数据结构课程设计

项目说明文档

排课软件

作者姓名: 汪明杰

学　　号: 1851055

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 项目分析 3
1.1 项目背景 3
1.2 项目需求分析 3
1.3 项目要求 4
1.3.1 功能要求 4
1.3.2 输入格式 4
1.3.3 输出格式 4
1.3.4 排课要求 4
1.3.5 课程信息 5
2 项目设计 6
2.1 数据结构设计 6
2.2 类设计 6
2.2.1 向量类（Vector） 6
2.2.2 双向链表类（List） 8
2.2.3 队列类（Queue） 9
3 项目实现 11
3.1 项目整体功能的实现 11
3.1.1 项目整体功能流程图 11
3.1.2 项目整体功能代码 12
3.2 拓扑排序的实现 14
3.2.1 拓扑排序流程图 14
3.2.2 拓扑排序代码 15
4 项目测试 16
4.1 项目结果 16
4.2 安排课程表 17

1 项目分析

1.1 项目背景

学校在学年开学之初必须要进行的一项工作是对班级、教师、课程及学校教学资源合理安排，制定各种各样课程表。排课就是将各班的课程、教师任课排列对应，形成表格。

由于计算机的崛起，排课工作已经可以通过计算机完成。利用计算机来完成繁琐的排课过程。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。在这样的前提下设计一个教学计划编制程序。

1.2 项目需求分析

针对于排课软件这一系统，本项目在实现的过程中，考虑并且满足了以下的需求：

- ✓ **功能完善**

  系统应该能够针对所输入的不同课表，合理的考虑前置课程并且完成排课任务。

- ✓ **执行效率高**

  针对数据量比较大的情况，本系统也应该具有在较短时间内求解出正确答案的能力。

- ✓ **代码可读性强**

  本项目在实现过程中，将代码根据功能的不同划分为了不同的代码块，同时进行了合理封装。

- ✓ **健壮性**

  当用户输入的数据不合理时，系统应当给予相应的提示而非直接报错。

- ✓ **可视化**

  该系统通过输出当前正在执行的操作内容，使得用户可以直观的了解到当前操作内容。

1.3 项目要求

1.3.1 功能要求

1.输入数据包括：八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。

1.3.2 输入格式

读取文件的地址。

1.3.3 输出格式

存储文件的地址。

1.3.4 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）

1.3.5 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 项目设计

2.1 数据结构设计

排课软件在排课时，各课程间存在前后关系，需要经过**拓扑排序**（Topo Sort）后才能够安排课程。

对一个有向无环图(Directed Acyclic Graph)G 进行拓扑排序，是将 G 中所有顶点排成一个线性序列，使得图中任意一对顶点 u 和 v，若边<u,v>∈E(G)，则 u 在线性序列中出现在 v 之前。通常，这样的线性序列称为满足拓扑次序的序列，简称拓扑序列。简单的说，由某个集合上的一个偏序得到该集合上的一个全序，这个操作称之为拓扑排序。

在拓扑排序过程中，需要借助**队列**（Queue）来完成。队列中存储当前入度为 0 的顶点，从队列中出队并且将该课程安排至课程表中。

2.2 类设计

本项目中使用到**队列**（Queue）这一数据结构，而队列的底层数据类型则采用了链表进行实实现。经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（ListNode）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。

为了实现代码的复用性，本系统实现了一个**链表**。采用 struct 描述链表结点类（ListNode），这样使得链表结点类（List）可以直接访问链表结点而不需要定义友元关系。本系统实现的链表结构各种操作的时间复杂度如下：

- ✓ 插入操作：O(n)
- ✓ 删除操作：O(n)
- ✓ 查询操作：O(n)
- ✓ 遍历操作：O(n)

2.2.1 向量类（Vector）

向量（Vector）是一个封装了动态大小数组的顺序容器（Sequence Container）。跟任意其它类型容器一样，它能够存放各种类型的对象。可以简单的认为，向量是一个能够存放任意类型的动态数组。需要注意到的是，向量具有以下两个特性：

