

**课 程 设 计 报 告**

**题目： 排课评估与分析助手**

**课程名称： 数据结构课程设计**

**专业班级： 校交1601**

**学 号： U201612696**

**姓 名： 陈淏睿**

**指导教师： 周全**

**报告日期： 2018.6.4**

**计算机科学与技术学院**

## 任 务 书

**设计内容**

设计有效的逻辑数据结构与存储结构表示课表信息，教师信息，课程信息，班级信息，学生信息，教室信息等，由时间、教室、课程名、班级、教师名、上课周期及课堂ID等构成课堂信息。根据课堂及其复杂相关性，依据合理的逻辑与准则评价排课课表的教师安排合理性、课程安排合理性、班级与学生课程学习合理性、教室安排合理性与利用率、能效情况等。本设计只对现有排课进行评估，不去求解最优排课方案。

**设计要求**

（1）交互式操作界面(并非一定指图形式界面)；

（2）教师、教室、班级、课程等的增加、删除、修改、查找与检索等；

（3）课堂的增加、删除、修改、查找与检索，包括按教师检索其任课课堂，按班级检索课堂，按教室、课程、时间等检索课堂；

（4）空闲教室查找与检索、教室利用率分析、能效分析；

（5）教师承担课堂合理性分析；

（6）班级参与课堂的合理性分析；

（7）主要数据对象的数据文件组织与存储。

**参考文献**

[1] 严蔚敏, 吴伟民. 数据结构（C语言版）. 北京: 清华大学出版社,1997

[2] 严蔚敏, 吴伟民, 米宁. 数据结构题集（C语言版）. 北京: 清华大学出版社,1999

[3] 王晓东. 计算机算法设计与分析. 北京: 电子工业出版社, 2007

[4]王秀焕. 基于图论的高校排课系统优化研究.重庆大学硕士学位论文，2012

[5] 洪文，朱广斌. 排课问题及其数学模型. 安徽电力职工大学学报，2002，7（3）：74-77

[6] 陶涛，谢卫星. 课表模型及排课算法应用. 计算机系统应用，2011，20（2）：198-201

[7] 徐晓.基于本体映射和规则推理的排课模型研究. 软件导刊，2009，8（2）：1-4

**目 录**

[任 务 书 I](#_Toc515962734)

[1 引言 1](#_Toc515962735)

[1.1 课题背景与意义 1](#_Toc515962736)

[1.1.1 背景 1](#_Toc515962737)

[1.1.2 意义 1](#_Toc515962738)

[1.2 国内外研究现状 1](#_Toc515962739)

[1.3 课程设计的主要研究工作 2](#_Toc515962740)

[2 系统需求分析与总体设计 3](#_Toc515962741)

[2.1 系统需求分析 3](#_Toc515962742)

[2.1.1 系统目标 3](#_Toc515962743)

[2.1.2 事务处理流程 3](#_Toc515962744)

[2.2 系统总体设计 5](#_Toc515962745)

[2.2.1 数据维护 5](#_Toc515962746)

[2.2.2 数据查询 6](#_Toc515962747)

[2.2.3 数据统计 6](#_Toc515962748)

[2.2.4 帮助 7](#_Toc515962749)

[3 系统详细设计 8](#_Toc515962750)

[3.1 有关数据结构的定义 8](#_Toc515962751)

[3.1.1 数据清单 8](#_Toc515962752)

[3.1.2 数据关系 9](#_Toc515962753)

[3.2 主要算法设计 10](#_Toc515962754)

[3.2.1 基础功能函数 10](#_Toc515962755)

[3.2.2 数据维护函数 12](#_Toc515962756)

[3.2.3 数据查找及统计函数 21](#_Toc515962757)

[3.2.4 文件存取及其他函数 25](#_Toc515962758)

[3.2.5 时空复杂度分析 28](#_Toc515962759)

[4 系统实现与测试 30](#_Toc515962760)

[4.1 系统实现 30](#_Toc515962761)

[4.1.1 头文件及预定义常量说明 30](#_Toc515962762)

[4.1.2 数据结构定义 30](#_Toc515962763)

[4.1.3 函数声明 34](#_Toc515962764)

[4.2 系统测试 37](#_Toc515962765)

[4.2.1 测试方法及原则 37](#_Toc515962766)

[4.2.2 测试计划 37](#_Toc515962767)

[4.2.3 测试 43](#_Toc515962768)

[5 总结与展望 65](#_Toc515962769)

[5.1 全文总结 65](#_Toc515962770)

[5.2 工作展望 65](#_Toc515962771)

[6 体 会 66](#_Toc515962772)

[参考文献 68](#_Toc515962773)

[附录 69](#_Toc515962774)

## 1 引言

1.1 课题背景与意义

排课问题是一个解决时间和空间资源矛盾的多因素优化决策问题，实现对班级、教师、时间、课程和教室（实验室）等几个相互制约的基本因素进行时空安排，这种安排需要满足一定的约束条件，如教室的容量、教师的教学效果等方面。

1.1.1 背景

随着近年高校招生规模的不断扩大，教师和学生的人数激增，所开设课程数目繁多，如果能充分利用有限的教学资源，高效地排出一个优质的课表，对于稳定教学秩序、提高教学质量有着积极的意义。排课问题是教学管理工作中的一个难题，它实际上是一类特设的资源调度问题，是将课程、教师和学生在合适的时间段内分配到合适的教室中，涉及到的因素较多，是一个多因素的整体优化问题，即时间表问题，应用领域非常广泛。

1.1.2 意义

教学管理工作中，排课是一项十分复杂、棘手的工作。如何利用有限的师资力量和有限教学资源，排出一个合理的课程安排结果，对稳定教学秩序、提高教学质量有着积极的意义。

1.2 国内外研究现状

60年代末，国外就有人开始研究课表编排问题。1962年，Gotlieb曾提出了一个课表问题的数学模型之后人们对课表问题的算法、解的存在性等问题做了很多深入探讨。但是大多数文献所用的数学模型都是Gotlieb的数学模型的简化或补充，而至今还没有一个可行的算法来解决课表问题。 40年来人们对课表问题的计算机解法做了许多尝试。其中课表编排的整数规划模型将问题归结为求一组0-1变量的解，但是其计算量非常大。解决0-1线性优化问题的分支一定界技术却只适用也规模较小的课表编排，Mihoc和Balas将课表公式化为一个优化问题，Krawczk则提出一种线性编程的方法。Junginger将课表问题简化为三维运输问题，而Tripathy则把课表问题视作整数线性编程问题并提出了课表的数学模型

此外，有些文献试图从图论的角度来求解排课表的问题，但是图的染色问题也是NP完全问题，只有在极为简单的情况下才可以将课表编排转化为二部图匹配问题，这样的数学模型与实际相差太远，所以对于大多数学校的课表编排问题来说没有实用价值。 目前，解决课表的方法有：模拟手工排课法，图论方法，拉格朗日法，二次分配法等多种方法。由于课表约束复杂，用数学方法进行描述时往往导致问题规模剧烈增大，这已经成为应用数学编程解决课表问题的巨大障碍。

1.3 课程设计的主要研究工作

从学校排课课表的EXCEL文件获取原始课堂排课数据，抽取其中的课堂信息，从其它资源获取教师、教室、班级、课程等数据。课堂数据可手动录入，鼓励利用相关编程接口自动抽取。使用线性表、查找表、图等结构建模所处理的数据，如有必要，鼓励使用排序算法优化操作。测试数据至少涵盖本院一学期全部课堂。合理性的逻辑与规则可访谈教师、后勤、管理、同学等相关人员。不允许使用数据库系统管理数据。在界面设计与其他功能上可自由发挥。

## 2 系统需求分析与总体设计

2.1 系统需求分析

2.1.1 系统目标

本系统首先应实现基于华中科技大学计算机学院2017-18第二学期课表的数据库相关操作。其中包括：（1）教师、教室、班级、课程等的增加、删除、修改、查找与检索等；（2）课堂的增加、删除、修改、查找与检索，包括按教师检索其任课课堂，按班级检索课堂，按教室、课程、时间等检索课堂。

其次，本系统应能够对目前现有的排课进行数据统计，包括：空闲教室查找与检索与教室利用率分析，同时对现有的排课方案方式以多种标准进行评价。在本系统中，评价方式具体包括：（1）特定时间范围内的教室能效比；（2）特定教师在一周内每天承担的课程的时间分布合理性；（3）特定教师在每周内承担的课程的时间分布合理性；（4）对高龄教师（男：55岁；女，50岁），其上课教室所在楼层的合理性；（5）特定班级在一周内每天上的课程的时间分布合理性；（6）特定班级每周上的的课程的时间分布及数量合理性；（7）特定班级本学期在修课程的合理性（是否之前已学过每门课程的先修课）。

2.1.2 事务处理流程

系统采用类似windows文件夹窗口的菜单界面（如图2-1）。进入操作界面之前，系统首先对要使用的各种结构体进行初始化。此工作完成后系统显示如图2-1。

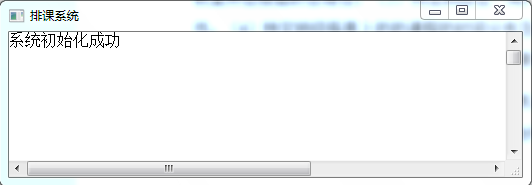


图2-1□系统初始化界面

进入系统后主界面如图2-2所示。最上方为主菜单栏，可用鼠标或键盘方向键实现在各个主菜单及子菜单选项间的切换及选择。



图2-2□系统主界面

根据功能的不同，系统可能会进入二级菜单，如图2-3所示。

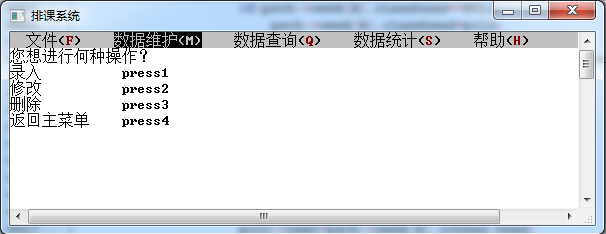


图2-3□系统二级菜单界面

从主菜单或二级菜单进入功能函数并执行完毕（或直接返回主菜单）后总会执行一次清屏并返回图2-2所示界面，并可继续进行其它操作。当选择退出选项时，进入图2-4所示界面。再次按任意键即可真正退出系统。



图2-4□退出系统界面

2.2 系统总体设计

整个系统分成5大功能模块：文件，数据维护，数据查询，数据统计与帮助。

文件部分共三个分支：数据备份，数据恢复与退出。其中数据备份将当前系统中的数据存入磁盘文件中；数据恢复可在系统运行任意时刻将之前由数据备份保存的数据录入系统并替换当前数据（若有）；退出即结束运行，退出系统。

2.2.1 数据维护

数据维护部分共五个分支：教师信息，教室信息，课程信息，班级信息与课堂信息。每个分支下的次级菜单均提供增加，修改，删除数据对象三种数据维护的方式。

添加教师时，可添加其姓名，性别及年龄；修改教师时，可修改其姓名，性别及年龄，在修改教师姓名的同时会同时修改所有该教师任课的课堂中的对应信息；删除教师时，同时删除该教师授课的所有课堂。添加教室时，可添加其编号及容量；修改教室时，可修改其编号及容量，在修改教室编号的同时会同时修改所有在该教室上课的课堂中的对应信息；删除教室时，同时删除在该教室上课的所有课堂。添加课程时，可添加其名称及各门先修课的名称；修改课程时，可修改其名称或添加/删除其先修课，修改课程名称的同时会修改所有课堂中的相应信息；删除课程时，会删除所有该课程的课堂。添加班级时，可添加其班号，人数及各门已修课程；修改班级时，可添加其班号，人数或添加/删除已修课程；删除班级时，一并删除其下属的班级课程表结构。添加课堂时，可添加其课程名称，上课教室，任课教师（这三项信息若上不存在于对应的查找表中，则自动创建），上课时间（包含节数，星期几及周数）及课堂中各个班的班号；修改课堂时，可修改其上课教室，任课教师及上课时间（包含节数，星期几及周数）；删除课堂时，可选择将某个班级从一个课堂中剔除或删除整个课堂。

2.2.2 数据查询

数据查询部分共五个分支：教师信息，教室信息，课程信息，班级信息与课堂信息。在教师信息中，提供按教师姓名，按年龄区间，按性别三种查找/检索方式；在教室信息中，提供按教室编号，按容量区间两种查找/检索方式；在课程信息中，提供按课程名称子串模糊查找的方式；在班级信息中，提供按班号，按人数范围两种查找/检索方式；在课堂信息中，提供按任课教师，按班级，按教室+课程+上课时间，按特定上课时间四种查找/检索方式。

2.2.3 数据统计

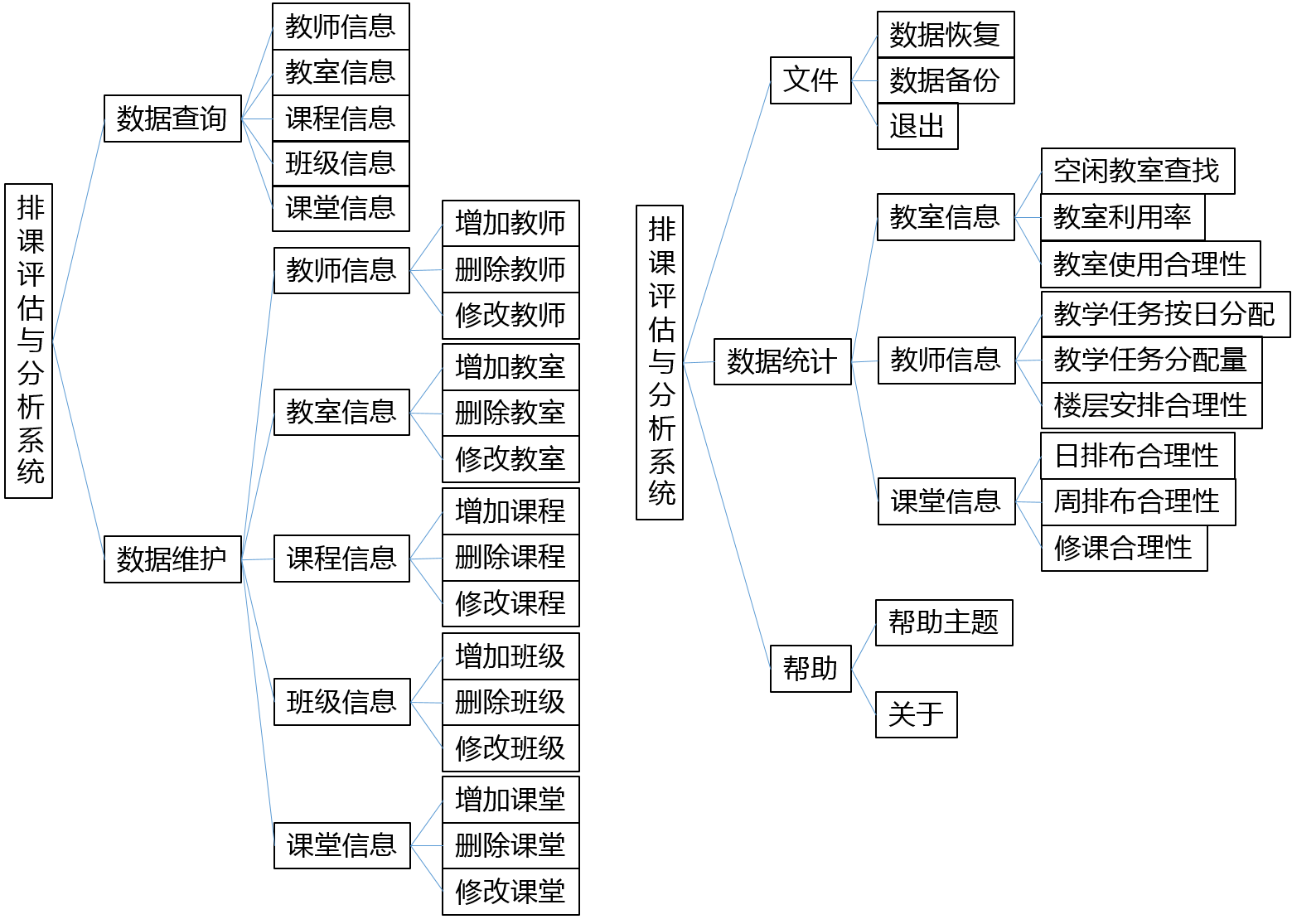
数据统计部分共三个分支：教师信息，教室信息与课堂信息。在教师信息中，提供教师的教学任务按日分配的合理性，分配给教师的教学任务总量的合理性，对高龄教师而言楼层设置的合理性三种评估方式；在教室信息中，提供特定时间段内空闲教室查找，特定时间段内教师利用率及教室与课堂适配的合理性三种统计/评估方式；在课堂信息中，提供特定班级的课程在每一天内设置的合理性，课程在每周内排布的合理性以及本学期修课合理性（即此班级本学期所修课程的先修课是否被包含在其已修课程清单中）三种评估方式。

图2-5□系统模块结构图

2.2.4 帮助

帮助部分共帮助主题和关于……两个分支。帮助主题会打开目录下名为“help.txt”的系统使用说明文档，关于……则打开显示本系统相关属性的对话框。

系统模块结构图如图2-5所示。

## 3 系统详细设计

3.1 有关数据结构的定义

3.1.1 数据清单

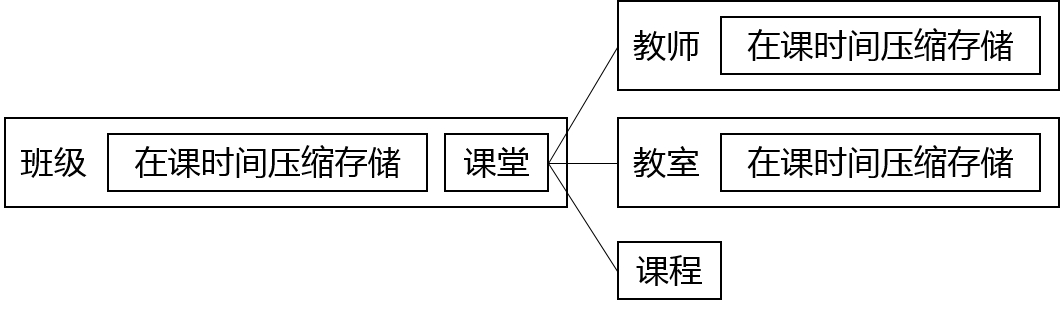
这系统中共处理五个数据对象，分别为教师，教室，课程，班级，课堂。其中教室，教师，班级结构中还包含在课时间三维数组的压缩存储表子对象。教师结构中包含教师姓名，性别代号，年龄，在课时间压缩存储结构表头数组；教师结构中包含包含教室编号，教师容量，在课时间压缩存储结构表头数组；课程结构中包含课程名称，先修课数目，先修课头结点指针；班级结构中包含班号，人数，已修课程数目，在修课程数目，已修课程字符串组，在修课程二维数组，，指向下一结点的指针及日期结构数组，其中包含在课时间压缩存储结构表头，课堂结构头结点，当日课堂数目；课堂结构中包含课程名称，教室编号，教师姓名，开始节，结束节，起始周数数组，结束周数数组；在课时间压缩存储结构包含上课周数与上课节数。具体数据类型如表3-1所示。

表3-1□数据类型清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 结构 | 变量名 | | 类型 | 注释 |
| 教师表头结构  TeList | length | | unsigned short | 查找表长 |
| listsize | | unsigned short | 查找表空间数量 |
| Elem | | Teacher\* | 表头指针 |
| 教师结构  Teacher | tname[20] | | char | 姓名 |
| gender | | unsigned short | 性别代号 |
| Age | | unsigned short | 年龄 |
| Ochead[6] | | Ochead | 在课时间压缩存储表头 |
|  | | | | |
| 教室表头结构  Rolist | length | | unsigned short | 查找表长 |
| listsize | | unsigned short | 查找表空间数量 |
| Elem | | Room\* | 表头指针 |
| 教室结构Room | code[20] | | char | 教室编号 |
| Rsize | | unsigned short | 容量 |
| ochead[6] | | Ochead | 在课时间压缩存储表头 |
|  | | | | |
| 课程拓扑图头结构  Cougraph | length | | unsigned short | 邻接表长 |
| listsize | | unsigned short | 邻接表空间数量 |
| coulist | | CouHead\* | 表头指针 |
| 课程逆邻接表头结构  CouHead | name[60] | | char | 课程名称 |
| arcnum | | unsigned short | 先修课数目 |
| archead | | CouArc\* | 先修课头指针 |
| 先修课结点  CouArc | prename | | char\* | 先修课名称 |
| nextarc | | CouArc\* | 指向下一先修课的指针 |
|  | | | | |
| 班级表头结点  Grade | gr\_num | | unsigned short | 该年级下班级结点数目 |
| schead | | Schedule\* | 班级节点头指针 |
| 班级节点  Schedule | classname[20] | | char | 班号 |
| clsize | | unsigned short | 人数 |
| do\_num | | unsigned short | 已修课程数目 |
| pr\_num | | unsigned short | 在修课程数目 |
| done [60][60] | | char | 已修课程 |
| progress [15][60] | | char | 在修课程 |
| next | | Schedule\* | 指向下一班级的指针 |
| week[6] | | struct weekday | 上课日结构 |
| 上课日结构  week | cl\_num | unsigned short | 当日课堂数量 |
| classhead | Class \* | 课堂头结点 |
| ochead | Ochead | 在课时间压缩存储表头 |
| 课堂结点  Class | coursename[60] | | char | 课程名称 |
| roomcode[20] | | char | 教室编号 |
| tname[20] | | char | 教师姓名 |
| cou\_st | | unsigned short | 开始节数 |
| cou\_ed | | unsigned short | 结束节数 |
| wk\_st[5] | | unsigned short | 开始周数 |
| wk\_ed[5] | | unsigned short | 结束周数 |
| next | | struct Class\* | 指向下一课堂的指针 |
|  | | | | |
| 在课时间压缩存储表头结点  Ochead | len | | unsigned short | 压缩存储结点长度 |
| head | | Occupy\* | 压缩存储结点头指针 |
| 在课时间压缩存储结点  Occupy | week | | unsigned short | 上课周数 |
| time | | unsigned short | 上课节数 |
| next | | struct Occupy\* | 指向下一结点的指针 |

3.1.2 数据关系

各个班级根据所在年级组织，课堂为其每个上课班级建一份拷贝结点并挂靠在对应班级的对应时间下。课堂中记录对应的课堂，教室，教师以实现关联。班级，教师与教师各有六组在课时间压缩存储查找表，记录其各自的周一至周六的上课时间。

图3-1□数据关系图

这部分要写的：（1）首先描述系统中要处理那些数据，每种类型的数据包括哪些数据项，每个数据项的数据类型，最后可用一个表格表示出来；（2) 描述这多种数据在系统中如何关联，可通过图直观的说明这多种数据间的关联。

3.2 主要算法设计

3.2.1 基础功能函数

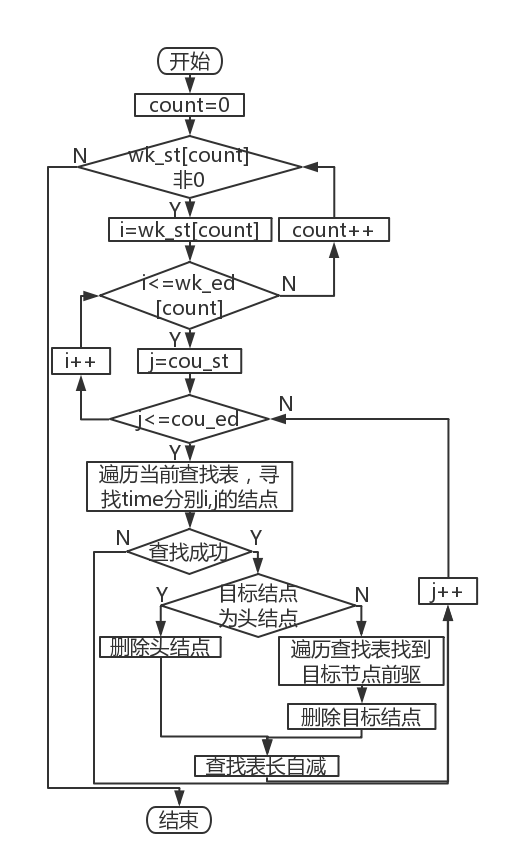
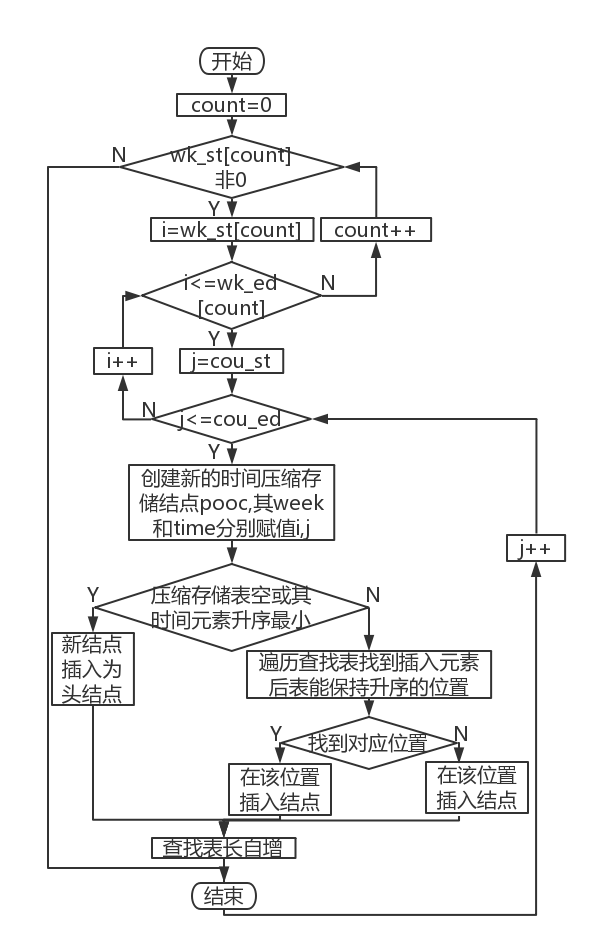
此部分的代码均在各个功能中反复使用多次，因此将其提取出来作为独立函数以精简代码，取消复用。  

图3-2□删除在课时间结点流程图 图3-3□添加在课时间结点流程图

1. 删除在课时间压缩存储结点函数DelTimeNode：函数参数为开始周数数组wk\_st[]，结束周数数组wk\_ed[]，开始节数cou\_st，结束节数cou\_ed及在课时间压缩存储表头指针ochead。第一重循环判定数组当前位置的开始/结束周数是否有效，由于两个数组的有效元素必定个数相同，因此只需判定开始周数数组的当前元素是否为0. 在以上条件有效时，用两重循环承接当前周数及节数值，在压缩存储查找表中找到符合条件的结点并删除，其中目标结点是否为头结点需分别讨论。流程图如图3-2所示。

2. 添加在课时间压缩存储结点函数AddTimeNode：函数参数为开始周数数组wk\_st[]，结束周数数组wk\_ed[]，开始节数cou\_st，结束节数cou\_ed及在课时间压缩存储表头指针ochead。第一重循环判定数组当前位置的开始/结束周数是否有效，由于两个数组的有效元素必定个数相同，因此只需判定开始周数数组的当前元素是否为0. 在以上条件有效时，用两重循环承接当前周数及节数值，遍历已存在的压缩存储查找表，若表空则直接以新节点为头结点，否则按时间升序查找符合要求的位置，标准为当前结点后继的周数大于新结点的周数，或当前结点后继的周数小于等于新结点的周数的同时当前结点后继的节数大于新结点的节数。找到相应位置则插入结点至目标位置，否则插入表尾。流程图如图3-3所示。

