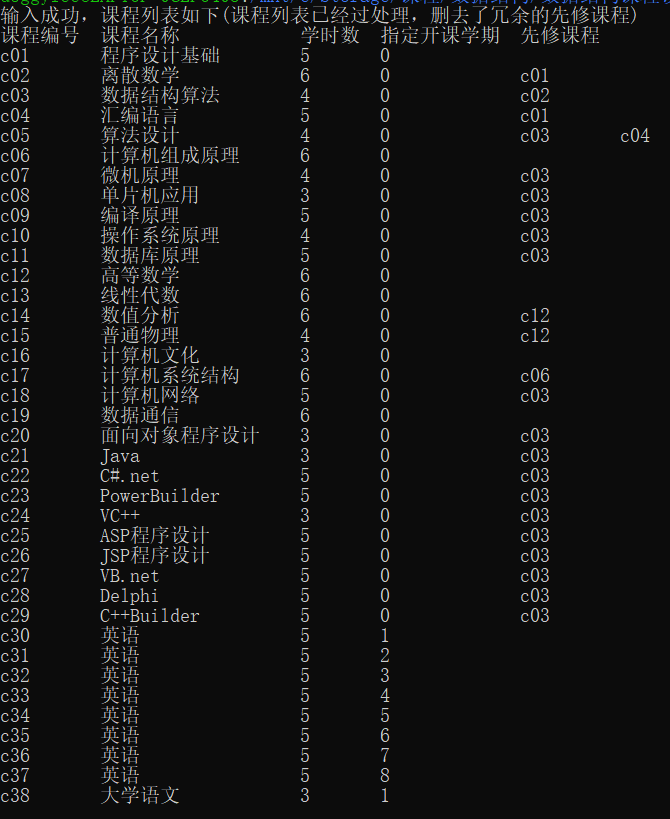
### 读取文件流程图



### 读取文件核心代码

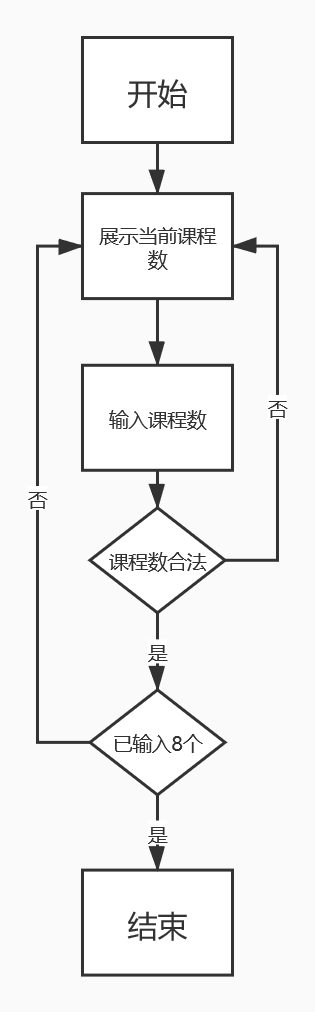


### 读取文件截屏示例

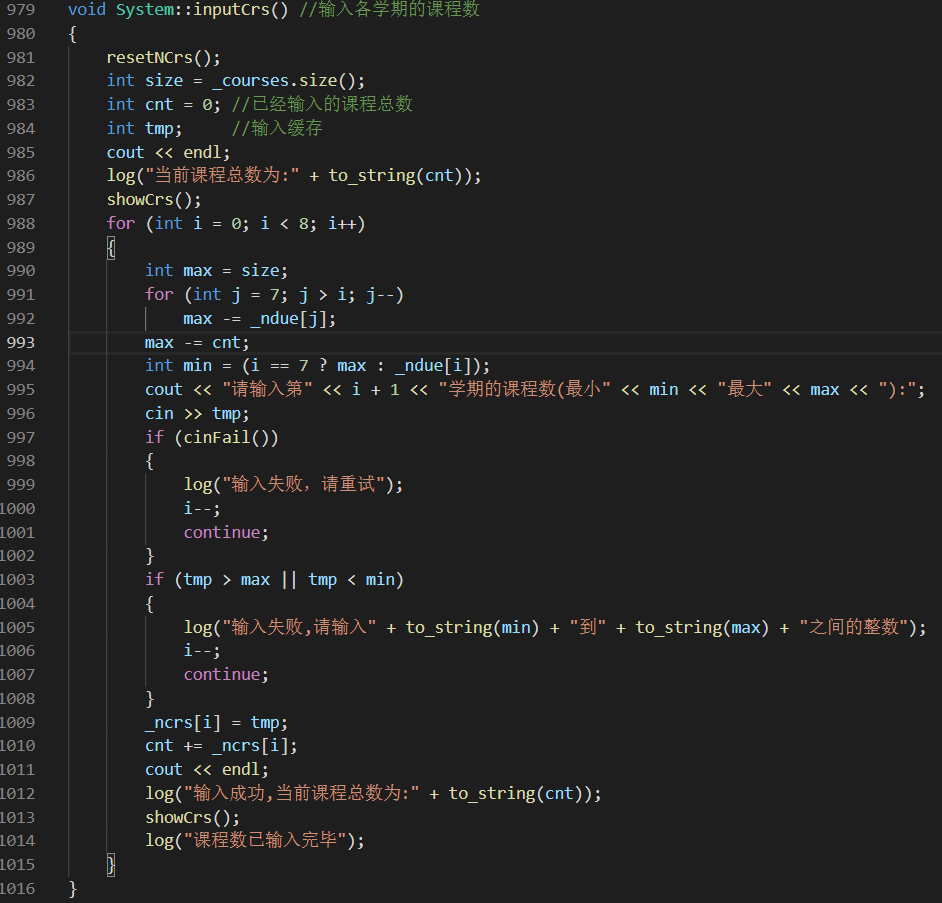


## 读取课程数的实现

### 读取课程数流程图



### 读取课程数核心代码

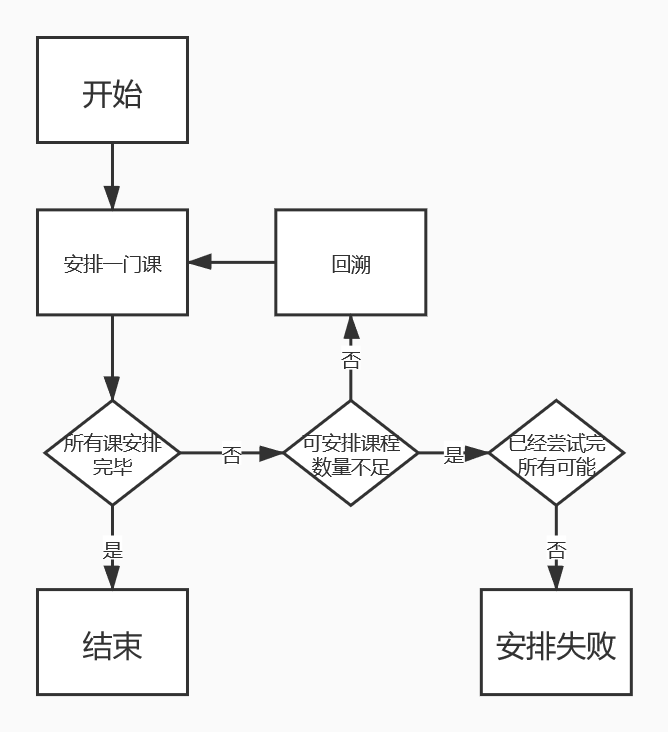


### 读取课程数截屏示例

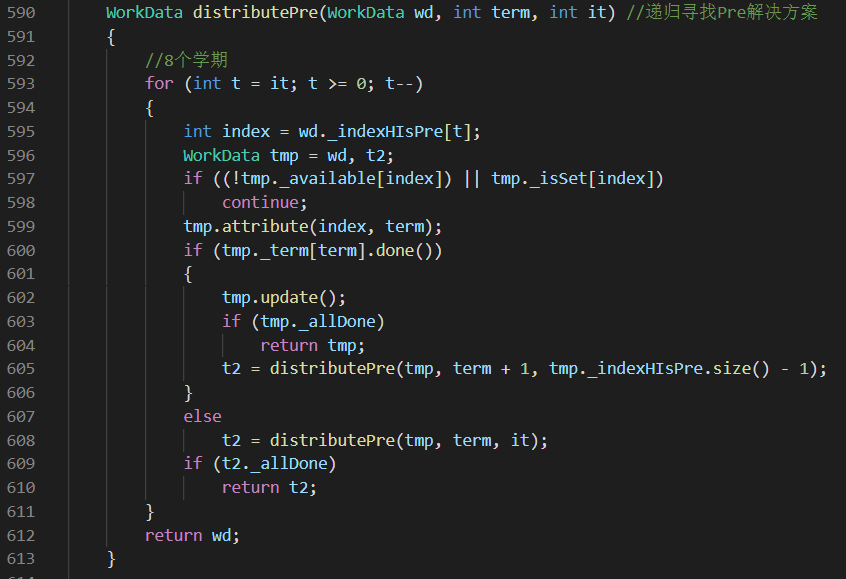


