项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名： 郑柯凡

学 号： 1950072

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 2](#_Toc13203)

[1.1背景分析 2](#_Toc31777)

[1.2项目前提假设 2](#_Toc24568)

[1.3功能要求 3](#_Toc32414)

[2 设计 3](#_Toc31765)

[2.1思路设计 3](#_Toc4007)

[2.2数据结构的选择 3](#_Toc10364)

[2.3类的设计 3](#_Toc15631)

[3 功能实现 6](#_Toc5397)

[3.1 输入功能的实现 6](#_Toc29428)

[3.1.1输入功能流程图 6](#_Toc29175)

[3.1.2输入功能核心代码 7](#_Toc3003)

[3.2 拓扑排序的实现 8](#_Toc6131)

[3.2.1拓扑排序流程图 8](#_Toc19697)

[3.2.2拓扑排序核心代码 9](#_Toc11897)

[3.3 生成课表功能的实现 10](#_Toc8202)

[3.3.1生成课表功能流程图 10](#_Toc458)

[3.3.2生成课表功能核心代码  11](#_Toc3679)

[3.4 输出功能的实现 13](#_Toc5010)

[3.4.1输出功能流程图 13](#_Toc473)

[3.4.2输出功能核心代码 13](#_Toc339)

[3.4.3输出功能截屏示例 15](#_Toc6181)

[4 测试 16](#_Toc30554)

[4.1排课测试 16](#_Toc1135)

[4.2报错测试 18](#_Toc23559)

**1 分析**

**1.1背景分析**

排课软件是指用来帮助学校编排课程表的[电脑软件](https://baike.baidu.com/item/%E7%94%B5%E8%84%91%E8%BD%AF%E4%BB%B6/1711577" \t "https://baike.baidu.com/item/%E6%8E%92%E8%AF%BE%E8%BD%AF%E4%BB%B6/_blank)。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。排课就是将各门课程排列对应，形成表格，且各课程在开设时间的安排必须满足先修关系。由于计算机的掘起，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。

**1.2项目前提假设**

前提假设：

a.任何专业都有固定的学习年限，每学年含两学期。

b.课程开设时间必须满足先修关系。

c.每门课恰好占一个学期。

d.每天上午与下午各有5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。

**1.3功能要求**

A.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，先修课程。若指定开课学期为0，表示自行指定开课学期。

B.若输入数据不合理，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。

C.用文本文件存储输入数据，并且读入计算机。

D.用文本文件存储产生的各学期的课表。

E.若一门课程有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。

F.若一门课程需要安排上两天，为教学效果较好，优先安排相隔2天上课

**2 设计**

**2.1思路设计**

各门课程及其之间的先修关系可以用网络关系来表示。在排课时，可以先从文本文件中读取课程信息存储在网络中，通过对网络的拓扑排序理清各课程之间的先修关系，并将拓扑排序的结果结合每周课时数的要求以及指定课程学期的要求，将课程分配进每个学期。最后再在每个学期中拆分课程，使有3节课的优先上三节课，且有需要两天上的安排相隔两天上课。

**2.2数据结构的选择**

使用图来存储课程之间的AOV网络。

**2.3类的设计**

顶点类，用于存储单门课程的信息：其私有成员包括课程的数字编号，课程的编号，名称，学时，学期，先修课程，先修课程数量等。

1. //邻接表顶点顺序表中的顶点结点
2. **class** Vertex {
3. **friend** **class** Graph;
4. **friend** **class** ArrangeSystem;
5. **private**:
6. //课程对应的数字编号
7. **int** num;
8. //课程的相关信息（编号，名称，学时，学期，先修课程，先修课程数量）
9. string number, name, learntime, semester;
10. string prerequisite[MaxVertices];
11. **int** prenum;
12. **public**:
13. Vertex()
14. {
15. num = prenum = 0;
16. number = name = learntime = semester = "0";
17. };
18. ~Vertex() {};
19. };

边结点类，存储邻接表中的每条边：其私有成员包括终点的数字编号。

1. //邻接表边链表中的边结点
2. **class** Edge {
3. **friend** **class** Graph;
4. **friend** **class** ArrangeSystem;
5. **private**:
6. **int** destnum;
7. **public**:
8. Edge() { destnum = -1; };
9. ~Edge() {};
10. };

图类，用来存储所有课程的网络表示：其私有成员包括用顺序表存储的顶点集合，用邻接表存储的边集合，顶点的数量，count数组（用于拓扑排序）和visited数组（用于判断课程是否已被安排）。

1. //用邻接表存储的图，用来存储所有课程的信息和先修关系
2. **class** Graph {
3. **friend** **class** ArrangeSystem;
4. **private**:
5. //顶点顺序表存储顶点信息
6. MyVector<Vertex>vertextable;
7. //邻接表存储边的信息
8. MyVector<MyVector<Edge>>edgelist;
9. //图顶点的数量
10. **int** vertexnum;
11. //count数组，拓扑排序时使用
12. **int** count[MaxVertices]{};
13. //visited访问数组，检测某门课程是否已被安排
14. **bool** visited[MaxVertices]{};
15. **public**:
16. Graph()
17. {
18. vertexnum = 0;
19. **for** (**int** i = 0; i < MaxVertices; i++)
20. {
21. count[i] = 0;
22. visited[i] = 0;
23. }
24. };
25. ~Graph() {};
26. //在邻接表中插入顶点和边的信息
27. **void** InsertVertex(Vertex x);
28. **void** InsertEdge(Vertex x);
29. //通过编号获取顶点位置
30. **int** GetVertexNum(string name);
31. };