（1） 顺序序列：

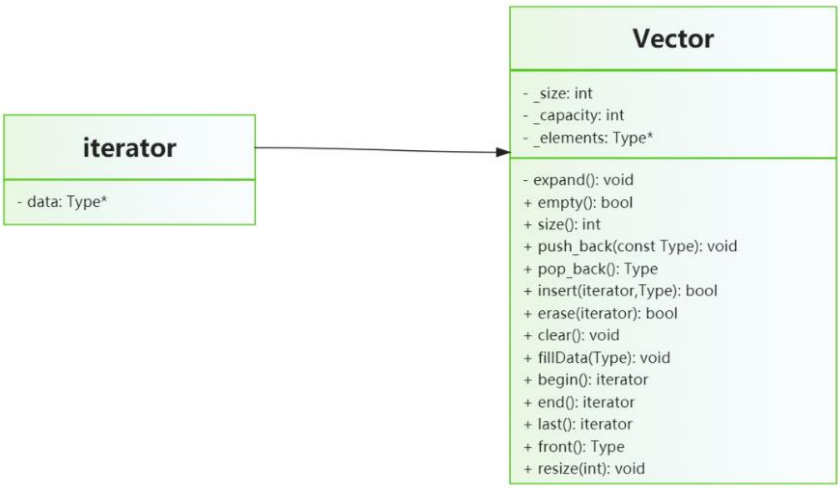
顺序容器中的元素按照严格的线性顺序排序，可以通过元素在序列中的位置访问对应的元素。本类中即通过重载取下标运算符[]实现了此功能。

（2） 动态数组：

支持对序列中的任意元素进行快速直接访问，甚至可以通过指针算术进行该操作，提供了在序列末尾相对快速地添加或者删除元素的操作。

本项目为了实现向量类，在内部定义了一个动态指针以及当前数组的大小。同时，封装了较多的函数以便使用。此外，为了便于对向量进行遍历、插入、删除、查找等操作，增加了一个 iterator 类。iterator 类内部存储一个数组元素指针，可以直接对向量的每个元素进行操作。同时，通过运算符重载相应的自增、自减、判等等操作。