## 安排课程的实现

### 安排课程流程图

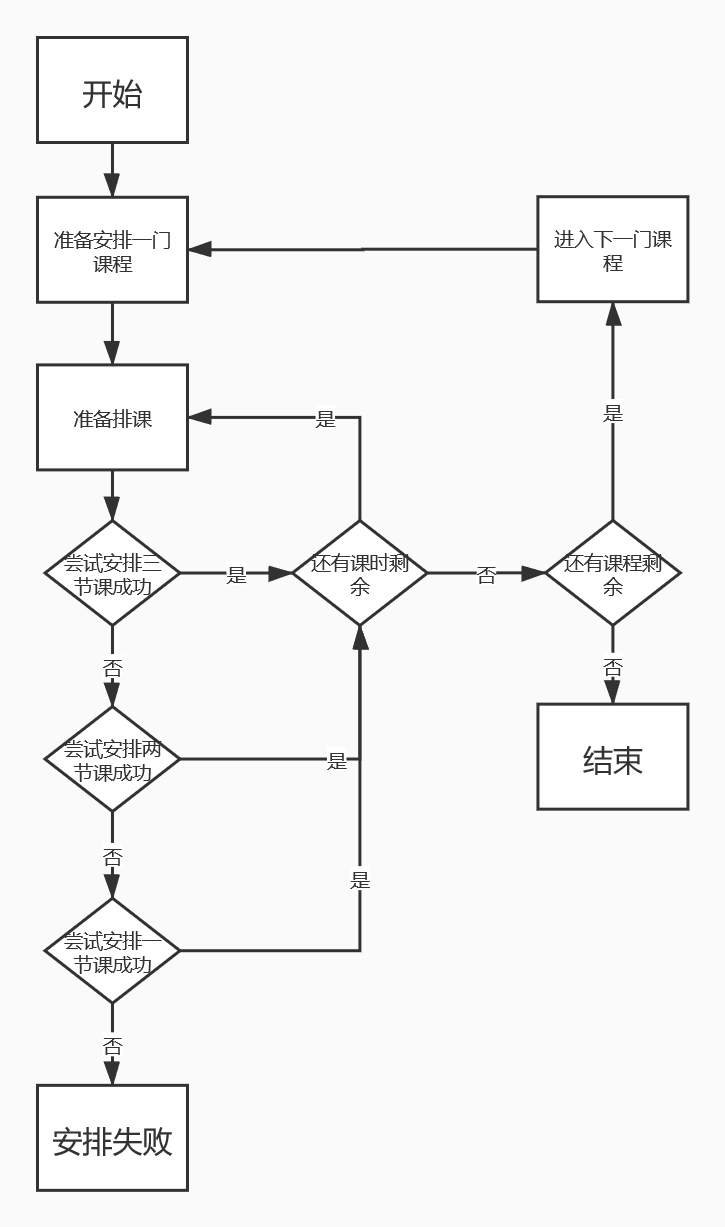


### 安排课程核心代码



## 安排课程表的实现

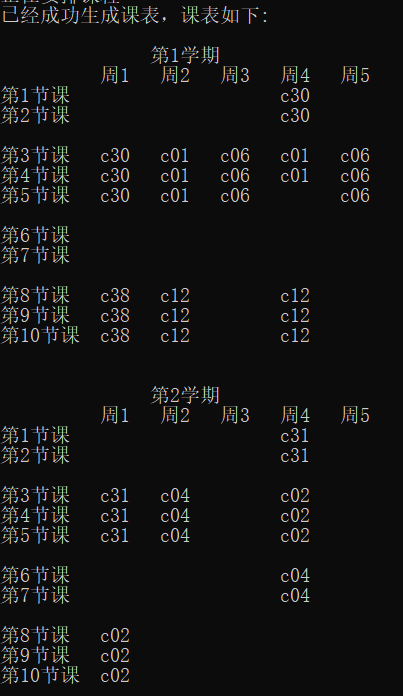
### 安排课程表流程图



### 安排课程表核心代码



### 安排课程表截屏示例



# 测试

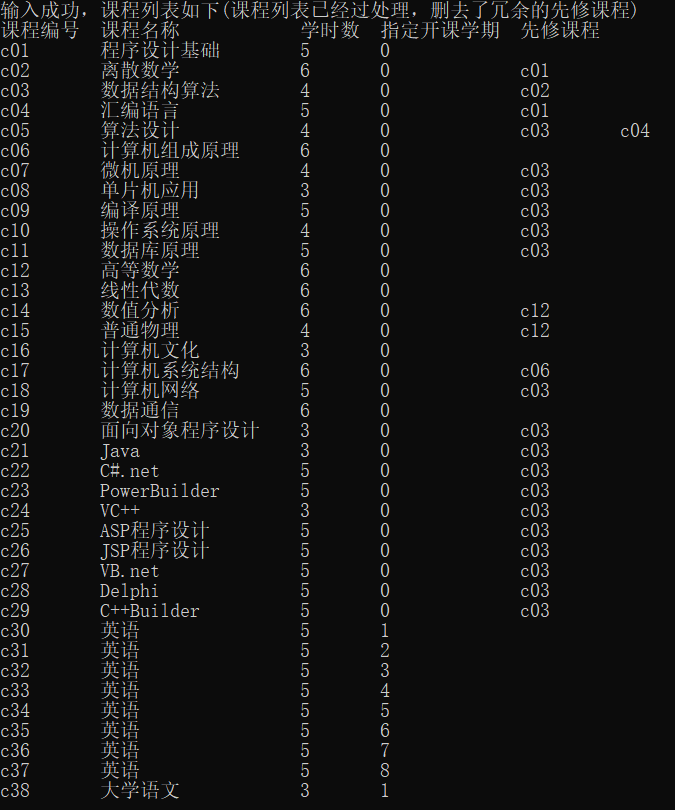
## 鲁棒性测试

### 正常案例测试

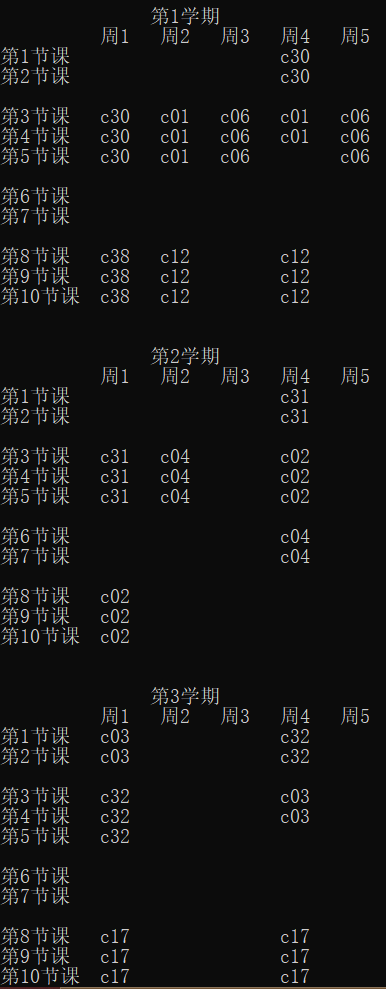
**测试用例：**一般情况

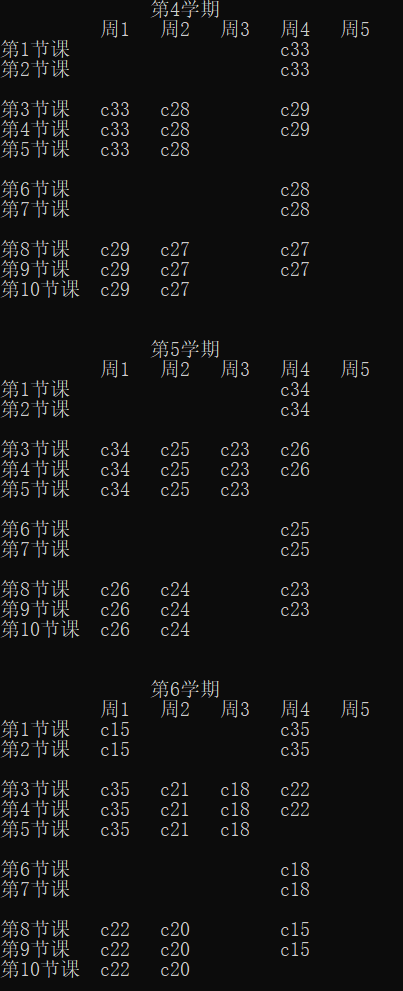
