项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名： 曹峰源

学 号： 1951328

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 3](#_Toc59809555)

[1.1 背景分析 3](#_Toc59809556)

[1.2 功能分析 3](#_Toc59809557)

[2. 设计 3](#_Toc59809558)

[2.1 数据结构设计 3](#_Toc59809559)

[2.2 类结构设计 3](#_Toc59809560)

[2.3 类成员函数 3](#_Toc59809561)

[3. 功能实现 5](#_Toc59809562)

[3.1 计算课程总数功能实现 5](#_Toc59809563)

[3.1.1 流程图 5](#_Toc59809564)

[3.1.2 代码实现 5](#_Toc59809565)

[3.2 字符串切割函数功能实现 6](#_Toc59809566)

[3.3 读取课程具体信息功能实现 6](#_Toc59809567)

[3.3.1 流程图 6](#_Toc59809568)

[3.3.2 代码实现 7](#_Toc59809569)

[3.4 拓扑排序课程分类功能实现 8](#_Toc59809570)

[3.4.1 流程图 8](#_Toc59809571)

[3.4.2 代码实现 9](#_Toc59809572)

[3.5 具体课表安排功能实现 10](#_Toc59809573)

[3.5.1 流程图 10](#_Toc59809574)

[3.5.2 代码实现 11](#_Toc59809575)

[3.5.3 说明 16](#_Toc59809576)

[3.6 输出功能实现 16](#_Toc59809577)

[4. 输入输出测试 17](#_Toc59809578)

[5. 改进方向 20](#_Toc59809579)

[5.1 文件输入错误提示的优化 20](#_Toc59809580)

[5.2 排课算法的优化 20](#_Toc59809581)

1. 分析
   1. 背景分析

当今教学系统离不开计算机知识的支持。除去之前的考试报名系统，选课与排课也是学生生活的重要部分，可以使用计算机系统来实现快速排课。

* 1. 功能分析

对于数字化的排课软件，首先能够实现课程的文件输入以及输出，用txt文件来保存。其次应能够对每学期要上的课程进行分类，并将其排到课表中，并且满足一些先修课程先后次序的特殊要求。

1. 设计
   1. 数据结构设计

因为众多课程所反映的关系复杂，所以使用图结构来存储课程信息数据。在实际排课的时候，因为要考虑到先修课程的先后顺序，所以采取拓扑排序来将课程分到各个学期当中。

* 1. 类结构设计

对于课程构成的图，因为要记录的课程数据很多，有编号，名字等等，所以课程的信息使用点类来存储。在图中要记录所有的点，于是使用vector<node>来记录所有点的信息，并使用逆邻接矩阵来记录边的关系。最后，使用三维数组来记录具体的课表。

* 1. 类成员函数

结点类 node

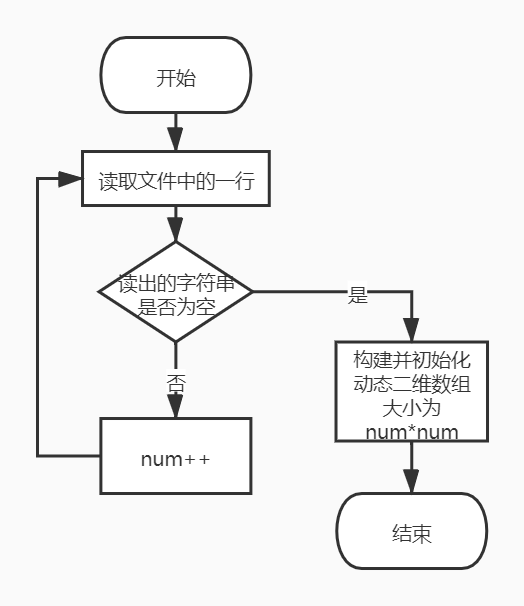
1. **class** node//存储课程信息的点类
2. {
3. **friend** **class** graph;//声明为图的友元类
4. **private**:
5. string index;//课程的编号
6. string name;//课程的名称
7. **int** time = 0;//课程的课时数
8. **int** term = 0;//课程应在的学期
9. **int** in = 0;//该课程的入度
10. **int** take = 0;//该课程是否已经被安排，用于后续拓扑排序
11. **public**:
12. node() {}//构造函数
13. ~node() {}//析构函数
14. };

