排课软件

作 者 姓 名： 王凌

学 号： 1951504

指 导 教 师： 张颖 \_\_

学院、 专业： 软件工程 \_\_

同济大学

Tongji University

目录

[1 分析 2](#_Toc18307)

[1.1项目简介 2](#_Toc29507)

[1.2项目要求 2](#_Toc29418)

[1.3排课要求 2](#_Toc26927)

[2 设计 2](#_Toc27869)

[2.0 部分全局变量的说明 2](#_Toc7117)

[2.1 course类的设计 4](#_Toc22718)

[2.1.1成员变量 4](#_Toc7575)

[2.1.2成员函数——重载输入函数 4](#_Toc4411)

[2.2 output函数的设计 5](#_Toc4152)

[2.3带拓扑排序的DFS深搜函数的设计 6](#_Toc25657)

[2.4主函数的设计 7](#_Toc6461)

[3实现 8](#_Toc15179)

[3.1 course类的实现 8](#_Toc21826)

[3.2 output函数的实现 9](#_Toc31913)

[3.3 带拓扑排序的DFS深搜函数的实现 12](#_Toc1410)

[3.4主函数的实现 15](#_Toc4347)

[4 测试 17](#_Toc21239)

[4.1 功能测试 18](#_Toc7293)

[4.1.1 测试1 18](#_Toc7020)

[4.1.2 测试2 18](#_Toc11846)

# 1 分析

## 1.1项目简介

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项目要求

1.输入数据包括：用户输入每个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，表示由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）。

# 2 设计

## 2.0 部分全局变量的说明

int maxCourseNum[17];//用户要求的每学期选课的门数

vector<course> courses;//存课程的不定长数组

vector<course> topoSorting[17];//记录每个学期选的哪些课程

vector<int> succeed\_course[10007];//记录每一个先修课程的所有后继课程

int indegree[10007];//入度

int hours[17];//记录每学期的学时

int compulsoryCourseNum[17];//记录的是:每一个学期 指定开课学期的课程的数目

bool visted[10007];//记录课程有没有上过

int semAfterCompulsoryCourse[10007];

//记录每门课至少应该在第几学期才能学，如果先修课在第n学期，那么后继课程必须至少在(n+1)学期后学习

int totalHours=0;//总学时

int curTotalHours=0;//当前已选学时

int tureDay[11]={0,1,3,5,2,4};

//优先安排周1，周3，周5，然后周2，周4。连续的课要隔着排课

int tureTime[5]={0,1,3,6,8};

//连上的大课，第一节课是1、2两节小课，第二节是3、4、5三节小课...

//所以枚举的四节大课开始的小课时间分别为1,3,6,8

int classTime[6][5]={{0,0,0,0,0},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3}};

//五个横行 一个星期有五天

//四个纵列 一天有四节可以连上的大课

bool emptyClass[6][5];//记录大课有没有被安排

int table[15][15];//最终的课表

## 2.1 course类的设计

### 2.1.1成员变量

所有成员变量均为公有变量。

string code;//课程编号

string name;//课程名称

string hours\_string;//string类型的学时数

string Semester\_string;//string类型的指定开学学期

int hours;//int类型的学时数

int Semester;//int类型的指定开学学期

vector<string> preCourse;//先修课程编码

### 2.1.2成员函数——重载输入函数

friend ifstream& operator>>(ifstream& input,course& c)//重载输入

输入的时候比较复杂，每一行前四个变量已知，分别为课程编号，课程名称，学时数，指定开课学期。但是先修课程的科目数量不知道，所以要进行特殊判断。

本函数通过空格判断读取的是什么内容，并将其中部分string类型的变量转化为int变量。

## 2.2 output函数的设计

output函数除了输出部分，主要难点在排课部分。

排课要求

1.如一门课程有3节课，则优先安排3节课连续上

2.如3节课连续无法按排，再优先安排两节课连续上

3.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

4.最后再安排单节课上的情况

我的做法如下：

1.定义了三个变量，用以下这三个变量实现了时间的跳转和同一节课相隔两天的安排。

int tureDay[11]={0,1,3,5,2,4}优先安排周1，周3，周5，然后周2，周4。连续的课要隔着排课。

int tureTime[5]={0,1,3,6,8};

//连上的大课，第一节课是1、2两节小课，第二节是3、4、5三节小课...