该类和其内部的 iterator 类的 UML 图如下所示：



本类中的主要函数如下所示：

◆ `inline int size()const`

返回向量中元素的个数，也即_size 的大小。

◆ `inline bool empty()const`

判断向量是否为空，也即_size 是否为 0。

◆ `void push_back(const Type data)`

在向量尾端加入一个元素。

◆ `Type pop_back()`

删除向量最后一个元素，并且加以返回。

◆ `bool insert(const Vector<Type>::iterator place, Type item)`

在指定迭代器的位置插入元素，返回是否插入成功。

◆ `bool erase(const Vector<Type>::iterator place)`

删除指定迭代器位置的元素，返回是否删除成功。

◆ `void clear()`

清空向量中所有元素。

◆ `void fillData(const Type data)`

将向量中的元素统一赋值为同一个值。

◆ `iterator begin()`

返回向量首元素的迭代器。

◆ `iterator end()`

返回向量尾元素的下一个位置的迭代器。

◆ `iterator last()`

返回向量尾元素的迭代器。

◆ `Type& front()const`

返回向量的首个元素值。

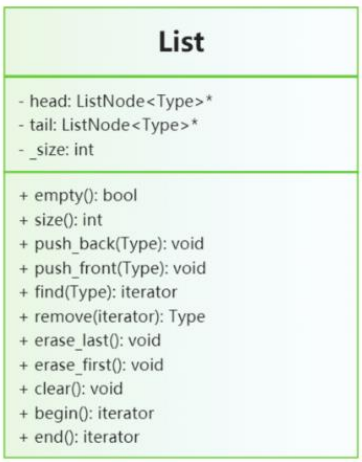
◆ `void resize(int sz)`

将向量重新设定大小，如果变小则删除多余元素。

2.2.2 双向链表类（List）

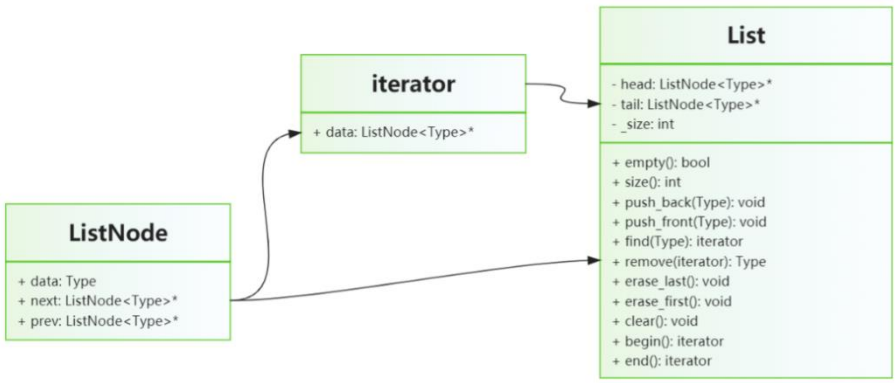
链表的实现原理大同小异，不同之处在于：是否带头结点、是否带尾结点、每一个结点是否带前驱结点等。为了使得链表中各种操作的时间复杂度都尽可能低，因此这里选择了带头结点和尾结点的双向链表来实现链表中的各种操作。也即：选择了牺牲空间来达到降低时间复杂度的效果。

链表的 UML 图如下所示：



为了便于对链表进行遍历、插入、删除、查找等操作，增加了一个 iterator 类。iterator 类内部存储一个链表节点指针，同时，通过运算符重载相应的自增、自减、判等等操作。

iterator 类、ListNode 类和 List 类的关系如下图所示：



其中，List 类中的主要函数如下所示：

◆ `inline int size()const`

返回链表中结点的个数，不包括头结点。

◆ `inline bool empty()const`

判断链表是否为空，也即链表中结点的个数是否为 0。

◆ `void push_back(Type data)`

在链表尾部插入新的数据，也即新增一个结点并且加入到链表末端。

◆ `void push_front(Type data)`

在链表头部插入新的数据，也即新增一个结点并且加入到链表的头部。

◆ `iterator find(const Type& data)const`

在链表中查找值为 data 的元素是否存在，返回该位置的迭代器，若查找失败返回空指针对应的迭代器。

◆ `Type remove(iterator index)`

移除迭代器所处位置的元素，返回移除位置元素的值。

◆ `void erase_last()`

移除链表末端的元素，即最后的结点。

◆ `void erase_first()`

移除链表首端的元素，即第一个结点。

◆ `void clear()`

清空链表，即删除链表中所有的结点。

◆ `iterator begin()`

返回链表第一个元素所在位置的迭代器。

◆ `iterator end()`

返回链表尾结点后的空结点的迭代器。

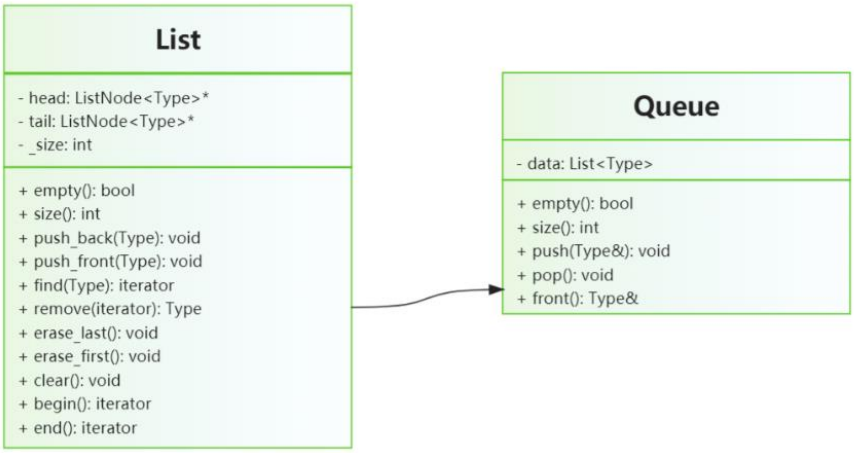
2.2.3 队列类（Queue）

队列是计算机程序中常用的数据结构，常常用于计算机模拟现实事务，如：排队等。队列是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）

进行删除操作，而在表的后端（rear）进行插入操作，因此是一种操作受限制的线性表。进行插入操作的端称为队尾，进行删除操作的端称为队头。队列中没有元素时，被称为空队列。

队列的数据元素又称为队列元素。在队列中插入一个队列元素称为入队，从队列中删除一个队列元素称为出队。因为队列只允许在一端插入，在另一端删除，所以只有最早进入队列的元素才能最先从队列中删除，故队列又称为先进先出（FIFO—first in first out）线性表。