3. 班级名称输入函数ClassInput：参数为新班级缓存串cbuf3。向串中输入新班级后，根据其年级决定查找头结点的位置，在对应班级名单中查找该班级的存在性，返回的指针记录符合要求的结点的位置，查找失败返回NULL。

4. 插入新教室函数AddNewRoom：参数为存放新教室的字符串cbuf2。若表长大于等于表的容量，则为查找表新分配数量为LISTINCREMENT（在程序中宏定义为10）的空间，若分配失败则输出提示信息并返回。在表尾录入该教师的信息并将六个压缩存储表长与表头指针分别初始化为0和NULL。

5. 插入新教师函数AddNewTeacher：思路与实现方法同插入新教室函数。

6. 插入新课程函数AddNewCourse：思路与实现方法同插入新教室函数。

7. 查询剩余课堂函数SeekRemClass：参数为班级结点指针psch及缓冲字符串cbuffer[]。在班级在修课清单中查找cbuffer中的课程，若找到，遍历该班级的所有课堂，查找是否还有此课程，若无，则将该课程从班级在修课清单中删除，在修课清单长度自减。流程图如图3-4所示。

8.删除课堂结点函数DelClassNode：参数为课堂结点指针pcls，班级结点指针psch，周数下标j。若pcls为psch->week[j]下课堂链表的头结点，删除pcls，若psch->week[j].classhead为空，则课堂链表长度psch->week[j].cl\_num置为0并直接返回。若pcls非头结点，则遍历课堂链表找到pcls的前驱，删除pcls后链表长度psch->week[j].cl\_num自减。

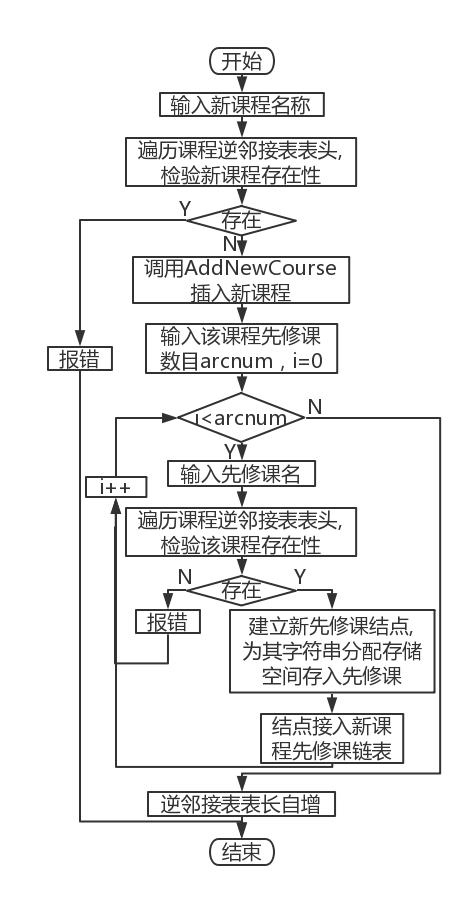
 

图3-4□查询剩余课堂流程图 图3-5□课程添加流程图

9. 插入时间函数TimeInput：参数为开始时间指针cou\_st，结束时间指针cou\_ed，开始周数数组wk\_st[]，结束周数数组wk\_ed[]。将两个数组元素全置为0，输入cou\_st和cou\_ed的值，之后循环输入各段开始及结束周数并判断是否继续。

3.2.2 数据维护函数

此部分函数实现系统中各数据对象（含教室，教师，课程，班级，课堂）的添加，修改及删除。

1. 教师添加函数AddTeacher：输入新教师，在教师线性表中寻找该教师是否已存在，若存在则输出提示信息并返回，否则调用AddNewTeacher插入新教师，输入其性别代码及年龄，线性表长自增。清屏。

2. 教室添加函数AddRoom：输入新教室，在教师线性表中寻找该教室是否已存在，若存在则输出提示信息并返回，否则调用AddNewRoom插入新教室，输入其容量，线性表长自增。清屏。

3. 课程添加函数AddCourse：输入新课程，在教师线性表中寻找该教室是否已存在，若存在则输出提示信息并返回，否则调用AddNewCourse插入新课程，输入先修课个数与相应数目的先修课。对每一门先修课，在课程线性表中检查其存在性若存在则为其创建先修课节点并接入新课程结点后，否则输出报错信息并返回。线性表长自增。清屏。流程图如图3-5所示。

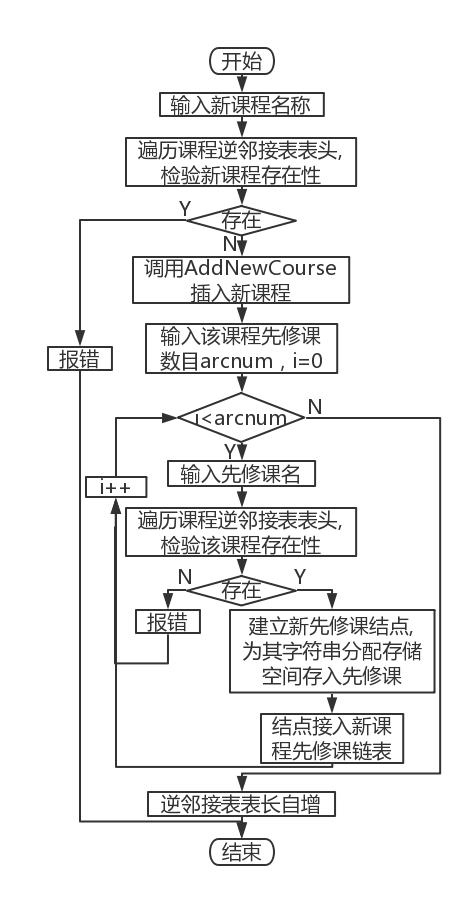


图3-5□添加课堂流程图

4. 班级添加函数AddGroup：输入新班级，调用ClassInput检查班级存在性，若已存在则输出报错信息并返回。新建班级结点psch，拷贝班级串，输入人数，已修课程数目与对应的各门已修课程，将6个week结构下的在课时间压缩存储表头指针与课堂头指针置NULL，在课时间压缩存储表长与课堂链表长置0。按新班级的年级将新节点插入为班级链表头指针，对应的课程链表长自增，清屏。

5. 教师删除函数DelTeacher：输入待删教师姓名，在教师线性表中检查其存在性，若不存在则输出报错信息并返回。二重循环删除该课程下所有在课时间压缩存储结点。在班级链表中查找所有任课教师为目标教师的课堂，用变量存储该结点的全部相关信息，包括教室编号，教师姓名，开始周数数组，结束周数数组，开始节数，结束节数。调用DelTimeNode在班级下属在课时间压缩存储表中删除该课堂对应的时间结点，调用SeekRemClass检查是否还有同名课堂，若没有则从班级在修课程清单中移除该课程。在教室查找表中定位该课堂使用的教室，调用DelTimeNode在教室下属在课时间压缩存储表中删除该课堂对应的时间结点，调用DelClassNode删除该课堂。所有相关课堂删除完成后，删除目标教师，线性表长自减，若减为0则输出相应提示信息。清屏。

6. 教室删除函数DelRoom：思路及实现方法同教师删除函数，唯一不同的是删除所有教室为目标教室的课堂时应调用调用DelTimeNode在教师下属在课时间压缩存储表中删除该课堂对应的时间结点。

7. 课程删除函数DelCourse：输入待删课程，在课程逆邻接表中检查其存在性，若不存在则输出报错信息并返回。一重循环删除目标课程的所有先修课结点。在班级链表中查找所有该课程的课堂，用变量存储该结点的全部相关信息，包括教室编号，教师姓名，开始周数数组，结束周数数组，开始节数，结束节数。调用DelTimeNode在班级下属在课时间压缩存储表中删除该课堂对应的时间结点，调用SeekRemClass检查是否还有同名课堂，若没有则从班级在修课程清单中移除该课程。在教室查找表中定位该课堂使用的教室，调用DelTimeNode在教室下属在课时间压缩存储表中删除该课堂对应的时间结点。在教师查找表中定位该课堂的任课教师，调用DelTimeNode在教师下属在课时间压缩存储表中删除该课堂对应的时间结点。调用DelClassNode删除该课堂。所有相关课堂删除完成后，删除目标课程，逆邻接表长自减，若减为0则输出相应提示信息。清屏。流程图如图3-6所示。

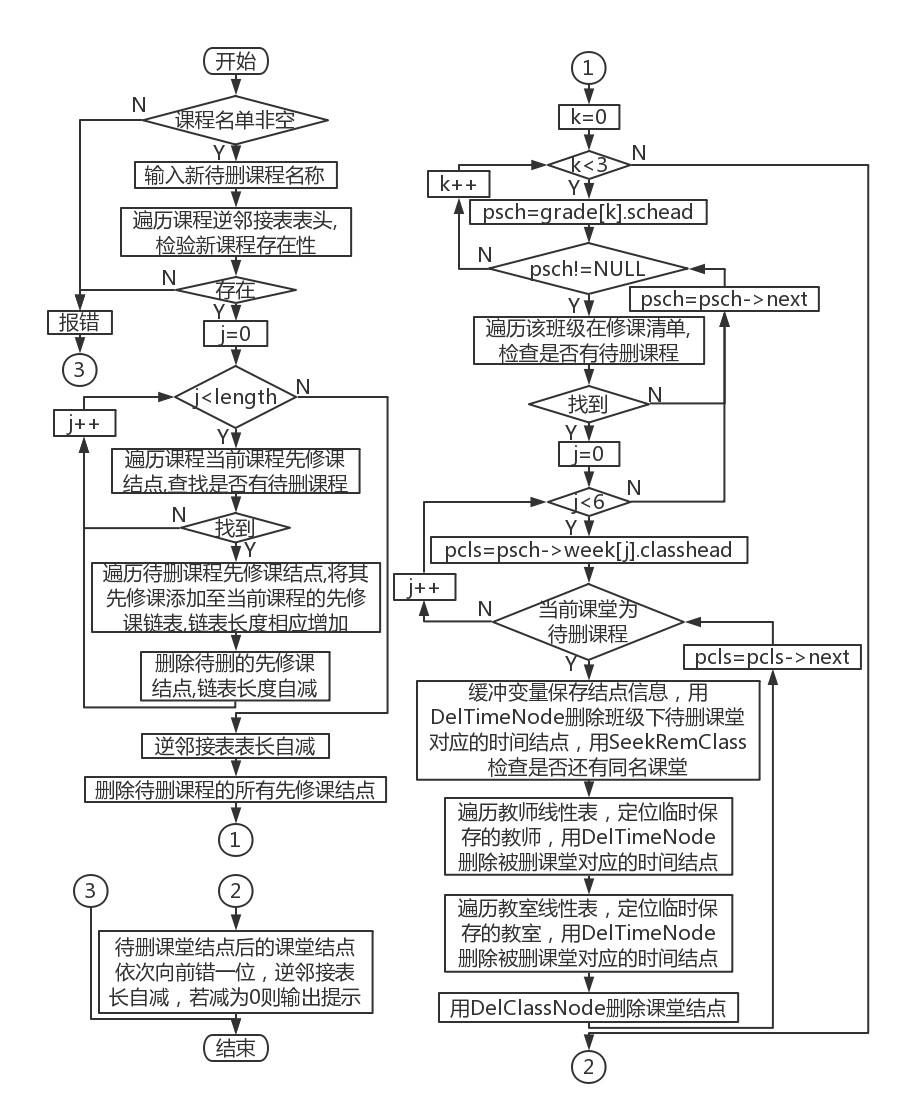


图3-6□课程删除流程图

8. 班级删除函数DelGroup：输入待删班级，进行确认删除后输入待删班级班号，调用ClassInput定位对应班级结点，二重循环删除班级下所有在课时间压缩存储结点及课堂结点。删除目标班级结点，当结点为年级下头结点时头指针直接指向其后继，否则需先遍历班级链表找到其前驱，用前驱指向后继。删除头贴点，清屏。

9. 教师修改函数AltTeacher: 首先判定教师线性表是否非空，为空则报错并返回。输入教师姓名，遍历线性表检查其存在性，若不存在则输出报错信息并返回。选择修改方式。修改教师姓名：输入新教师姓名并遍历线性表检查其存在性，若已存在则报错并返回。遍历班级链表，找到所有任课教师为待改教师的课堂并写入新教师姓名，最后将新姓名写入目标教师结点。修改性别：若原性别代号非0则改为0，若为0则改为1，然后输出提示信息。修改年龄：直接向教师结点中写入新年龄。清屏。

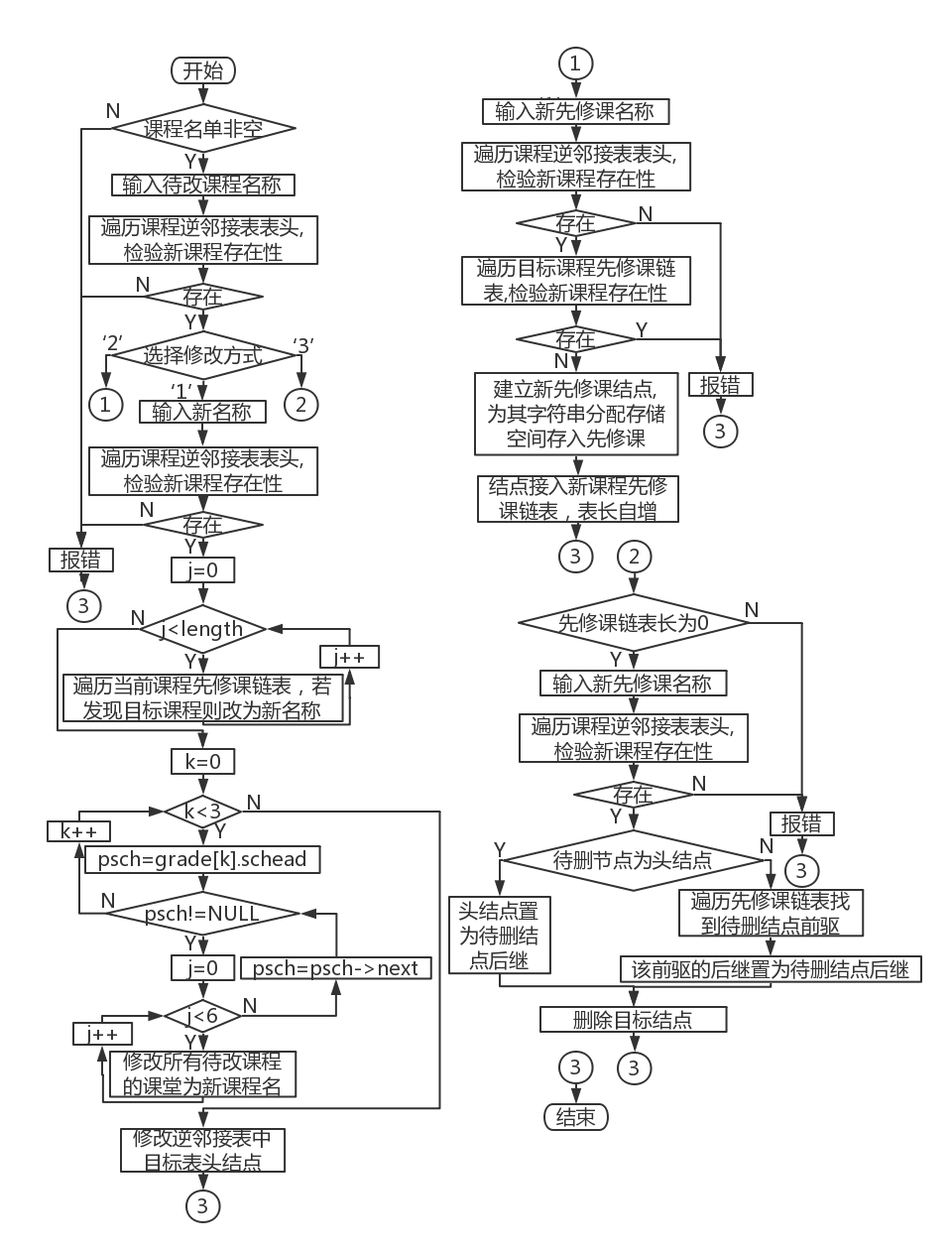


图3-7□课程删除流程图

10. 教室修改函数AltRoom：首先判定教室线性表是否非空，为空则报错并返回。输入教室编号，遍历线性表检查其存在性，若不存在则输出报错信息并返回。选择修改方式。修改教室编号：输入新编号并遍历线性表检查其存在性，若已存在则报错并返回。遍历班级链表，找到所有上课教室为待改教室的课堂并写入新编号，最后将新编号写入目标教室结点。修改容量：直接向教室结点中写入新容量。清屏。

11. 课程修改函数AltCourse：首先判定课程逆邻接表表是否非空，为空则报错并返回。输入课程名称，遍历逆邻接表表头检查其存在性，若不存在则输出报错信息并返回。选择修改方式。修改课程名称：输入新名称并遍历逆邻接表检查其存在性，若已存在则报错并返回。遍历逆邻接表，找到所有待该课程的先修课结点并修改课名。遍历班级链表，找到所有该课程课堂并写入新课名，最后将新名称写入目标课程结点。添加先修课：输入新先修课名称并遍历逆邻接表检查其存在性，若已存在则报错并返回。遍历目标课程的先修课链表，检查新先修课的存在性，若已存在则报错并返回。为新先修课建立先修课结点与空字符串，先修课写入空串并将新节点插入先修课链表。选择继续输入时返回函数起始位置，退出则进行清屏。流程图如图3-7所示。

12. 班级修改函数AltGroup：输入待该班级名称，调用ClassInput检查其存在性并返回结点地址。若结点不存在则报错并返回。选择修改方式。修改班号：输入新班号并检查其合法性。按班级所在年级遍历班级链表，若新班号已被占用则报错并返回。将新班号写入班级节点。添加已修课程：输入新的已修课名称，遍历已修课清单检查其存在性，若存在则报错返回。在表尾插入新课程，已修课清单长度自增。修改已修课程：输入待该课程名称，遍历已修课清单检查其存在性，若不存在则报错返回。输入新课名，遍历已修课清单检查其存在性，若存在则报错返回。用新课名覆盖原课名。删除已修课程：输入待删课名，遍历已修课清单检查其存在性，若不存在则报错返回。待删课之后的全部课程向前错一位以达到删除效果，同时已修课清单长度自减。修改人数：直接向班级结点中写入新人数。清屏。

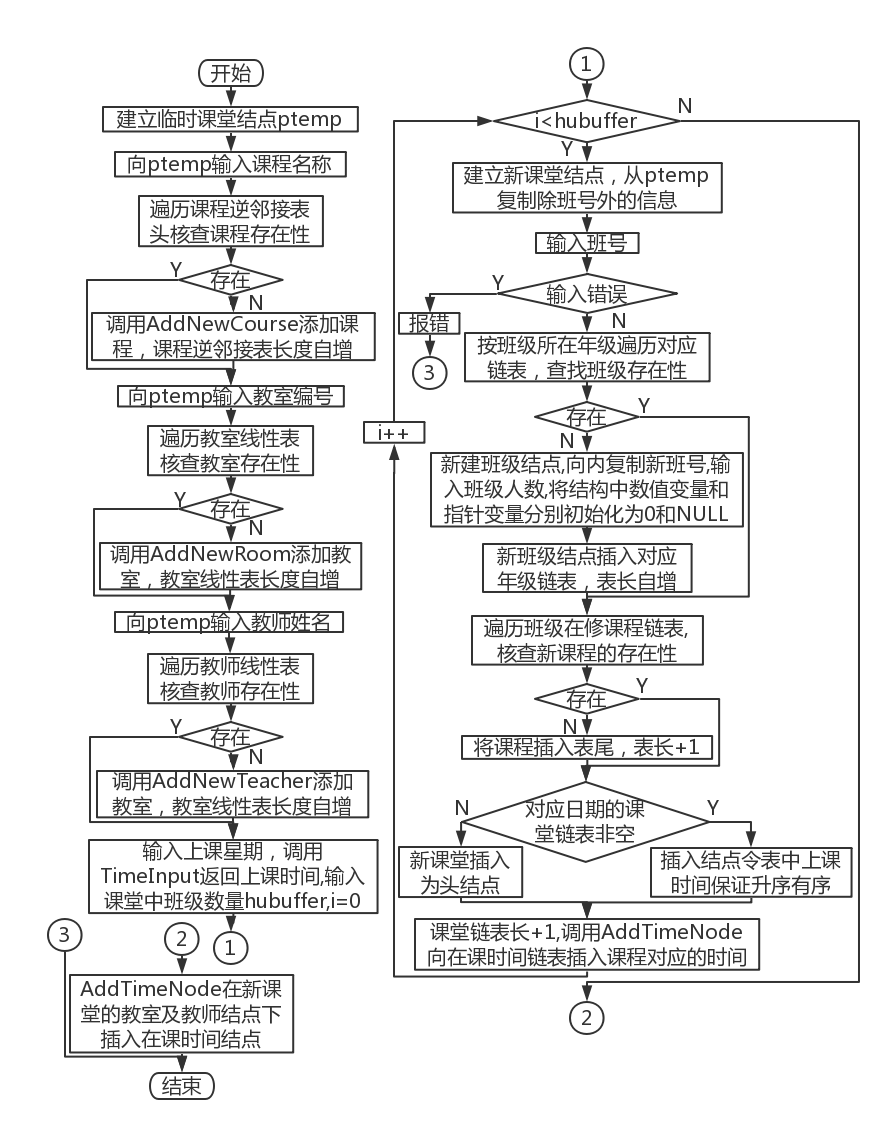


图3-8□课堂添加流程图

13. 课堂添加函数AddClass：建立临时课堂结点。向临时结点中输入课程名，遍历课程逆邻接表检查课程存在性，若不存在则调用AddNewCourse添加新课程，逆邻接表长度自增。按同样的流程向临时结点中输入教师编号及教师姓名。调用TimeInput向结点中写入上课时间。输入班级数量，在每一次循环中，建立新课程结点并将临时结点中的已录入信息向新结点拷贝。输入上课班级的班号，根据其所在年级搜索目标班级存在性，班号输入错误时报错返回。若不存在则新建一班级结点，将已修课程数与在修课程数均初始化为0，输入新班级人数，同时将其下属的六个日期结构下结点与数值均初始化为NULL和0，将新班级结点插入链表。将课堂结点按期上课日期依照时间升序插入课堂链表，课程链表长度自增。同时调用AddTimeNode在班级下插入在课时间压缩存储结点。全部班级添加完毕后，调用AddTimeNode分别在此课堂对应教室与教师的节点下插入在课时间压缩存储结点。清屏。流程图如图3-8所示。

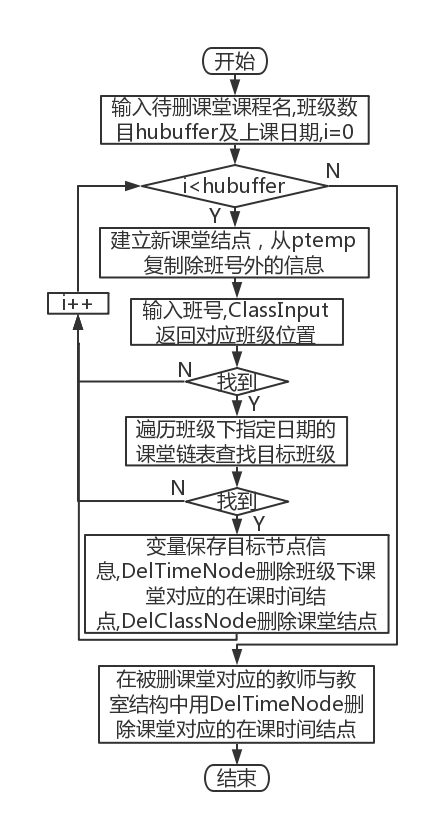


图3-9□课堂删除流程图

14. 课堂删除函数DelClass：输入待删课堂的课程名，班级数目及上课日。在每一次循环中，输入班号并调用ClassInput定位目标班级结点，未找到则报错并进入下一次循环。在班级下按上课日期在课堂链表中定位待删班级，未找到则报错并进入下一次循环。用变量存储该课堂的信息，包括教室编号，教师姓名，开始周数数组，结束周数数组，开始节数，结束节数。调用DelTimeNode删除对应的在课时间结点，调用DelClassNode删除目标课堂结点。在教师与教室线性表中分别定位被删课堂的教师与教室，调用DelTimeNode删除对应的在课时间结点。清屏。流程图如图3-9所示。

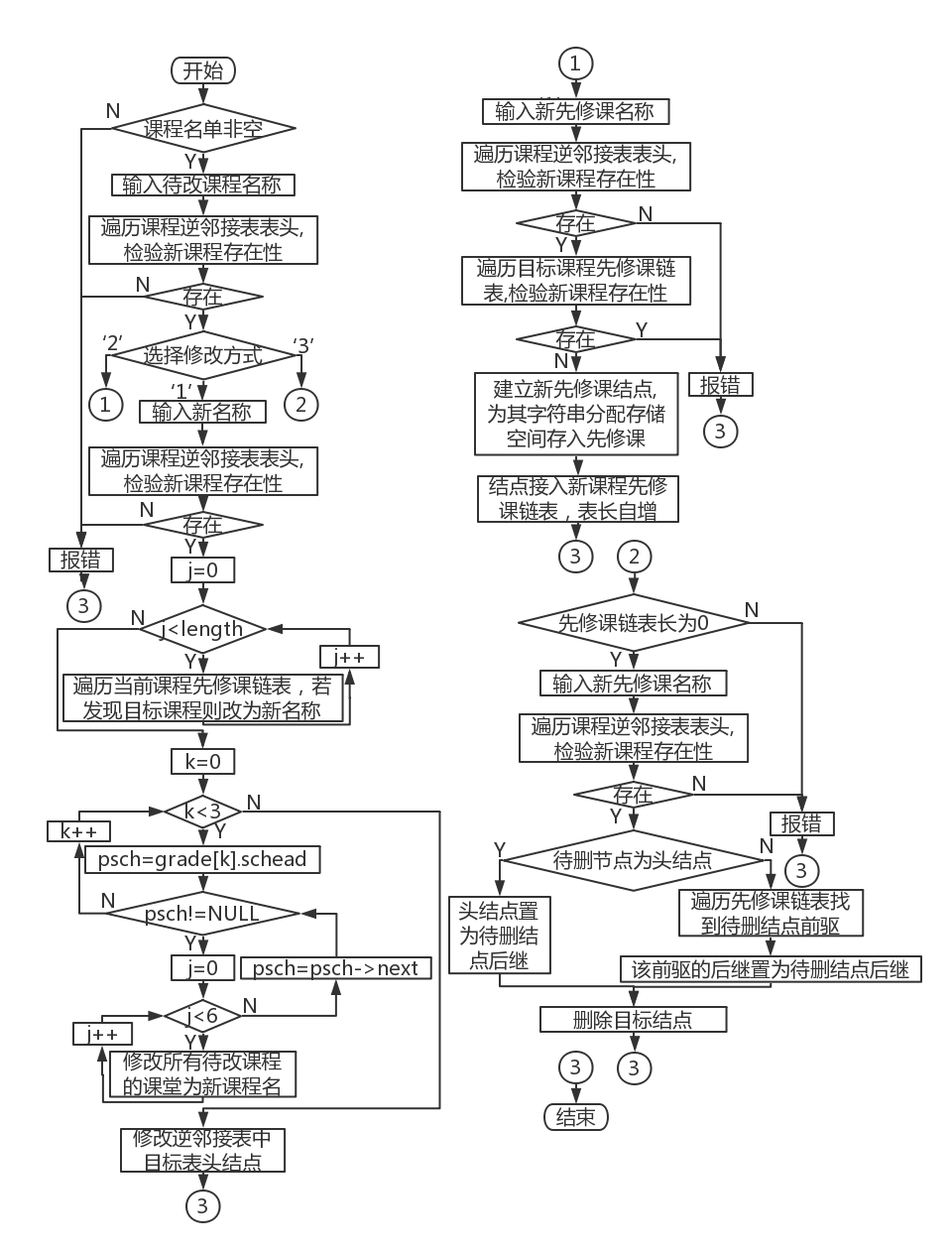


图3-10□课堂修改流程图

15. 课堂修改函数AltClass：输入待改课堂的课程名，班级数目及上课日，选择修改方式。修改教室：向缓冲串中写入新教室，遍历教室线性表检查其存在性，若不存在则用AddNewRoom添加，线性表长自增。修改教师：向缓冲串中写入新教师，遍历教师线性表检查其存在性，若不存在则用AddNewTeacher添加，线性表长自增。修改上课时间：用TimeInpu输入新上课时间。在每一次循环中，用ClassInput定位目标班级结点，未找到则报错并进入下一次循环。在班级下按上课日期在课堂链表中定位待删班级，未找到则报错并进入下一次循环。用变量存储该课堂的信息，包括教室编号，教师姓名，开始周数数组，结束周数数组，开始节数，结束节数。修改教室：将新教师编号从缓冲串复制入课堂结点。修改教师：将新教师姓名从缓冲串复制入课堂结点。修改上课时间：用DelTimeNode删除原在课时间结点，用变量复制该课堂占用的教室及教师。向课堂结点赋值新上课时间，用AddTimeNode插入新在课时间结点。循环结束后，修改教室：遍历教室线性表找到原教室，用DelTimeNode删除目标课程在课时间结点，用AddTimeNode在新教室下添加在课时间结点。修改教师：遍历教师线性表找到原教师，用DelTimeNode删除目标课程在课时间结点，用AddTimeNode在新教师下添加在课时间结点。修改上课时间：遍历教室线性表找到原教室，用DelTimeNode删除原在课时间结点，用AddTimeNode添加新在课时间结点。遍历教师线性表找到原教师，用DelTimeNode删除原在课时间结点，用AddTimeNode添加新在课时间结点。清屏。流程图如图3-10所示。