**预期结果：**对应的正确结果。

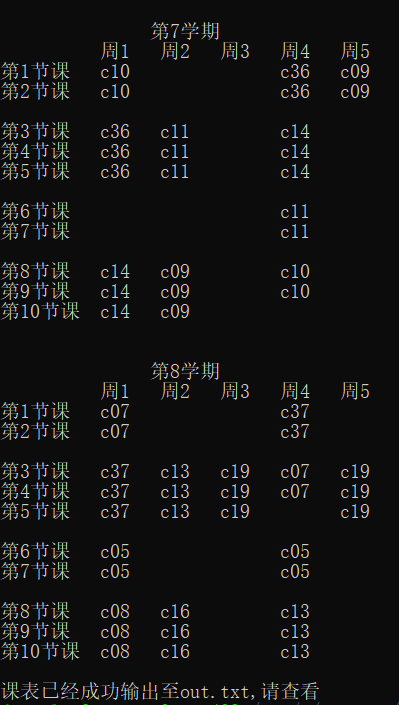
**实验结果：**











### 课程输入测试

**测试用例：**in.txt文件不存在

**预期结果：**给出提示，程序结束

**实验结果：**



**测试用例：**某课程信息不足

**预期结果：**给出提示，程序结束

**实验结果：**





**测试用例：**某学时数不为非负整数

**预期结果：**给出提示，程序结束

**实验结果：**





**测试用例：**某学时数大于50

**预期结果：**给出提示，程序结束

**实验结果：**





**测试用例：**某指定开课学期不为正整数

**预期结果：**给出提示，程序结束

**实验结果：**