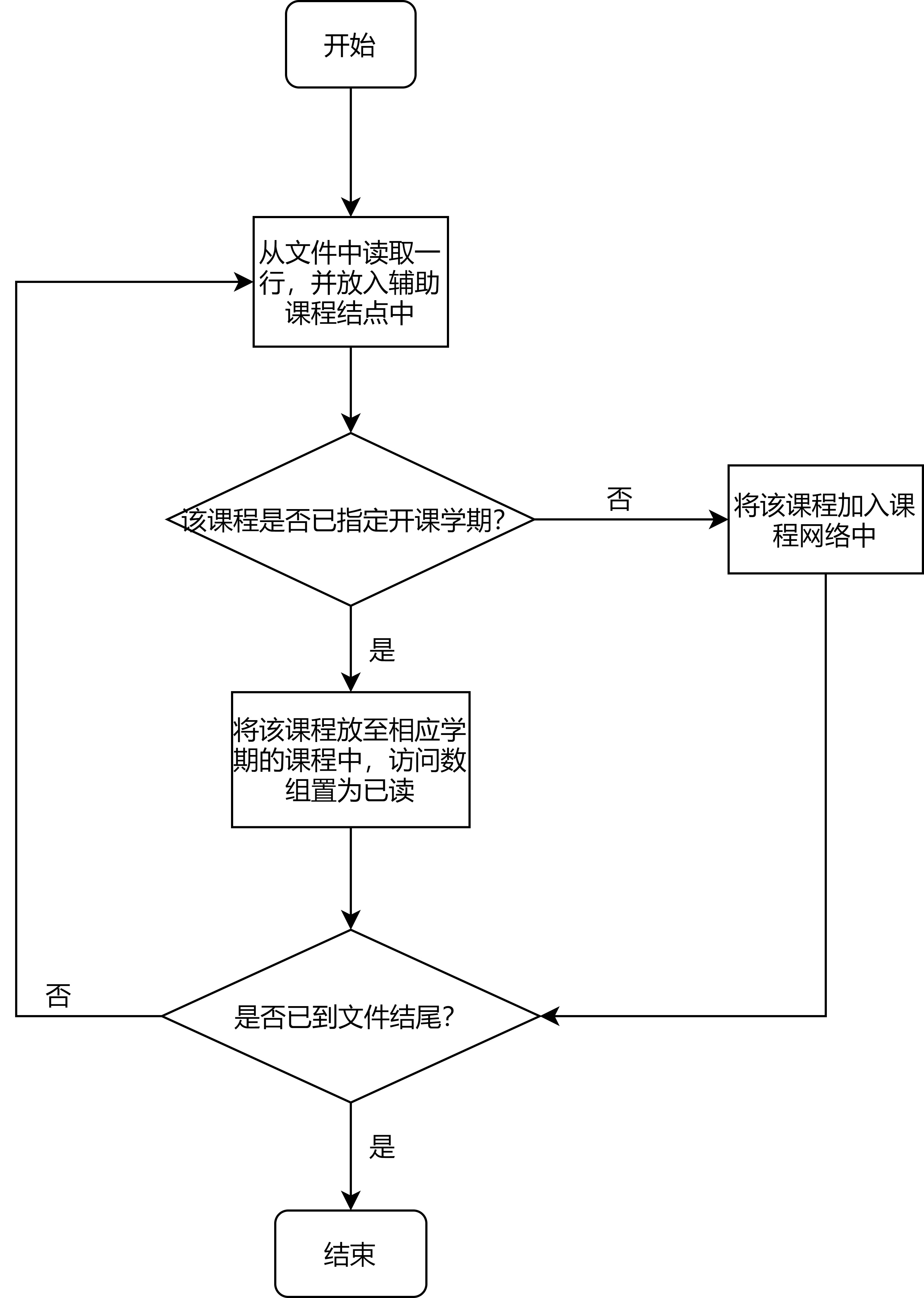
排课系统类：其私有成员包括图存储的课程网络，顺序表数组存储的每个学期课程，以及三维数组实现的最终课表。其公有成员包括输入函数，输出函数，拓扑排序函数，生成课表函数，以及辅助生成课表的课程插入函数（包含插入1节课的，2节课的和3节课的）。

1. //排课系统
2. **class** ArrangeSystem {
3. **private**:
4. //所有课程的图表示
5. Graph all\_courses;
6. //存储每个学期的课程
7. MyVector<Vertex>schedule[8];
8. //生成的8个学期的课表
9. **int** final\_schedule[8][5][10]{};
10. **public**:
11. ArrangeSystem()
12. {
13. **for** (**int** i = 0; i < 8; i++)
14. {
15. **for** (**int** j = 0; j < 5; j++)
16. {
17. **for** (**int** k = 0; k < 10; k++)
18. {
19. final\_schedule[i][j][k] = -1;
20. }
21. }
22. }
23. };
24. ~ArrangeSystem() {};
25. //从文本文件中输入数据
26. **void** Input();
27. //将数据输出到文本文件中
28. **void** Output();
29. //拓扑排序，并把课程分配到相应的学期中
30. **void** Topologicalsort();
31. //将每个学期内的课分配到各个课时，并生成课表
32. **bool** GetFinalSch();
33. **void** InsertCourse(**int** semester, **int** j, **int** courselen);
34. **bool** InsertThree(**int** semester,**int** weekday,Vertex x);
35. **bool** InsertTwo(**int** semester,**int** weekday, Vertex x);
36. **bool** InsertOne(**int** semester,**int** weekday, Vertex x);
37. };