3.2.3 数据查找及统计函数

1. 教师查找函数SeekTeacher：首先检查教师线性表是否为空，空则报错返回。选择查找方式。按姓名查找：输入待查教师姓名，遍历教师线性表检查目标教师存在性，未找到返回，否则输出性别及年龄信息。按年龄区间查找：输入年龄上下限，遍历教师线性表，输出所有符合要求的教师的姓名，性别及年龄。按性别查找：输入性别代码，遍历教师线性表，输出所有符合要求的教师的姓名及年龄。清屏。

2. 教室查找函数SeekRoom：首先检查教室线性表是否为空，空则报错返回。选择查找方式。按编号查找：输入待查教室编号，遍历教室线性表检查目标教室存在性，未找到返回，否则输出容量信息。按容量区间查找：输入容量上下限，遍历教室线性表，输出所有符合要求的教室的编号及容量。清屏。

3. 课程查找函数SeekCourse：首先检查课程逆邻接表是否为空，空则报错返回。输入待查课程子串编号，遍历教室线性表进行模糊查找，若某课程含有目标子串则输出目标信息，遍历其先修课邻接表输出各先修课；遍历整个逆邻接表，若，某课程先修课为目标课程则输出此课程作为后继课，若未找到输出提示信息。清屏。

4. 班级查找函数SeekGroup：首先检查按年级划分的班级链表是否为空，空则报错返回。选择查找方式。按班号查找：输入待查班级班号，用ClassInput定位其结点，未找到则报错返回。输出班级人数，已修课程清单与在修课程清单。按人数区间查找：输入人数上下限，遍历班级链表，输出所有满足要求的班级的班号，人数，已修课程清单与在修课程清单。清屏。

5. 课堂查找函数SeekClass：选择查找方式。按任课教师查找：输入目标教师，四重循环遍历班级链表，若某课程任课教师为目标教师，输出该课程的班级，上课教室与上课时间。按班级查找：输入待查班号，用ClassInput返回其结点位置，若未找到则报错返回。二重循环遍历班级下的课堂链表，输出每个班级的任课教师，上课教师与时间。按教室+课程+时间查找：输入待查课堂的课程名，教室编号，上课日期，用TimeInput返回其上课周数与节数。三重循环遍历每个班级指定日期的课堂链表，若上课时间，教室与课程完全符合则输出该班级与课堂的任课教师。按时间查找：输出始末节数，上课日期及始末周数。四重循环遍历指定日期的课堂链表，访问其下在课时间结点，若找到某节点记录的时间在给出的目标时间以内，则输出该课堂的课程，教室，班级及教师。

6. 教室排课评估及统计函数RoomUsage：输入分析方式。查找空闲教室：输入目标周数，日期及节数范围，二重循环遍历每个教室下指定日期的在课时间链表，若当前结点记录的时间在给定时间范围内，则说明该教室在目标时间段内被占用，跳过该教室，否则输出该教室的编号。统计教室利用率：输入始末周数，始末日期，始末节数，三重循环遍历每个教室在给定日期范围内的在课时间链表，若结点记录的时间信息在给定时间范围内，则说明该课堂被占用。用计数器统计这样的课堂作为分子，以教师数目与给定的时间范围的乘积作为分子，结果即为目标时间范围内的教室利用率。教室使用合理性：输入上课教室，遍历课堂线性表检查目标课堂存在性，若不存在则报错返回。输出目标教室的容量。输入待分析课堂的课程名称与上课日期，用TimeInput返回上课时间，三重循环遍历每个班级在指定日期下的课堂链表，若有该课堂则记录此班级的人数，若未找到符合要求的班级则报错返回。统计在目标课堂内的班级的总人数并与教室容量进行比较，评价分为四档，占用座位比小于0.5为过于稀疏，建议更换更小的教室；大于0.5小于0.8为合适；大于0.8小于1为过于拥挤，建议更换更大的教室；大于1为容量过小，需更换更大的教室。清屏。流程图如图3-11所示。

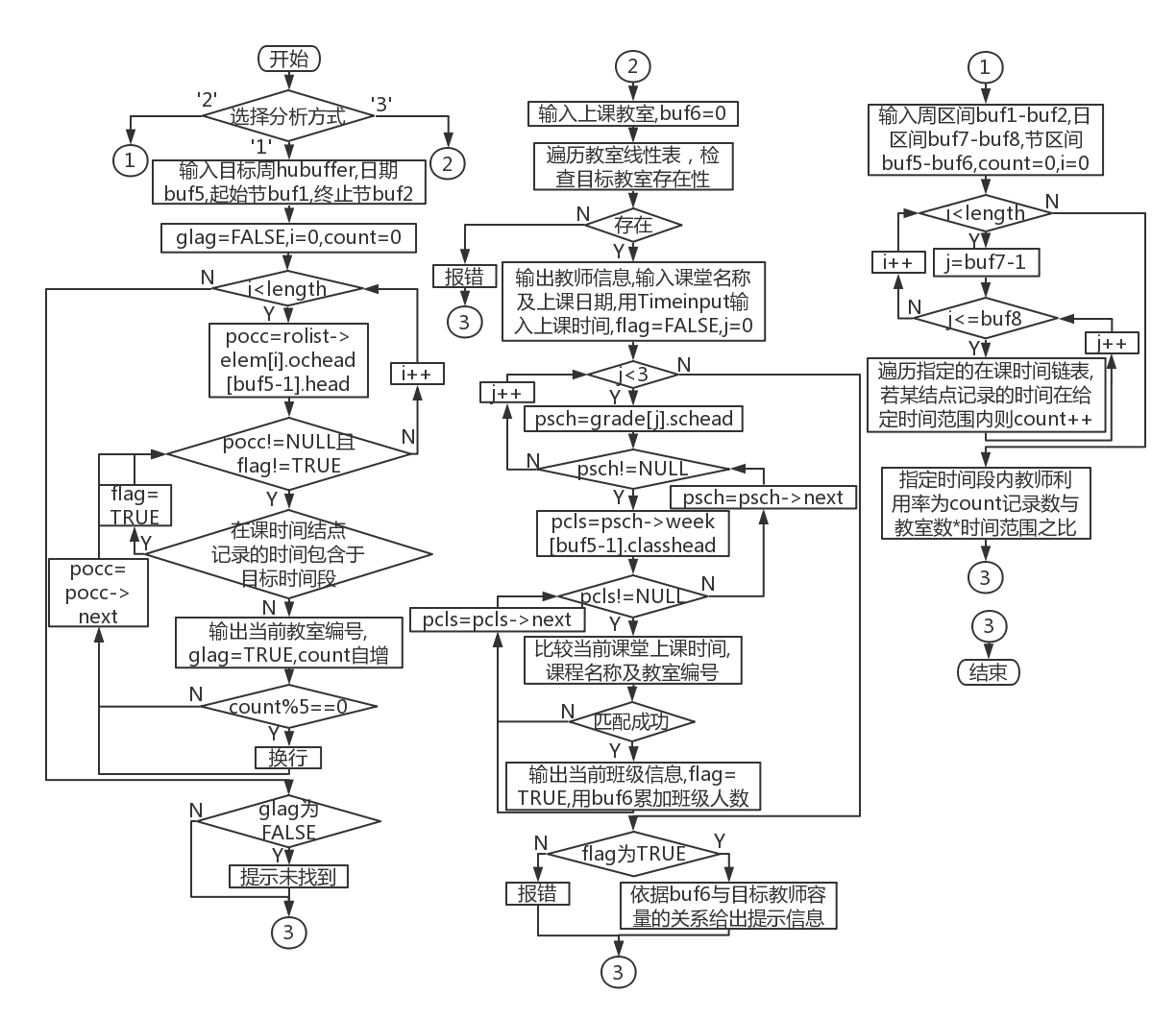


图3-11□教室排课评估及统计流程图

7. 班级排课合理性评估函数ClassAnalysis：用ClassInput返回待分析班级结点的位置。若未找到该班级则报错返回。选择分析方式。分析课程在每天内排布的合理性：评判标准为是否做到把所有课尽量安排在白天。三重循环遍历该班级下的各在课时间链表，分别累加每周上课时间为1-4节，5-8节，9-12节的结点数目，若晚课节数大于上午节数或下午节数的一半，则视为本周晚课过多，否则称本周晚课比重合适。分析课程在每周内按日排布的合理性：输入本学期有效学期数，二重循环遍历班级下各在课时间链表，用提前元素置0的occupy二维数组统计每周中每天课程节数。对本学期每个有效周，两两比较每天的课程节数，若两天课程相差大于等于6节，则判定课程按日排布过于悬殊。若一周中无这样的判定则认为该周课程按日排布较为合理。再根据occupy矩阵统计第一、二周和倒数第一、二周的课程数目。正常情况下这四周的课程数目不宜过多，若第一周安排超过20节课，第二周安排超过22节课，倒数第二周安排超过10节课，最后一周安排超过4节课，则认为对应周课程安排过多，输出提示信息。分析修课合理性：对于班级在修课程清单中的每门课程，在课程逆邻接表中遍历其先修课，对每一门先修课，若该课程未出现在班级已修课程清单中，则认为该班应修过先修课后再修本课程，判定该课程不应在本学期修，输出提示信息。清屏。流程图如图3-12所示。

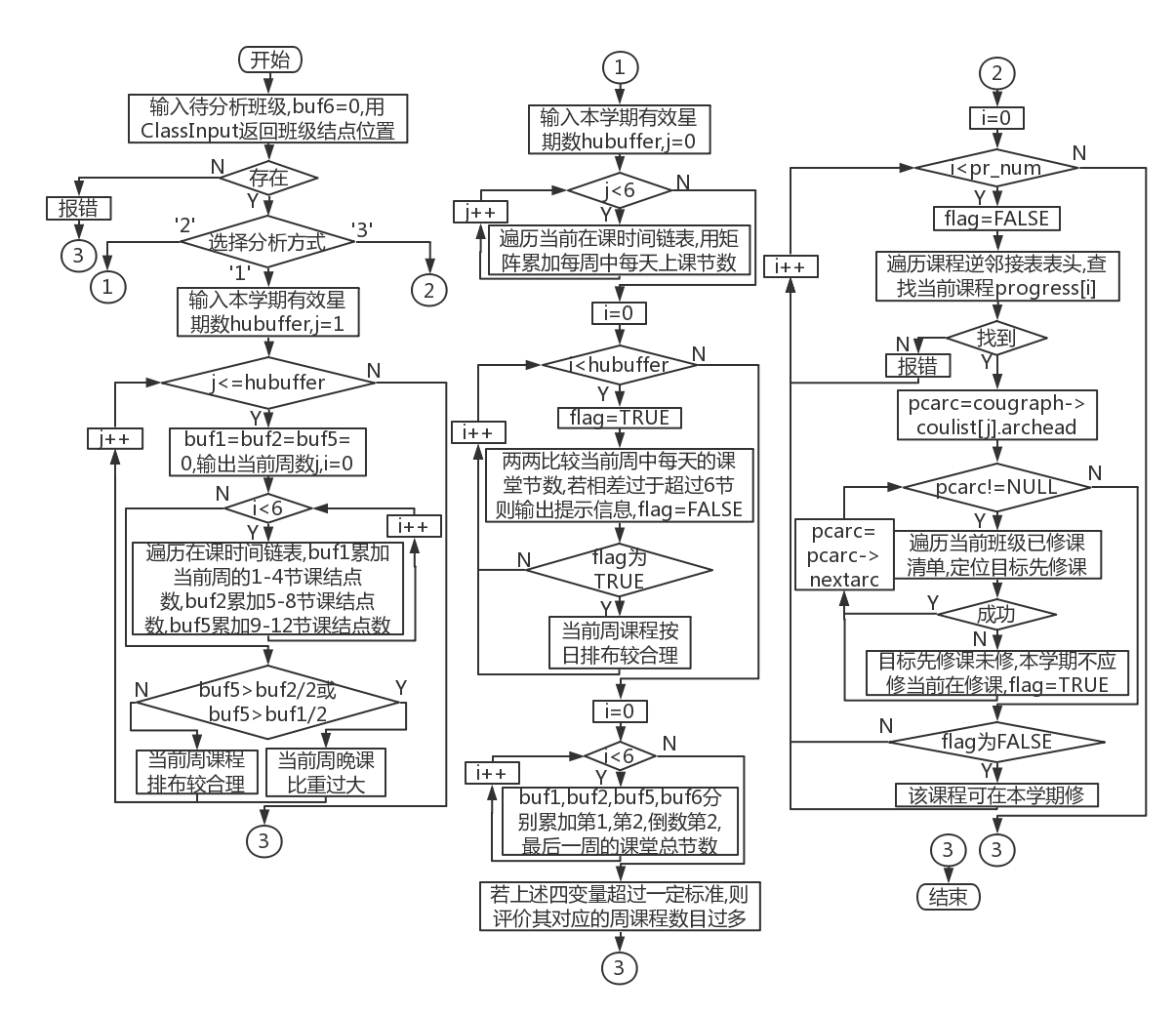


图3-12□班级排课合理性评估流程图

8. 教师排课合理性评估函数TeacherAnalysis：输入待分析教师，遍历教师线性表检查其存在性，若不存在则报错返回。选择分析方式。分析教学任务按日分配合理性：评判标准为是否做到把所有课尽量安排在白天。三重循环遍历该班级下的各在课时间链表，分别累加每周上课时间为1-4节，5-8节，9-12节的结点数目，若晚课节数大于上午节数或下午节数的一半，则视为本周晚课过多，否则称本周晚课比重合适。分析教学任务分配量：评判标准为大于一定年龄（男女标准不同）的教师是否在某一周中承担了过多的课堂。输入本学期有效星期数，二重循环遍历该教师结点下各在课时间链表，统计每周授课节数。在本程序中，判定女教师年龄小于50岁，男教师年龄小于55岁时，若某周授课超过14节，则判断该周其教学任务过重；当其年龄大于此标准时，若某周授课超过10节，则判断该周其教学任务过重。分析授课楼层合理性：四重循环遍历各班级-课堂链表，当该课堂授课教师为目标教师时，若其年龄高于上述标准且上课教室为四层或五层，则判断该教室对于教师而言过高，教室安排不合理。若遍历完成后仍无类似判定则称该教师任课的教室均较为合理。清屏。流程图如图3-13所示。

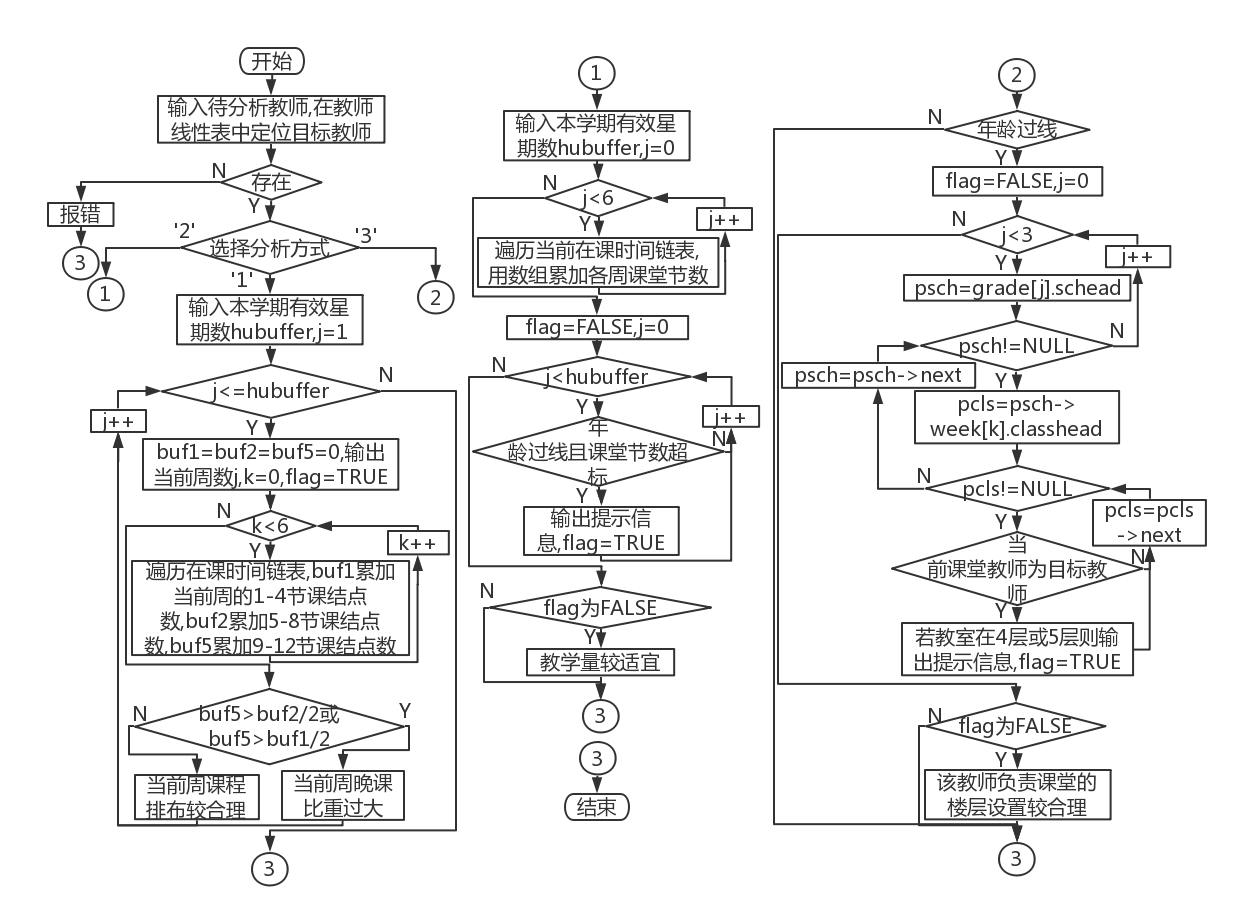


图3-13□教师排课合理性评估流程图

3.2.4 文件存取及其他函数

1. 文件保存函数FileOut：以执行操作的系统时间为基础自动生成文本文件名，与后缀“.txt”拼接形成完整文件名。写入教师线性表长。遍历教师线性表，对每个教师结点，向文件中写入其姓名、性别代码及年龄；遍历六个在课时间链表，写入每个链表的长度，并遍历当前链表写入每个结点的周数及节数值。写入教室线性表长。遍历教室线性表，对每个教室结点，向文件中写入其编号及容量；遍历六个在课时间链表，写入每个链表的长度，并遍历当前链表写入每个结点的周数及节数值。写入课程逆邻接表头长度。对每个课程结点，写入该课程的名称及先修课节点个数。遍历先修课链表，写入各先修课名称。遍历按年级划分的三个班级链表组。对每个链表，写入表长（即班级个数），遍历当前链表。对每个班级结点，写入其班号，人数，已修课个数及各已修课程，在修课个数及各在修课



图3-14□文件保存流程图

程，对其下属的六个日期结构，写入每个结构中在课时间链表长度并遍历当前在课时间链表写入每个结点的周数及节数值；写入课堂链表长度并遍历当前课堂链表，写入每个课堂的课程名称，教室编号，教师姓名，开始节数，结束节数，开始周数数组及结束周数数组。关闭文件，输出提示信息，清屏。流程图如图3-14所示。

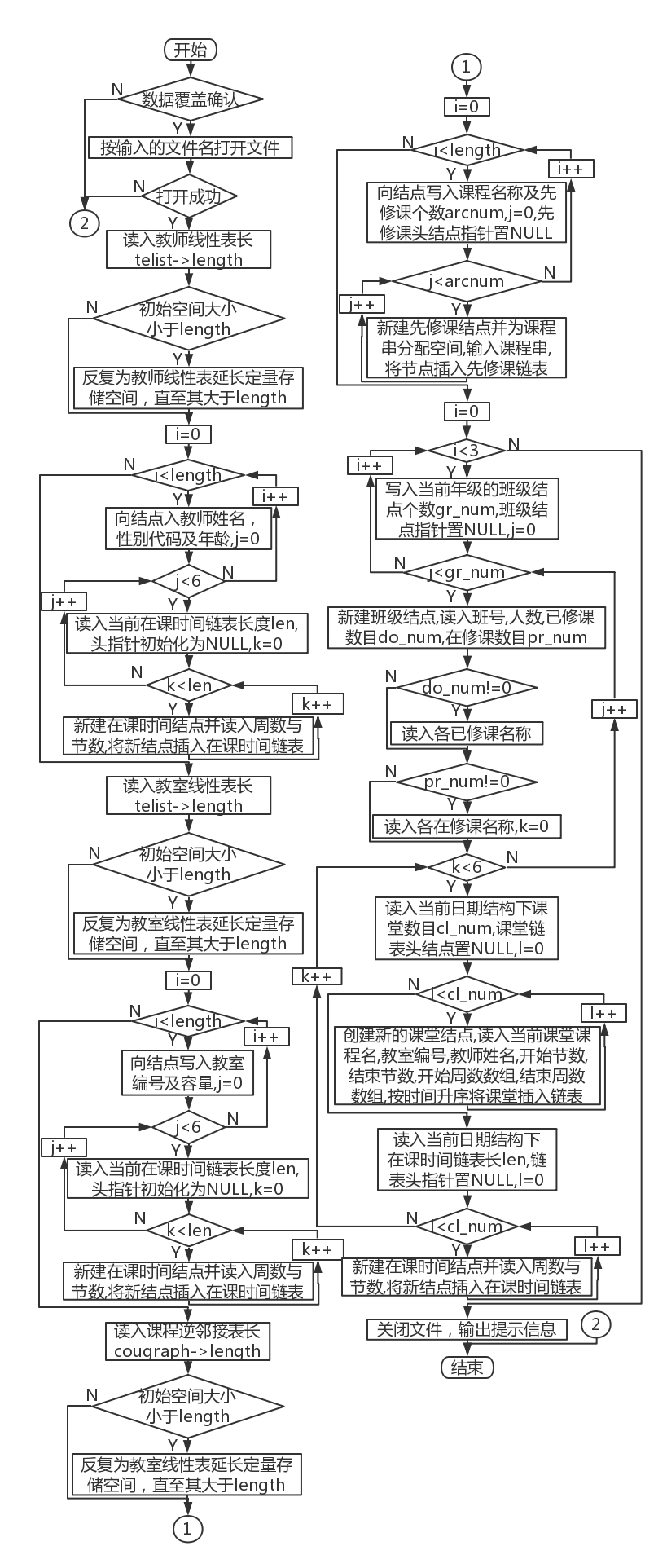


图3-15□文件恢复流程图

2. 文件恢复函数FileIn：首先进行数据覆盖的操作确认。不覆盖（不恢复）则返回主菜单。选择回复时，按输入的文件名打开程序目录下的对应文件，开始向系统中恢复数据。读入教师线性表长。对每个教师结点，读入其姓名、性别代码及年龄；对六个在课时间链表，读入每个链表的长度，循环创建在课时间结点，读入每个结点的周数及节数值并按时间升序插入链表，如此读入与教师线性表长等同数目的教师数据后完成。写入教室线性表长。对每个教室结点，读入其编号及容量；对六个在课时间链表，读入每个链表的长度，循环创建在课时间结点，读入每个结点的周数及节数值并按时间升序插入链表，如此读入与教室线性表长等同数目的教师数据后完成。读入课程逆邻接表头长度。对每个课程结点，读入该课程的名称及先修课节点个数。循环创建先修课结点并为先修课字符串分配空间，读入各先修课名称。如此读入与课程逆邻接表长等同数目的课程数据后完成。读取按年级划分的三个班级链表的表长（即班级个数），循环创建班级结点，读入其班号，人数，已修课个数及各已修课程，在修课个数及各在修课程，对其下属的六个日期结构，读入每个结构中在课时间链表长度，循环创建在课时间结点，并读入每个结点的周数及节数值并按时间升序插入链表；读入课堂链表长度，循环创建课堂结点，读入每个课堂的课程名称，教室编号，教师姓名，开始节数，结束节数，开始周数数组及结束周数数组并按时间升序插入链表。完成上述工作后一个班级结点的读取工作结束，将其按年级插入对应链表。关闭文件，输出提示信息，清屏。流程图如图3-15所示。

3. 系统初始化函数Initial：创建教师线性表，教室线性表，课程逆邻接表的表头结构结点并将长度初始化为0，初始空间大小为LIST\_INIT\_SIZE（宏定义为100）。为教师线性表，教室线性表，课程逆邻接表分配初始大小为LIST\_INIT\_SIZE的空间。将三个班级链表的头指针置为NULL，长度置为0，输出提示信息，清屏。

3.2.5 时空复杂度分析

需要说明的是，以上诸多函数中有很多看起来有着高度的循环嵌套，然而数据的实际特性决定了有很多的循环次数上限均是固定的。比如本学期课程表中共三个年级，因此在遍历以年级为基础构造的班级链表时最多循环三次；一个班在一周中最多上六天课，因此遍历班级结构下以一周中的日期为基础构造的链表组时也只需对六个链表进行遍历。然而，虽然排课评估这一具体问题决定了一个年级中的班级，一个班级所上的课的数目一定是有限的，然而这个限制不甚明确，即是说理论上而言只要数据合法便可以向年级中无限添加班级，向班级下无限添加课堂，假如将排课评估对象扩大到全校甚至多所学校，则上述数据的量确实可以任意增大，因此不适合作为常数级处理。