//所以枚举的四节大课开始的小课时间分别为1,3,6,8

int classTime[6][5]={{0,0,0,0,0},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3},

{0,2,3,2,3}};

//五个横行 一个星期有五天

//四个纵列 一天有四节可以连上的大课

2.用3个while循环，依次处理连上三节课，连上两节课和单独上一节课的情况。

## 2.3带拓扑排序的DFS深搜函数的设计

这是一个带拓扑排序的深搜，有两个形参。形参currentSemester为当前学期，形参count为当前已选课程数目。

深度优先搜索的终止条件为以下四个:

1. if(count==courses.size()) output();如果所有课排完了，就输出
2. if(currentSemester==9) return 0;总共四学年八学期，超过了就说明排错了
3. if(curTotalHours+(8-currentSemester+1)\*50<totalHours) return 0;如果当前总学时加上未来几个学期满课的学时数还是小于所要求的总学时数，说明排错了
4. if(hours[currentSemester]>50) return 0;如果课报满的话，一周一共5天\*10节课， 所以一学期超过50学时就说明排错了

继续搜索的条件为：

1.if(topoSorting[currentSemester].size()<maxCourseNum[currentSemester])还没有选满操作者意愿所选择的课程数目那么继续选课

2.if(courses[i].Semester==0 || courses[i].Semester==currentSemester)这节课没有指定开课学期，或者现在就是指定的开课学期就继续

3.if(indegree[i]==0 && !visted[i] && currentSemester>=semAfterCompulsoryCourse[i]) 这节课入度为0(先修课已经上完了) 并且 之前没有选过 并且 在必须要学的先修课的学期之后

4.if(compulsoryCourseNum[currentSemester]==0) DFS(currentSemester+1,count);这学期没有强制这学期学习的课了，那么直接学下学期的课

标记的过程为：

1. //————以下五步为标记与修改操作，如果选了这门课会发生哪些变化————
2. **for**(**int** j=0;j<succeed\_course[i].size();j++)
3. {
4. indegree[succeed\_course[i][j]]--;//选了课程i之后，课程i的后继课程入度减一
5. //succeed\_course[i][j]意思是先修课程i的后面的第j个后继课程
6. tempLeast[succeed\_course[i][j]]=semAfterCompulsoryCourse[succeed\_course[i][j]];
7. //记录原来的二维不定长变量，方便还原
8. semAfterCompulsoryCourse[succeed\_course[i][j]]=currentSemester+1;
9. //先修课程i的后面的第j个后继课程，这个课程必须在(currentSemester+1)学期或以后才能选
10. }
11. visted[i]=**true**;//标记这门课已经选过
12. curTotalHours+=courses[i].hours;//当前课时数加上所选课程的课时
13. hours[currentSemester]+=courses[i].hours;//当前学期的课时数加上所选课程的课时
14. **if**(courses[i].Semester==currentSemester) compulsoryCourseNum[currentSemester]--;
15. //如果是当前学期所选课指定的开课学期， 那么选了这节课之后，指定这一学期的必修课数量减一
16. topoSorting[currentSemester].push\_back(courses[i]);
17. //这节课被记录在排序课表之中
18. //———————————————————————————————————

回溯的过程为：

1. //————以下五步是回溯，还原未选这门课之前的状态——————————
2. //由于是还原以上步骤，就是上述操作的反操作，不再赘述
3. visted[i]=**false**;
4. curTotalHours-=courses[i].hours;
5. hours[currentSemester]-=courses[i].hours;
6. **if**(courses[i].Semester==currentSemester) compulsoryCourseNum[currentSemester]++;
7. topoSorting[currentSemester].pop\_back();
8. **for**(**int** j=0;j<succeed\_course[i].size();j++)
9. {
10. indegree[succeed\_course[i][j]]++;
11. semAfterCompulsoryCourse[succeed\_course[i][j]]=tempLeast[succeed\_course[i][j]];
12. }
13. //———————————————————————————————————

## 2.4主函数的设计

流程如下；

1. 判断自定义的文件的地址是否正确，如果不正确重新输入input.txt的文件
2. 之后从文件中读取课程，并记录总课程数
3. 在操作框中提示输入用户想要每个学期(共8个学期)选修的课程数，如果课程总数与文件中的总课程数不同则报错退出
4. 如果输入没有问题，那么初始化每个节点（每节课）的入度（先修课）和后继结点（先修课之后能学什么课）。
5. 深搜开始，搜索过程搜到结果直接退出
6. 没搜到结果，输出“当前课表无法编排！”并退出