**3 功能实现**

**3.1 输入功能的实现**

**3.1.1输入功能流程图**



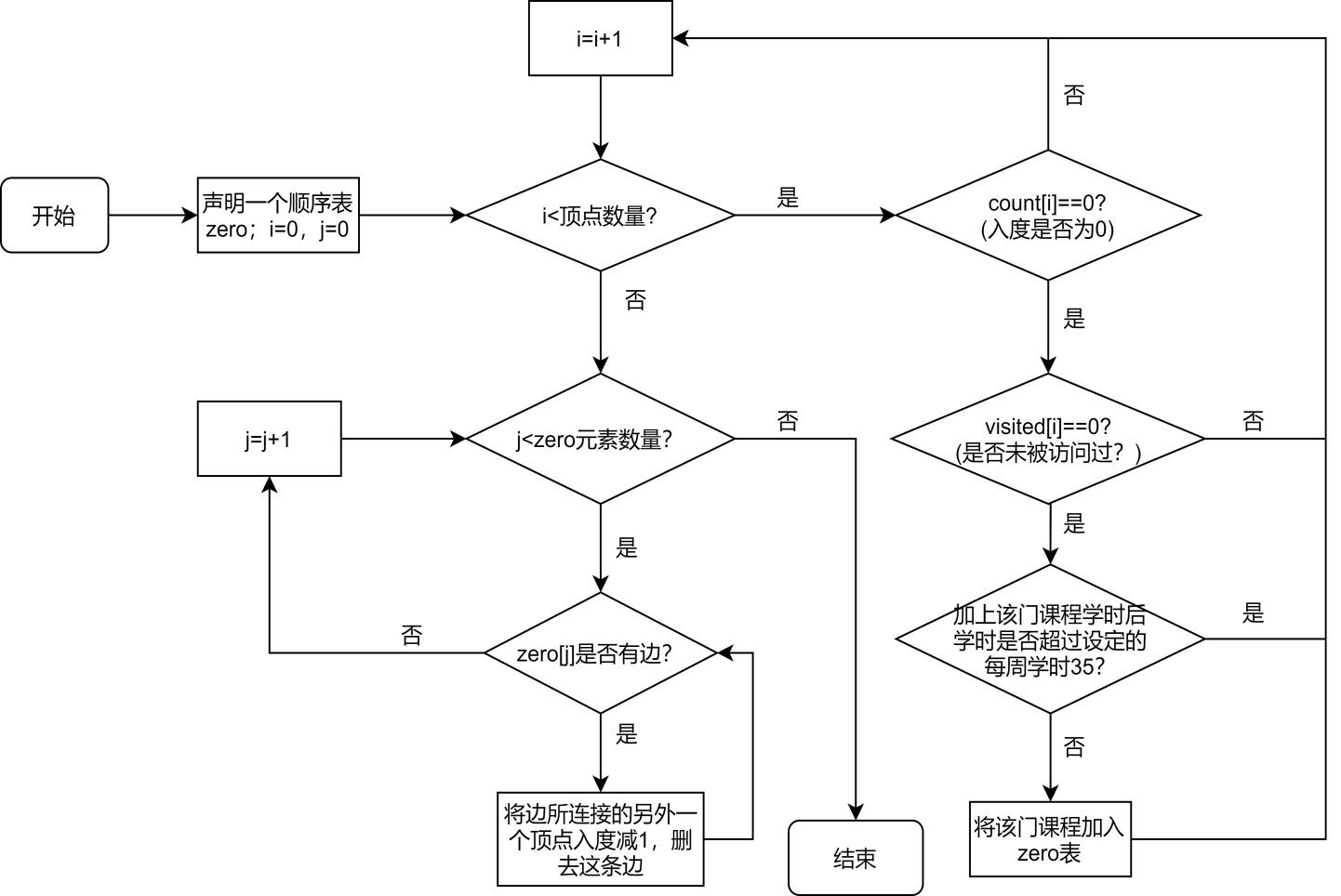
**3.1.2输入功能核心代码**

从文件中一行一行读取数据，将读取到的一行数据（即一门课程）用一个辅助结点存储，若该门课程已被指定了学期数，则直接将该门课程放进对应学期的课程中，并将对应的visited数组置为1。若没有指定学期数，则将该辅助结点插入网络中。

1. //输入函数
2. **void** ArrangeSystem::Input()
3. {
4. //读取文件
5. fstream in;
6. in.open("in.txt",ios::in);
7. **int** num = 0;
8. **char** str[1024];
9. in.getline(str, 1024);
10. **while** (!in.eof())
11. {
12. //临时课程变量，辅助输入
13. Vertex curcourse;
14. //课程数字编号
15. curcourse.num = num;
16. num++;
17. //从文本中按行读取数据并分割字符串
18. **int** cnt=0;
19. in >> curcourse.number >> curcourse.name >> curcourse.learntime >> curcourse.semester;
20. **while** (!in.eof())
21. {
22. **if** (in.get() != '\n')
23. {
24. in >> curcourse.prerequisite[cnt];
25. cnt++;
26. }
27. **else**
28. {
29. **break**;
30. }
31. }
32. curcourse.prenum = cnt;
33. //---------------------------------------------------------------------------------------//
34. //将顶点和边插入图
35. //优先处理已确定学期的课程
36. **switch** (curcourse.semester[0])
37. {
38. **case**'1':schedule[0].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
39. **case**'2':schedule[1].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
40. **case**'3':schedule[2].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
41. **case**'4':schedule[3].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
42. **case**'5':schedule[4].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
43. **case**'6':schedule[5].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
44. **case**'7':schedule[6].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
45. **case**'8':schedule[7].push\_back(curcourse); all\_courses.visited[curcourse.num] = 1; **break**;
46. **default**:**break**;
47. }
48. //通用插入操作
49. all\_courses.InsertVertex(curcourse);
50. **if** (curcourse.prenum != 0)
51. {
52. all\_courses.InsertEdge(curcourse);
53. }
54. }
55. in.close();
56. }