表3-2□时空复杂度分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 函数名 | 时间复杂度 | 空间复杂度 | 函数名 | 时间复杂度 | 空间复杂度 |
| 1. DelTimeNode | O(n2) | O(1) | 19.AltRoom | O(n2) | O(1) |
| 2. AddTimeNode | O(n) | O(n) | 20.AltCourse | O(n2) | O(1) |
| 3. ClassInput | O(1) | O(1) | 21.AltGroup | O(n) | O(1) |
| 4. AddNewRoom | O(1) | O(1) | 22.AddClass | O(n2) | O(n) |
| 5.AddNewTeacher | O(1) | O(1) | 23.DelClass | O(n2) | O(1) |
| 6.AddNewCourse | O(1) | O(1) | 24.AltClass | O(n2) | O(n) |
| 7.SeekRemClass | O(n) | O(1) | 25.SeekTeacher | O(n) | O(1) |
| 8.DelClassNode | O(n) | O(1) | 26.SeekRoom | O(n) | O(1) |
| 9.TimeInput | O(n) | O(1) | 27.SeekCourse | O(n3) | O(1) |
| 10.AddTeacher | O(n) | O(1) | 28.SeekGroup | O(n2) | O(1) |
| 11.AddRoom | O(n) | O(1) | 29.SeekClass | O(n2) | O(1) |
| 12.AddCourse | O(n2) | O(n) | 30.RoomUsage | O(n2) | O(1) |
| 13.AddGroup | O(n) | O(1) | 31.ClassAnalysis | O(n2) | O(1) |
| 14.DelTeacher | O(n3) | O(1) | 32.TeacherAnalysis | O(n2) | O(1) |
| 15.DelRoom | O(n3) | O(1) | 33.FileOut | O(n2) | O(1) |
| 16.DelCourse | O(n3) | O(1) | 34.FileIn | O(n2) | O(n2) |
| 17.DelGroup | O(n) | O(1) | 35.Initial | O(1) | O(1) |
| 18.AltTeacher | O(n2) | O(1) |  |  |  |

## 4 系统实现与测试

4.1 系统实现

本次实验采用Codeblocks编程软件编写，并用Codeblocks在Windows系统中进行编译运行通过，项目名称是DTCoursework。

4.1.1 头文件及预定义常量说明

1. 头文件

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <wincon.h>

#include <conio.h>

#include <string.h>

#include <io.h>

#include <fcntl.h>

#include <sys\stat.h>

#include <ctype.h>

#include <time.h>

2. 预定义常量

#define SCR\_ROW 50 /\*屏幕行数\*/

#define SCR\_COL 80 /\*屏幕列数\*/

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

4.1.2 数据结构定义

/\*\*

\*在课时间压缩存储链表结点结构

\*/

typedef struct Occupy{

unsigned short week; //上课周数

unsigned short time; //上课节数

struct Occupy \*next;

}Occupy;

/\*\*

\*在课时间压缩存储链表头结构

\*/

typedef struct ochead{

unsigned short len;

Occupy \*head;

}Ochead;

/\*\*

\*教师结点结构

\*/

typedef struct{

char tname[20]; //教师姓名

unsigned short gender, age; //性别：女为0，男为1

Ochead ochead[6]; //教师占用情况,三维:星期-周-课堂

}Teacher;

/\*\*

\*教师线性表头结构

\*/

typedef struct {

Teacher \*elem; //线性表

unsigned short length, listsize; //表长，表容量

}TeList;

/\*\*

\*教室结点结构

\*/

typedef struct{

char code[20]; //教室编号

unsigned short rsize; //教师容量

Ochead ochead[6]; //教室占用情况,三维:星期-周-课堂

}Room;

/\*\*

\*教室线性表头结构

\*/

typedef struct {

Room \*elem; //线性表

unsigned short length, listsize; //表长，表容量

}RoList;

/\*\*

\*先修课程结点结构

\*/

typedef struct CouArc{

char \*prename; //先修课程名称

struct CouArc \*nextarc; //指向下一条弧的指针

}CouArc;

/\*\*

\*课程逆邻接表弧结点结构

\*/

typedef struct {

char name[60]; //课程名称

unsigned short arcnum; //先修课数目

CouArc \*archead; //先修课头指针

}CouHead;

/\*\*

\*课程逆邻接表头结构

\*/

typedef struct { //课程逆邻接表头结构

CouHead \*coulist; //逆邻接表

unsigned short length, listsize; //总课程数目

}CouGraph;

/\*\*

\*课堂结点结构

\*/

typedef struct Class{

char coursename[60], roomcode[20], tname[20]; //课堂名称,教室编号,教师姓名

unsigned short cou\_st, cou\_ed, wk\_st[5], wk\_ed[5]; //开始节数,结束节数,开始周数,持续周数

struct Class \*next;

}Class;

/\*\*

\*班级结点结构

\*/

typedef struct Schedule{

char classname[20]; //班号

unsigned short do\_num, pr\_num, clsize; //已修/在修课程数目

char done[60][60], progress[15][60]; //已修/在修课程

struct weekday{

Class \*classhead; //课堂链表头指针

Ochead ochead; //在课时间链表头结构

unsigned short cl\_num; //课堂数目

}week[6]; //周日无课，每周只用考虑六天

struct Schedule \*next;

}Schedule;

/\*\*

\*班级链表头结构

\*/

typedef struct {

unsigned short gr\_num; //班级数目

Schedule \*schead; //表头指针

}Grade;

4.1.3 函数声明

void InitInterface(void); /\*系统界面初始化\*/

void ClearScreen(void); /\*清屏\*/

void ShowMenu(void); /\*显示菜单栏\*/

void PopMenu(int num); /\*显示下拉菜单\*/

void PopPrompt(int num); /\*显示弹出窗口\*/

void PopUp(SMALL\_RECT \*, WORD, LABEL\_BUNDLE \*, HOT\_AREA \*); /\*弹出窗口屏幕信息维护\*/

void PopOff(void); /\*关闭顶层弹出窗口\*/

void DrawBox(SMALL\_RECT \*parea); /\*绘制边框\*/

void LocSubMenu(int num, SMALL\_RECT \*parea); /\*主菜单下拉菜单定位\*/

void ShowState(void); /\*显示状态栏\*/

void TagMainMenu(int num); /\*标记被选中的主菜单项\*/

void TagSubMenu(int num); /\*标记被选中的子菜单项\*/

int DealConInput(HOT\_AREA \*phot\_area, int \*pihot\_num); /\*控制台输入处理\*/

void SetHotPoint(HOT\_AREA \*phot\_area, int hot\_num); /\*设置热区\*/

BOOL ShowModule(char \*\*pString, int n); /\*弹出信息窗口\*/

BOOL Message(char \*\*); /\*弹出提示窗口\*/

BOOL LoadData(void); /\*数据加载\*/

void RunSys(void); /\*系统功能模块的选择和运行\*/

BOOL ExeFunction(int main\_menu\_num, int sub\_menu\_num); /\*功能模块的调用\*/

void CloseSys(void); /\*退出系统\*/

void ClearMemory(void); /\*清空内存数据\*/

BOOL Initial(void); /\*数据结构初始化\*/

BOOL AddTeacher(void); /\*插入教师\*/

BOOL AddRoom(void); /\*插入教室\*/

BOOL AddCourse(void); /\*插入课程\*/

BOOL AddGroup(void); /\*插入班级\*/

BOOL DelTeacher(void); /\*删除教师\*/

BOOL DelRoom(void); /\*删除教室\*/

BOOL DelCourse(void); /\*删除课程\*/

BOOL DelGroup(void); /\*删除班级\*/

BOOL AltTeacher(void); /\*修改教师\*/

BOOL AltRoom(void); /\*修改教室\*/

BOOL AltCourse(void); /\*修改课程\*/

BOOL AltGroup(void); /\*修改班级\*/

BOOL SeekTeacher(void); /\*查找教师\*/

BOOL SeekRoom(void); /\*查找教室\*/

BOOL SeekCourse(void); /\*查找课程\*/

BOOL SeekGroup(void); /\*查找班级\*/

BOOL MaintainTeacherInfo(void); /\*维护教师信息\*/

BOOL MaintainRoomInfo(void); /\*维护教室信息\*/

BOOL MaintainGroupInfo(void); /\*维护班级信息\*/

BOOL MaintainCourseInfo(void); /\*维护课程信息\*/

BOOL MaintainClassInfo(void); /\*维护课堂信息\*/

BOOL AddClass(void); /\*插入课堂\*/

BOOL DelClass(void); /\*删除课堂\*/

BOOL AltClass(void); /\*修改课堂\*/

BOOL SeekClass(void); /\*查找课堂\*/

BOOL RoomUsage(void); /\*教室利用情况\*/

BOOL TeacherAnalysis(void); /\*教室排课分析\*/

BOOL ClassAnalysis(void); /\*班级排课分析\*/

BOOL FileOut(void); /\*数组备份\*/

BOOL FileIn(void); /\*数组恢复\*/

BOOL ExitSys(void); /\*退出系统\*/

BOOL HelpTopic(void); /\*帮助主题\*/

BOOL AboutDorm(void); /\*关于系统\*/

BOOL DelTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead); /\*删除在课时间结点\*/

BOOL AddTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead); /\*添加在课时间结点\*/

Schedule \*ClassInput(char cbuf[]); /\*班级名称输入\*/

BOOL TimeInput(unsigned short \*cou\_st, unsigned short \*cou\_ed, unsigned short wk\_st[], unsigned short wk\_ed[]); /\*班级名称输入\*/

BOOL AddNewRoom(char cbuf[]); /\*插入新教室\*/

BOOL AddNewTeacher(char cbuf[]); /\*插入新教师\*/

BOOL AddNewCourse(char cbuf[]); /\*插入新课程\*/

BOOL SeekRemClass(Schedule \*ps, char cbuf[]); /\*查找剩余课堂\*/

BOOL DelClassNode(Class \*pc, Schedule \*ps, unsigned short j); /\*删除课堂结点\*/

各功能函数的调用关系已在图2-5中给出，同时每个功能函数都调用了清屏ClearScreen以在返回主菜单时重置界面。此外，部分功能函数还调用了3.2.1节中的基础模块函数，调用关系如图4-1所示。

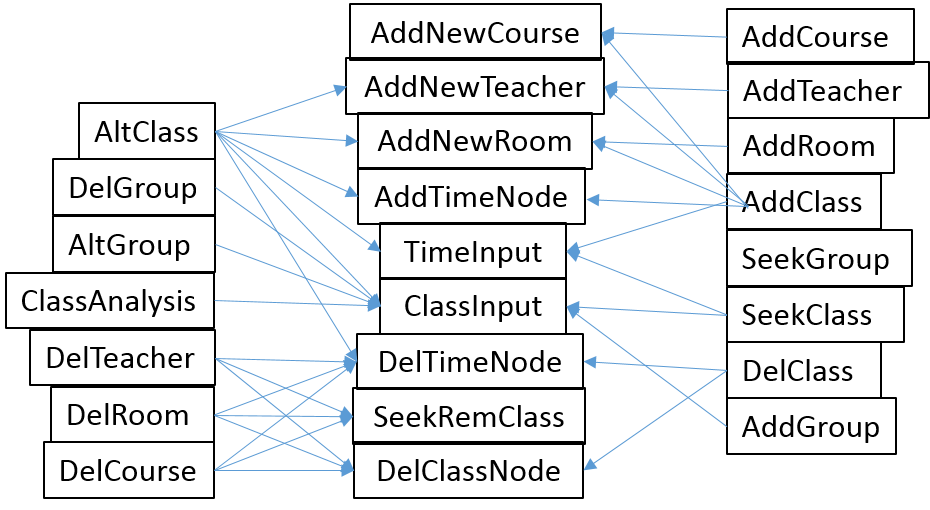


图4-1□函数调用关系图

4.2 系统测试

4.2.1 测试方法及原则

算法的正确性要求对精心选择的有刁难性质的数据得出满足规格说明要求的结果，因此在测试时除正确数据外，还需故意进行各种输入错误，以测试算法在面对边缘条件时能否在不崩溃的基础上返回正确的值。同时，选取数据时还应尽量达到“牵一发而动全身”的效果，即修改或删除数据时让尽可能多的其他数据受到其影响并更改为相应的正确结果，同时用多种方法检查修改数据后系统运行的正确性。

4.2.2 测试计划

表4-1□测试计划大纲

|  |  |  |  |
| --- | --- | --- | --- |
| 待测试功能 | 测试顺序 | 测试输入 | 预计输出 |
| 1.添加课堂 | 1 | 课程名：数据库系统结构  教室：西十二楼S209  教师：潘鹏  5-16周 周一3-4节上课  课堂中共2个班级：  计算机1501；计算机1502 | 导入了新课程，新教室与新教师；新建了两个班级并填入了人数信息 |
| 2.修改课程名称 | 2. | 原名称：数据库系统结构  新名称：数据库系统基础 | 修改成功 |
| 3. 按课程+时间+教室查找课堂 | 3 | 课程：数据库系统基础  教室：西十二楼S209  上课时间：5-16周，周一3-4节 | 班级：计算机1501，计算机1502；教师均为潘鹏 |
| 4.查找某时间段空闲教室 | 4,9,10 | 4&9：第7周周一3-4节（上述课程的上课时间段内）  10：16周周一7-8节（新的上课时间段内） | 4：不出现被占用的教室西十二楼S209；  9：出现被解除占用的西十二楼S209及新教室西十二楼N511；  10：不出现被占用的教室西十二楼N511 |
| 5.按教师查找课堂 | 5,11,28 | 5&11：秦磊华  28：闫帅 | 5：该教师负责的所有课堂；  11：增加了“数据库系统结构”  12：显示修改了时间与上课教室后的班级信息 |
| 6.修改课堂使用的教室 | 6,27 | 6：课程：数据库系统结构  上课日期：周一  上课班级：计算机1501；计算机1502  新教室：西十二楼N511  7：课程：数据库系统结构  上课日期：周一  上课班级：计算机1701；计算机1702；计算机1703；计算机1704；计算机1705  新教室：西十二楼N511 | 6&27：修改成功 |
| 7.修改课堂的任课教师 | 7. | 课程：数据库系统结构  上课日期：周一  上课班级：计算机1501；计算机1502  新教师：秦磊华 | 修改成功 |
| 8.修改课堂的上课时间 | 8 | 课程：数据库系统结构  上课日期：周一  上课班级：计算机1501；计算机1502  新节数：7-8节  新周数：1-3,5-7,9-9,11-13,15-17周 | 修改成功 |
| 9.按班级查找课堂 | 12 | 计算机1502 | 12：该班级其它课堂的信息及“数据库系统结构”课堂修改后的新信息 |
| 10.添加课程的先修课 | 17,19 | 17&19：目标课程：综合英语（二）  目标课程：综合英语（一） | 17：课程清单中无综合英语（一）这门课，添加失败  19：添加成功 |
| 11.分析班级修课合理性 | 16,23 | 16&23：目标班级：计算机1704 | 16：微积分（一）（下）及大学物理（一）两门课不应在本学期修，因为该班级已修课清单中没有先修课微积分（一）（上）  23：上述两门课变为可修，但综合英语（二）变为不可修 |
| 12.插入新课程 | 18 | 新课程：综合英语（一）  先修课个数：0 | 插入成功 |
| 13.为班级添加新的已修课 | 20 | 目标班级：计算机1704  添加课程：微积分（一）（上） | 添加成功 |
| 13.模糊查找课程 | 15,21 | 15：课程子串：大学  21：课程子串：微积分（一） | 15：课程：大学物理（一）  先修课：微积分（一）（上）  后继课：无  21：课程：微积分（一）（上）  先修课：无  后继课：微积分（一）（下）  大学物理（一）  信号与线性系统  课程：微积分（一）（下）  先修课：微积分（一）（上）  后继课：微积分（一）（下）  大学物理（一）  信号与线性系统 |
| 14.按班号查找班级 | 22,43,49 | 22：计算机1704  43&49：计算机1711 | 22：显示人数与在修课程，添加了已修课程“微积分（一）（上）”  43：显示改后的班级人数与已修及在修课程  49：查找失败 |
| 15.按姓名查找教师信息 | 24 | 姓名：闫帅 | 性别：女  年龄：0 |
| 16.修改教师性别 | 25 | 姓名：闫帅 | 性别修改为男 |
| 17.修改教师年龄 | 26,31 | 26：姓名：闫帅  新年龄：57  31：姓名：闫帅  新年龄：61 | 26&31：修改成功 |
| 18.评估教师的教学任务在一天内早晚分布的合理性 | 29 | 姓名：闫帅 | 其它周均合理（无课），第15周白天无课但有晚课，晚课比重过大 |
| 19. 评估教师的上课教师安排的合理性 | 30,32 | 30&32：姓名：闫帅 | 30：该教师无特别楼层要求（评价标准在2.3.3中给出）  32：思政课所在教室N511楼层过高 |
| 20.评估课堂使用班级的合理性（能效比） | 33,34,36,39 | 33：教室：西操场  34：  教室：西十二楼S409  课程：汇编语言程序设计  上课时间：3-8周，周一3-4节 | 33：未找到该教室  1601班人数26；1602班人数27；1603班人数26；校交班人数38  34：教师容量150，座位紧缺，需更换更大的教室  36：教师容量230，教室合适  39：教师容量300，空余座位过多，考虑合并课堂或换至更小的教室 |
| 21.修改教室容量 | 35,38, | 教室：西十二楼S409  35：容量:230  38：容量:300 | 修改成功 |
| 22.修改教室编号 | 37 | 教室：西十二楼S409  新编号：西十二楼N409 | 修改成功 |
| 23.查找目标时间段存在的课堂 | 40 | 11-15周周二第3-4节 | 课程：Verilog语言  教室：南一楼803  教师：卢萍  班级：计算机1601，计算机1602，计算机1603  课程：逻辑与计算机设计基础  教室：西十二楼N203  教师：秦磊华  班级：信息安全1601，信息安全1602，信息安全1603  课程：离散数学（一）  教室：西十二楼N411  教师：张爱华  班级：计算机1701，计算机1702，计算机1703 |
| 24.修改班级的班号 | 41 | 待该班级：计算机1701  新班号：计算机1711 | 修改成功 |
| 25.修改班级人数 | 42 | 待该班级：计算机1711  新人数：32 | 修改成功 |
| 26.修改教师姓名 | 44，45 | 44：待改教师：张爱华  新姓名：秦磊华  45：待改教师：张爱华  新姓名：王立军 | 44：新教师已存在，修改失败  45：修改成功 |
| 27.删除教师 | 46 | 教师姓名：王立军 | 删除成功 |
| 28.删除教室 | 47 | 教室编号：西五楼505 | 删除成功 |
| 29.删除班级 | 48 | 班号：计算机1711 | 删除成功 |
| 30.按容量检索教室 | 50 | 容量区间：100-150 | 显示所有容量符合要求的教室及具体容量 |
| 31.按人数检索班级 | 51 | 人数区间：25-26 | 显示所有人数符合要求的班级的班号，具体人数，已修课清单与在修课清单 |
| 32.按年龄检索教师 | 52 | 年龄区间：34-40 | 显示所有年龄符合要求的教师的姓名，年龄及性别 |
| 33.按性别检索教师 | 53 | 性别：女 | 显示所性别符合要求的教师的姓名及年龄 |
| 34.计算目标时间区间的教室利用率 | 54 | 时间区间：2-3周周一至周二 第4-5节 | 教师利用率为0.072368 |

4.2.3 测试

首先叙述一下常用的软件测试方法，在选择几个主要的功能模块（自行掌握数量，关键要体现你的水平的一些模块）描述测试过程，（1）先明确模块的功能、设计目标等。（2）分析、叙述如何选取测试数据，要求有完整的测试大纲。（3）运行结果（这时可用截图）。（4）分析运行结果、确认程序满足该模块的设计目标。

1. 插入新课堂并按课程+时间+教室查找。

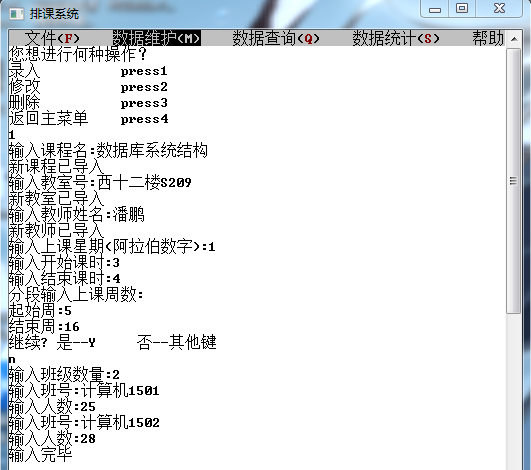
****

图4-2 □插入新课堂

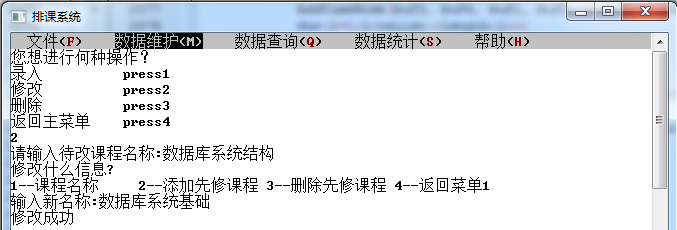


图4-3 □修改课程名称

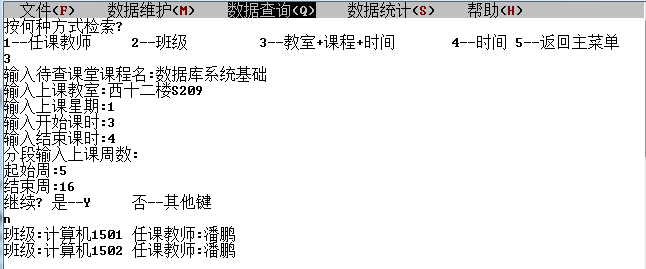


图4-4 □按课程+时间+教室查找新课堂

1. 查找该时间段的空闲教室。

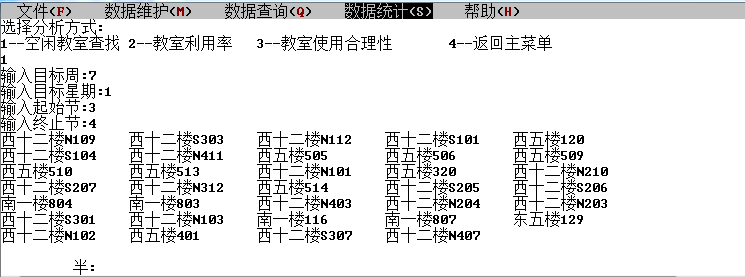


图4-5 □查找空闲教室

1. 按教师查找课堂。

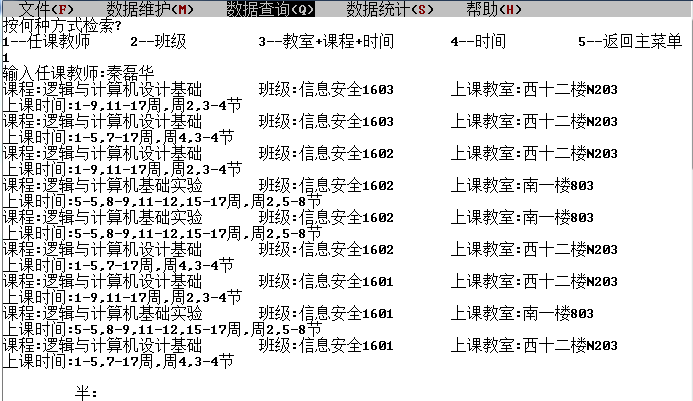


图4-6 □按教师查找课堂

1. 将课程名改回“数据库系统结构”后，修改该课堂上课教室，教师及时间。

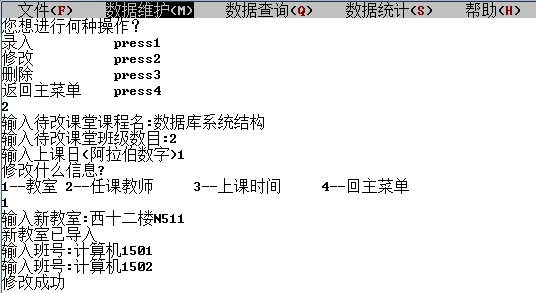


图4-7 □修改课堂使用的教室

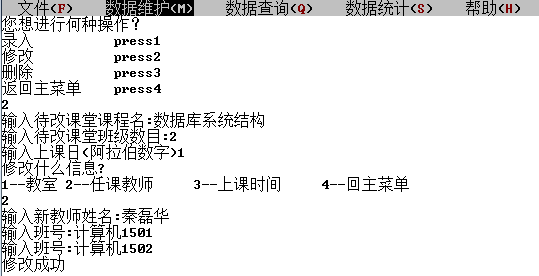


图4-8 □修改课堂任课教师

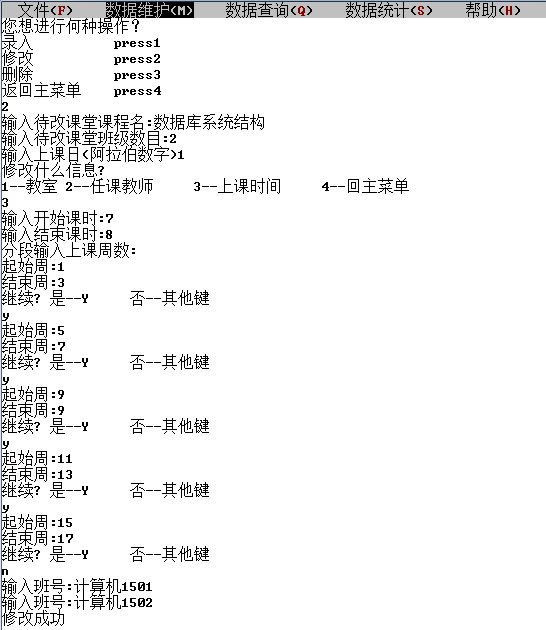


图4-9 □修改课堂上课时间

1. 检查修改效果：按步骤3中的时间段输入相关信息，可以看到出现了修改时间之前被占用的西十二楼S209与修改教室时输入的系统中的新教室西十二楼N511；在输入一新时间段（包含在新的上课时间以内），观察到西十二楼N511消失，证明时间及教室修改成功。再按教师查找课堂，发现搜索修改后的教师时发现目标课堂，证明教师修改成功。