队列的包括了顺序队列和循环队列，实现存储的底层数据可以通过向量或者链表完成。本项目中的队列以链表作为底层数据结构，实现了顺序队列。其 UNL 图如下所示：



队列中主要函数如下所示：

◆ `inline bool empty()const`

判断队列是否为空，也即队列内部链表是否为空。

◆ `inline int size()const`

返回队列中链表节点的个数。

◆ `void push(const Type& i)`

在队列尾部加入一个元素，也即入队。

◆ `void pop()`

删除队列头部的元素，也即出队。

◆ `const Type& front()const`

获取队首的元素值。

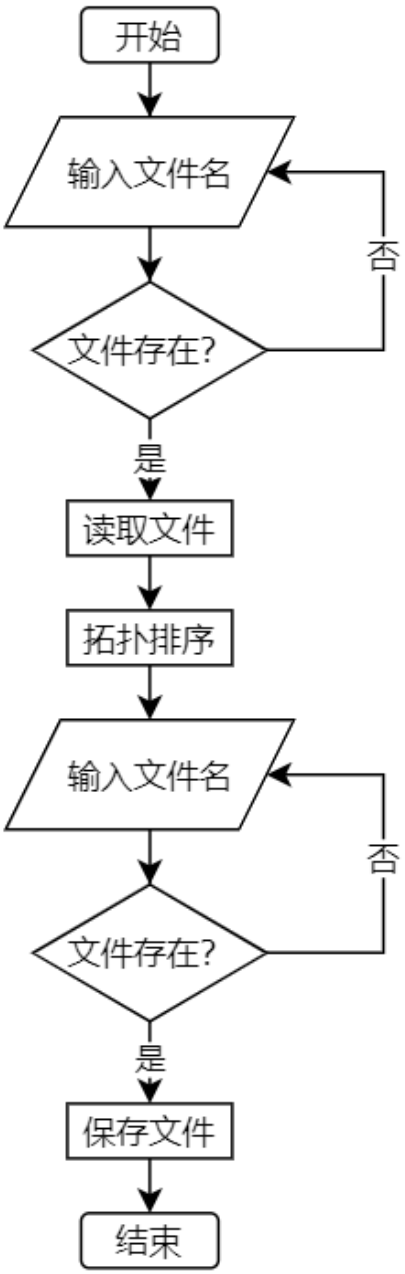
通过上述函数操作，即完成了一个队列所需要的最基本操作。其中，队的各个操作的时间复杂度如下所示：

- ✓ 入队操作：O(1)
- ✓ 出队操作：O(1)
- ✓ 读取操作：O(1)