**3.2 拓扑排序**的实现

**3.2.**1拓扑排序流程图



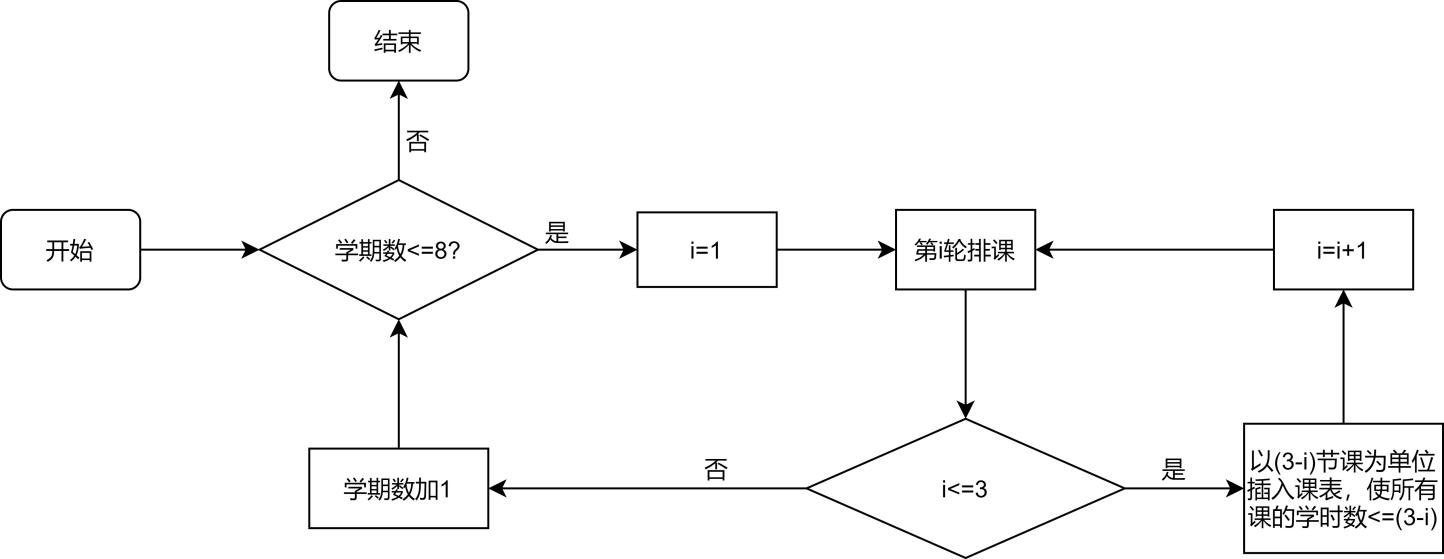
**3.2.2拓扑排序核心代码**

声明一个zero顺序表，用来存储本学期要上的课程顶点。遍历顶点表，若该课程入度为0（表明没有先修课程或先修课程已修完）且未被访问且加入该门课程学时后该学期总学时不超过35，则将该门课程放入zero中。当本学期可以上的课程全部确定后，遍历zero表更新count数组，将与已选顶点存在边的顶点入度减1。然后重复以上操作获得8个学期对应的课程。

1. //根据课程学时进行拓扑排序
2. **void** ArrangeSystem::Topologicalsort()
3. {
4. **for**(**int** semester=0;semester<8;semester++)
5. {
6. //记录学时并初始化
7. **int** learnedtime=0;
8. **for** (**int** i = 0; i < schedule[semester].get\_size(); i++)
9. {
10. learnedtime += schedule[semester].get\_element(i)->GetData().learntime[0]-48;
11. }
12. //存储被选中课程，在最后进行入度更新
13. MyVector<Vertex>zero;
14. **for** (**int** i = 0; i < all\_courses.vertexnum; i++)
15. {
16. **if** (learnedtime >= MaxCourseTime)//学时数已满，进入下一个学期排课
17. {
18. **break**;
19. }
20. **if** (all\_courses.visited[i] == 0)//未确定学期
21. {
22. **if** (all\_courses.count[i] == 0)//入度为0
23. {
24. **if** (learnedtime + all\_courses.vertextable.get\_element(i)->GetData().learntime[0]-48 <= 35)//未超过学时数
25. {
26. //更新学时
27. learnedtime += all\_courses.vertextable.get\_element(i)->GetData().learntime[0] - 48;
28. //将该课程加入该学期,并进行标记
29. Vertex xxxx;
30. xxxx = all\_courses.vertextable.get\_element(i)->GetData();
31. schedule[semester].push\_back(xxxx);
32. all\_courses.visited[xxxx.num] = 1;
33. //将该课程加入零度表
34. zero.push\_back(xxxx);
35. }
36. }
37. }
38. }
39. //对count表进行更新
40. **for** (**int** i = 0; i < zero.get\_size(); i++)
41. {
42. **for** (**int** j = 0; j < all\_courses.edgelist.get\_element(zero.get\_element(i)[0].GetData().num)[0].GetData().get\_size(); j++)
43. {
44. **int** curvertexnum = all\_courses.edgelist.get\_element(zero.get\_element(i)->GetData().num)->GetData().get\_element(j)->GetData().destnum;
45. all\_courses.count[curvertexnum]--;
46. }
47. }
48. }
49. }

**3.3 生成课表功能的实现**

**3.3.1生成课表功能流程图**

****

**3.3.2生成课表功能核心代码**

根据题目要求，三节课优先安排，因此安排3轮排课顺序。第一轮，先安排学时>=3的课程,使这一轮下来，所有该学期内要上的课程学时数都<3;接着第二轮排课，使所有课程的剩余学时数<2;最后第三轮只剩下了学时数为1的课程，因此见缝插针，有空余的时间就安排课程。