图4-10 □按修改上课时间之前的时间段查找空闲教室



图4-11 □按修改上课时间之后的时间段查找空闲教室

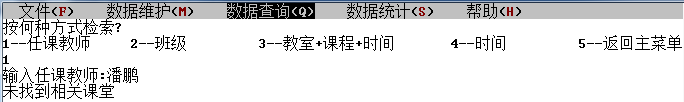


图4-12 □按原任课教师查找课堂



图4-13 □按新任课教师查找课堂

1. 按班级查找课堂。

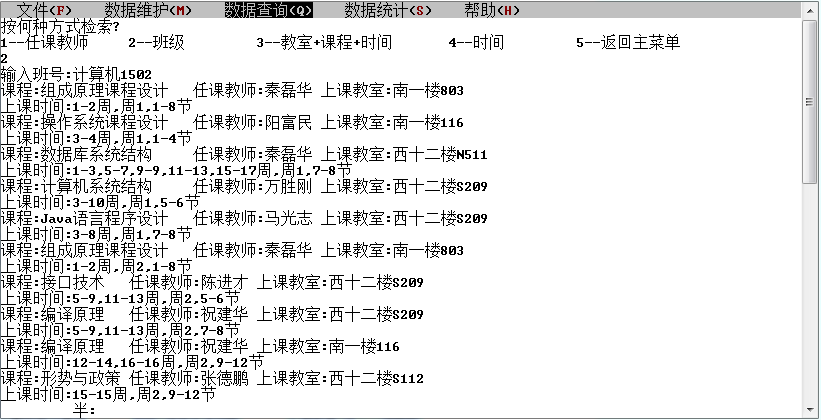


图4-14 □按班级查找课堂

1. 为课程微积分（一）（下）添加先修课并检查大学物理（一）的先修课。

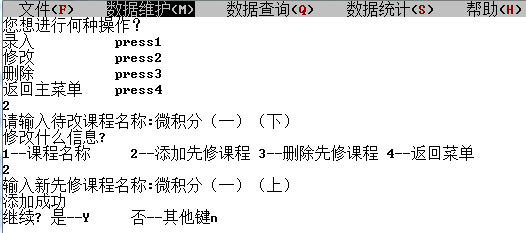


图4-15 □为课程微积分（一）（下）添加先修课微积分（一）（上）



图4-16 □查找课堂大学物理（一）

1. 分析17级班级的修课合理性。

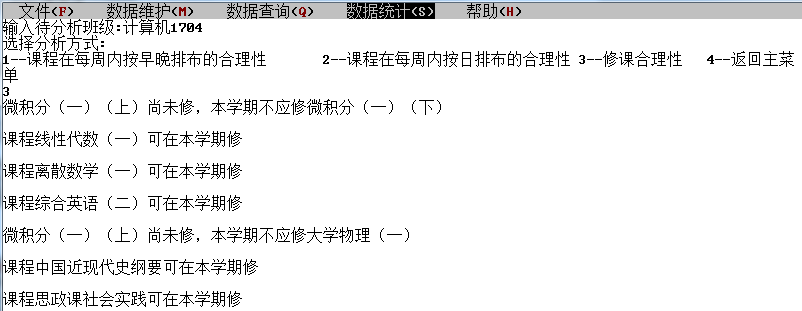


图4-17 □分析1704班修课合理性

1. 添加新课程综合英语（一）并添加其为课程综合英语（二）的先修课，同时为计算机1704班添加已修课程微积分（一）（上）。

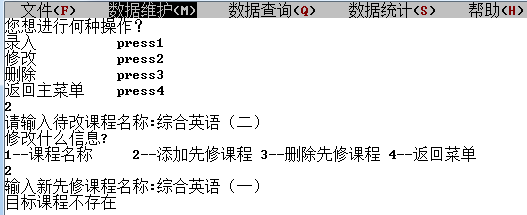


图4-18 □为课程添加课程清单中没有的先修课时报错

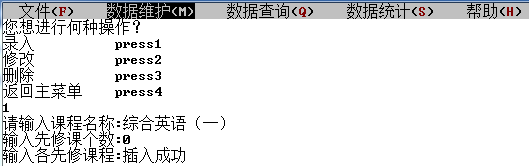


图4-19 □添加新课程

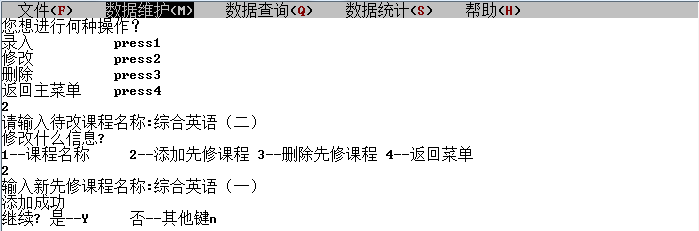


图4-20 □为课程添加新的先修课

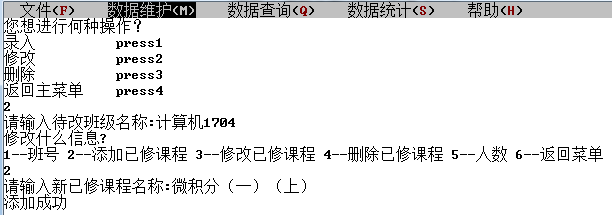


图4-21 □为班级添加新的已修课

1. 上述操作后检查目标班级的修课情况并再次评估该班级修课合理性，发现由于该班级在本学期修综合英语（二）前并未修综合英语（一），因此认定其本学期不应修综合英语（二）；同时微积分（一）（上）已成为其已修课程，因此作为其后继课程的微积分（一）（下）和大学物理（一）均变得可修。

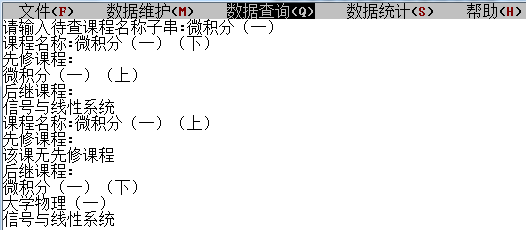


图4-22 □模糊查找课程信息

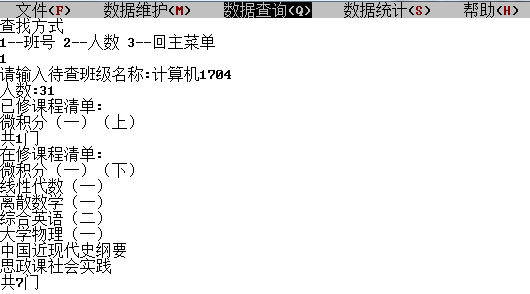


图4-23 □按班号查找班级信息

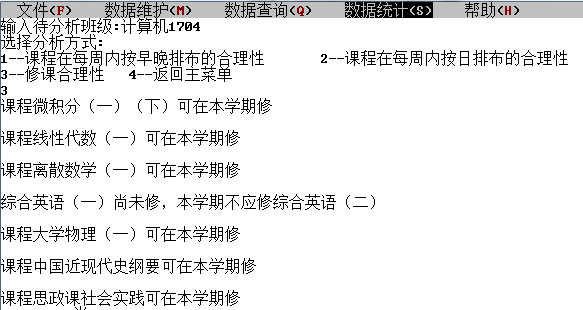


图4-24 □再次评估班级修课合理性

1. 修改教师信息，并评估该教师的教学人物在一天中时间安排的合理性与课堂所在楼层安排的合理性。评价标准参见3.2.3节。

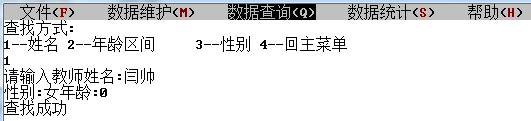


图4-25 □查找教师信息



图4-26 □修改教师性别

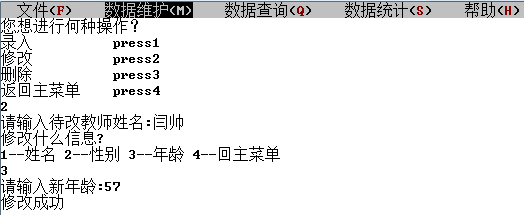


图4-27 □修改教师年龄

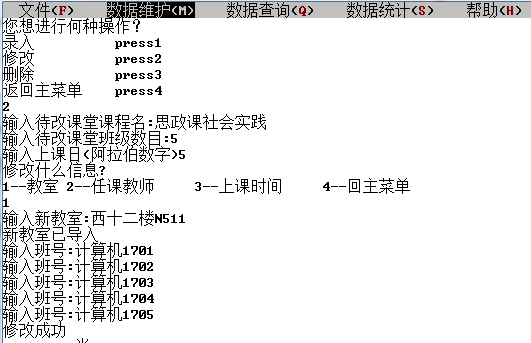


图4-28 □修改课堂所在教室



图4-29 □按教师查找课堂

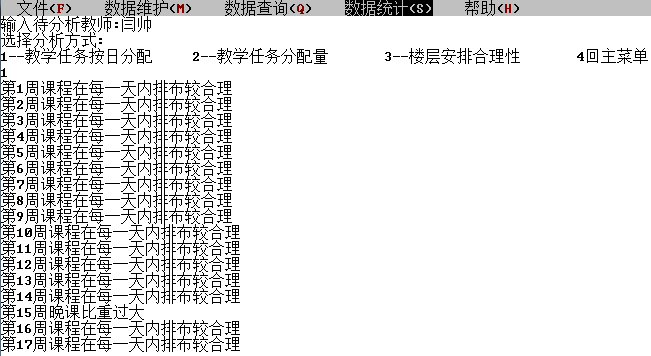


图4-30 □评估该教师的教学任务在一天中按早晚分布的合理性

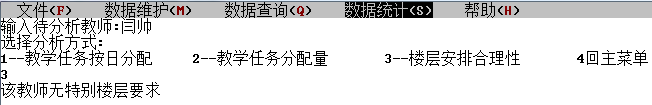


图4-31 □评估该教师的上课教室安排合理性



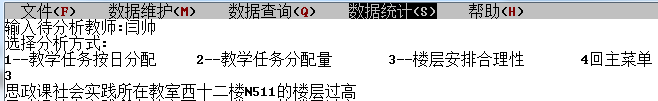
图4-32 □进一步修改教师信息

图4-33 □再次评估该教师的上课教室安排合理性

1. 反复修改目标教师的容量并修改教室编号，观察指定课堂与目标教室的适配性变化与教师编号修改的有效性。评价标准见3.2.3节。

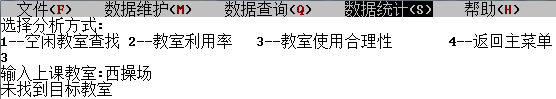


图4-33 □查找教室错误

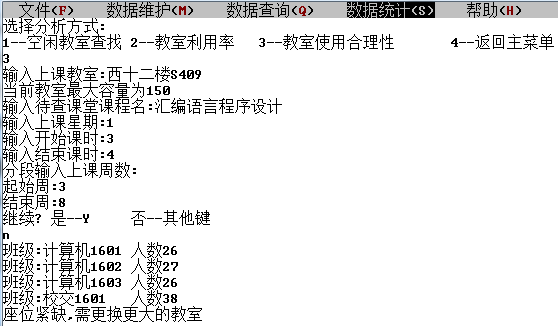


图4-34 □评估教室与课堂的适配性

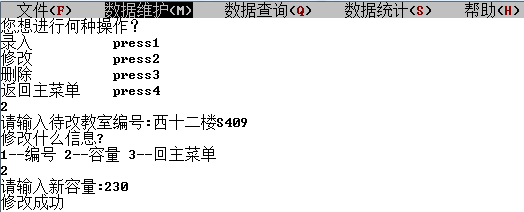


图4-35 □修改教室容量

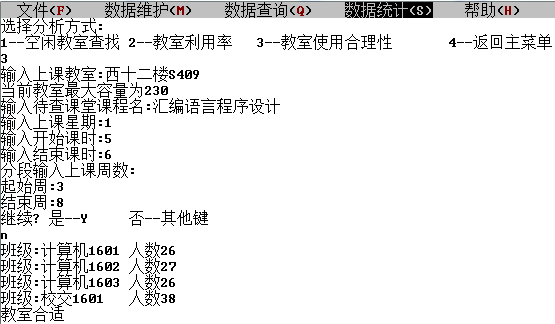


图4-36 □评估教室与课堂的适配性



图4-37 □修改教室编号

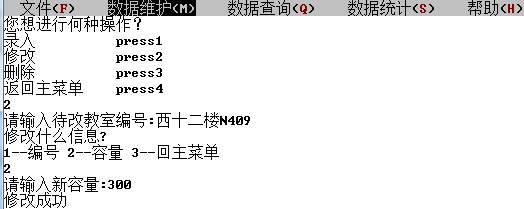


图4-38 □修改教室容量

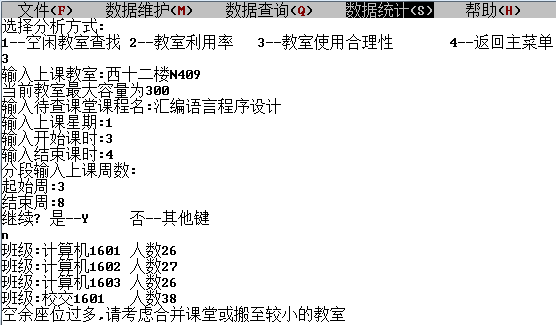


图4-39 □评估教室与课堂的适配性

1. 修改班级班号与人数，通过多个查找与分析函数对目标班级的新信息正确显示与否检查修改的准确性。

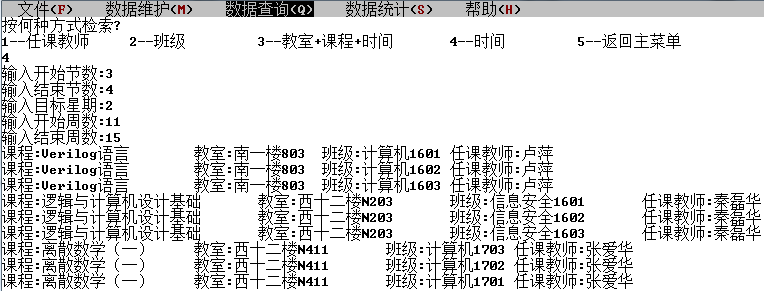


图4-40 □查找目标时间段存在的课堂

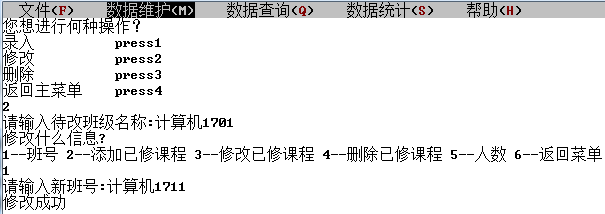


图4-41 □修改班级的班号

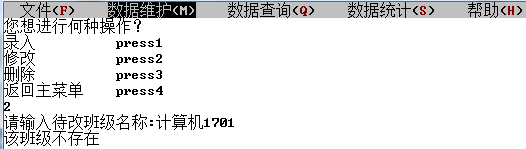


图4-42 □查找原班级失败

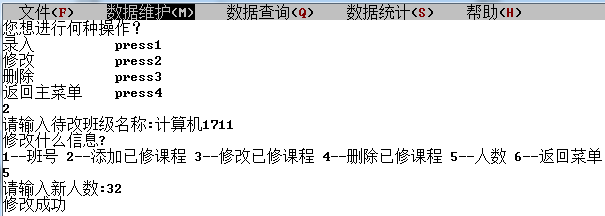


图4-43 □修改班级的人数

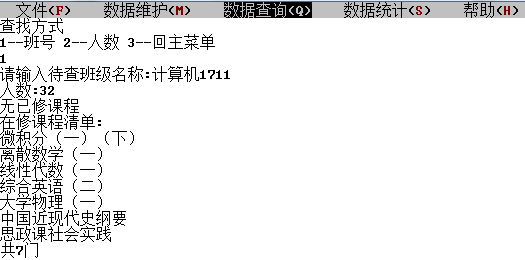


图4-44 □按新班号查找班级信息

1. 修改教师姓名，通过同一种查找方式前后显示信息的对比检查修改有效性。

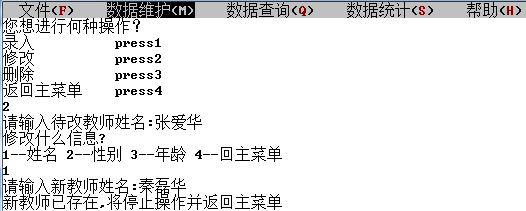


图4-45 □修改教师姓名失败

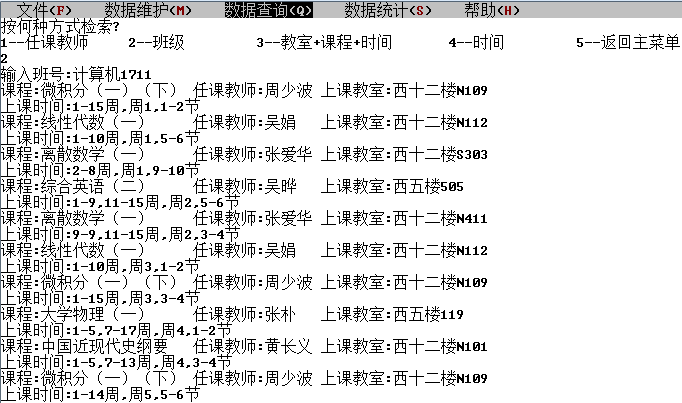


图4-46 □查找班级下属的课堂

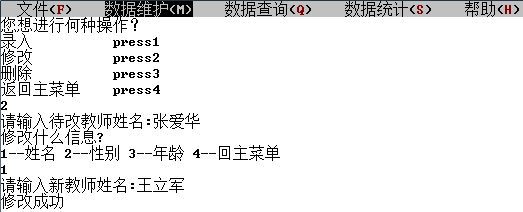


图4-47 □修改教师姓名

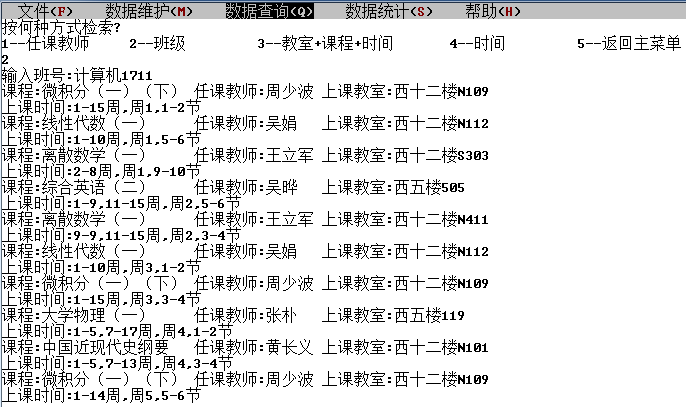


图4-48 □查找相同班级的课堂，重点关注修改后的教师

1. 沿用上一步中查找的信息，删除其中某个教室与教师，仍通过同一种查找方式前后显示信息的对比检查删除有效性。



图4-49 □删除教师



图4-50 □删除教室

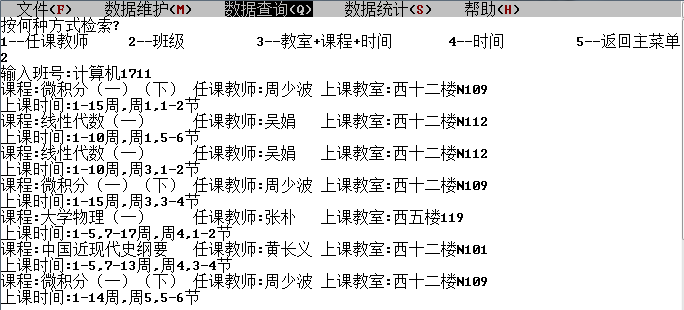


图4-51 □查找相同班级的课堂，重点关注被删除的教师与教室对应的课堂

1. 删除上述班级，再查找该班级并按教室+课程+时间方式查找课堂，均未找到目标班级的信息，证明删除成功。

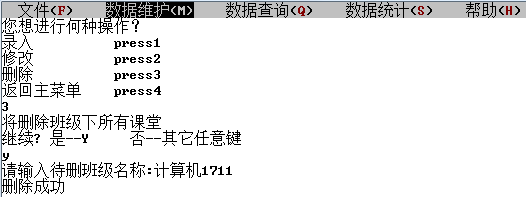


图4-52 □删除班级



图4-53 □按被删除的班级下属的一个课堂查找，未找到原班级

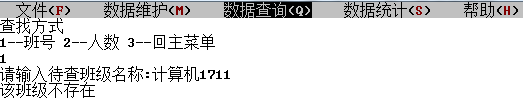


图4-54 □查找班级失败

1. 按教室容量检索教室。

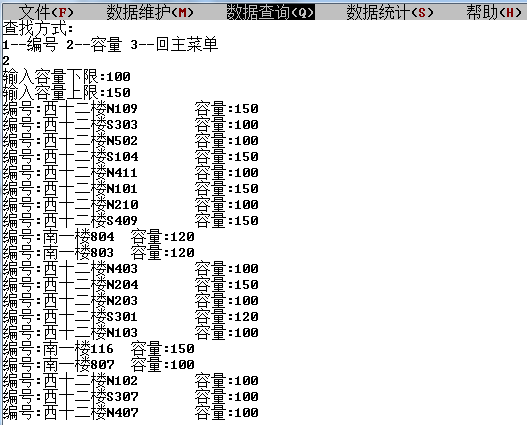


图4-55 □按容量检索教室

1. 按人数检索班级。

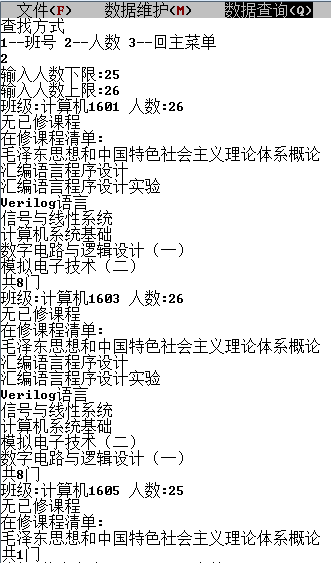
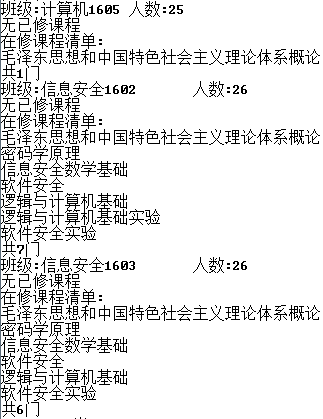
 

图4-56 □按人数检索班级

1. 按年龄检索教师。

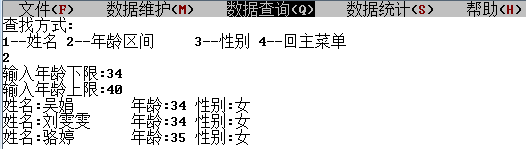


图4-57 □按年龄检索教师

1. 按性别检索教师。

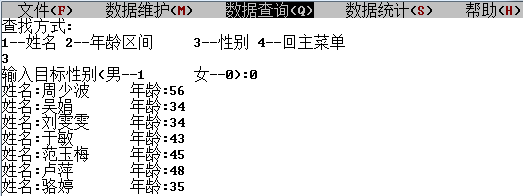


图4-58 □按性别检索教师

1. 分析指定时间段的教师利用率。

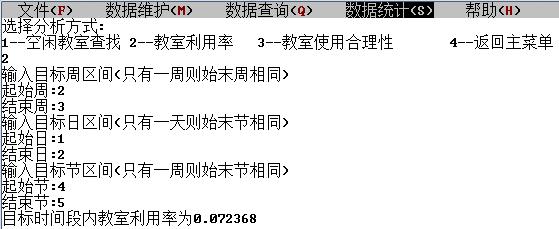


图4-59 □分析目标时间段的教师利用率

1. 插入课堂。

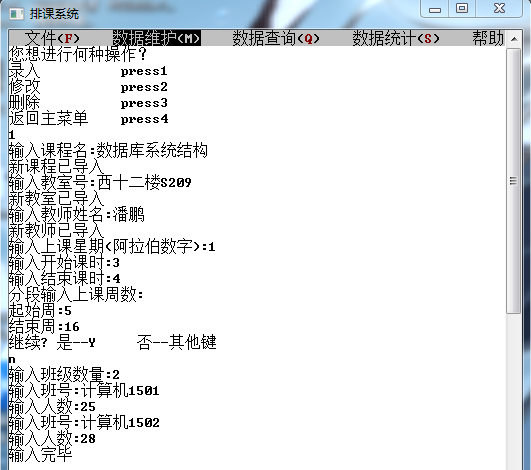


图4-60 □插入课堂

## 5 总结与展望

5.1 全文总结

对自己的工作做个总结，主要工作如下：

（1）分析设计题目、设计内容和设计要求，到图书馆和网上查阅相关资料，制定设计计划，为设计工作做准备。

（2）按照设计要求进行设计工作，设计合适的数据结构和算法，收集并处理相关的资料，编程实现。

（3）按照设计要求测试系统，调试程序BUG，优化、完善系统。

（4）撰写课程设计报告。

5.2 工作展望

在今后的研究中，围绕着如下几个方面开展工作。。。。。。。。

（1）学习中尽量多运用数据结构中所学的内容，在遇到一些数据，多想想用怎样的结构去存储和管理它们。

（2）多实践，将数据结构带到即将要学习的专业课中，不断巩固提升。

（3）多看一些算法，提高自己的编程效率。

（4）设计的系统功能还是比较单一。所以应该完善和扩充系统功能，使系 统能为用户提供更多、更优质的服务。

## 6 体 会

本次我遇到的第一个难点是数据组织结构问题。由于有上学期的实验课打下的基础，我对于各个数据结构的基本操作即使是经过了考试周以及半个假期也还算较为熟悉；真正难住我较长时间的数据关系的问题，尤其是课堂与同其相关的教室、教师及班级的问题。在第一次进入总调试的过程中，偷懒使用高维数组保存时间信息的我又发现对高维数组的存储在排课问题中根本无法强行全部存储，因此我又了一个下午加一个晚上将高位矩阵全部改成了压缩存储二元组表。

文件输入输出之前一直是我的一块短板，而想尽量简化对文件输入输出的操作，结合数据结构，刚开始我选择了二进制存储，要用到fread和fwrite两个函数，同时对于相关的读写方式也要进行复习。这就说明人的记忆能力终究是有限的，要是想熟练掌握并保持高效率的编程能力，需要一个程序员在日常堆砌大量的代码操作。一方面这是为了持续记忆一些硬性的语法知识，另一方面也有助于保持编程手感。同时作为一个程序员，必须要具备强大的自主学习能力，不仅是为了面对陌生的任务快速上手之前没接触过的新知识，更是为了跟上计算机领域的瞬息万变。

然而，如果说在上学期的C语言课设时还能够使用二进制文件保存数据，检查数据工作不算多，单步调试便足以胜任；数据结构的课设数据量可以说翻了几番。在这种情况下，用文本文件保存便成了不二选择，而说实话，习惯了二进制文件存储的我对文本文件存储格式仍然十分陌生，一直以来令我不敢用文本存储的原因就是在字符串与数字交叉保存时不能确定一个变量或串的读取在什么地方停止，数据间的空格何时应添加，何时又不行。但这一次，我逼迫自己用各种各样的变量及字符串进行了混合读写实验，算是补上了C语言时欠下的债；在文本文件中对数据的观察分析也使我在debug时速度有所提升。

在调试程序的过程中，我体会到了作为一名编程者的艰辛。编程者需要谨慎、认真的态度和作风，对每个细微的细节都必须十分注意，不能马虎大意。在开始编写程序的时候，我看到别人的程序功能非常的详细，而且界面非常美观，总是希望自己的程序也非常的完善，但是，发现编一个好的程序不是一蹴而就的事情，需要经过长期的编程经验积累。

在最后将写好的函数主体与给定的文字界面进行对接也是一处大难点。我用了两天时间对文字界面的示例代码进行理解，在进行拼接的同时我也逐渐产生了一些增加用户友好程度，丰富系统功能的想法，由此我也接触到了一些新函数。在这个过程中程序调试着实占用了很大环节，有些是因为变成疏漏，而有一些确实是由于缺乏知识造成的：比如system(“pause”)语句执行完成后鼠标输入莫名其妙失灵的特点我完全没有了解，但在不清楚这一特点之前我只能猜想是之前在鼠标输入相关函数上进行了误操作，为此我基本把全部的文字界面相关函数重新查了一遍。