3 项目实现

3.1 项目整体功能的实现

3.1.1 项目整体功能流程图



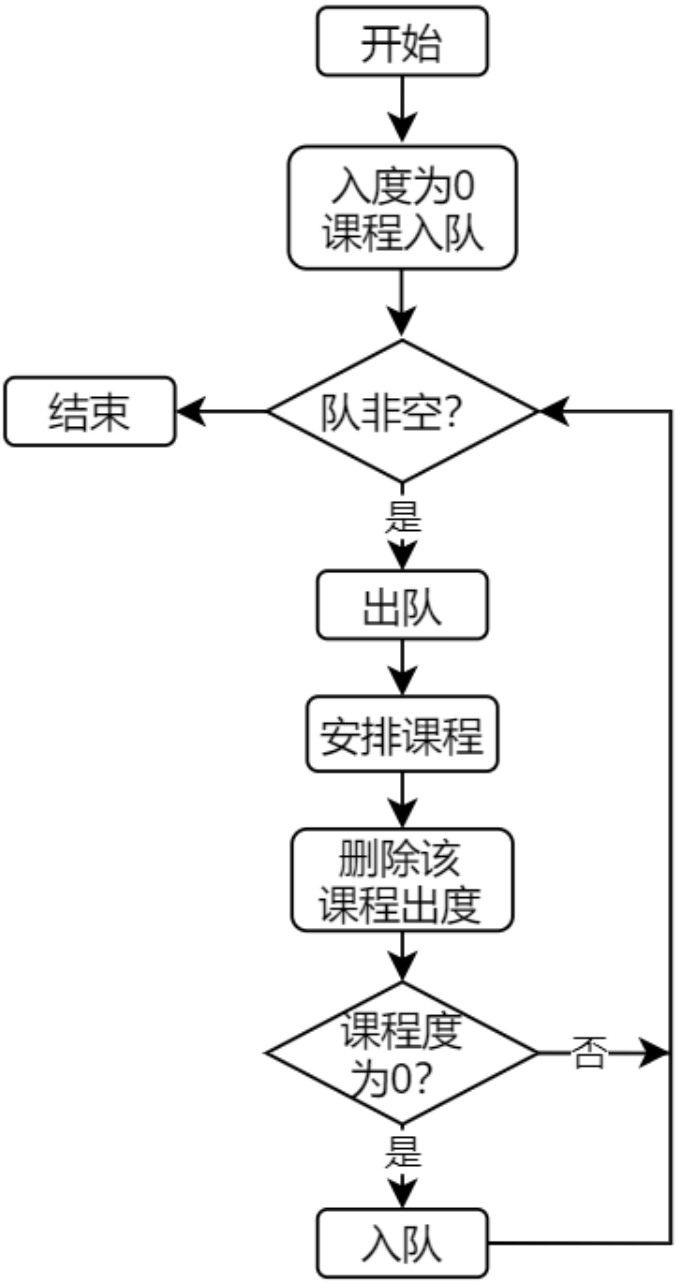
3.1.2 项目整体功能代码

```
int main()
{
    //读取文件
    cout << "请输入读取文件的位置(输入 0 读取默认文件):";
    string fileLoad;
    ifstream file(fileLoad);
    //读取文件
    while (true)
    {
        cin >> fileLoad;
        if (fileLoad.length() == 0 || fileLoad == "0")
        {
            cout << "读取默认文件 input.txt";
            fileLoad = "input.txt";
        }
        file.open(fileLoad);
        //查看文件是否存在
        if (file.is_open())
            break;
        cout << endl << fileLoad << "不存在! " << endl;
    }
    //读取课程总数
    file >> courseNum;
    courses.resize(courseNum);
    preCourse.resize(courseNum);
    //读取每一门课程信息
    for (int i = 0; i < courseNum; ++i)
    {
        //读取到文件末尾
        if (file.eof())
            break;
        getline(file, fileLoad);
        //定义字符串流
        stringstream sr(fileLoad);
        sr >> courses[i].id >> courses[i].name >>
            courses[i].number >> courses[i].term;
        //读取前置课程信息
        while (!sr.eof())
        {
            sr >> fileLoad;
            preCourse[i].push_back(fileLoad);
```

```
        }
    }
    file.close();
    //检查输入是否正确
    if (courses.size() != courseNum)
    {
        cout << "课程总数有误，请检查数据" << endl;
        courseNum = courses.size();
    }

    topoSort();

    //写文件
    cout << "请输入输出文件位置(输入0存储到默认位置): ";
    string saveFile;
    cin >> saveFile;
    if (saveFile == "0")
        saveFile = "output.txt";
    ofstream save(saveFile);
    for (int curterm = 0; curterm < 8; curterm++)
    {
        save << "第" << curterm + 1 << "学期课表" << endl;
        for (int curcourse = 0; curcourse < 10; curcourse++)
        {
            for (int curday = 0; curday < 5; curday++)
            {
                int index = courseTable[curterm][curday][curcourse];
                if (index == -1)
                    save << left << setw(20) << "--------------
" << "\t";
                else
                    save << left << setw(20) << courses[index].name + c
ourses[index].id << "\t";
            }
            save << endl;
        }
        save << endl;
    }
    cout << "已存储至" << saveFile << endl;
    save.close();

    return 0;
}
```