1. //生成课表
2. **bool** ArrangeSystem::GetFinalSch()
3. {
4. Topologicalsort();
5. **for** (**int** i = 0; i < all\_courses.vertexnum; i++)
6. {
7. **if** (all\_courses.visited[i] == 0)
8. {
9. cout << "课程输入有误，请检查课程的学期和先修关系!"<<endl;
10. **return** 0;
11. }
12. }
13. **for** (**int** semester = 0; semester < 8; semester++)
14. {
15. //先处理三节的大课，两节次之，一节最末
16. **for** (**int** turn = 0; turn < 3; turn++)
17. {
18. **for** (**int** j = 0; j < schedule[semester].get\_size(); j++)
19. {
20. **if** (schedule[semester].get\_element(j)->GetData().learntime[0] == 48)
21. {
22. **continue**;
23. }
24. **switch** (turn)
25. {
26. **case** 0:InsertCourse(semester, j, 3); **break**;
27. **case** 1:InsertCourse(semester, j, 2); **break**;
28. **case** 2:InsertCourse(semester, j, 1); **break**;
29. }
30. }
31. }
32. }
33. **return** 1;
34. }

根据轮次的不同，初始安排的课程时间也不同。第一轮安排3节课的从周一开始，第二轮安排2节课的从周3开始，第三轮安排1节课的从周五开始。使不同课程之间的时间相对交错。安排时，学时数为4的课是一个特例，因为它分成2+2节课上更加合适，因此把它安排至下一轮排课。然后根据日期排课，若当天假设是weekday排课成功了，则令weekday=(weekday - 3 >= 0) ? (weekday - 3) : (weekday + 2);保证相同的课是隔天上的；若当天weekday排课失败，则令weekday=weekday+1，从后一天开始继续尝试。

1. //插入课程
2. **void** ArrangeSystem::InsertCourse(**int** semester, **int** j,**int** courselen)
3. {
4. **int** weekday = 0;
5. **if** (courselen == 3)
6. {
7. weekday = 0;
8. }
9. **else** **if** (courselen == 2)
10. {
11. weekday = 2;
12. }
13. **else** **if** (courselen == 1)
14. {
15. weekday = 4;
16. }
17. Vertex curvertex = schedule[semester].get\_element(j)->GetData();
18. **int** learntime = schedule[semester].get\_element(j)->GetData().learntime[0] - 48;
19. **while** (learntime >= courselen)
20. {
21. **if** (courselen == 3 && learntime == 4)
22. {
23. **break**;
24. }
25. **int** sign = 0;
26. **for** (**int** fre = 0; fre < 10; fre++)
27. {
28. **bool** judge=0;
29. **if**(courselen==3)
30. {
31. judge = InsertThree(semester, weekday, curvertex);
32. }
33. **else** **if** (courselen == 2)
34. {
35. judge = InsertTwo(semester, weekday, curvertex);
36. }
37. **else** **if** (courselen == 1)
38. {
39. judge = InsertOne(semester, weekday, curvertex);
40. }
41. **if** (judge==1)
42. {
43. Vertex newvertex= schedule[semester].get\_element(j)->GetData();
44. newvertex.learntime[0] -=  courselen;
45. schedule[semester].get\_element(j)->ModifyData(newvertex);
46. learntime -= courselen;
47. weekday = (weekday - 3 >= 0) ? (weekday - 3) : (weekday + 2);
48. sign = 1;
49. **break**;
50. }
51. weekday++;
52. weekday = weekday % 5;
53. }
54. **if** (sign == 0)
55. {
56. **break**;
57. }
58. }
59. }