在反反复复的调试过程中，我终于作出一个简单的排课评估系统，虽然这个程序的功能仍非常基础，未能照顾到排课合理性的方方面面。由于我现在知识浅薄，经验不足和阅历尚浅，对这个排课评估系统设计得还有很多不足之处，比如界面没有那么美观，功能较单一等。我相信在以后的学习中，随着知识的增长和经验的积累，我可以做出更好的成果。

这次课设给我的经验是：今后在编程时，碰上只动脑想不清楚的过程，最好的办法就是在纸上确认其执行情况，这样慎之又慎的行为不仅让诸如交换指针域一类的操作变得更为直观，同时也更有可能在之后的调试过程中节省大量时间。但反过来说，一遍遍确认执行情况也是我面对一段代码不能顺利抽象化的表现，这在一定程度上拉低了我的编程效率。于是话又说了回来，只有日常大量的代码堆砌才能换来任务中的高质量高效率编程。

## 参考文献

[1] 严蔚敏, 吴伟民. 数据结构（C语言版）. 北京: 清华大学出版社,1997

[2] 严蔚敏, 吴伟民, 米宁. 数据结构题集（C语言版）. 北京: 清华大学出版社,1999

[3] 王晓东. 计算机算法设计与分析. 北京: 电子工业出版社, 2007

[4]王秀焕. 基于图论的高校排课系统优化研究.重庆大学硕士学位论文，2012

[5] 洪文，朱广斌. 排课问题及其数学模型. 安徽电力职工大学学报，2002，7（3）：74-77

[6] 陶涛，谢卫星. 课表模型及排课算法应用. 计算机系统应用，2011，20（2）：198-201

[7] 徐晓.基于本体映射和规则推理的排课模型研究. 软件导刊，2009，8（2）：1-4

## 附录

1. head.h

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <wincon.h>

#include <conio.h>

#include <string.h>

#include <io.h>

#include <fcntl.h>

#include <sys\stat.h>

#include <ctype.h>

#include <time.h>

#ifndef TEST\_H

#define TEST\_H

#endif

#ifndef TYPE\_H\_INCLUDED

#define TYPE\_H\_INCLUDED

#define SCR\_ROW 60 /\*屏幕行数\*/

#define SCR\_COL 100 /\*屏幕列数\*/

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

/\*\*

\*在课时间压缩存储链表结点结构

\*/

typedef struct Occupy{

unsigned short week; //上课周数

unsigned short time; //上课节数

struct Occupy \*next;

}Occupy;

/\*\*

\*在课时间压缩存储链表头结构

\*/

typedef struct ochead{

unsigned short len;

Occupy \*head;

}Ochead;

/\*\*

\*教师结点结构

\*/

typedef struct{

char tname[20]; //教师姓名

unsigned short gender, age; //性别：女为0，男为1

Ochead ochead[6]; //教师占用情况,三维:星期-周-课堂

}Teacher;

/\*\*

\*教师线性表头结构

\*/

typedef struct {

Teacher \*elem; //线性表

unsigned short length, listsize; //表长，表容量

}TeList;

/\*\*

\*教室结点结构

\*/

typedef struct{

char code[20]; //教室编号

unsigned short rsize; //教师容量

Ochead ochead[6]; //教室占用情况,三维:星期-周-课堂

}Room;

/\*\*

\*教室线性表头结构

\*/

typedef struct {

Room \*elem; //线性表

unsigned short length, listsize; //表长，表容量

}RoList;

/\*\*

\*课程结点结构

\*/

typedef struct CouArc{

char \*prename; //先修课程名称

struct CouArc \*nextarc; //指向下一条弧的指针

}CouArc; //先修关系结点结构

/\*\*

\*课程逆邻接表弧结点结构

\*/

typedef struct {

char name[60]; //课程名称

unsigned short arcnum;

CouArc \*archead; //先修课头指针

}CouHead;

/\*\*

\*课程逆邻接表头结构

\*/

typedef struct { //课程逆邻接表头结构

CouHead \*coulist; //逆邻接表

unsigned short length, listsize; //总课程数目

}CouGraph;

/\*\*

\*课堂结点结构

\*/

typedef struct Class{

char coursename[60], roomcode[20], tname[20]; //课堂名称,教室编号,教师姓名

unsigned short cou\_st, cou\_ed, wk\_st[5], wk\_ed[5]; //开始节数,结束节数,开始周数,持续周数

struct Class \*next;

}Class;

/\*\*

\*班级结点结构

\*/

typedef struct Schedule{

char classname[20]; //班号

unsigned short do\_num, pr\_num, clsize; //已修/在修课程数目

char done[60][60], progress[15][60]; //已修/在修课程

struct weekday{

Class \*classhead; //课堂链表头指针

Ochead ochead; //在课时间链表头结构

unsigned short cl\_num; //课堂数目

}week[6]; //周日无课，每周只用考虑六天

struct Schedule \*next;

}Schedule;

/\*\*

\*班级链表头结构

\*/

typedef struct {

unsigned short gr\_num; //班级数目

Schedule \*schead; //表头指针

}Grade;

/\*\*

\*屏幕窗口信息链结点结点结构

\*/

typedef struct layer\_node {

char LayerNo; /\*\*< 弹出窗口层数\*/

SMALL\_RECT rcArea; /\*\*< 弹出窗口区域坐标\*/

CHAR\_INFO \*pContent; /\*\*< 弹出窗口区域字符单元原信息存储缓冲区\*/

char \*pScrAtt; /\*\*< 弹出窗口区域字符单元原属性值存储缓冲区\*/

struct layer\_node \*next; /\*\*< 指向下一结点的指针\*/

} LAYER\_NODE;

/\*\*

\*标签束结构

\*/

typedef struct labe1\_bundle {

char \*\*ppLabel; /\*\*< 标签字符串数组首地址\*/

COORD \*pLoc; /\*\*< 标签定位数组首地址\*/

int num; /\*\*< 标签个数\*/

} LABEL\_BUNDLE;

/\*\*

\*热区结构

\*/

typedef struct hot\_area {

SMALL\_RECT \*pArea; /\*\*< 热区定位数组首地址\*/

char \*pSort; /\*\*< 热区类别(按键、文本框、选项框)数组首地址\*/

char \*pTag; /\*\*< 热区序号数组首地址\*/

int num; /\*\*< 热区个数\*/

} HOT\_AREA;

LAYER\_NODE \*gp\_top\_layer = NULL; /\*弹出窗口信息链链头\*/

char \*gp\_sys\_name = "排课系统"; /\*系统名称\*/

char \*ga\_main\_menu[] = {"文件(F)", /\*系统主菜单名\*/

"数据维护(M)",

"数据查询(Q)",

"数据统计(S)",

"帮助(H)"

};

char \*ga\_sub\_menu[] = {"[S] 数据备份", /\*系统子菜单名\*/

"[R] 数据恢复",

"[X] 退出 Alt+X",

"[S] 教师信息",

"[T] 教室信息",

"[P] 班级信息",

"[S] 课程信息",

"[U] 课堂信息",

"[S] 教师信息",

"[T] 教室信息",

"[P] 班级信息",

"[S] 课程信息",

"[U] 课堂信息",

"[G] 教室分析",

"[T] 教师分析",

"[C] 课堂分析",

"[T] 帮助主题",

"",

"[A] 关于..."

};

int ga\_sub\_menu\_count[] = {3, 5, 5, 3, 3}; /\*各主菜单项下子菜单的个数\*/

int gi\_sel\_menu = 1; /\*被选中的主菜单项号,初始为1\*/

int gi\_sel\_sub\_menu = 0; /\*被选中的子菜单项号,初始为0,表示未选中\*/

CHAR\_INFO \*gp\_buff\_menubar\_info = NULL; /\*存放菜单条屏幕区字符信息的缓冲区\*/

CHAR\_INFO \*gp\_buff\_stateBar\_info = NULL; /\*存放状态条屏幕区字符信息的缓冲区\*/

char \*gp\_scr\_att = NULL; /\*存放屏幕上字符单元属性值的缓冲区\*/

char gc\_sys\_state = '\0'; /\*用来保存系统状态的字符\*/

HANDLE gh\_std\_out; /\*标准输出设备句柄\*/

HANDLE gh\_std\_in; /\*标准输入设备句柄\*/

void InitInterface(void); /\*系统界面初始化\*/

void ClearScreen(void); /\*清屏\*/

void ShowMenu(void); /\*显示菜单栏\*/

void PopMenu(int num); /\*显示下拉菜单\*/

void PopPrompt(int num); /\*显示弹出窗口\*/

void PopUp(SMALL\_RECT \*, WORD, LABEL\_BUNDLE \*, HOT\_AREA \*); /\*弹出窗口屏幕信息维护\*/

void PopOff(void); /\*关闭顶层弹出窗口\*/

void DrawBox(SMALL\_RECT \*parea); /\*绘制边框\*/

void LocSubMenu(int num, SMALL\_RECT \*parea); /\*主菜单下拉菜单定位\*/

void ShowState(void); /\*显示状态栏\*/

void TagMainMenu(int num); /\*标记被选中的主菜单项\*/

void TagSubMenu(int num); /\*标记被选中的子菜单项\*/

int DealConInput(HOT\_AREA \*phot\_area, int \*pihot\_num); /\*控制台输入处理\*/

void SetHotPoint(HOT\_AREA \*phot\_area, int hot\_num); /\*设置热区\*/

BOOL ShowModule(char \*\*pString, int n); /\*弹出信息窗口\*/

BOOL Message(char \*\*); /\*弹出提示窗口\*/

BOOL LoadData(void); /\*数据加载\*/

void RunSys(void); /\*系统功能模块的选择和运行\*/

BOOL ExeFunction(int main\_menu\_num, int sub\_menu\_num); /\*功能模块的调用\*/

void CloseSys(void); /\*退出系统\*/

void ClearMemory(void); /\*清空内存数据\*/

BOOL Initial(void); /\*数据结构初始化\*/

BOOL AddTeacher(void); /\*插入教师\*/

BOOL AddRoom(void); /\*插入教室\*/

BOOL AddCourse(void); /\*插入课程\*/

BOOL AddGroup(void); /\*插入班级\*/

BOOL DelTeacher(void); /\*删除教师\*/

BOOL DelRoom(void); /\*删除教室\*/

BOOL DelCourse(void); /\*删除课程\*/

BOOL DelGroup(void); /\*删除班级\*/

BOOL AltTeacher(void); /\*修改教师\*/

BOOL AltRoom(void); /\*修改教室\*/

BOOL AltCourse(void); /\*修改课程\*/

BOOL AltGroup(void); /\*修改班级\*/

BOOL SeekTeacher(void); /\*查找教师\*/

BOOL SeekRoom(void); /\*查找教室\*/

BOOL SeekCourse(void); /\*查找课程\*/

BOOL SeekGroup(void); /\*查找班级\*/

BOOL MaintainTeacherInfo(void); /\*维护教师信息\*/

BOOL MaintainRoomInfo(void); /\*维护教室信息\*/

BOOL MaintainGroupInfo(void); /\*维护班级信息\*/

BOOL MaintainCourseInfo(void); /\*维护课程信息\*/

BOOL MaintainClassInfo(void); /\*维护课堂信息\*/

BOOL AddClass(void); /\*插入课堂\*/

BOOL DelClass(void); /\*删除课堂\*/

BOOL AltClass(void); /\*修改课堂\*/

BOOL SeekClass(void); /\*查找课堂\*/

BOOL RoomUsage(void); /\*教室利用情况\*/

BOOL TeacherAnalysis(void); /\*教室排课分析\*/

BOOL ClassAnalysis(void); /\*班级排课分析\*/

BOOL FileOut(void); /\*数组备份\*/

BOOL FileIn(void); /\*数组恢复\*/

BOOL ExitSys(void); /\*退出系统\*/

BOOL HelpTopic(void); /\*帮助主题\*/

BOOL AboutDorm(void); /\*关于系统\*/

BOOL DelTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead); /\*删除在课时间结点\*/

BOOL AddTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead); /\*添加在课时间结点\*/

Schedule \*ClassInput(char cbuf[]); /\*班级名称输入\*/

BOOL TimeInput(unsigned short \*cou\_st, unsigned short \*cou\_ed, unsigned short wk\_st[], unsigned short wk\_ed[]); /\*班级名称输入\*/

BOOL AddNewRoom(char cbuf[]); /\*插入新教室\*/

BOOL AddNewTeacher(char cbuf[]); /\*插入新教师\*/

BOOL AddNewCourse(char cbuf[]); /\*插入新课程\*/

BOOL SeekRemClass(Schedule \*ps, char cbuf[]); /\*查找剩余课堂\*/

BOOL DelClassNode(Class \*pc, Schedule \*ps, unsigned short j); /\*删除课堂结点\*/

BOOL Add(void);

TeList \*telist=NULL;

RoList \*rolist=NULL;

CouGraph \*cougraph=NULL;

Grade grade[3];

CouArc \*pcarc=NULL, \*qcarc=NULL;

Occupy \*pocc=NULL, \*qocc=NULL;

Schedule \*psch=NULL, \*qsch=NULL;

Class \*pcls=NULL, \*qcls=NULL;

unsigned long ul;

unsigned short choice, hubuffer, buf1, buf2, buf3[5], buf4[5], buf5, buf6;

unsigned char i, j, k, l, m, n, b=' ', count;

char cbuffer[60], cbuf1[60], cbuf2[60];

BOOL flag;

char \*pError[]={"未找到目标信息！返回主菜单？", "确定", "取消"};

char \*pContinue[]={"返回主菜单？", "确定", "取消"};

#endif /\*\*< TYPE\_H\_INCLUDED\*/

1. main.c

#include"head.h"

int main()

{

COORD size = {SCR\_COL, SCR\_ROW}; /\*窗口缓冲区大小\*/

gh\_std\_out = GetStdHandle(STD\_OUTPUT\_HANDLE); /\* 获取标准输出设备句柄\*/

gh\_std\_in = GetStdHandle(STD\_INPUT\_HANDLE); /\* 获取标准输入设备句柄\*/

SetConsoleTitle(gp\_sys\_name); /\*设置窗口标题\*/

SetConsoleScreenBufferSize(gh\_std\_out, size); /\*设置窗口缓冲区大小80\*25\*/

// FileIn();

Initial(); /\*数据加载\*/

InitInterface(); /\*界面初始化\*/

RunSys(); /\*系统功能模块的选择及运行\*/

CloseSys(); /\*退出系统\*/

return 0;

}

/\*\*< 第一部分 文字界面函数 \*/

/\*\*

\* 函数名称: InitInterface

\* 函数功能: 初始化界面.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void InitInterface()

{

ClearScreen(); /\* 清屏\*/

/\*创建弹出窗口信息堆栈，将初始化后的屏幕窗口当作第一层弹出窗口\*/

gp\_scr\_att = (char \*)calloc(SCR\_COL \* SCR\_ROW, sizeof(char));/\*屏幕字符属性\*/

gp\_top\_layer = (LAYER\_NODE \*)malloc(sizeof(LAYER\_NODE));

gp\_top\_layer->LayerNo = 0; /\*弹出窗口的层号为0\*/

gp\_top\_layer->rcArea.Left = 0; /\*弹出窗口的区域为整个屏幕窗口\*/

gp\_top\_layer->rcArea.Top = 0;

gp\_top\_layer->rcArea.Right = SCR\_COL - 1;

gp\_top\_layer->rcArea.Bottom = SCR\_ROW - 1;

gp\_top\_layer->pContent = NULL;

gp\_top\_layer->pScrAtt = gp\_scr\_att;

gp\_top\_layer->next = NULL;

ShowMenu(); /\*显示菜单栏\*/

ShowState(); /\*显示状态栏\*/

return;

}

/\*\*

\* 函数名称: ClearScreen

\* 函数功能: 清除屏幕信息.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void ClearScreen(void)

{

CONSOLE\_SCREEN\_BUFFER\_INFO bInfo;

COORD home = {0, 1}; /\*仅清空从第一行开始的信息，以便被各个功能函数调用\*/

unsigned long size;

GetConsoleScreenBufferInfo( gh\_std\_out, &bInfo );/\*取屏幕缓冲区信息\*/

size = bInfo.dwSize.X \* (bInfo.dwSize.Y); /\*计算屏幕缓冲区字符单元数\*/

/\*将屏幕缓冲区所有单元的字符属性设置为当前屏幕缓冲区字符属性\*/

FillConsoleOutputAttribute(gh\_std\_out, bInfo.wAttributes, size, home, &ul);

/\*将屏幕缓冲区所有单元填充为空格字符\*/

FillConsoleOutputCharacter(gh\_std\_out, ' ', size, home, &ul);

SetConsoleCursorPosition(gh\_std\_out, home); /\*光标置于窗口左上角\*/

return;

}

/\*\*

\* 函数名称: ShowMenu

\* 函数功能: 在屏幕上显示主菜单, 并设置热区, 在主菜单第一项上置选中标记.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void ShowMenu()

{

CONSOLE\_SCREEN\_BUFFER\_INFO bInfo;

CONSOLE\_CURSOR\_INFO lpCur;

COORD size, pos = {0, 0};

int PosA = 2, PosB, i, j;

char ch;

GetConsoleScreenBufferInfo( gh\_std\_out, &bInfo );

size.X = bInfo.dwSize.X; size.Y = 1;

SetConsoleCursorPosition(gh\_std\_out, pos);

for (i=0; i < 5; i++) /\*在窗口第一行第一列处输出主菜单项\*/

{

printf(" %s ", ga\_main\_menu[i]);

}

GetConsoleCursorInfo(gh\_std\_out, &lpCur);

lpCur.bVisible = FALSE;

SetConsoleCursorInfo(gh\_std\_out, &lpCur); /\*隐藏光标\*/

/\*申请动态存储区作为存放菜单条屏幕区字符信息的缓冲区\*/

gp\_buff\_menubar\_info = (CHAR\_INFO \*)malloc(size.X \* size.Y \* sizeof(CHAR\_INFO));

SMALL\_RECT rcMenu ={0, 0, size.X-1, 0} ;

/\*将窗口第一行的内容读入到存放菜单条屏幕区字符信息的缓冲区中\*/

ReadConsoleOutput(gh\_std\_out, gp\_buff\_menubar\_info, size, pos, &rcMenu);

/\*将这一行中英文字母置为红色，其他字符单元置为白底黑字\*/

for (i=0; i<size.X; i++)

{

(gp\_buff\_menubar\_info+i)->Attributes = BACKGROUND\_BLUE | BACKGROUND\_GREEN

| BACKGROUND\_RED;

ch = (char)((gp\_buff\_menubar\_info+i)->Char.AsciiChar);

if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))

{

(gp\_buff\_menubar\_info+i)->Attributes |= FOREGROUND\_RED;

}

}

/\*修改后的菜单条字符信息回写到窗口的第一行\*/

WriteConsoleOutput(gh\_std\_out, gp\_buff\_menubar\_info, size, pos, &rcMenu);

COORD endPos = {0, 1};

SetConsoleCursorPosition(gh\_std\_out, endPos); /\*将光标位置设置在第2行第1列\*/

/\*将菜单项置为热区，热区编号为菜单项号，热区类型为0(按钮型)\*/

i = 0;

do

{

PosB = PosA + strlen(ga\_main\_menu[i]); /\*定位第i+1号菜单项的起止位置\*/

for (j=PosA; j<PosB; j++)

{

gp\_scr\_att[j] |= (i+1) << 2; /\*设置菜单项所在字符单元的属性值\*/

}

PosA = PosB + 4; i++;

} while (i<5);

TagMainMenu(gi\_sel\_menu); /\*在选中主菜单项上做标记，gi\_sel\_menu初值为1\*/

return;

}

/\*\*

\* 函数名称: ShowState

\* 函数功能: 显示状态条.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明: 状态条字符属性为白底黑字, 初始状态无状态信息.

\*/

void ShowState()

{

CONSOLE\_SCREEN\_BUFFER\_INFO bInfo;

COORD size, pos = {0, 0};

int i;

GetConsoleScreenBufferInfo( gh\_std\_out, &bInfo );

size.X = bInfo.dwSize.X; size.Y = 1;

SMALL\_RECT rcMenu ={0, bInfo.dwSize.Y-1, size.X-1, bInfo.dwSize.Y-1};

if (gp\_buff\_stateBar\_info == NULL)

{

gp\_buff\_stateBar\_info = (CHAR\_INFO \*)malloc(size.X \* size.Y \* sizeof(CHAR\_INFO));

ReadConsoleOutput(gh\_std\_out, gp\_buff\_stateBar\_info, size, pos, &rcMenu);

}

for (i=0; i<size.X; i++)

{

(gp\_buff\_stateBar\_info+i)->Attributes = BACKGROUND\_BLUE | BACKGROUND\_GREEN

| BACKGROUND\_RED;

}

WriteConsoleOutput(gh\_std\_out, gp\_buff\_stateBar\_info, size, pos, &rcMenu);

return;

}

/\*\*

\* 函数名称: TagMainMenu

\* 函数功能: 在指定主菜单项上置选中标志.

\* 输入参数: num 选中的主菜单项号

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void TagMainMenu(int num)

{

CONSOLE\_SCREEN\_BUFFER\_INFO bInfo;

COORD size, pos = {0, 0};

int PosA = 2, PosB, i;

char ch;

if (num == 0) /\*num为0时，将会去除主菜单项选中标记\*/

{ PosA = PosB = 0; }

else /\*否则，定位选中主菜单项的起止位置: PosA为起始位置, PosB为截止位置\*/

{

for (i=1; i<num; i++)

{

PosA += strlen(ga\_main\_menu[i-1]) + 4;

}

PosB = PosA + strlen(ga\_main\_menu[num-1]);

}

GetConsoleScreenBufferInfo( gh\_std\_out, &bInfo );

size.X = bInfo.dwSize.X; size.Y = 1;

/\*去除选中菜单项前面的菜单项选中标记\*/

for (i=0; i<PosA; i++)

{

(gp\_buff\_menubar\_info+i)->Attributes = BACKGROUND\_BLUE | BACKGROUND\_GREEN

| BACKGROUND\_RED;

ch = (gp\_buff\_menubar\_info+i)->Char.AsciiChar;

if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))

{

(gp\_buff\_menubar\_info+i)->Attributes |= FOREGROUND\_RED;

}

}

/\*在选中菜单项上做标记，黑底白字\*/

for (i=PosA; i<PosB; i++)

{

(gp\_buff\_menubar\_info+i)->Attributes = FOREGROUND\_BLUE | FOREGROUND\_GREEN

| FOREGROUND\_RED;

}

/\*去除选中菜单项后面的菜单项选中标记\*/

for (i=PosB; i<bInfo.dwSize.X; i++)

{

(gp\_buff\_menubar\_info+i)->Attributes = BACKGROUND\_BLUE | BACKGROUND\_GREEN

| BACKGROUND\_RED;

ch = (char)((gp\_buff\_menubar\_info+i)->Char.AsciiChar);

if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))

{

(gp\_buff\_menubar\_info+i)->Attributes |= FOREGROUND\_RED;

}

}

/\*将做好标记的菜单条信息写到窗口第一行\*/

SMALL\_RECT rcMenu ={0, 0, size.X-1, 0};

WriteConsoleOutput(gh\_std\_out, gp\_buff\_menubar\_info, size, pos, &rcMenu);

return;

}

/\*\*

\* 函数名称: PopMenu

\* 函数功能: 在子菜单并设置热区, 随光标运动置选中标记.

\* 输入参数: num 子菜单从上向下的项数

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void PopMenu(int num)

{

LABEL\_BUNDLE labels;

HOT\_AREA areas;

SMALL\_RECT rcPop;

COORD pos;

WORD att;

char \*pCh;

int i, j, loc = 0;

if (num != gi\_sel\_menu) /\*如果指定主菜单不是已选中菜单\*/

{

if (gp\_top\_layer->LayerNo != 0) /\*如果此前已有子菜单弹出\*/

{

PopOff(); gi\_sel\_sub\_menu = 0;

}

}

else if (gp\_top\_layer->LayerNo != 0) /\*若已弹出该子菜单，则返回\*/

{ return; }

gi\_sel\_menu = num; /\*将选中主菜单项置为指定的主菜单项\*/

TagMainMenu(gi\_sel\_menu); /\*在选中的主菜单项上做标记\*/

LocSubMenu(gi\_sel\_menu, &rcPop); /\*计算弹出子菜单的区域位置, 存放在rcPop中\*/

/\*计算该子菜单中的第一项在子菜单字符串数组中的位置(下标)\*/

for (i=1; i<gi\_sel\_menu; i++)

{

loc += ga\_sub\_menu\_count[i-1];

}

/\*将该组子菜单项项名存入标签束结构变量\*/

labels.ppLabel = ga\_sub\_menu + loc; /\*标签束第一个标签字符串的地址\*/

labels.num = ga\_sub\_menu\_count[gi\_sel\_menu-1]; /\*标签束中标签字符串的个数\*/

COORD aLoc[labels.num];/\*定义一个坐标数组，存放每个标签字符串输出位置的坐标\*/

for (i=0; i<labels.num; i++) /\*确定标签字符串的输出位置，存放在坐标数组中\*/

{

aLoc[i].X = rcPop.Left + 2; aLoc[i].Y = rcPop.Top + i + 1;

}

labels.pLoc = aLoc; /\*使标签束结构变量labels的成员pLoc指向坐标数组的首元素\*/

/\*设置热区信息\*/

areas.num = labels.num; /\*热区的个数，等于标签的个数，即子菜单的项数\*/

SMALL\_RECT aArea[areas.num]; /\*定义数组存放所有热区位置\*/

char aSort[areas.num]; /\*定义数组存放所有热区对应类别\*/

char aTag[areas.num]; /\*定义数组存放每个热区的编号\*/

for (i=0; i<areas.num; i++)

{

aArea[i].Left = rcPop.Left + 2; /\*热区定位\*/

aArea[i].Top = rcPop.Top + i + 1;

aArea[i].Right = rcPop.Right - 2;

aArea[i].Bottom = aArea[i].Top;

aSort[i] = 0; /\*热区类别都为0(按钮型)\*/

aTag[i] = i + 1; /\*热区按顺序编号\*/

}

areas.pArea = aArea;/\*使热区结构变量areas的成员pArea指向热区位置数组首元素\*/

areas.pSort = aSort;/\*使热区结构变量areas的成员pSort指向热区类别数组首元素\*/

areas.pTag = aTag; /\*使热区结构变量areas的成员pTag指向热区编号数组首元素\*/

att = BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED; /\*白底黑字\*/

PopUp(&rcPop, att, &labels, &areas);

DrawBox(&rcPop); /\*给弹出窗口画边框\*/

pos.X = rcPop.Left + 2;

for (pos.Y=rcPop.Top+1; pos.Y<rcPop.Bottom; pos.Y++)

{ /\*此循环用来在空串子菜项位置画线形成分隔，并取消此菜单项的热区属性\*/

pCh = ga\_sub\_menu[loc+pos.Y-rcPop.Top-1];

if (strlen(pCh)==0) /\*串长为0，表明为空串\*/

{ /\*首先画横线\*/

FillConsoleOutputCharacter(gh\_std\_out, '-', rcPop.Right-rcPop.Left-3, pos, &ul);

for (j=rcPop.Left+2; j<rcPop.Right-1; j++)

{ /\*取消该区域字符单元的热区属性\*/

gp\_scr\_att[pos.Y\*SCR\_COL+j] &= 3; /\*按位与的结果保留了低两位\*/

}

}

}

/\*将子菜单项的功能键设为白底红字\*/

pos.X = rcPop.Left + 3;

att = FOREGROUND\_RED | BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED;

for (pos.Y=rcPop.Top+1; pos.Y<rcPop.Bottom; pos.Y++)

{

if (strlen(ga\_sub\_menu[loc+pos.Y-rcPop.Top-1])==0)

{ continue; /\*跳过空串\*/ }

FillConsoleOutputAttribute(gh\_std\_out, att, 1, pos, &ul);

}

return;

}

/\*\*

\* 函数名称: PopUp

\* 函数功能: 在指定区域输出弹出窗口信息, 同时设置热区, 将弹出窗口位置信息入栈.