# 3实现

## 3.1 course类的实现

输入的时候比较复杂，每一行前四个变量已知，分别为课程编号，课程名称，学时数，指定开课学期。但是先修课程的科目数量不知道，所以要进行特殊判断。

本函数通过空格判断读取的是什么内容，并将其中部分string类型的变量转化为int变量。

1. **class** course
2. {
3. **public**:
4. string code;//课程编号
5. string name;//课程名称
6. string hours\_string;//string类型的学时数
7. string Semester\_string;//string类型的指定开学学期
8. **int** hours;//int类型的学时数
9. **int** Semester;//int类型的指定开学学期
10. vector<string> preCourse;//先修课程编码
11. **friend** ifstream& operator>>(ifstream& input,course& c)//重载输入
12. {
13. string strline="";
14. string temp="";//初始化
15. getline(inf,strline);
16. **int** choice=1;//choice决定存哪个变量
17. **for**(**int** i=0;i<strline.length();i++)
18. {
19. **if**(strline[i]!=' ')//通过空格判断读取的是什么
20. {
21. **if**(choice<=4)
22. //前四个分别是课程编号，课程名称，学时数，指定开课学期
23. {
24. **switch**(choice)
25. {
26. **case** 1:
27. c.code+=strline[i];
28. **break**;
29. **case** 2:
30. c.name+=strline[i];
31. **break**;
32. **case** 3:
33. c.hours\_string+=strline[i];
34. **break**;
35. **case** 4:
36. c.Semester\_string+=strline[i];
37. **break**;
38. **default**:
39. **break**;
40. }
41. }
42. **else**//其它的是先修课程
43. {
44. temp+=strline[i];
45. }
46. }
47. **else** **if**(choice<=4) choice++;
48. //如果是前四个变量的话直接更新choice
49. **else** **if**(choice>=5)
50. //如果是读取的是先修课程的话，更新choice的同时，记录并清空temp
51. {
52. c.preCourse.push\_back(temp);
53. temp="";
54. choice++;
55. }
56. }
57. **if**(temp!="") c.preCourse.push\_back(temp);
58. //最后一位没有空格，无法通过空格判断，如果是先修课程的话需要记录
59. //转化char数组再转化为int
60. c.hours=atoi(c.hours\_string.c\_str());
61. c.Semester=atoi(c.Semester\_string.c\_str());
62. **return** input;
64. }
65. };

## 3.2 output函数的实现

我的做法如下：

1.定义了三个变量，用以下这三个变量实现了时间的跳转和同一节课相隔两天的安排。

int tureDay[11]={0,1,3,5,2,4}优先安排周1，周3，周5，然后周2，周4。连续的课要隔着排课。

int tureTime[5]={0,1,3,6,8};

//连上的大课，第一节课是1、2两节小课，第二节是3、4、5三节小课...