3.2 拓扑排序的实现

3.2.1 拓扑排序流程图



3.2.2 拓扑排序代码

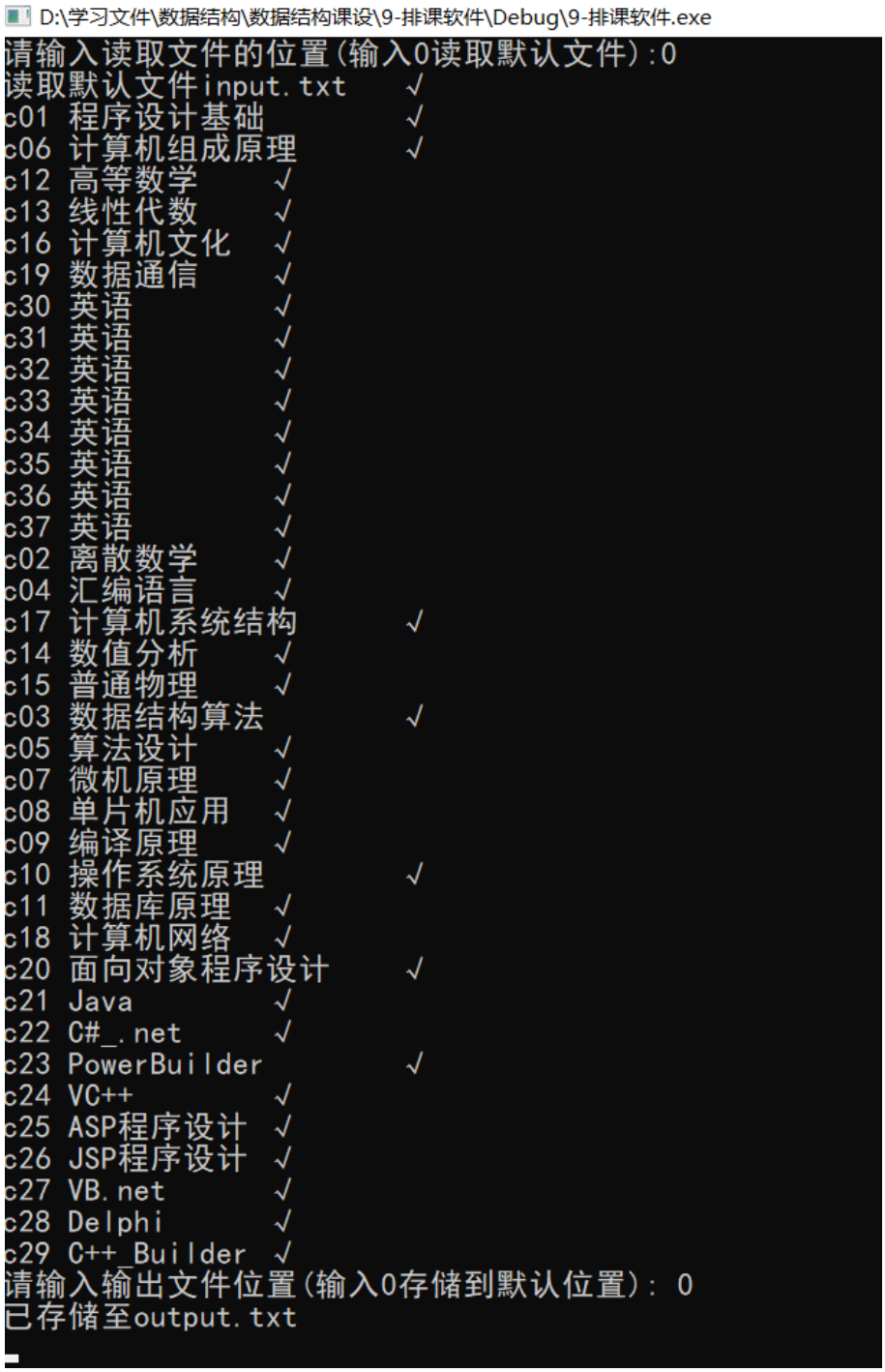
```
void topoSort()
{
    //入度为0 顶点组成的队列
    Queue<int> procedure;
    //记录入度数
    Vector<int> inDegrees(courseNum);
    //遍历，找出入度为0 的点
    for (int i = 0; i < courseNum; ++i)
    {
        inDegrees[i] = preCourse[i].size();
        if (inDegrees[i] == 0)
        {
            procedure.push(i);
        }
    }
    //拓扑过程
    while (!procedure.empty())
    {
        int index = procedure.front();
        procedure.pop();
        arrangeCourse(index); //对该课程进行安排
        //将所以该课程的入度设为0
        for (int i = 0; i < courseNum; ++i)
        {
            for (int j = 0; j < preCourse[i].size(); ++j)
            {
                if (preCourse[i][j] == courses[index].id)
                {
                    --inDegrees[i];
                    if (inDegrees[i] == 0) //如果入度为0
                    {
                        //将最早安排学期置为下一学期
                        if (courses[i].term < courses[index].term + 1)
                            courses[i].term = courses[index].term + 1;
                        procedure.push(i);
                    }
                }
            }
        }
    }
}
```