图类 graph

1. **class** graph
2. {
3. **private**:
4. vector<node> nodetable;//课程信息的存储容器
5. **int**\*\* arrey;//边数组存储入边情况
6. **int** num;//输入的文件中课程数
7. string classtable[8][5][10];//存储具体课表
8. **int** classjudge[8][5][4];//具体课表时间是否被占用，用于后续判断
9. vector<vector<node>> t;//存储每学期的课程
10. **public**:
11. graph()
12. {
13. num = 0;
14. **for** (**int** i = 0; i < 8; i++)//将三维数组classjudge初始化
15. {
16. **for** (**int** j = 0; j < 5; j++)
17. {
18. **for** (**int** k = 0; k < 4; k++)
19. {
20. classjudge[i][j][k] = 0;
21. }
22. }
23. }
24. **for** (**int** i = 0; i < 8; i++)//对课表进行初始化
25. {
26. **for** (**int** j = 0; j < 5; j++)
27. {
28. **for** (**int** k = 0; k < 10; k++)
29. {
30. classtable[i][j][k] = "0";
31. }
32. }
33. }
34. t.resize(8);//将容器t的大小设置为8
35. }
36. ~graph()
37. {
38. **for** (**int** i = 0; i < num; i++)
39. {
40. **delete**[]arrey[i];
41. }
42. **delete** arrey;
43. }
44. **void** getnum(ifstream& OpenFile);//得到文件中有多少课程的函数
45. **void** get\_data(ifstream& OpenFile);//将文件中课程信息提取出来
46. **void** sort();//拓扑排序
47. **void** schedule();//排课函数
48. **void** three(**int** i,**int** j,**int** q);//在课表中安排三个学时的课（i代表学期，j代表课程编号，q代表从哪一天开始寻找空课时）
49. **void** two(**int** i,**int** j,**int** q);
50. **void** out();//文件输出
51. };
52. 功能实现
    1. 计算课程总数功能实现

要进行排课，首先要能知道输入文件中课程的总数，这个函数就是用于对文件中课程的统计，得到课程数num后再对边的动态二维数组进行初始化操作。

* + 1. 流程图



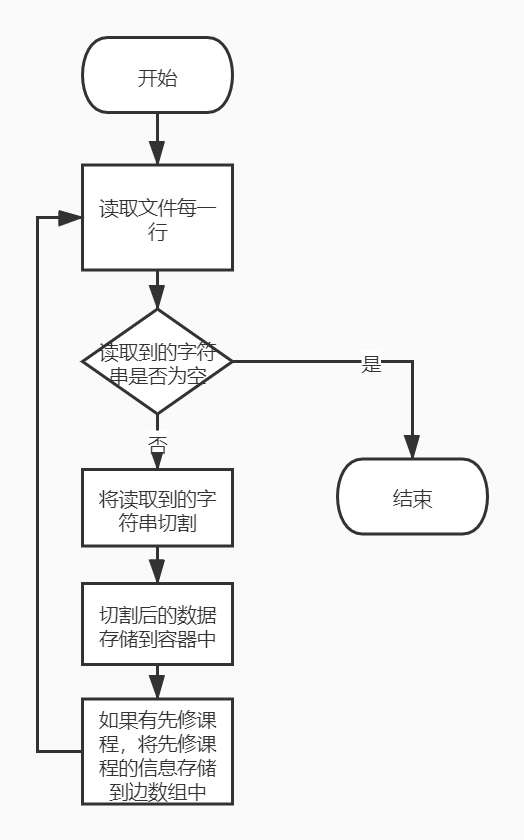
* + 1. 代码实现

1. **void** graph::getnum(ifstream& OpenFile)//得到文件中数据的行数并且初始化边数组
2. {
3. string line;
4. **while** (getline(OpenFile, line))//读取文件中的每一行
5. {
6. **if** (line == "") num--;//如果读出的line字符串为空，则跳出循环
7. num++;//每读一次，num++
8. }
9. arrey = **new** **int**\* [num];
10. **for** (**int** i = 0; i < num; i++)//创建边的动态二维数组
11. {
12. arrey[i] = **new** **int**[num];
13. }
14. **if** (!arrey) cout << "数组开辟失败";
15. **else** {
16. **for** (**int** i = 0; i < num; i++)//初始化数组，将每个边先赋值为0
17. {
18. **for** (**int** j = 0; j < num; j++)
19. {
20. arrey[i][j] = 0;
21. }
22. }
23. }
24. OpenFile.close();
25. }
    1. 字符串切割函数功能实现