**3.4 输出功能的实现**

**3.4.1输出功能流程图**

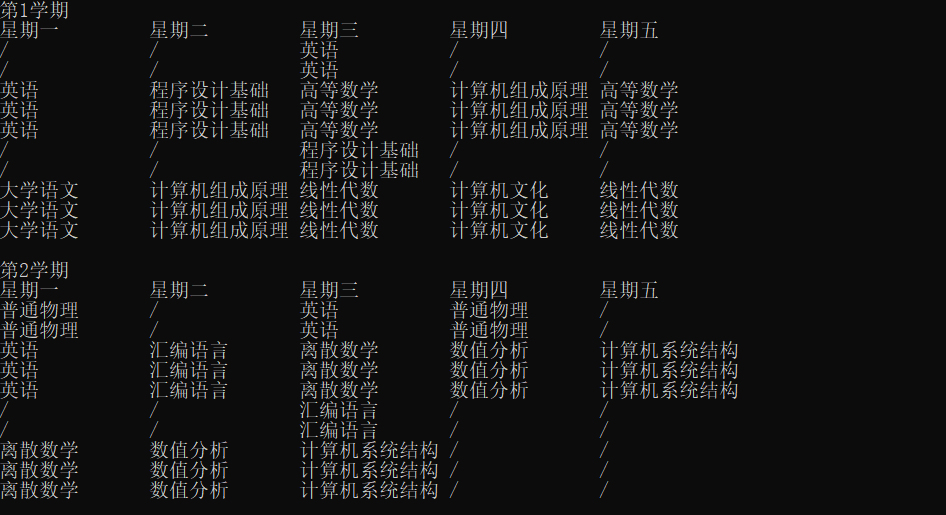
****

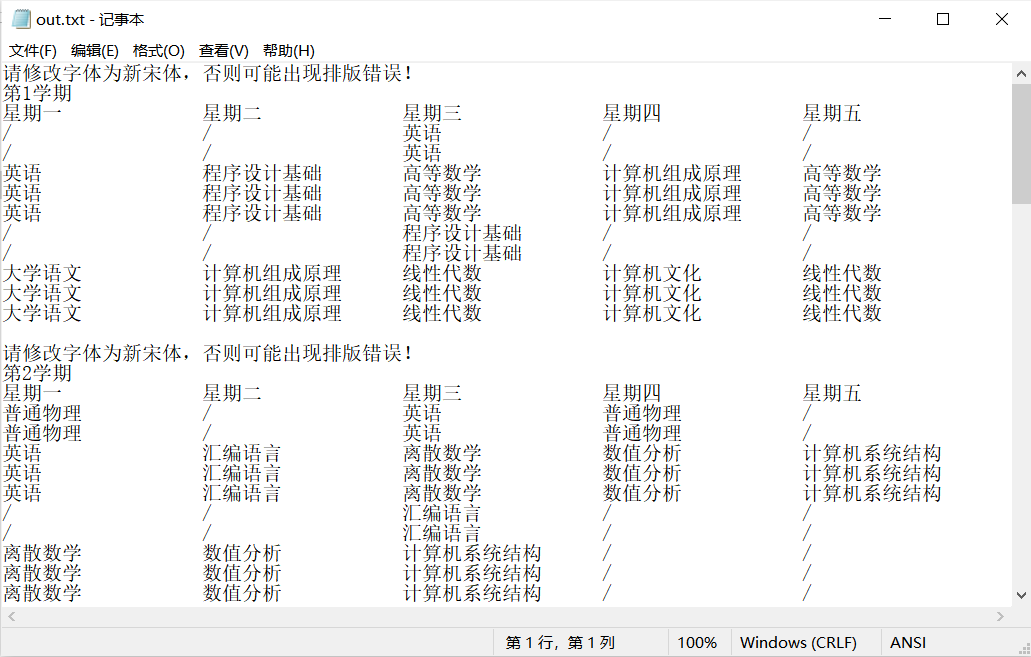
**3.4.2输出功能核心代码**

将课表打印在控制台应用程序和文件中。（控制台应用程序未作要求，但方便观察排课的情况）

1. //输出函数
2. **void** ArrangeSystem::Output()
3. {
4. ofstream out("out.txt");
5. //控制台应用程序打印，方便观察
6. **for** (**int** i = 0; i < 8; i++)
7. {
8. cout << setiosflags(ios::left);
9. cout << "第" << i + 1 << "学期" <<endl << setw(15) << "星期一" << setw(15) << "星期二"
10. << setw(15) << "星期三" << setw(15) << "星期四" << setw(15) << "星期五" << endl ;
11. **for** (**int** j = 0; j < 10; j++)
12. {
13. **for** (**int** k = 0; k < 5; k++)
14. {
15. **int** num = final\_schedule[i][k][j];
16. **if** (num == -1)
17. {
18. cout << setw(15) << "/";
19. }
20. **else**
21. {
22. cout << setw(15) << all\_courses.vertextable.get\_element(num)->GetData().name;
23. }
24. }
25. cout << endl;
26. }
27. cout << endl;
28. }
29. //输出到文件中
30. out << "请修改字体为新宋体，否则可能出现排版错误！" << endl;
31. **for** (**int** i = 0; i < 8; i++)
32. {
33. out << setiosflags(ios::left);
34. out << "第" << i + 1 << "学期" << endl << setw(20) << "星期一" << setw(20) << "星期二"
35. << setw(20) << "星期三" << setw(20) << "星期四" << setw(20) << "星期五" << endl;
36. **for** (**int** j = 0; j < 10; j++)
37. {
38. **for** (**int** k = 0; k < 5; k++)
39. {
40. **int** num = final\_schedule[i][k][j];
41. **if** (num == -1)
42. {
43. out << setw(20) << "/";
44. }
45. **else**
46. {
47. out << setw(20) << all\_courses.vertextable.get\_element(num)->GetData().name;
48. }
49. }
50. out << endl;
51. }
52. out << endl;
53. }
54. out.close();
55. }