//所以枚举的四节大课开始的小课时间分别为1,3,6,8

int classTime[6][5]//五个横行，一个星期有五天；四个纵列，一天有四节可以连上的大课

2.用3个while循环，依次处理连上三节课，连上两节课和单独上一节课的情况。

由于源代码注释较多，若word中缩进部分难以阅读，不妨直接看源代码。

1. **void** output()//输出
2. {
3. **for**(**int** i=1;i<=8;i++)
4. {
5. memset(&table[0][0],0,**sizeof**(table));
6. memset(emptyClass,1,**sizeof**(emptyClass));
7. outf<<endl<<"第"<<i<<"学期"<<endl;
8. **for**(**int** j=0;j<topoSorting[i].size();j++)
9. {
10. /\*
11. 此处为了实现以下目的：
12. 如一门课程有3节课，则优先安排3节课连续上
13. 如3节课连续无法按排，再优先安排两节课连续上
14. 如果一门课程需要安排上两天，为教学效果较好，
15. 最好不安排在相邻的两天，比如优先安排相隔2天上课
16. \*/
17. **while**(topoSorting[i][j].hours>=3)
18. {
19. **bool** f\_find=0;
20. **for**(**int** Day=1;Day<=5;Day++)
21. **for**(**int** Time=2;Time<=4;Time+=2)
22. {
23. //[tureDay[d]][Time]意思是第tureDay[d]天的第t节课，这节课可以连上classTime[tureDay[d]][Time]节小课
24. **if**(emptyClass[tureDay[Day]][Time] && topoSorting[i][j].hours>=3)
25. //如果[tureDay[d]][Time]这节大课的时间没有被安排 并且 且 这节课的剩余学时数>=3
26. {
27. f\_find=1;//找到了尚未安排的大课
28. emptyClass[tureDay[Day]][Time]=0;
29. //安排这节课
30. topoSorting[i][j].hours-=3;
31. //这节课的剩余学时数减3
32. **for**(**int** k=0;k<classTime[tureDay[Day]][Time];k++) table[tureDay[Day]][tureTime[Time]+k]=j+1;//把这节课录入课表
33. //j+1是因为只有要用0来判断，如果标号从1开始就不会出错
34. **break**;
35. }
36. }
37. **if**(f\_find==0) **break**;
38. }
39. **while**(topoSorting[i][j].hours>=2)
40. {
41. **bool** f\_find=0;
42. **for**(**int** Day=1;Day<=5;Day++)
43. **for**(**int** Time=1;Time<=3;Time+=2)
44. {
45. **if**(emptyClass[tureDay[Day]][Time] && topoSorting[i][j].hours>=2)
46. //如果[tureDay[d]][Time]这节大课的时间没有被安排 并且 这节课的剩余学时数>=2
47. {
48. f\_find=1;//找到了尚未安排的大课
49. emptyClass[tureDay[Day]][Time]=0;//安排这节课
50. topoSorting[i][j].hours-=2;//这节课的剩余学时数减2
51. **for**(**int** k=0;k<classTime[tureDay[Day]][Time];k++) table[tureDay[Day]][tureTime[Time]+k]=j+1;//把这节课录入课表
52. **break**;
53. }
54. }
55. **if**(f\_find==0) **break**;
56. }
57. }
58. //最后再安排单节课上的情况
59. **for**(**int** j=0;j<topoSorting[i].size();j++)
60. **while**(topoSorting[i][j].hours>=1)
61. //当选的几门的学时还没有上完的时候（剩余学时大于零）
62. {
63. **for**(**int** time=0;time<=49;time++)
64. //枚举这一周所有可以上课的时间，一节课一节课地枚举(即一个学时一个学时地枚举)
65. **if**(table[(time/10)+1][time%10+1]==0 && topoSorting[i][j].hours>=1)
66. //如果这周第[(k/10)+1]天的第[k%10+1]节课，还没有安排课
67. {
68. table[(time/10)+1][time%10+1]=j+1;//那么直接把课安排在此处
69. topoSorting[i][j].hours--;//这节课的剩余学时减一
70. }
71. }
73. **for**(**int** p=1;p<=10;p++)
74. {
75. **for**(**int** j=1;j<=5;j++)
76. **if**(table[j][p]==0) outf<<setw(5)<<setiosflags(ios::left)<<"无  ";
77. **else** outf<<setw(5)<<setiosflags(ios::left)<<topoSorting[i][table[j][p]-1].name<<"  ";
78. outf<<endl;
79. }
80. outf<<endl;
81. }
82. cout<<endl<<"课表保存！"<<endl;
83. inf.close();
84. outf.close();
85. exit(0);
86. }

## 3.3带拓扑排序的DFS深搜函数的实现

这是一个带拓扑排序的深搜，有两个形参。形参currentSemester为当前学期，形参count为当前已选课程数目。

深度优先搜索的终止条件为以下四个:

1.如果所有课排完了，就输出

2.总共四学年八学期，超过了就说明排错了

3.如果当前总学时加上未来几个学期满课的学时数还是小于所要求的总学时数，说明排错了

4.如果课报满的话，一周一共5天\*10节课， 所以一学期超过50学时就说明排错了

继续搜索的条件为：

1.还没有选满操作者意愿所选择的课程数目那么继续选课

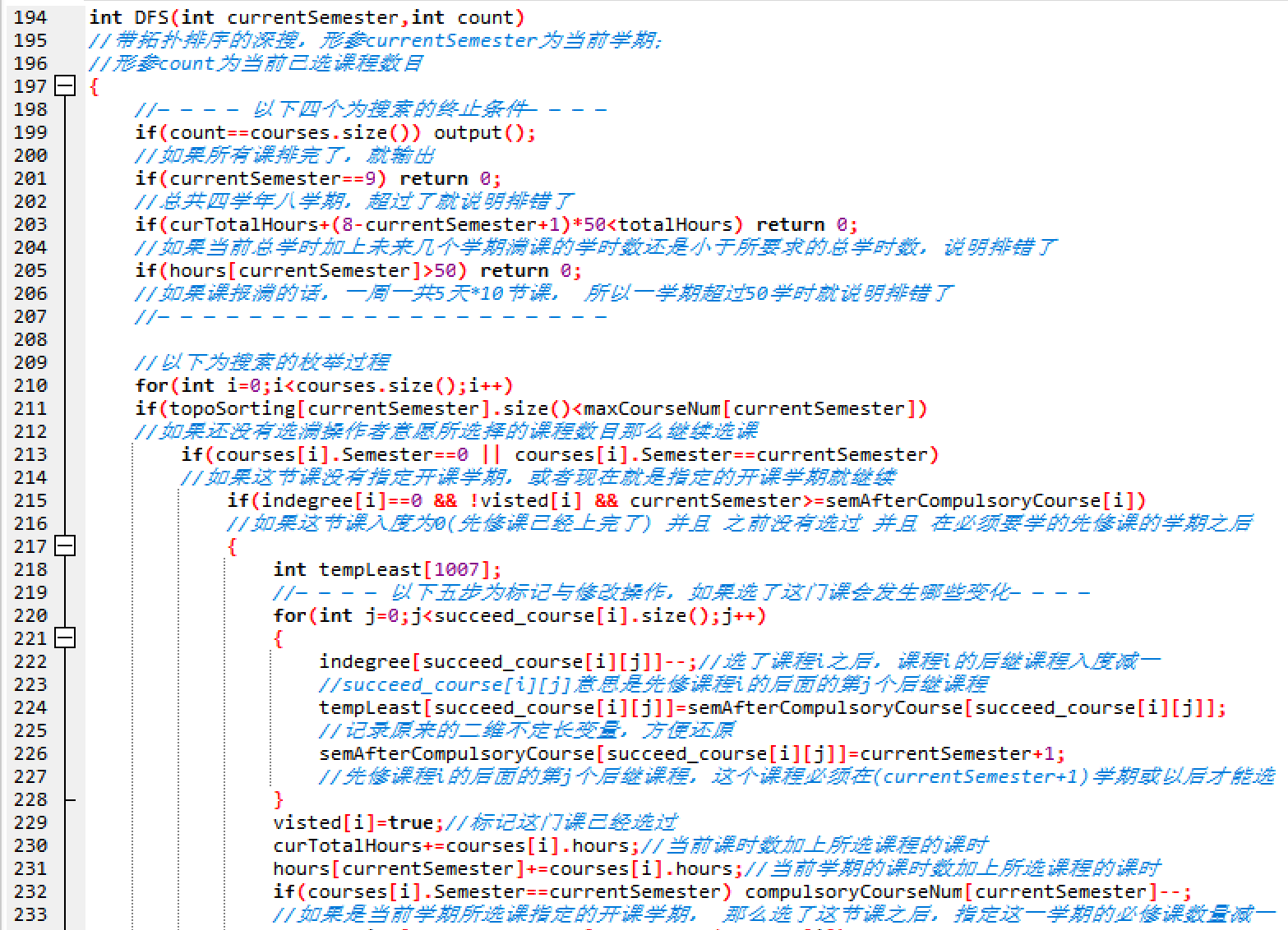
2.这节课没有指定开课学期，或者现在就是指定的开课学期就继续

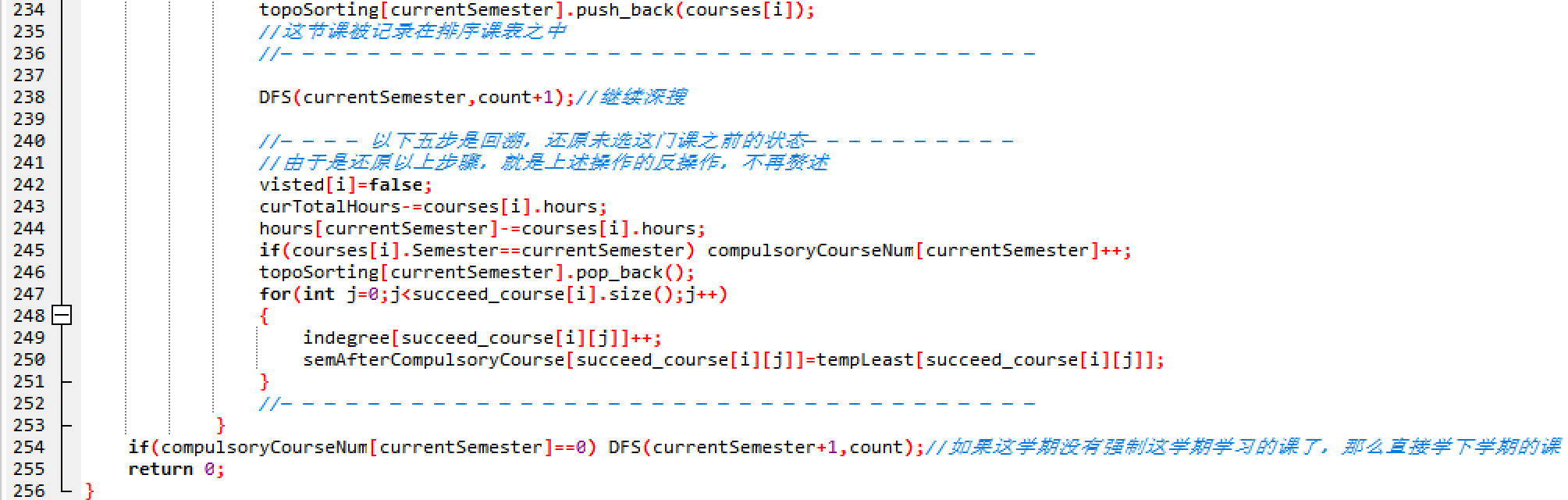
3.这节课入度为0(先修课已经上完了) 并且 之前没有选过 并且 在必须要学的先修课的学期之后