在读取文件时，一整行进行阅读，这就需要对读到的一整行数据进行切割得到具体的数据，所以使用了String\_Split()函数

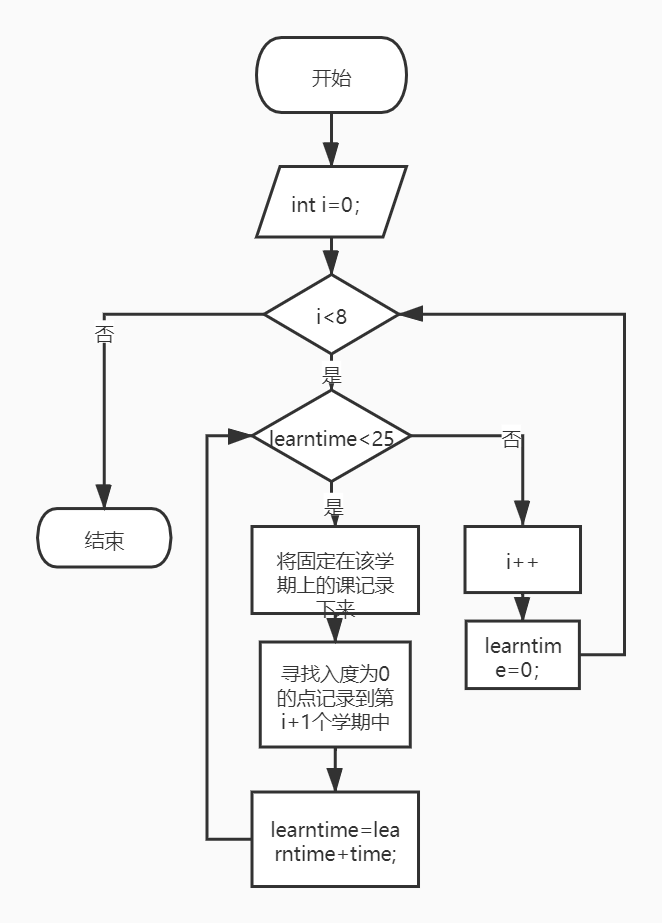
代码实现：

1. **void** String\_Split(string s, string delim, vector<string>& ans)//用于将从文件中读出的string类分割存到vector中
2. {
3. string::size\_type pos\_1, pos\_2 = 0;
4. **while** (pos\_2 != s.npos) {
5. pos\_1 = s.find\_first\_not\_of(delim, pos\_2);
6. **if** (pos\_1 == s.npos) **break**;
7. pos\_2 = s.find\_first\_of(delim, pos\_1);
8. ans.push\_back(s.substr(pos\_1, pos\_2 - pos\_1));
9. }
10. }
    1. 读取课程具体信息功能实现
       1. 流程图



* + 1. 代码实现

1. **void** graph::get\_data(ifstream& OpenFile)//得到文件中的数据
2. {
3. string line;
4. string index, name;
5. **int** i = 0;
6. **while** (getline(OpenFile, line))//对文件中数据逐行提取
7. {
8. **if** (line == "") **break**;//如果得到空字符串跳出循环
9. node data;//新建一个node类
10. vector<string> ans;//建立一个容器ans用于保存切割后的字符串
11. String\_Split(line, " \t", ans);//将得到的字符串切割，每个数据保存到ans中
12. data.index = ans[0];//对新建node类中的每一个数据赋值
13. data.name = ans[1];
14. data.time = stod(ans[2]);
15. data.term = stod(ans[3]);
16. **int** j = 4;
17. **while** (j < ans.size())//如果存在先修课程，在容器中找到该先修课程，将它们边的情况保存到边数组中
18. {
19. **for** (**int** k = 0; k < nodetable.size(); k++)
20. {
21. **if** (nodetable[k].index == ans[j])
22. {
23. arrey[i][k] = 1;
24. }
25. }
26. j++;
27. data.in++;//该点中的入度记录加1
28. }
29. nodetable.push\_back(data);
30. i++;
31. }
32. }
    1. 拓扑排序课程分类功能实现
       1. 流程图