\* 输入参数: pRc 弹出窗口位置数据存放的地址

\* att 弹出窗口区域字符属性

\* pLabel 弹出窗口中标签束信息存放的地址

pHotArea 弹出窗口中热区信息存放的地址

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void PopUp(SMALL\_RECT \*pRc, WORD att, LABEL\_BUNDLE \*pLabel, HOT\_AREA \*pHotArea)

{

LAYER\_NODE \*nextLayer;

COORD size, pos = {0, 0};

char \*pCh;

int i, j, row;

/\*弹出窗口所在位置字符单元信息入栈\*/

size.X = pRc->Right - pRc->Left + 1; /\*弹出窗口的宽度\*/

size.Y = pRc->Bottom - pRc->Top + 1; /\*弹出窗口的高度\*/

/\*申请存放弹出窗口相关信息的动态存储区\*/

nextLayer = (LAYER\_NODE \*)malloc(sizeof(LAYER\_NODE));

nextLayer->next = gp\_top\_layer;

nextLayer->LayerNo = gp\_top\_layer->LayerNo + 1;

nextLayer->rcArea = \*pRc;

nextLayer->pContent = (CHAR\_INFO \*)malloc(size.X\*size.Y\*sizeof(CHAR\_INFO));

nextLayer->pScrAtt = (char \*)malloc(size.X\*size.Y\*sizeof(char));

pCh = nextLayer->pScrAtt;

/\*将弹出窗口覆盖区域的字符信息保存，用于在关闭弹出窗口时恢复原样\*/

ReadConsoleOutput(gh\_std\_out, nextLayer->pContent, size, pos, pRc);

for (i=pRc->Top; i<=pRc->Bottom; i++)

{ /\*此二重循环将所覆盖字符单元的原先属性值存入动态存储区，便于以后恢复\*/

for (j=pRc->Left; j<=pRc->Right; j++)

{

\*pCh = gp\_scr\_att[i\*SCR\_COL+j]; pCh++;

}

}

gp\_top\_layer = nextLayer; /\*完成弹出窗口相关信息入栈操作\*/

/\*设置弹出窗口区域字符的新属性\*/

pos.X = pRc->Left;

pos.Y = pRc->Top;

for (i=pRc->Top; i<=pRc->Bottom; i++)

{

FillConsoleOutputAttribute(gh\_std\_out, att, size.X, pos, &ul);

pos.Y++;

}

/\*将标签束中的标签字符串在设定的位置输出\*/

for (i=0; i<pLabel->num; i++)

{

pCh = pLabel->ppLabel[i];

if (strlen(pCh) != 0)

{

WriteConsoleOutputCharacter(gh\_std\_out, pCh, strlen(pCh),

pLabel->pLoc[i], &ul);

}

}

/\*设置弹出窗口区域字符单元的新属性\*/

for (i=pRc->Top; i<=pRc->Bottom; i++)

{ /\*此二重循环设置字符单元的层号\*/

for (j=pRc->Left; j<=pRc->Right; j++)

{

gp\_scr\_att[i\*SCR\_COL+j] = gp\_top\_layer->LayerNo;

}

}

for (i=0; i<pHotArea->num; i++)

{ /\*此二重循环设置所有热区中字符单元的热区类型和热区编号\*/

row = pHotArea->pArea[i].Top;

for (j=pHotArea->pArea[i].Left; j<=pHotArea->pArea[i].Right; j++)

{

gp\_scr\_att[row\*SCR\_COL+j] |= (pHotArea->pSort[i] << 6)

| (pHotArea->pTag[i] << 2);

}

}

return;

}

/\*\*

\* 函数名称: PopOff

\* 函数功能: 关闭顶层弹出窗口, 恢复覆盖区域原外观和字符单元原属性.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void PopOff(void)

{

LAYER\_NODE \*nextLayer;

COORD size, pos = {0, 0};

char \*pCh;

int i, j;

if ((gp\_top\_layer->next==NULL) || (gp\_top\_layer->pContent==NULL))

{ /\*栈底存放的主界面屏幕信息，不用关闭\*/

return;

}

nextLayer = gp\_top\_layer->next;

/\*恢复弹出窗口区域原外观\*/

size.X = gp\_top\_layer->rcArea.Right - gp\_top\_layer->rcArea.Left + 1;

size.Y = gp\_top\_layer->rcArea.Bottom - gp\_top\_layer->rcArea.Top + 1;

WriteConsoleOutput(gh\_std\_out, gp\_top\_layer->pContent, size, pos, &(gp\_top\_layer->rcArea));

/\*恢复字符单元原属性\*/

pCh = gp\_top\_layer->pScrAtt;

for (i=gp\_top\_layer->rcArea.Top; i<=gp\_top\_layer->rcArea.Bottom; i++)

{

for (j=gp\_top\_layer->rcArea.Left; j<=gp\_top\_layer->rcArea.Right; j++)

{

gp\_scr\_att[i\*SCR\_COL+j] = \*pCh; pCh++;

}

}

free(gp\_top\_layer->pContent); /\*释放动态存储区\*/

free(gp\_top\_layer->pScrAtt);

free(gp\_top\_layer);

gp\_top\_layer = nextLayer;

gi\_sel\_sub\_menu = 0;

return;

}

/\*\*

\* 函数名称: DrawBox

\* 函数功能: 在指定区域画边框.

\* 输入参数: pRc 存放区域位置信息的地址

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void DrawBox(SMALL\_RECT \*pRc)

{

char chBox[] = {'+','-','|'}; /\*画框用的字符\*/

COORD pos = {pRc->Left, pRc->Top}; /\*定位在区域的左上角\*/

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[0], 1, pos, &ul);/\*画边框左上角\*/

for (pos.X = pRc->Left + 1; pos.X < pRc->Right; pos.X++)

{ /\*此循环画上边框横线\*/

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[1], 1, pos, &ul);

}

pos.X = pRc->Right;

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[0], 1, pos, &ul);/\*画边框右上角\*/

for (pos.Y = pRc->Top+1; pos.Y < pRc->Bottom; pos.Y++)

{ /\*此循环画边框左边线和右边线\*/

pos.X = pRc->Left;

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[2], 1, pos, &ul);

pos.X = pRc->Right;

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[2], 1, pos, &ul);

}

pos.X = pRc->Left; pos.Y = pRc->Bottom;

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[0], 1, pos, &ul);/\*画边框左下角\*/

for (pos.X = pRc->Left + 1; pos.X < pRc->Right; pos.X++)

{ /\*画下边框横线\*/

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[1], 1, pos, &ul);

}

pos.X = pRc->Right;

WriteConsoleOutputCharacter(gh\_std\_out, &chBox[0], 1, pos, &ul);/\*画边框右下角\*/

return;

}

/\*\*

\* 函数名称: TagSubMenu

\* 函数功能: 在指定子菜单项上做选中标记.

\* 输入参数: num 选中的子菜单项号

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void TagSubMenu(int num)

{

SMALL\_RECT rcPop;

COORD pos;

WORD att;

int width;

LocSubMenu(gi\_sel\_menu, &rcPop); /\*计算弹出子菜单的区域位置, 存放在rcPop中\*/

if ((num<1) || (num == gi\_sel\_sub\_menu) || (num>rcPop.Bottom-rcPop.Top-1))

{ /\*如果子菜单项号越界，或该项子菜单已被选中，则返回\*/

return;

}

pos.X = rcPop.Left + 2;

width = rcPop.Right - rcPop.Left - 3;

if (gi\_sel\_sub\_menu != 0) /\*首先取消原选中子菜单项上的标记\*/

{

pos.Y = rcPop.Top + gi\_sel\_sub\_menu;

att = BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED; /\*白底黑字\*/

FillConsoleOutputAttribute(gh\_std\_out, att, width, pos, &ul);

pos.X += 1;

att |= FOREGROUND\_RED;/\*白底红字\*/

FillConsoleOutputAttribute(gh\_std\_out, att, 1, pos, &ul);

}

/\*在制定子菜单项上做选中标记\*/

pos.X = rcPop.Left + 2; pos.Y = rcPop.Top + num;

att = FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED; /\*黑底白字\*/

FillConsoleOutputAttribute(gh\_std\_out, att, width, pos, &ul);

gi\_sel\_sub\_menu = num; /\*修改选中子菜单项号\*/

return;

}

/\*\*

\* 函数名称: LocSubMenu

\* 函数功能: 计算弹出子菜单区域左上角和右下角的位置.

\* 输入参数: num 选中的主菜单项号

\* 输出参数: rc 存放区域位置信息的地址

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void LocSubMenu(int num, SMALL\_RECT \*rc)

{

int i, len, loc = 0;

rc->Top = 1; /\*区域的上边定在第2行，行号为1\*/

rc->Left = 1;

for (i=1; i<num; i++)

{ /\*计算区域左边界位置, 同时计算第一个子菜单项在子菜单字符串数组中的位置\*/

rc->Left += strlen(ga\_main\_menu[i-1]) + 4;

loc += ga\_sub\_menu\_count[i-1];

}

rc->Right = strlen(ga\_sub\_menu[loc]);/\*暂时存放第一个子菜单项字符串长度\*/

for (i=1; i<ga\_sub\_menu\_count[num-1]; i++)

{ /\*查找最长子菜单字符串，将其长度存放在rc->Right\*/

len = strlen(ga\_sub\_menu[loc+i]);

if (rc->Right < len)

{ rc->Right = len; }

}

rc->Right += rc->Left + 3; /\*计算区域的右边界\*/

rc->Bottom = rc->Top + ga\_sub\_menu\_count[num-1] + 1;/\*计算区域下边的行号\*/

if (rc->Right >= SCR\_COL) /\*右边界越界的处理\*/

{

len = rc->Right - SCR\_COL + 1;

rc->Left -= len;

rc->Right = SCR\_COL - 1;

}

return;

}

/\*\*

\* 函数名称: DealInput

\* 函数功能: 在弹出窗口区域设置热区, 等待并响应用户输入.

\* 输入参数: pHotArea 热区参数指针

\* piHot 焦点热区编号的存放地址, 即指向焦点热区编号的指针

\* 输出参数: pHotArea 热区参数

\* piHot 用鼠标单击、按回车或空格时返回当前热区编号

\* 返 回 值: iRet 回车虚拟键码

\*

\* 调用说明:

\*/

int DealInput(HOT\_AREA \*pHotArea, int \*piHot)

{

INPUT\_RECORD inRec;

DWORD res;

COORD pos = {0, 0};

int num, arrow, iRet = 0;

int cNo, cTag, cSort;/\*cNo:层号, cTag:热区编号, cSort: 热区类型\*/

char vkc, asc; /\*vkc:虚拟键代码, asc:字符的ASCII码值\*/

SetHotPoint(pHotArea, \*piHot);

while (TRUE)

{ /\*循环\*/

ReadConsoleInput(gh\_std\_in, &inRec, 1, &res);

if ((inRec.EventType == MOUSE\_EVENT) &&

(inRec.Event.MouseEvent.dwButtonState

== FROM\_LEFT\_1ST\_BUTTON\_PRESSED))

{

pos = inRec.Event.MouseEvent.dwMousePosition;

cNo = gp\_scr\_att[pos.Y \* SCR\_COL + pos.X] & 3;

cTag = (gp\_scr\_att[pos.Y \* SCR\_COL + pos.X] >> 2) & 15;

cSort = (gp\_scr\_att[pos.Y \* SCR\_COL + pos.X] >> 6) & 3;

if ((cNo == gp\_top\_layer->LayerNo) && cTag > 0)

{

\*piHot = cTag;

SetHotPoint(pHotArea, \*piHot);

if (cSort == 0)

{ iRet = 13; break; }

}

}

else if (inRec.EventType == KEY\_EVENT && inRec.Event.KeyEvent.bKeyDown)

{

vkc = inRec.Event.KeyEvent.wVirtualKeyCode;

asc = inRec.Event.KeyEvent.uChar.AsciiChar;;

if (asc == 0)

{

arrow = 0;

switch (vkc)

{ /\*方向键(左、上、右、下)的处理\*/

case 37: arrow = 1; break;

case 38: arrow = 2; break;

case 39: arrow = 3; break;

case 40: arrow = 4; break;

}

if (arrow > 0)

{

num = \*piHot;

while (TRUE)

{

if (arrow < 3) { num--; }

else { num++; }

if ((num < 1) || (num > pHotArea->num) ||

((arrow % 2) && (pHotArea->pArea[num-1].Top

== pHotArea->pArea[\*piHot-1].Top)) || ((!(arrow % 2))

&& (pHotArea->pArea[num-1].Top

!= pHotArea->pArea[\*piHot-1].Top)))

{ break; }

}

if (num > 0 && num <= pHotArea->num)

{

\*piHot = num;

SetHotPoint(pHotArea, \*piHot);

}

}

}

else if (vkc == 27)

{ /\*ESC键\*/ iRet = 27; break; }

else if (vkc == 13 || vkc == 32)

{ /\*回车键或空格表示按下当前按钮\*/ iRet = 13; break; }

}

}

return iRet;

}

/\*\*

\* 函数名称: SetHotPoint

\* 函数功能: 在弹出窗口区域设置热区, 等待并响应用户输入.

\* 输入参数: pHotArea 热区参数指针

\* iHot 焦点热区编号

\* 输出参数: pHotArea 热区参数

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void SetHotPoint(HOT\_AREA \*pHotArea, int iHot)

{

CONSOLE\_CURSOR\_INFO lpCur;

COORD pos = {0, 0};

WORD att1, att2;

int i, width;

att1 = FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED; /\*黑底白字\*/

att2 = BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED; /\*白底黑字\*/

for (i=0; i<pHotArea->num; i++)

{ /\*将按钮类热区置为白底黑字\*/

pos.X = pHotArea->pArea[i].Left;

pos.Y = pHotArea->pArea[i].Top;

width = pHotArea->pArea[i].Right - pHotArea->pArea[i].Left + 1;

if (pHotArea->pSort[i] == 0)

{ /\*热区是按钮类\*/

FillConsoleOutputAttribute(gh\_std\_out, att2, width, pos, &ul);

}

}

pos.X = pHotArea->pArea[iHot-1].Left;

pos.Y = pHotArea->pArea[iHot-1].Top;

width = pHotArea->pArea[iHot-1].Right - pHotArea->pArea[iHot-1].Left + 1;

if (pHotArea->pSort[iHot-1] == 0)

{ /\*被激活热区是按钮类\*/

FillConsoleOutputAttribute(gh\_std\_out, att1, width, pos, &ul);

}

else if (pHotArea->pSort[iHot-1] == 1)

{ /\*被激活热区是文本框类\*/

SetConsoleCursorPosition(gh\_std\_out, pos);

GetConsoleCursorInfo(gh\_std\_out, &lpCur);

lpCur.bVisible = TRUE;

SetConsoleCursorInfo(gh\_std\_out, &lpCur);

}

}

/\*\*

\* 函数名称: ShowModule

\* 函数功能: 从指定数据文件中恢复系统代码表和三类基础数据.

\* 输入参数: pString 文字信息二维数组首地址

\* n 字符串数量

\* 输出参数:

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL ShowModule(char \*\*pString, int n)

{

LABEL\_BUNDLE labels;

HOT\_AREA areas;

BOOL bRet = TRUE;

SMALL\_RECT rcPop;

COORD pos;

WORD att;

int i, maxlen, str\_len, iHot = 1;

for (i=0,maxlen=0; i<n; i++) {

str\_len = strlen(pString[i]);

if (maxlen < str\_len) {

maxlen = str\_len;

}

}

pos.X = maxlen + 16; pos.Y = n + 15;

rcPop.Left = (SCR\_COL - pos.X) / 2 - 10;

rcPop.Right = rcPop.Left + pos.X - 1;

rcPop.Top = (SCR\_ROW - pos.Y) / 2 - 12;

rcPop.Bottom = rcPop.Top + pos.Y - 1;

att = BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED; /\*白底黑字\*/

labels.num = n;

labels.ppLabel = pString;

COORD aLoc[n];

for (i=0; i<n; i++) {

aLoc[i].X = rcPop.Left + 3;

aLoc[i].Y = rcPop.Top + (pos.Y-n)/2 + i;

}

str\_len = strlen(pString[n-1]);

aLoc[n-1].X = rcPop.Left + 3 + (maxlen-str\_len)/2;

aLoc[n-1].Y = aLoc[n-1].Y + 2;

labels.pLoc = aLoc;

areas.num = 1;

SMALL\_RECT aArea[] = {{aLoc[n-1].X, aLoc[n-1].Y,

aLoc[n-1].X + 3, aLoc[n-1].Y}};

char aSort[] = {0}, aTag[] = {1};

areas.pArea = aArea;

areas.pSort = aSort;

areas.pTag = aTag;

PopUp(&rcPop, att, &labels, &areas);

pos.X = rcPop.Left + 1; pos.Y = rcPop.Top + (pos.Y-n)/2 + n;

FillConsoleOutputCharacter(gh\_std\_out, '-', rcPop.Right-rcPop.Left-1, pos, &ul);

DealInput(&areas, &iHot);

PopOff();

return bRet;

}

/\*\*

\* 函数名称: Message

\* 函数功能: 弹出对话框

\* 输入参数: \*\*pCh 对话框信息数组首地址

\* 输出参数: 无

\* 返 回 值: BOOL类型，退出为TRUE，取消为FALSE

\*

\* 调用说明:

\*/

BOOL Message(char \*\*pCh)

{

LABEL\_BUNDLE labels;

HOT\_AREA areas;

SMALL\_RECT rcPop;

COORD pos;

WORD att;

int iHot = 1;

char len[3] = { (char)strlen(pCh[0]), (char)strlen(pCh[1]), (char)strlen(pCh[2])};

char lmax = (len[0] > len[1]+len[2]+4) ? len[0] : len[1]+len[2]+4;

pos.X = lmax + 6;

pos.Y = 7;

char blanc = (pos.X - (len[1]+len[2]+4)) / 2;

rcPop.Left = (SCR\_COL - pos.X) / 2 - 10;

rcPop.Right = rcPop.Left + pos.X - 1;

rcPop.Top = (SCR\_ROW - pos.Y) / 2 - 12;

rcPop.Bottom = rcPop.Top + pos.Y - 1;

att = BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED; /\*白底黑字\*/

labels.num = 3;

labels.ppLabel = pCh;

COORD aLoc[] = {{rcPop.Left+3, rcPop.Top+2},

{rcPop.Left+blanc, rcPop.Top+5},

{rcPop.Right-blanc-(len[2]-1), rcPop.Top+5}};

labels.pLoc = aLoc;

areas.num = 2;

SMALL\_RECT aArea[] = {{rcPop.Left+blanc, rcPop.Top+5,

rcPop.Left+blanc+(len[1]-1), rcPop.Top + 5},

{rcPop.Right-blanc-(len[2]-1), rcPop.Top + 5,

rcPop.Right - blanc, rcPop.Top + 5}};

char aSort[] = {0, 0}, aTag[] = {1, 2};

areas.pArea = aArea;

areas.pSort = aSort;

areas.pTag = aTag;

PopUp(&rcPop, att, &labels, &areas);

pos.X = rcPop.Left + 1; pos.Y = rcPop.Top + 4;

FillConsoleOutputCharacter(gh\_std\_out, '-', rcPop.Right-rcPop.Left-1, pos, &ul);

if (DealInput(&areas, &iHot) == 13 && iHot == 1)

{ return FALSE; }

else{ return TRUE; }

}

/\*\*< 第二部分 系统骨架函数 \*/

/\*\*

\* 函数名称: RunSys

\* 函数功能: 运行系统, 在系统主界面下运行用户所选择的功能模块.

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void RunSys(void)

{

INPUT\_RECORD inRec;

DWORD res;

COORD pos = {0, 0};

BOOL bRet = TRUE;

int i, loc, num;

int cNo, cAtt; /\*cNo:字符单元层号, cAtt:字符单元属性\*/

char vkc, asc; /\*vkc:虚拟键代码, asc:字符的ASCII码值\*/

while (bRet)

{

/\*从控制台输入缓冲区中读一条记录\*/

ReadConsoleInput(gh\_std\_in, &inRec, 1, &res);

if (inRec.EventType == MOUSE\_EVENT) /\*如果记录由鼠标事件产生\*/

{

pos = inRec.Event.MouseEvent.dwMousePosition; /\*获取鼠标坐标位置\*/

cNo = gp\_scr\_att[pos.Y \* SCR\_COL + pos.X] & 3; /\*取该位置的层号\*/

cAtt = gp\_scr\_att[pos.Y \* SCR\_COL + pos.X] >> 2;/\*取该字符单元属性\*/

if (cNo == 0) /\*层号为0，表明该位置未被弹出子菜单覆盖\*/

{

/\* cAtt > 0 表明该位置处于热区(主菜单项字符单元)

\* cAtt != gi\_sel\_menu 表明该位置的主菜单项未被选中

\* gp\_top\_layer->LayerNo > 0 表明当前有子菜单弹出

\*/

if (cAtt > 0 && cAtt != gi\_sel\_menu && gp\_top\_layer->LayerNo > 0)

{

PopOff(); /\*关闭弹出的子菜单\*/

gi\_sel\_sub\_menu = 0; /\*将选中子菜单项的项号置为0\*/

PopMenu(cAtt); /\*弹出鼠标所在主菜单项对应的子菜单\*/

}

}

else if (cAtt > 0) /\*鼠标所在位置为弹出子菜单的菜单项字符单元\*/

{

TagSubMenu(cAtt); /\*在该子菜单项上做选中标记\*/

}

if (inRec.Event.MouseEvent.dwButtonState

== FROM\_LEFT\_1ST\_BUTTON\_PRESSED) /\*如果按下鼠标左边第一键\*/

{

if (cNo == 0) /\*层号为0，表明该位置未被弹出子菜单覆盖\*/

{

if (cAtt > 0) /\*如果该位置处于热区(主菜单项字符单元)\*/

{

PopMenu(cAtt); /\*弹出鼠标所在主菜单项对应的子菜单\*/

}

/\*如果该位置不属于主菜单项字符单元，且有子菜单弹出\*/

else if (gp\_top\_layer->LayerNo > 0)

{

PopOff(); /\*关闭弹出的子菜单\*/

gi\_sel\_sub\_menu = 0; /\*将选中子菜单项的项号置为0\*/

}

}

else /\*层号不为0，表明该位置被弹出子菜单覆盖\*/

{

if (cAtt > 0) /\*如果该位置处于热区(子菜单项字符单元)\*/

{

PopOff(); /\*关闭弹出的子菜单\*/

gi\_sel\_sub\_menu = 0; /\*将选中子菜单项的项号置为0\*/

/\*执行对应功能函数:gi\_sel\_menu主菜单项号,cAtt子菜单项号\*/

bRet = ExeFunction(gi\_sel\_menu, cAtt);

}

}

}

else if (inRec.Event.MouseEvent.dwButtonState

== RIGHTMOST\_BUTTON\_PRESSED) /\*如果按下鼠标右键\*/

{

if (cNo == 0) /\*层号为0，表明该位置未被弹出子菜单覆盖\*/

{

PopOff(); /\*关闭弹出的子菜单\*/

gi\_sel\_sub\_menu = 0; /\*将选中子菜单项的项号置为0\*/

}

}

}

else if (inRec.EventType == KEY\_EVENT /\*如果记录由按键产生\*/

&& inRec.Event.KeyEvent.bKeyDown) /\*且键被按下\*/

{

vkc = inRec.Event.KeyEvent.wVirtualKeyCode; /\*获取按键的虚拟键码\*/

asc = inRec.Event.KeyEvent.uChar.AsciiChar; /\*获取按键的ASC码\*/

/\*系统快捷键的处理\*/

if (vkc == 112) /\*如果按下F1键\*/

{

if (gp\_top\_layer->LayerNo != 0) /\*如果当前有子菜单弹出\*/

{

PopOff(); /\*关闭弹出的子菜单\*/

gi\_sel\_sub\_menu = 0; /\*将选中子菜单项的项号置为0\*/

}

bRet = ExeFunction(5, 1); /\*运行帮助主题功能函数\*/

}

else if (inRec.Event.KeyEvent.dwControlKeyState

& (LEFT\_ALT\_PRESSED | RIGHT\_ALT\_PRESSED))

{ /\*如果按下左或右Alt键\*/

switch (vkc) /\*判断组合键Alt+字母\*/

{

case 88: /\*Alt+X 退出\*/

if (gp\_top\_layer->LayerNo != 0)

{ PopOff(); gi\_sel\_sub\_menu = 0; }

bRet = ExeFunction(1,4); break;

case 70: /\*Alt+F\*/ PopMenu(1); break;

case 77: /\*Alt+M\*/ PopMenu(2); break;

case 81: /\*Alt+Q\*/ PopMenu(3); break;

case 83: /\*Alt+S\*/ PopMenu(4); break;

case 72: /\*Alt+H\*/ PopMenu(5); break;

}

}

else if (asc == 0) /\*其他控制键的处理\*/

{

if (gp\_top\_layer->LayerNo == 0) /\*如果未弹出子菜单\*/

{

switch (vkc) /\*处理方向键(左、右、下)，不响应其他控制键\*/

{

case 37:

gi\_sel\_menu--;

if (gi\_sel\_menu == 0)

{ gi\_sel\_menu = 5; }

TagMainMenu(gi\_sel\_menu); break;

case 39:

gi\_sel\_menu++;

if (gi\_sel\_menu == 6)

{ gi\_sel\_menu = 1; }

TagMainMenu(gi\_sel\_menu); break;

case 40:

PopMenu(gi\_sel\_menu);

TagSubMenu(1); break;

}

}

else /\*已弹出子菜单时\*/

{

for (loc=0,i=1; i<gi\_sel\_menu; i++)

{

loc += ga\_sub\_menu\_count[i-1];

} /\*计算该子菜单中的第一项在子菜单字符串数组中的位置(下标)\*/

switch (vkc) /\*方向键(左、右、上、下)的处理\*/

{

case 37:

gi\_sel\_menu--;

if (gi\_sel\_menu < 1)

{ gi\_sel\_menu = 5; }

TagMainMenu(gi\_sel\_menu);

PopOff(); PopMenu(gi\_sel\_menu);

TagSubMenu(1); break;

case 38:

num = gi\_sel\_sub\_menu - 1;

if (num < 1)

{

num = ga\_sub\_menu\_count[gi\_sel\_menu-1];

}

if (strlen(ga\_sub\_menu[loc+num-1]) == 0)

{ num--; }

TagSubMenu(num); break;

case 39:

gi\_sel\_menu++;

if (gi\_sel\_menu > 5)

{ gi\_sel\_menu = 1; }

TagMainMenu(gi\_sel\_menu);

PopOff(); PopMenu(gi\_sel\_menu);

TagSubMenu(1); break;

case 40:

num = gi\_sel\_sub\_menu + 1;

if (num > ga\_sub\_menu\_count[gi\_sel\_menu-1])

{ num = 1; }

if (strlen(ga\_sub\_menu[loc+num-1]) == 0)

{ num++; }

TagSubMenu(num); break;

}

}

}

else if ((asc-vkc == 0) || (asc-vkc == 32)){ /\*按下普通键\*/

if (gp\_top\_layer->LayerNo == 0) /\*如果未弹出子菜单\*/

{

switch (vkc)

{

case 70: /\*f或F\*/ PopMenu(1); break;

case 77: /\*m或M\*/ PopMenu(2); break;

case 81: /\*q或Q\*/ PopMenu(3); break;

case 83: /\*s或S\*/ PopMenu(4); break;

case 72: /\*h或H\*/ PopMenu(5); break;

case 13: /\*回车\*/

PopMenu(gi\_sel\_menu);

TagSubMenu(1); break;

}

}

else /\*已弹出子菜单时的键盘输入处理\*/

{

if (vkc == 27) /\*如果按下ESC键\*/

{

PopOff(); gi\_sel\_sub\_menu = 0;

}

else if(vkc == 13) /\*如果按下回车键\*/

{

num = gi\_sel\_sub\_menu;

PopOff(); gi\_sel\_sub\_menu = 0;

bRet = ExeFunction(gi\_sel\_menu, num);

}

else /\*其他普通键的处理\*/

{

/\*计算该子菜单中的第一项在子菜单字符串数组中的位置(下标)\*/

for (loc=0,i=1; i<gi\_sel\_menu; i++)

{

loc += ga\_sub\_menu\_count[i-1];

}

/\*依次与当前子菜单中每一项的代表字符进行比较\*/

for (i=loc; i<loc+ga\_sub\_menu\_count[gi\_sel\_menu-1]; i++)

{

if (strlen(ga\_sub\_menu[i])>0 && vkc==ga\_sub\_menu[i][1])

{ /\*如果匹配成功\*/

PopOff(); gi\_sel\_sub\_menu = 0;

bRet = ExeFunction(gi\_sel\_menu, i-loc+1);

}

}

}

}

}

}

}

}

/\*\*

\* 函数名称: ExeFunction

\* 函数功能: 执行由主菜单号和子菜单号确定的功能函数.