**3.4.3输出功能截屏示例**

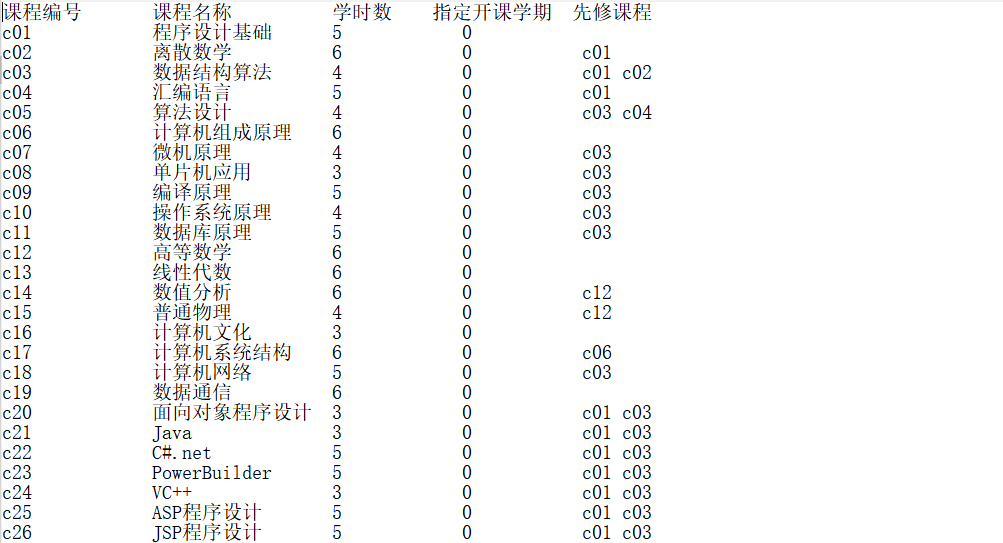


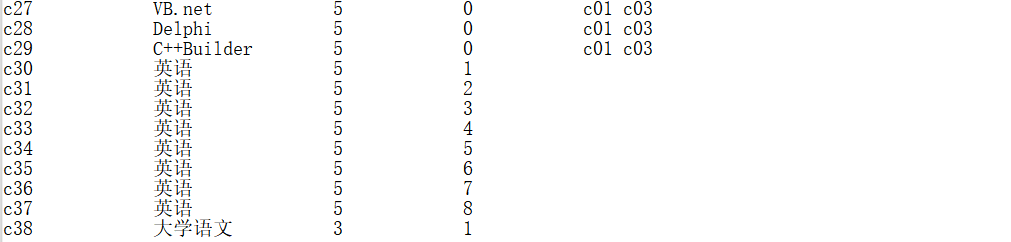


**4 测试**

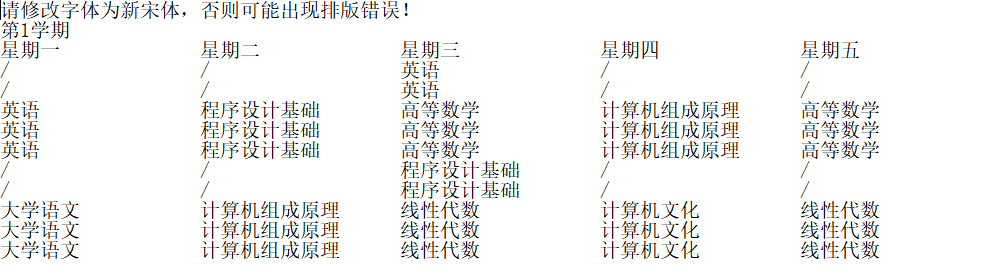
**4.1排课测试**

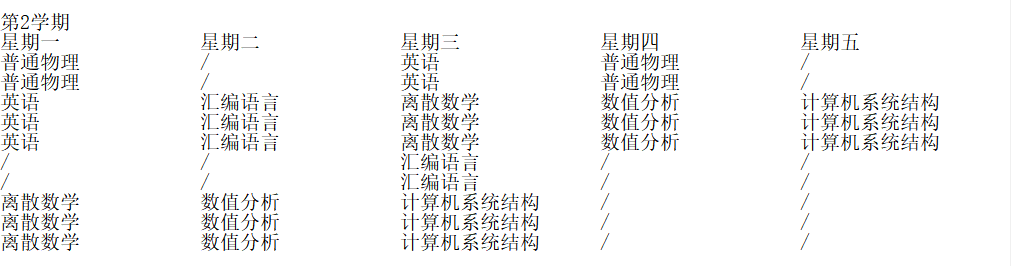
**测试用例：**题目中给出的课程

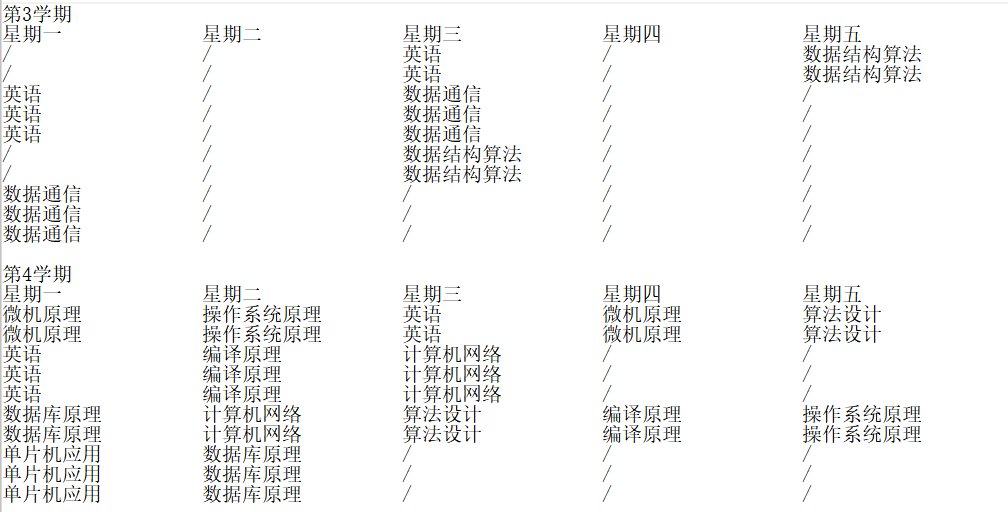


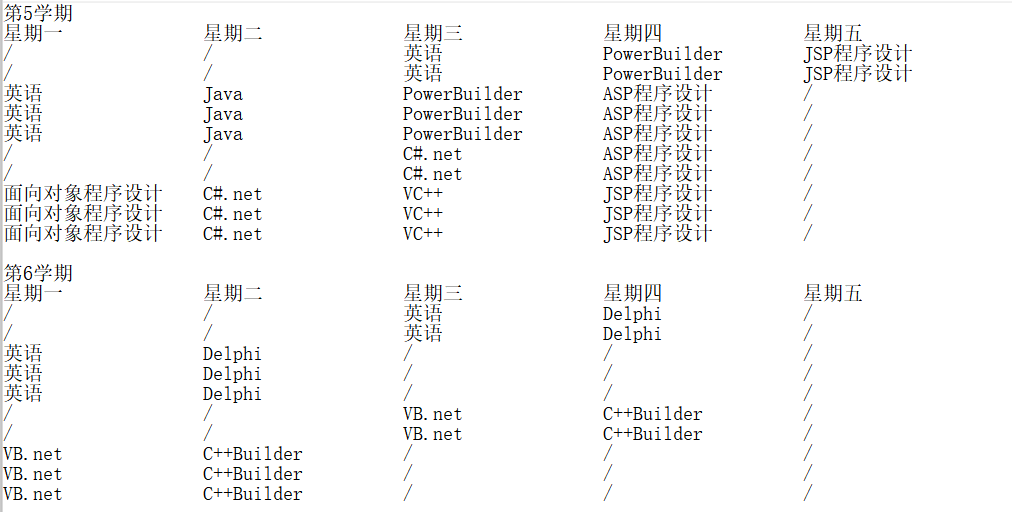


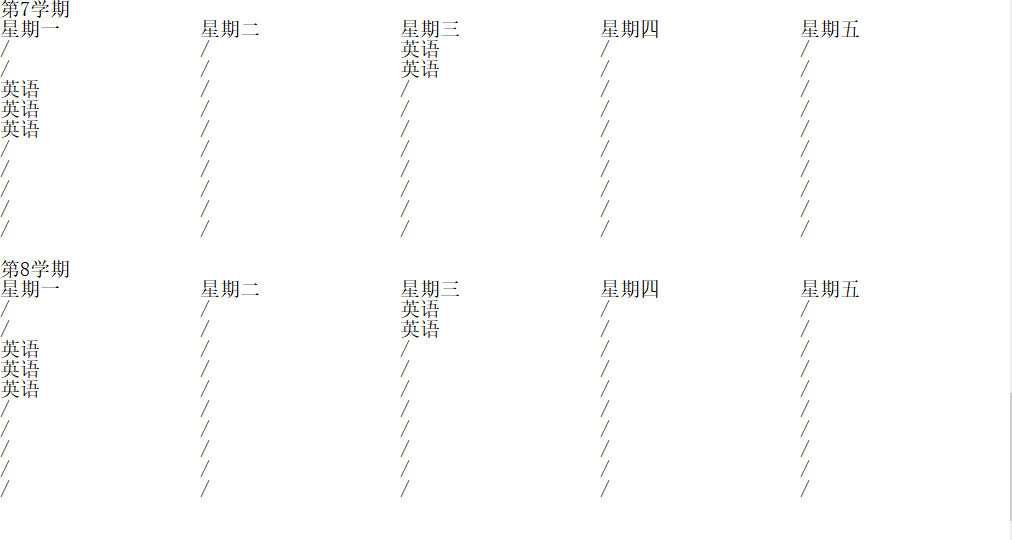
**实验结果：**











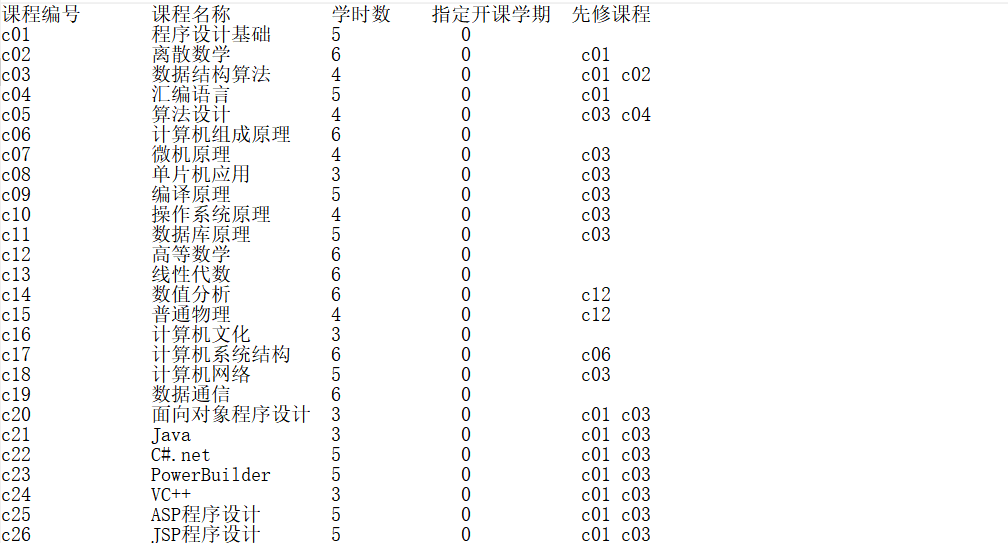