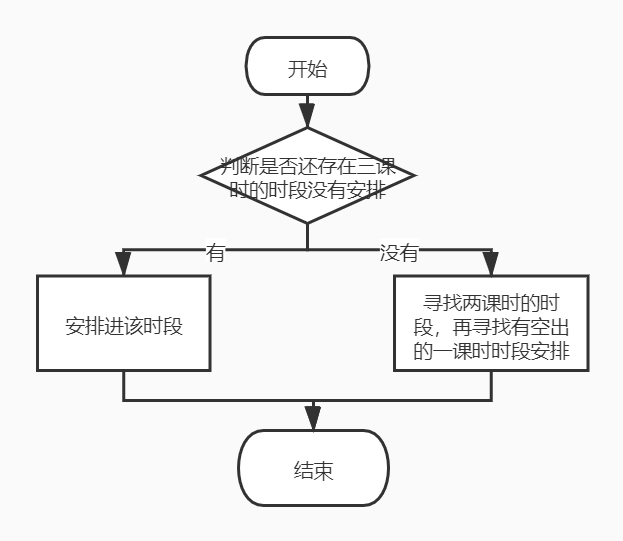
* + 1. 代码实现

1. **void** graph::sort()
2. {
3. **int** learntime = 0;//定义learntime用于计算每学期的学时数
4. **for** (**int** i = 0; i < 8; i++)
5. {
6. vector<**int**> cur;
7. **while** (learntime < 25)//规定每学期的学时数不超过25
8. {
9. **int** j = 0;
10. **for** (j = 0; j < num; j++)
11. {
12. **if** (nodetable[j].term == i + 1 && nodetable[j].take == 0)//如果图中有特定学期的课程将它提取出来
13. {
14. node temp;
15. temp.index = nodetable[j].index;
16. temp.name = nodetable[j].name;
17. temp.time = nodetable[j].time;
18. t[i].push\_back(temp);
19. learntime = learntime + nodetable[j].time;
20. nodetable[j].take = 1;
21. }
22. }
23. j = 0;
24. **for** (j = 0; j < num; j++)//接着寻找图中入度为0的点将其取出且没有规定学期的课程
25. {
26. **if** (nodetable[j].in == 0)
27. {
28. **if** (nodetable[j].term == 0 && nodetable[j].take == 0)
29. {
30. node temp;
31. temp.index = nodetable[j].index;
32. temp.name = nodetable[j].name;
33. temp.time = nodetable[j].time;
34. t[i].push\_back(temp);
35. learntime = learntime + nodetable[j].time;
36. nodetable[j].take = 1;
37. cur.push\_back(j);//用容器cur将这学期的课程存放
38. **break**;
39. }
40. **else** **continue**;
41. }
42. }
43. **if** (j == num) **break**;
44. }
45. **for** (**int** k = 0; k < cur.size(); k++)//遍历cur的数据，将以cur中元素为先修的课程入度减1
46. {
47. **for** (**int** p = 0; p < num; p++)
48. {
49. **if** (arrey[p][cur[k]] == 1)
50. {
51. nodetable[p].in--;
52. }
53. }
54. }
55. learntime = 0;
56. }
57. }
    1. 具体课表安排功能实现

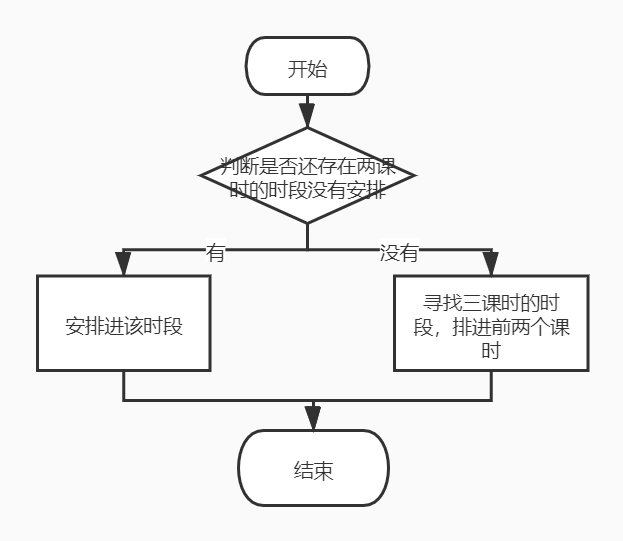
对于具体每学期的课表实现，使用了两个函数，一个three()函数，一个two()函数，因为考虑到课程的学时数都为3、4、5、6，这样使用three()函数对三个学时进行排课，two()函数对两个学时进行排课，这样经过这两个函数的组合之后就可以实现多种课时的排课了。