4 项目测试

测试用例：1.3.5 中课程信息

预期结果：合理安排的课程表

4.1 项目结果



4.2 安排课程表

第 1 学期课表					
上午			程序设计基础 c01		
			程序设计基础 c01		
	程序设计基础 c01	线性代数 c13	计算机组成原理 c06	数据通信 c19	高等数学 c12
	程序设计基础 c01	线性代数 c13	计算机组成原理 c06	数据通信 c19	高等数学 c12
	程序设计基础 c01	线性代数 c13	计算机组成原理 c06	数据通信 c19	高等数学 c12
下午					
	计算机组成原理 c06	计算机文化 c16	高等数学 c12	数据通信 c19	线性代数 c13
	计算机组成原理 c06	计算机文化 c16	高等数学 c12	数据通信 c19	线性代数 c13
	计算机组成原理 c06	计算机文化 c16	高等数学 c12	数据通信 c19	线性代数 c13

第 2 学期课表					
上午	普通物理 c15		英语 c30		汇编语言 c04
			英语 c30		汇编语言 c04
	英语 c30	计算机系统结构 c17	离散数学 c02	普通物理 c15	计算机系统结构 c17
	英语 c30	计算机系统结构 c17	离散数学 c02	普通物理 c15	计算机系统结构 c17
	英语 c30	计算机系统结构 c17	离散数学 c02	普通物理 c15	计算机系统结构 c17
下午					
	离散数学 c02	数值分析 c14	汇编语言 c04		数值分析 c14
	离散数学 c02	数值分析 c14	汇编语言 c04		数值分析 c14
	离散数学 c02	数值分析 c14	汇编语言 c04		数值分析 c14

第 3 学期课表					
上午			英语 c31		
			英语 c31		
	英语 c31		数据结构算法 c03		
	英语 c31				
	英语 c31				
下午					
	数据结构算法 c03				
	数据结构算法 c03				
	数据结构算法 c03				

第 4 学期课表					
上午	计算机网络 c18	编译原理 c09	英语 c32	操作系统原理 c10	微机原理 c07
	计算机网络 c18	编译原理 c09	英语 c32		
	英语 c32	操作系统原理 c10	算法设计 c05	计算机网络 c18	单片机应用 c08
	英语 c32	操作系统原理 c10		计算机网络 c18	单片机应用 c08
	英语 c32	操作系统原理 c10		计算机网络 c18	单片机应用 c08
下午					
	算法设计 c05	数据库原理 c11	微机原理 c07	数据库原理 c11	编译原理 c09
	算法设计 c05	数据库原理 c11	微机原理 c07	数据库原理 c11	编译原理 c09
	算法设计 c05	数据库原理 c11	微机原理 c07		编译原理 c09

第 5 学期课表					
上午	VB.netc27	PowerBuilderc23	英语 c33	ASP 程序设计 c25	C#_.netc22
	VB.netc27	PowerBuilderc23	英语 c33	ASP 程序设计 c25	C#_.netc22
	英语 c33	ASP 程序设计 c25	Javac21	VB.netc27	PowerBuilderc23
	英语 c33	ASP 程序设计 c25	Javac21	VB.netc27	PowerBuilderc23
	英语 c33	ASP 程序设计 c25	Javac21	VB.netc27	PowerBuilderc23
下午					
	面向对象程序设计 c20	JSP 程序设计 c26	C#_.netc22	JSP 程序设计 c26	VC++c24
	面向对象程序设计 c20	JSP 程序设计 c26	C#_.netc22	JSP 程序设计 c26	VC++c24
	面向对象程序设计 c20	JSP 程序设计 c26	C#_.netc22		VC++c24

第 6 学期课表					
上午			英语 c34		C++_Builderc29
			英语 c34		C++_Builderc29
	英语 c34		C++_Builderc29		
	英语 c34		C++_Builderc29		
	英语 c34		C++_Builderc29		
下午					
	Delphic28		Delphic28		
	Delphic28		Delphic28		
	Delphic28				

第 7 学期课表					
上午			英语 c35		
			英语 c35		
	英语 c35				
	英语 c35				
	英语 c35				
下午					

第 78 学期课表					
上午			英语 c36		
			英语 c36		
	英语 c36				
	英语 c36				
	英语 c36				
下午					