**测试用例：**某指定开课学期大于8

**预期结果：**给出提示，程序结束

**实验结果：**

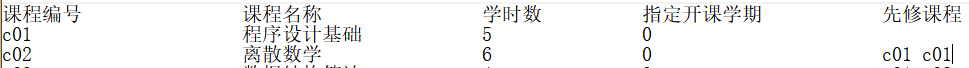


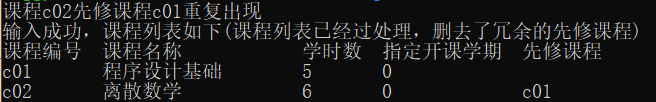


**测试用例：**先修课程重复出现

**预期结果：**给出提示，程序继续运行

**实验结果：**

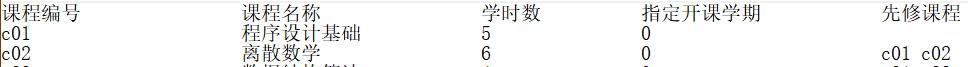


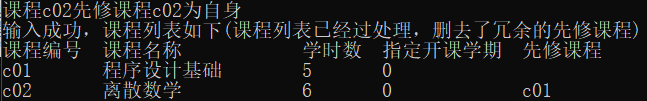


**测试用例：**先修课程为自身

**预期结果：**给出提示，程序继续运行

**实验结果：**



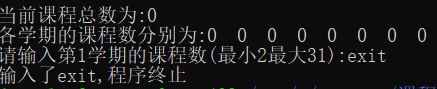


### 课程输入测试

**测试用例：**输入exit

**预期结果：**程序退出

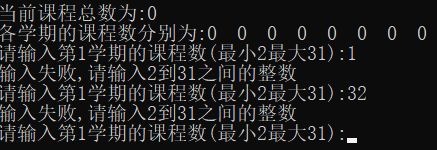
**实验结果：**



**测试用例：**输入非范围内的数字

**预期结果：**给出提示，程序继续运行

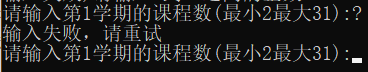
**实验结果：**



**测试用例：**输入非法字符

**预期结果：**给出提示，程序继续运行

**实验结果：**



**测试用例：**输入的8个课程数不存在可行解

**预期结果：**给出提示，重新输入课程数，程序继续运行

**实验结果：**