* + 1. 流程图

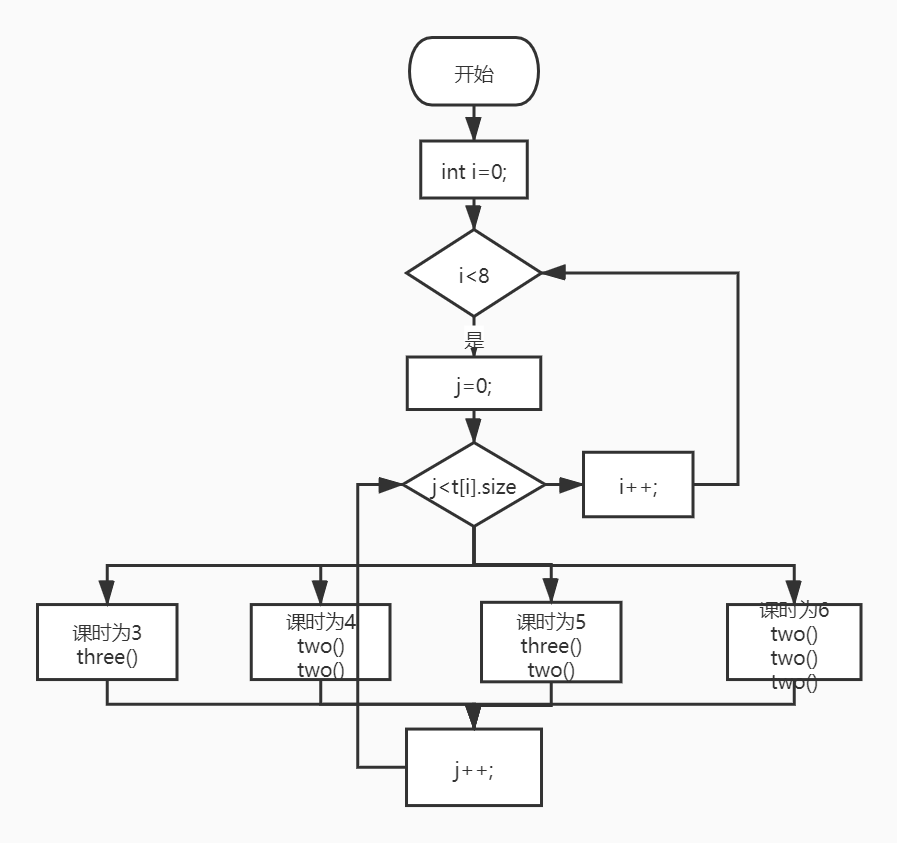
three()函数的流程图



two()函数的流程图



总体排课的流程图



* + 1. 代码实现

three()函数的代码实现

1. **void** graph::three(**int** i,**int** j,**int** q)
2. {
3. **int** count = 0;
4. **for** (**int** p=0;p<5;p++)
5. {
6. **if** (classjudge[i][p][1]==0||classjudge[i][p][3]==0)
7. {
8. count = 1;
9. }
10. }
11. **int** k = q;
12. **if** (count == 1)//如果三个课时仍然存在的情况
13. {
14. **while**(1)
15. {
16. **if** (classjudge[i][k][1] == 0 || classjudge[i][k][3] == 0)
17. {
18. **if** (classjudge[i][k][1] == 0)
19. {
20. classjudge[i][k][1] = 3;
21. classtable[i][k][2] = t[i][j].name;
22. classtable[i][k][3] = t[i][j].name;
23. classtable[i][k][4] = t[i][j].name;
24. }
25. **else**
26. {
27. classjudge[i][k][3] = 3;
28. classtable[i][k][7] = t[i][j].name;
29. classtable[i][k][8] = t[i][j].name;
30. classtable[i][k][9] = t[i][j].name;
31. }
32. **break**;
33. }
34. k++;
35. **if** (k >= 5)k = k - 5;
36. }
37. }
38. **else**//没有三节课时的安排，放到两节课时中
39. {
40. **while**(1)
41. {
42. **if** (classjudge[i][k][0] == 0 || classjudge[i][k][2] == 0)
43. {
44. **if** (classjudge[i][k][2] == 0)//优先考虑下午的课，使得课表相对分散
45. {
46. classjudge[i][k][2] = 2;
47. classtable[i][k][5] = t[i][j].name;
48. classtable[i][k][6] = t[i][j].name;
49. }
50. **else**
51. {
52. classjudge[i][k][0] = 2;
53. classtable[i][k][0] = t[i][j].name;
54. classtable[i][k][1] = t[i][j].name;
55. }
56. **break**;
57. }
58. k++;
59. **if** (k >= 5)k = k - 5;
60. }
61. k = q;
62. **while**(1)//还需要再安排一节课时
63. {
64. **if** (classjudge[i][k][0] == 0 || classjudge[i][k][2] == 0 || classjudge[i][k][0] == 1 || classjudge[i][k][2] == 1)
65. {
66. **if** (classjudge[i][k][0] == 0)
67. {
68. classjudge[i][k][0] = 1;
69. classtable[i][k][0] = t[i][j].name;
70. }
71. **else** **if** (classjudge[i][k][0] == 1)
72. {
73. classjudge[i][k][0] = 2;
74. classtable[i][k][1] = t[i][j].name;
75. }
76. **else** **if** (classjudge[i][k][2] == 0)
77. {
78. classjudge[i][k][2] = 1;
79. classtable[i][k][5] = t[i][j].name;
80. }
81. **else**
82. {
83. classjudge[i][k][2] = 2;
84. classtable[i][k][6] = t[i][j].name;
85. }
86. **break**;
87. }
88. k++;
89. **if** (k >= 5)k = k - 5;
90. }
91. }
92. }