4.这学期没有强制这学期学习的课了，那么直接学下学期的课

由于源代码注释较多，若word中缩进部分难以阅读，不妨直接看源代码。

1. **int** DFS(**int** currentSemester,**int** count)
2. //带拓扑排序的深搜，形参currentSemester为当前学期；
3. //形参count为当前已选课程数目
4. {
5. //————以下四个为搜索的终止条件————
6. **if**(count==courses.size()) output();
7. //如果所有课排完了，就输出
8. **if**(currentSemester==9) **return** 0;
9. //总共四学年八学期，超过了就说明排错了
10. **if**(curTotalHours+(8-currentSemester+1)\*50<totalHours) **return** 0;
11. //如果当前总学时加上未来几个学期满课的学时数还是小于所要求的总学时数，说明排错了
12. **if**(hours[currentSemester]>50) **return** 0;
13. //如果课报满的话，一周一共5天\*10节课， 所以一学期超过50学时就说明排错了
14. //————————————————————
16. //以下为搜索的枚举过程
17. **for**(**int** i=0;i<courses.size();i++)
18. **if**(topoSorting[currentSemester].size()<maxCourseNum[currentSemester])
19. //如果还没有选满操作者意愿所选择的课程数目那么继续选课
20. **if**(courses[i].Semester==0 || courses[i].Semester==currentSemester)
21. //如果这节课没有指定开课学期，或者现在就是指定的开课学期就继续
22. **if**(indegree[i]==0 && !visted[i] && currentSemester>=semAfterCompulsoryCourse[i])
23. //如果这节课入度为0(先修课已经上完了) 并且 之前没有选过 并且 在必须要学的先修课的学期之后
24. {
25. **int** tempLeast[1007];
26. //————以下五步为标记与修改操作，如果选了这门课会发生哪些变化————
27. **for**(**int** j=0;j<succeed\_course[i].size();j++)
28. {
29. indegree[succeed\_course[i][j]]--;//选了课程i之后，课程i的后继课程入度减一
30. //succeed\_course[i][j]意思是先修课程i的后面的第j个后继课程
31. tempLeast[succeed\_course[i][j]]=semAfterCompulsoryCourse[succeed\_course[i][j]];
32. //记录原来的二维不定长变量，方便还原
33. semAfterCompulsoryCourse[succeed\_course[i][j]]=currentSemester+1;
34. //先修课程i的后面的第j个后继课程，这个课程必须在(currentSemester+1)学期或以后才能选
35. }
36. visted[i]=**true**;//标记这门课已经选过
37. curTotalHours+=courses[i].hours;//当前课时数加上所选课程的课时
38. hours[currentSemester]+=courses[i].hours;//当前学期的课时数加上所选课程的课时
39. **if**(courses[i].Semester==currentSemester) compulsoryCourseNum[currentSemester]--;
40. //如果是当前学期所选课指定的开课学期， 那么选了这节课之后，指定这一学期的必修课数量减一
41. topoSorting[currentSemester].push\_back(courses[i]);
42. //这节课被记录在排序课表之中
43. //———————————————————————————————————
45. DFS(currentSemester,count+1);//继续深搜
47. //————以下五步是回溯，还原未选这门课之前的状态——————————
48. //由于是还原以上步骤，就是上述操作的反操作，不再赘述
49. visted[i]=**false**;
50. curTotalHours-=courses[i].hours;
51. hours[currentSemester]-=courses[i].hours;
52. **if**(courses[i].Semester==currentSemester) compulsoryCourseNum[currentSemester]++;
53. topoSorting[currentSemester].pop\_back();
54. **for**(**int** j=0;j<succeed\_course[i].size();j++)
55. {
56. indegree[succeed\_course[i][j]]++;
57. semAfterCompulsoryCourse[succeed\_course[i][j]]=tempLeast[succeed\_course[i][j]];
58. }
59. //———————————————————————————————————
60. }
61. **if**(compulsoryCourseNum[currentSemester]==0) DFS(currentSemester+1,count);//如果这学期没有强制这学期学习的课了，那么直接学下学期的课
62. **return** 0;
63. }





## 3.4主函数的实现

流程如下；

1. 判断自定义的文件的地址是否正确，如果不正确重新输入input.txt的文件
2. 之后从文件中读取课程，并记录总课程数
3. 在操作框中提示输入用户想要每个学期(共8个学期)选修的课程数，如果课程总数与文件中的总课程数不同则报错退出
4. 如果输入没有问题，那么初始化每个节点（每节课）的入度（先修课）和后继结点（先修课之后能学什么课）。
5. 深搜开始，搜索过程搜到结果直接退出
6. 没搜到结果，输出“当前课表无法编排！”并退出
7. **int** main()
8. {
9. string temp;
10. cout<<"使用默认文件地址吗?(Y/N)";
11. cin>>temp;
12. **if**(temp=="N")
13. {
14. cout<<"输入input.txt文件的地址: " ;
15. cin>>INPUT\_PATH;
16. inf.open(INPUT\_PATH);
17. **if**(!inf)
18. {
19. cerr<<"\n地址错误，打不开文件！"<<endl;
20. **return** 0;
21. }
22. }
23. **int** totalCourseNum=0;
24. **while**(!inf.eof())
25. {
26. course tempCourse;
27. inf>>tempCourse;
28. totalCourseNum++;
29. courses.push\_back(tempCourse);
30. totalHours+=tempCourse.hours;
31. compulsoryCourseNum[tempCourse.Semester]++;
32. //指定学期必修课程 所在的学期位置加一
33. }
34. cout<<"输入你想要每个学期(共8个学期)选修的课程数！注意：课程数总和为"<<totalCourseNum<<endl;
35. **int** tempsum=0;
36. **for**(**int** i=1;i<=8;i++)
37. {
38. cin>>maxCourseNum[i];
39. tempsum+=maxCourseNum[i];
40. }
41. **if**(tempsum!=totalCourseNum)
42. {
43. cout<<"与课程总数不相等！"<<endl;
44. **return** 0;
45. }
46. **for**(**int** i=0;i<courses.size();i++)
47. {
48. indegree[i]=courses[i].preCourse.size();
49. //入度定义为这门课先修课的门数
50. **for**(**int** j=0;j<courses.size();j++)
51. **for**(**int** k=0;k<courses[j].preCourse.size();k++)
52. **if**(courses[j].preCourse[k]==courses[i].code)
53. //如果课程i是课程j的先修课程
54. {
55. succeed\_course[i].push\_back(j);
56. //课程i的后继课程加入课程j
57. **break**;
58. }
59. }
60. DFS(1,0);//深搜，搜出结果直接输出
61. //如果搜不出结果，说明这种课表无法编排无解
62. cerr<<"\n当前课表无法编排！"<<endl;
63. **return** 0;
64. }