**实验结论：**

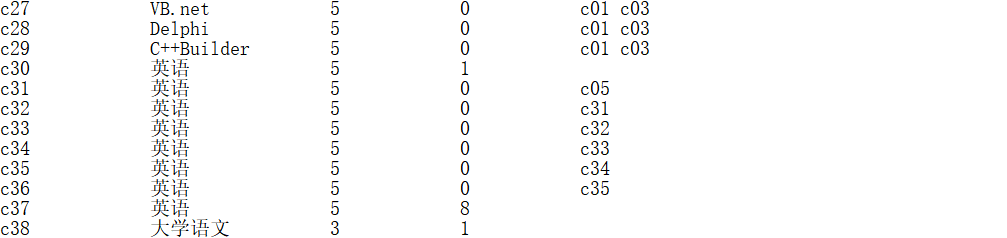
课表每周学时在35个学时以下，不会给学生造成太大负担。主要的课程也基本在前两个学年学习，后两个学年课较少，基本符合现实情况。同时，有三节课的基本安排了三节课一起上，部分课程安排了两节课上，本次排课中没有出现一节课的情况，较为理想。需要两天上的课也基本间隔了两天安排，与实验预期相符。

**4.2报错测试**

**测试用例：**

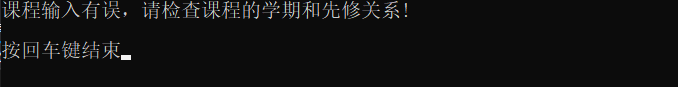
将题目给出的课程稍作修改，使最后几节英语课存在错误的先修关系，无法在8个学期内完成英语课的学习。





**预期结果：**系统提示“课程输入有误，请检查课程的学期和先修关系!”。

**实验结果：**



**实验结论：**排课软件按照预期，识别出了输入数据中错误的先修关系。