two()函数的代码实现

1. **void** graph::two(**int** i,**int** j,**int** q)
2. {
3. **int** count = 0;
4. **for** (**int** p = 0; p < 5; p++)
5. {
6. **if** (classjudge[i][p][0] == 0 || classjudge[i][p][2] == 0)
7. {
8. count = 1;
9. }
10. }
11. **int** k = q;
12. **if** (count==1)//两个课时依然存在的情况
13. {
14. **while**(1)
15. {
16. **if** (classjudge[i][k][0] == 0 || classjudge[i][k][2] == 0)
17. {
18. **if** (classjudge[i][k][2] == 0)
19. {
20. classjudge[i][k][2] = 2;
21. classtable[i][k][5] = t[i][j].name;
22. classtable[i][k][6] = t[i][j].name;
23. }
24. **else**
25. {
26. classjudge[i][k][0] = 2;
27. classtable[i][k][0] = t[i][j].name;
28. classtable[i][k][1] = t[i][j].name;
29. }
30. **break**;
31. }
32. k++;
33. **if** (k >= 5)k = k - 5;
34. }
35. }
36. **else**//没有两节课时的课了，需要将其安排到三个课时的时间里
37. {
38. **while**(1)
39. {
40. **if** (classjudge[i][k][1] == 0 || classjudge[i][k][3] == 0)
41. {
42. **if** (classjudge[i][k][1] == 0)
43. {
44. classjudge[i][k][1] = 2;
45. classtable[i][k][2] = t[i][j].name;
46. classtable[i][k][3] = t[i][j].name;
47. }
48. **else**
49. {
50. classjudge[i][k][3] = 2;
51. classtable[i][k][7] = t[i][j].name;
52. classtable[i][k][8] = t[i][j].name;
53. }
54. **break**;
55. }
56. k++;
57. **if** (k >= 5)k = k - 5;
58. }
59. }
60. }

总体排课的代码实现

1. **void** graph::schedule()
2. {
3. **for** (**int** i = 0; i < 8; i++)//对每一个学期进行遍历进行排课
4. {
5. **for** (**int** j = 0; j < t[i].size(); j++)//对每个学期内的每个课程进行考虑
6. {
7. **switch** (t[i][j].time)
8. {
9. **case** 3:
10. {
11. three(i,j,1);
12. **break**;
13. }
14. **case** 4:
15. {
16. two(i,j,0);
17. two(i,j,2);
18. **break**;
19. }
20. **case** 5:
21. {
22. three(i,j,2);
23. two(i,j,4);
24. **break**;
25. }
26. **case** 6:
27. {
28. two(i,j,0);
29. two(i,j,1);
30. two(i,j,3);
31. **break**;
32. }
33. }
34. }
35. }
36. }
    * 1. 说明

在这两个函数的实现中，如果都是从第一天开始寻找的话，将使得整个课表前半段排的特别满，但是后半段非常的空，这是我们排课不希望看到的结果。所以在实际排课中，three()和two()函数分别从上午和下午优先考虑，schedule()函数中不同的课时情况从不同的天数开始优先考虑，如果有分成两个课的情况，第二节课从两天后开始优先考虑排课。这样能够尽可能使得课表排的相对均匀。

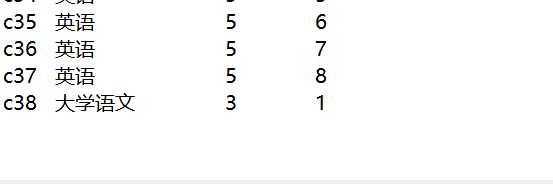
* 1. 输出功能实现

代码实现：

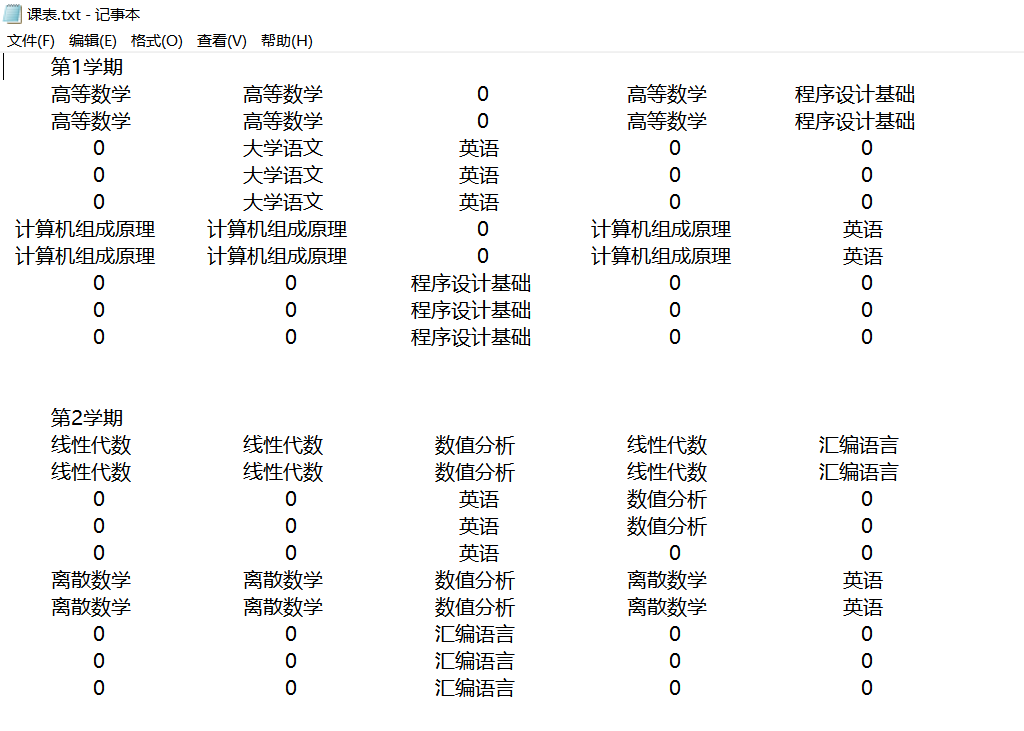
1. **void** graph::out()
2. {
3. ofstream OutFile("课表.txt");
4. **for** (**int** i=0;i<8;i++)//学期循环
5. {
6. OutFile <<setw(10)<< "第"<<i+1<<"学期"<<endl;
7. **for** (**int** j=0;j<10;j++)//课的循环
8. {
9. **for** (**int** k=0;k<5;k++)//天的循环
10. {
11. OutFile<<setw(16)<<classtable[i][k][j]<<"\t";
12. }
13. OutFile << endl;
14. }
15. OutFile << endl << endl;
16. }
17. }
18. 输入输出测试