# 4 测试

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

c221 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

## 4.1 功能测试

**测试数据：**

Y

5 5 5 5 5 5 4 4

**结果：**

第1学期

程序设计基础 无 无 无 无

程序设计基础 无 无 无 无

程序设计基础 大学语文 计算机组成原理 无 高等数学

程序设计基础 大学语文 计算机组成原理 无 高等数学

程序设计基础 大学语文 计算机组成原理 无 高等数学

英语 无 无 无 无

英语 无 无 无 无

计算机组成原理 无 高等数学 无 英语

计算机组成原理 无 高等数学 无 英语

计算机组成原理 无 高等数学 无 英语

第2学期

汇编语言 无 无 无 无

汇编语言 无 无 无 无

离散数学 数值分析 离散数学 无 线性代数

离散数学 数值分析 离散数学 无 线性代数

离散数学 数值分析 离散数学 无 线性代数

英语 无 无 无 无

英语 无 无 无 无

汇编语言 英语 线性代数 无 数值分析

汇编语言 英语 线性代数 无 数值分析

汇编语言 英语 线性代数 无 数值分析

第3学期

英语 无 无 无 无

英语 无 无 无 无

数据结构算法 无 计算机文化 无 计算机系统结构

数据结构算法 无 计算机文化 无 计算机系统结构

数据结构算法 无 计算机文化 无 计算机系统结构

数据结构算法 无 无 无 无

普通物理 无 无 无 无

普通物理 无 计算机系统结构 无 英语

普通物理 无 计算机系统结构 无 英语

普通物理 无 计算机系统结构 无 英语

第4学期

编译原理 算法设计 无 无 无

编译原理 微机原理 无 无 无

算法设计 无 单片机应用 无 英语

算法设计 无 单片机应用 无 英语

算法设计 无 单片机应用 无 英语

英语 无 无 无 无

英语 无 无 无 无

微机原理 无 编译原理 无 无

微机原理 无 编译原理 无 无

微机原理 无 编译原理 无 无

第5学期

数据库原理 操作系统原理 英语 无 无

数据库原理 无 英语 无 无

操作系统原理 无 计算机网络 无 数据通信

操作系统原理 无 计算机网络 无 数据通信

操作系统原理 无 计算机网络 无 数据通信

计算机网络 无 无 无 无

计算机网络 无 无 无 无

数据库原理 无 数据通信 无 英语

数据库原理 无 数据通信 无 英语

数据库原理 无 数据通信 无 英语

第6学期

C#.net 无 英语 无 无

C#.net 无 英语 无 无

面向对象程序设计 无 C#.net 无 英语

面向对象程序设计 无 C#.net 无 英语

面向对象程序设计 无 C#.net 无 英语

PowerBuilder 无 无 无 无

PowerBuilder 无 无 无 无

Java 无 PowerBuilder 无 无

Java 无 PowerBuilder 无 无

Java 无 PowerBuilder 无 无

第7学期

ASP程序设计 无 英语 无 无

ASP程序设计 无 英语 无 无

VC++ 无 JSP程序设计 无 无

VC++ 无 JSP程序设计 无 无

VC++ 无 JSP程序设计 无 无

JSP程序设计 无 无 无 无

JSP程序设计 无 无 无 无

ASP程序设计 无 英语 无 无

ASP程序设计 无 英语 无 无

ASP程序设计 无 英语 无 无

第8学期

VB.net 无 C++Builder 无 无

VB.net 无 C++Builder 无 无

VB.net 无 C++Builder 无 无

VB.net 无 C++Builder 无 无

VB.net 无 C++Builder 无 无

Delphi 无 英语 无 无

Delphi 无 英语 无 无

Delphi 无 英语 无 无

Delphi 无 英语 无 无

Delphi 无 英语 无 无

## 4.2 错误判定

### 4.2.1 输入课程数之和不等于总数