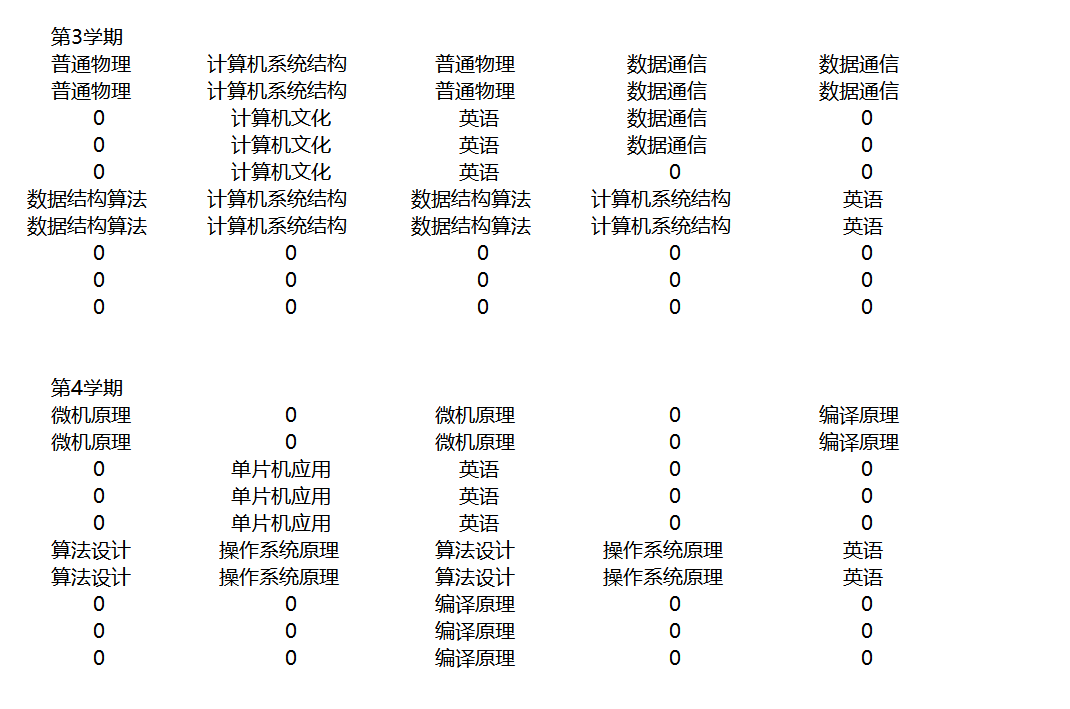
输入文件：

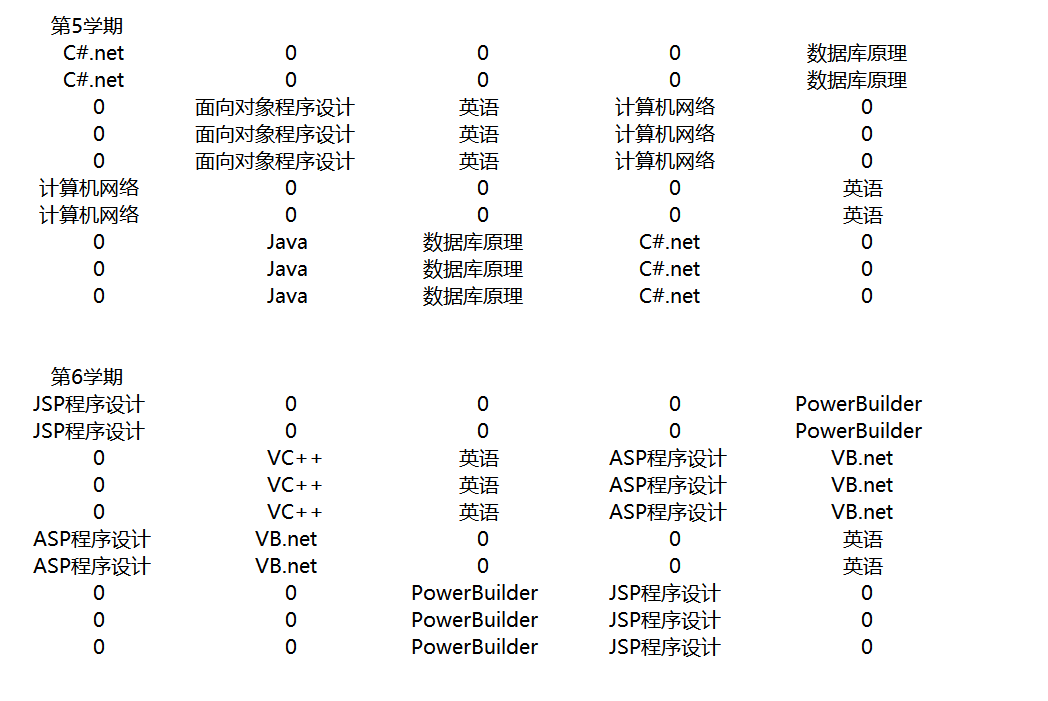


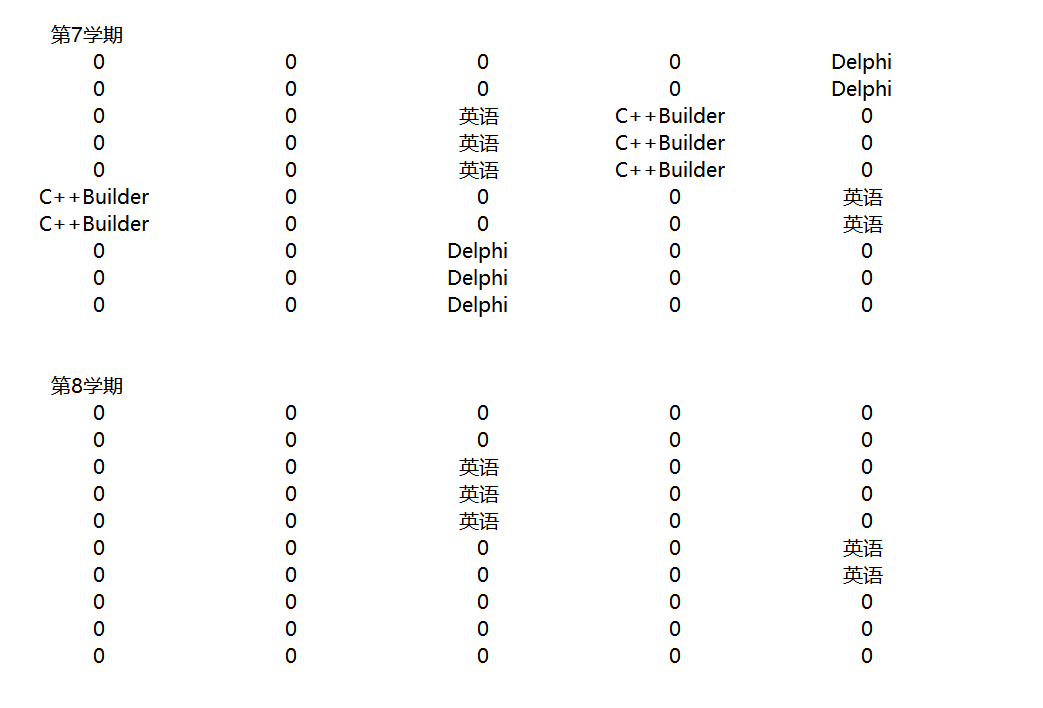


输出文件：









1. 改进方向

在这个程序中有一些优化的方向

* 1. 文件输入错误提示的优化

对于文件输入不可避免会有一些错误不规范的信息，但这在这段程序中没有进行处理，优化方向应该放在字符串转数字时，如果不规范，提示报错信息。

* 1. 排课算法的优化

在这个课程文件中，有一些课程属于极为关键的课程，是很多后续课程的先修课程，这会使得如果没有选择恰当的学期课时安排，会有一些学期课程数非常少，只有将这学期的先修课程修完后才能开始后续学习，这样安排也是不够合理的，应当对一些学期过少的课时数进行判断，如果后续还有很多课程，可以将后续课程提前放到同一学期内进行学习，这就是可以优化的方向。