\* 输入参数: m 主菜单项号

\* s 子菜单项号

\* 输出参数: 无

\* 返 回 值: BOOL类型, TRUE 或 FALSE

\*

\* 调用说明: 仅在执行函数ExitSys时, 才可能返回FALSE, 其他情况下总是返回TRUE

\*/

BOOL ExeFunction(int m, int s)

{

BOOL bRet = TRUE;

/\*函数指针数组，用来存放所有功能函数的入口地址\*/

BOOL (\*pFunction[ga\_sub\_menu\_count[0]+ga\_sub\_menu\_count[1]+ga\_sub\_menu\_count[2]+ga\_sub\_menu\_count[3]+ga\_sub\_menu\_count[4]])(void);

int i, loc;

/\*将功能函数入口地址存入与功能函数所在主菜单号和子菜单号对应下标的数组元素\*/

pFunction[0] = FileOut;

pFunction[1] = FileIn;

pFunction[2] = ExitSys;

pFunction[3] = MaintainTeacherInfo;

pFunction[4] = MaintainRoomInfo;

pFunction[5] = MaintainGroupInfo;

pFunction[6] = MaintainCourseInfo;

pFunction[7] = MaintainClassInfo;

pFunction[8] = SeekTeacher;

pFunction[9] = SeekRoom;

pFunction[10] = SeekGroup;

pFunction[11] = SeekCourse;

pFunction[12] = SeekClass;

pFunction[13] = RoomUsage;

pFunction[14] = TeacherAnalysis;

pFunction[15] = ClassAnalysis;

pFunction[16] = HelpTopic;

pFunction[17] = NULL;

pFunction[18] = AboutDorm;

for (i=1,loc=0; i<m; i++) /\*根据主菜单号和子菜单号计算对应下标\*/

{ loc += ga\_sub\_menu\_count[i-1]; }

loc += s - 1;

if (pFunction[loc] != NULL)

{

bRet = (\*pFunction[loc])(); /\*用函数指针调用所指向的功能函数\*/

}

return bRet;

}

/\*\*

\* 函数名称: CloseSys

\* 函数功能: 关闭系统.

\* 输入参数: hd 主链头指针

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void CloseSys(void)

{

ClearMemory(); ClearScreen(); /\*清屏\*/

/\*释放存放菜单条、状态条、性别代码和学生类别代码等信息动态存储区\*/

free(gp\_buff\_menubar\_info);

free(gp\_buff\_stateBar\_info);

/\*关闭标准输入和输出设备句柄\*/

CloseHandle(gh\_std\_out); CloseHandle(gh\_std\_in);

/\*将窗口标题栏置为运行结束\*/

SetConsoleTitle("运行结束");

return;

}

/\*\*

\* 函数名称: ExitSys

\* 函数功能: 弹出退出系统的选项

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型，退出为TRUE，取消为FALSE

\*

\* 调用说明:

\*/

BOOL ExitSys(void)

{

char \*pCh[] = {"确认退出系统吗？", "确定", "取消"};

BOOL bRet = Message(pCh);

PopOff();

return bRet;

}

/\*\*

\* 函数名称: ClearMemory

\* 函数功能: 清空内存

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: 无

\*

\* 调用说明:

\*/

void ClearMemory(void){

for(i=0;i<telist->length;i++){

for(j=0;j<6;j++)

for(pocc=telist->elem[i].ochead[j].head;pocc!=NULL;pocc=telist->elem[i].ochead[j].head){

telist->elem[i].ochead[j].head=pocc->next; free(pocc); pocc=NULL;

}

}

free(telist->elem); telist->elem=NULL; telist->length=0;

for(i=0;i<rolist->length;i++){

for(j=0;j<6;j++)

for(pocc=rolist->elem[i].ochead[j].head;pocc!=NULL;pocc=rolist->elem[i].ochead[j].head){

rolist->elem[i].ochead[j].head=pocc->next; free(pocc); pocc=NULL;

}

}

free(rolist->elem); rolist->elem=NULL; rolist->length=0;

for(i=0;i<cougraph->length;i++){ //删除所有先修课节点

for(pcarc=cougraph->coulist[i].archead;pcarc!=NULL;pcarc=cougraph->coulist[i].archead){

cougraph->coulist[i].archead=pcarc->nextarc;

free(pcarc->prename); free(pcarc);

}

}

free(cougraph->coulist); cougraph->coulist=NULL; cougraph->length=0;

for(i=0;i<3;i++){

for(psch=grade[i].schead;psch!=NULL;psch=grade[i].schead){

for(j=0;j<6;j++){

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=psch->week[j].classhead){

psch->week[j].classhead=pcls->next; free(pcls); pcls=NULL;

}

for(pocc=psch->week[j].ochead.head;pocc!=NULL;pocc=psch->week[j].ochead.head){

psch->week[j].ochead.head=pocc->next; free(pocc); pcls=NULL;

}

}

grade[i].schead=psch->next; free(psch); psch=NULL;

}

}

}

BOOL SaveData(void)

{

BOOL bRet = TRUE;

char \*plabel\_name[] = {"主菜单项：文件",

"子菜单项：数据保存",

"确认"

};

ShowModule(plabel\_name, 3);

return bRet;

}

BOOL Initial(void){ //pass

//初始化教室、教师，课程清单结构

telist=(TeList \*)malloc(sizeof(TeList));

rolist=(RoList \*)malloc(sizeof(RoList));

cougraph = (CouGraph \*)malloc(sizeof(CouGraph));

//初始化教室，教师，课程线性查找表

telist->elem = (Teacher \*)malloc(LIST\_INIT\_SIZE\*sizeof(Teacher));

rolist->elem = (Room \*)malloc(LIST\_INIT\_SIZE\*sizeof(Room));

cougraph->coulist = (CouHead \*)malloc(LIST\_INIT\_SIZE\*sizeof(CouHead));

if(!(telist->elem && rolist->elem && cougraph->coulist)) return FALSE;

//各表长度置0，指针置NULL

telist->length = rolist->length = cougraph->length = 0;

telist->listsize = rolist->listsize = cougraph->listsize = LIST\_INIT\_SIZE;

cougraph->length = 0;

grade[0].gr\_num=grade[1].gr\_num=grade[2].gr\_num=0;

grade[0].schead = grade[1].schead = grade[2].schead =NULL;

printf("系统初始化成功 ");

getchar(); ClearScreen();

return TRUE;

}

BOOL AddTeacher(void){ //pass

fflush(stdin);

printf("输入教师姓名:");

scanf("%s", cbuffer); getchar();

for(i=0;i<telist->length;i++)

if(!strcmp(telist->elem[i].tname,cbuffer)){

printf("该教师已存在");

getchar(); ClearScreen(); return TRUE;

}

AddNewTeacher(cbuffer);

printf("输入性别(男--1\t女--0):");

scanf("%hu",&telist->elem[telist->length].gender); getchar();

printf("输入年龄:");

scanf("%hu",&telist->elem[telist->length].age); getchar();

telist->length++; printf("插入成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL AddRoom(void){ //pass

fflush(stdin);

printf("请输入教室编号:");

scanf("%s", cbuffer); getchar();

for(i=0;i<rolist->length;i++)

if(!strcmp(rolist->elem[i].code,cbuffer)){

printf("该教室已存在");

getchar(); ClearScreen(); return TRUE;

}

AddNewRoom(cbuffer);

printf("输入容量:");

scanf("%hu",&rolist->elem[rolist->length].rsize); getchar();

rolist->length++; printf("插入成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL AddCourse(void){ //pass

fflush(stdin);

printf("请输入课程名称:");

scanf("%s", cbuffer); getchar();

for(i=0;i<cougraph->length;i++)

if(!strcmp(cougraph->coulist[i].name,cbuffer)){

printf("该课程已存在");

getchar(); ClearScreen(); return TRUE;

}

AddNewCourse(cbuffer);

printf("输入先修课个数:");

scanf("%hu", &(cougraph->coulist[cougraph->length].arcnum)); getchar();

printf("输入各先修课程:");

for(j=0;j<cougraph->coulist[cougraph->length].arcnum;j++){

scanf("%s", cbuffer); getchar();

for(i=0;i<cougraph->length;i++)

if(!strcmp(cougraph->coulist[i].name, cbuffer)){ //在课程清单中找到该课程

pcarc = (CouArc \*)malloc(sizeof(CouArc));

pcarc->prename=(char \*)malloc(sizeof(cbuffer)\*sizeof(char));

strcpy(pcarc->prename, cbuffer);

pcarc->nextarc = cougraph->coulist[cougraph->length].archead;

cougraph->coulist[cougraph->length].archead = pcarc;

}

else printf("清单中无此课程,放弃对其操作");

}

cougraph->length++; printf("插入成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL AddGroup(void){ //pass

char cbuf3[20];

fflush(stdin);

printf("输入班号:");

psch=ClassInput(cbuf3);

if(psch!=NULL){

printf("该班级已存在");

getchar(); ClearScreen(); return TRUE;

}

//不存在时建立该班级结点

psch=(Schedule \*)malloc(sizeof(Schedule));

strcpy(psch->classname, cbuf3);

printf("输入已修课程数目:");

scanf("%hu",&psch->do\_num); getchar();

psch->pr\_num=0; //在修课程不可手动输入，需配合新课程一并导入

printf("输入各已修课程:\n");

for(i=0;i<psch->do\_num;i++){

scanf("%s",psch->done[i]); getchar();

}

printf("输入人数:");

scanf("%hu",&psch->clsize); getchar();

for(i=0;i<6;i++){ //新班级下课堂结构初始化

psch->week[i].classhead=NULL;

psch->week[i].cl\_num=0;

psch->week[i].ochead.head=NULL;

psch->week[i].ochead.len=0;

}

//按年级将新班级插入基于班级的课程表

if(strstr(cbuf3,"15")){

psch->next=grade[0].schead; grade[0].schead=psch;

grade[0].gr\_num++;

}

else if(strstr(cbuf3,"16")){

psch->next=grade[1].schead; grade[1].schead=psch;

grade[1].gr\_num++;

}

else if(strstr(cbuf3,"17")){

psch->next=grade[2].schead; grade[2].schead=psch;

grade[2].gr\_num++;

}

printf("插入成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL DelTeacher(void){

fflush(stdin);

if(telist->length==0){

printf("教师名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("输入待删教师姓名:");

scanf("%s", cbuffer); getchar();

//定位目标教师，同时确认其存在性

for(i=0;i<telist->length;i++){

if(!strcmp(telist->elem[i].tname,cbuffer)) break;

}

if(i==telist->length){

printf("未找到相应教师");

getchar(); ClearScreen(); return TRUE;

}

//删除目标教师结构下时间占位链表

for(j=0;j<6;j++)

for(pocc=telist->elem[i].ochead[j].head;pocc!=NULL;pocc=telist->elem[i].ochead[j].head){

telist->elem[i].ochead[j].head = pocc->next; free(pocc);

}

//删除所有有相应教师的课堂

for(k=0;k<3;k++)

for(psch=grade[k].schead;psch!=NULL;psch=psch->next)

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(telist->elem[i].tname,pcls->tname)){ //检查教师姓名是否吻合

strcpy(cbuf1, pcls->roomcode);

memcpy(buf3, pcls->wk\_st, sizeof(buf3)); memcpy(buf4, pcls->wk\_ed, sizeof(buf4));

buf5=pcls->cou\_st; buf6=pcls->cou\_ed; //将课程结点信息复制进临时创建的课程结点

//在班级下属数组中删除相关时间段

DelTimeNode(pcls->wk\_st,pcls->wk\_ed,pcls->cou\_st,pcls->cou\_ed,&(psch->week[j].ochead));

SeekRemClass(psch, cbuffer); //检查目标课程是否还在该班级下有课堂

//在教室结构中删除占位时间

for(l=0;l<rolist->length;l++)

if(!strcmp(rolist->elem[l].code, cbuf1)) break;

if(l!=rolist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(rolist->elem[l].ochead[j]));

DelClassNode(pcls, psch, (unsigned short)j); //删除课程结点

}

//删除目标教师

for(j=i;j<telist->length-1;j++)

telist->elem[j]=telist->elem[j+1];

telist->length--; printf("删除成功");

if(telist->length==0) printf("表中已无教师");

getchar(); ClearScreen();

return TRUE;

}

BOOL DelRoom(void){

fflush(stdin);

if(rolist->length==0){

printf("教室名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入待删教室编号:");

scanf("%s", cbuffer); getchar();

//定位目标教室，同时确认其存在性

for(i=0;i<rolist->length;i++){

if(!strcmp(rolist->elem[i].code,cbuffer)) break;

}

if(i==rolist->length){

printf("未找到相应教室");

getchar(); ClearScreen(); return TRUE;

}

//删除所有先修课节点

for(j=0;j<6;j++)

for(pocc=rolist->elem[i].ochead[j].head;pocc!=NULL;pocc=rolist->elem[i].ochead[j].head){

rolist->elem[i].ochead[j].head = pocc->next; free(pocc);

}

//删除所有有相应教室的课堂

for(k=0;k<3;k++)

for(psch=grade[k].schead;psch!=NULL;psch=psch->next)

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(rolist->elem[i].code,pcls->roomcode)){

strcpy(cbuf2, pcls->tname); //将课程结点信息复制进临时创建的课程结点

memcpy(buf3, pcls->wk\_st, sizeof(buf3)); memcpy(buf4, pcls->wk\_ed, sizeof(buf4));

buf5=pcls->cou\_st; buf6=pcls->cou\_ed;

//在班级下属数组中删除相关时间段

DelTimeNode(pcls->wk\_st,pcls->wk\_ed,pcls->cou\_st,pcls->cou\_ed,&(psch->week[j].ochead));

SeekRemClass(psch, cbuffer); //检查目标课程是否还在该班级下有课堂

//在教师结构中删除占位时间

for(l=0;l<telist->length;l++)

if(!strcmp(telist->elem[l].tname, cbuf2)) break;

if(l!=telist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(telist->elem[l].ochead[j]));

DelClassNode(pcls, psch, (unsigned short)j); //删除课程结点

}

for(j=i;j<rolist->length-1;j++) //删除教室结构

rolist->elem[j]=rolist->elem[j+1];

rolist->length--; printf("删除成功");

if(rolist->length==0) printf("表中已无教室");

getchar(); ClearScreen();

return TRUE;

}

BOOL DelCourse(void){

fflush(stdin);

if(cougraph->length==0){

printf("课程名单为空");

getchar(); ClearScreen(); return TRUE;

}

CouArc \*rcarc;

printf("输入待删课程名称:");

scanf("%s", cbuffer); getchar();

//定位目标课程，同时确认其存在性

for(i=0;i<cougraph->length;i++)

if(!strcmp(cougraph->coulist[i].name,cbuffer)) break;

if(i==cougraph->length){

printf("未找到相应课程");

getchar(); ClearScreen(); return TRUE;

}

//若课程清单中某门课的先修课为目标课程，则将目标课程结点删除，同时添加目标课程的先修课为新的先修课

for(j=0; j<cougraph->length; j++){

if(j==i) continue;

qcarc = cougraph->coulist[j].archead;

while(qcarc){ //有弧或边

if(!strcmp(qcarc->prename, cbuffer)){ //该弧被待删结点所指

pcarc = cougraph->coulist[i].archead;

while(pcarc){ //将先修课程的先修课程指向当前课程

rcarc = (CouArc \*)malloc(sizeof(CouArc));

strcpy(rcarc->prename, pcarc->prename);

rcarc->nextarc = cougraph->coulist[j].archead;

cougraph->coulist[j].archead = rcarc;

pcarc = pcarc->nextarc;

}

rcarc = cougraph->coulist[j].archead;

while(rcarc->nextarc!=qcarc) rcarc=rcarc->nextarc; //删除弧头为指定课程的弧

rcarc->nextarc = qcarc->nextarc;

free(qcarc->prename); free(qcarc);

qcarc = rcarc->nextarc;

} } }

cougraph->length--; //课程清单长度自减

for(pcarc=cougraph->coulist[i].archead;pcarc!=NULL;pcarc=cougraph->coulist[i].archead){ //删除目标课程的所有先修课

cougraph->coulist[i].archead=pcarc->nextarc;

free(pcarc->prename); free(pcarc);

}

//删除所有有相应课程的课堂

for(k=0;k<3;k++)

for(psch=grade[k].schead;psch!=NULL;psch=psch->next){

for(j=0;j<psch->pr\_num;j++)

if(!strcmp(cougraph->coulist[i].name,psch->progress[j])) break;

if(j==psch->pr\_num) continue;

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(cougraph->coulist[i].name,pcls->coursename)){

strcpy(cbuf1, pcls->roomcode); strcpy(cbuf2, pcls->tname); //将课程结点信息复制进临时创建的课程结点

memcpy(buf3, pcls->wk\_st, sizeof(buf3)); memcpy(buf4, pcls->wk\_ed, sizeof(buf4));

buf5=pcls->cou\_st; buf6=pcls->cou\_ed;

//在班级下属数组中删除相关时间段

DelTimeNode(pcls->wk\_st,pcls->wk\_ed,pcls->cou\_st,pcls->cou\_ed,&(psch->week[j].ochead));

SeekRemClass(psch, cbuffer); //检查目标课程是否还在该班级下有课堂

//在教室结构中删除目标课程占位时间

for(l=0;l<rolist->length;l++)

if(!strcmp(rolist->elem[l].code, cbuf1)) break;

if(l!=rolist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(rolist->elem[l].ochead[j]));

//在教师结构中删除占位时间

for(l=0;l<telist->length;l++)

if(!strcmp(telist->elem[l].tname, cbuf2)) break;

if(l!=telist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(telist->elem[l].ochead[j]));

//删除目标课程

DelClassNode(pcls, psch, (unsigned short)j); //删除课程结点

}

}

for(j=i; j<cougraph->length-1; j++) //顶点v后面的顶点前移

cougraph->coulist[j] = cougraph->coulist[j+1];

cougraph->length--; printf("删除成功");

if(cougraph->length==0) printf("图中已无顶点");

getchar(); ClearScreen();

return TRUE;

}

BOOL DelGroup(void){ //pass

char cbuf3[20];

fflush(stdin);

Schedule \*psch=NULL, \*qsch=NULL;

Class \*pcls=NULL;

printf("将删除班级下所有课堂\n继续? 是--Y\t否--其它任意键\n");

j=getchar(); getchar();

if(j!='y' && j!='Y'){

ClearScreen(); return TRUE;

}

printf("请输入待删班级名称:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("该班级不存在");

getchar(); ClearScreen(); return TRUE;

}

for(i=0;i<6;i++){

for(pcls=psch->week[i].classhead;pcls!=NULL;pcls=psch->week[i].classhead){ //删除班级下所有课堂

psch->week[i].classhead=pcls->next; free(pcls);

}

for(pocc=psch->week[i].ochead.head;pocc!=NULL;pocc=psch->week[i].ochead.head){ //删除班级下所有时间占位结点

psch->week[i].ochead.head=pocc->next; free(pocc);

}

}

//根据年级找到班级节点并删除

if(strstr(cbuf3,"15")){

if(psch==grade[0].schead){ //目标班级为头结点

grade[0].schead=psch->next; free(psch);

}

else{ //目标班级非头结点

for(qsch=grade[0].schead;qsch->next!=psch;qsch=qsch->next) ;

qsch->next=psch->next; free(psch);

}

grade[0].gr\_num--;

if(grade[0].gr\_num==0) printf("2015级已无班级");

}

else if(strstr(cbuf3,"16")){ //目标班级为头结点

if(psch==grade[1].schead){

grade[1].schead=psch->next; free(psch);

}

else{ //目标班级非头结点

for(qsch=grade[1].schead;qsch->next!=psch;qsch=qsch->next) ;

qsch->next=psch->next; free(psch);

}

grade[1].gr\_num--;

if(grade[1].gr\_num==0) printf("2016级已无班级");

}

else if(strstr(cbuf3,"17")){ //目标班级为头结点

if(psch==grade[2].schead){

grade[2].schead=psch->next; free(psch);

}

else{ //目标班级非头结点

for(qsch=grade[2].schead;qsch->next!=psch;qsch=qsch->next) ;

qsch->next=psch->next; free(psch);

}

grade[2].gr\_num--;

if(grade[2].gr\_num==0) printf("2017级已无班级");

}

printf("删除成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL AltTeacher(void){ //pass

fflush(stdin);

if(telist->length==0){

printf("教师名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入待改教师姓名:");

scanf("%s", cbuffer); getchar();

for(i=0;i<telist->length;i++)

if(!strcmp(telist->elem[i].tname,cbuffer)) break; //i保存目标教师在清单中的位置

if(i==telist->length){

printf("未找到相应教师");

getchar(); ClearScreen(); return TRUE;

}

count=0;

//选择修改方式

do{

if(count++) printf("选项越界,重新输入");

printf("修改什么信息?\n1--姓名\t2--性别\t3--年龄\t4--回主菜单\n");

scanf("%hu", &choice); getchar();

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1:

printf("请输入新教师姓名:");

scanf("%s", cbuffer); getchar();

//验证新名字的合法性

for(j=0;j<telist->length;j++){

if(j==i) continue;

if(!strcmp(telist->elem[j].tname,cbuffer)){

printf("新教师已存在,将停止操作并返回主菜单");

getchar(); ClearScreen(); return TRUE;

}

}

for(k=0;k<3;k++) //修改所有有相应教师的课堂

for(psch=grade[k].schead;psch!=NULL;psch=psch->next)

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next) //找到课堂并修改

if(!strcmp(telist->elem[i].tname,pcls->tname)) strcpy(pcls->tname,cbuffer);

strcpy(telist->elem[i].tname, cbuffer);

printf("修改成功");

break;

case 2:

//性别只有两种可能，互相改动即可

if(telist->elem[i].gender==0) telist->elem[i].gender=1;

else telist->elem[i].gender=0;

printf("性别修改成功,当前性别为");

if(telist->elem[i].gender==0) printf("女");

else printf("男");

break;

case 3:

printf("请输入新年龄:");

scanf("%hu", &telist->elem[i].age); getchar();

printf("修改成功");

case 4:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL AltRoom(void){ //pass

fflush(stdin);

if(rolist->length==0){

printf("教室名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入待改教室编号:");

scanf("%s", cbuffer); getchar();

//查找目标教室

for(i=0;i<rolist->length;i++)

if(!strcmp(rolist->elem[i].code,cbuffer)) break; //i保存目标教室在清单中的位置

if(i==rolist->length){

printf("未找到相应教室");

getchar(); ClearScreen(); return TRUE;

}

//选择修改方式

count = 0;

do{

if(count) printf("选项越界,重新输入");

printf("修改什么信息?\n1--编号\t2--容量\t3--回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==3){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>3);

switch(choice){

case 1:

printf("请输入新教室编号:");

scanf("%s", cbuffer); getchar();

//检查新编号的合法性

for(j=0;j<rolist->length;j++){

if(j==i) continue;

if(!strcmp(rolist->elem[j].code,cbuffer)){

printf("目标教室已存在,将停止操作并返回主菜单");

getchar(); ClearScreen(); return TRUE;

}

}

for(k=0;k<3;k++) //修改所有有相应教师的课堂

for(psch=grade[k].schead;psch!=NULL;psch=psch->next)

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(rolist->elem[i].code,pcls->roomcode)) strcpy(pcls->roomcode,cbuffer);

//修改教室结点

strcpy(rolist->elem[i].code, cbuffer);

printf("修改成功");

break;

case 2:

printf("请输入新容量:");

scanf("%hu", &rolist->elem[i].rsize); getchar();

printf("修改成功");

case 3:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL AltCourse(void){ //pass

fflush(stdin);

if(cougraph->length==0){

printf("课程名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入待改课程名称:");

scanf("%s", cbuffer); getchar();

//查找目标课程

for(i=0;i<cougraph->length;i++)

if(!strcmp(cougraph->coulist[i].name,cbuffer)) break; //i保存目标课程在清单中的位置,之后不可用i做累加器

if(i==cougraph->length){

printf("未找到相应课程");

getchar(); ClearScreen(); return TRUE;

}

//选择修改方式

count = 0;

do{

if(count++) printf("选项越界,重新输入");

printf("修改什么信息?\n1--课程名称\t2--添加先修课程\t3--删除先修课程\t4--返回菜单\n");

scanf("%hu", &choice); getchar();

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1:

printf("输入新名称:");

scanf("%s", cbuffer); getchar();

//检查新课名的合法性

for(j=0;j<cougraph->length;j++){

if(j==i) continue;

if(!strcmp(cougraph->coulist[j].name,cbuffer)){

printf("该课程已存在");

getchar(); ClearScreen(); return TRUE;

}

}

for(j=0;j<cougraph->length;j++){

if(j==i) continue;

for(pcarc=cougraph->coulist[j].archead;pcarc!=NULL;pcarc=pcarc->nextarc) //修改所有对应的先修课弧节点

if(!strcmp(cougraph->coulist[i].name,pcarc->prename))

strcpy(pcarc->prename,cbuffer);

}

for(k=0;k<3;k++) //修改所有有相应课程的课堂

for(psch=grade[k].schead;psch!=NULL;psch=psch->next)

for(j=0;j<6;j++)

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(cougraph->coulist[i].name,pcls->coursename)) strcpy(pcls->coursename,cbuffer);

//修改目标课程结点

strcpy(cougraph->coulist[i].name, cbuffer);

printf("修改成功");

break;

case 2:

do{

printf("输入新先修课程名称:");

scanf("%s", cbuffer); getchar();

for(j=0;j<cougraph->length;j++){ //在课程查找表中查找对应课程

if(j==i) continue;

if(!strcmp(cougraph->coulist[j].name, cbuffer)) break;

}

if(j==cougraph->length){

printf("目标课程不存在");

getchar(); ClearScreen(); return TRUE;

}

pcarc = cougraph->coulist[i].archead;

while(pcarc){ //检查该课程是否为已有的先修课

if(!strcmp(pcarc->prename, cbuffer)){

printf("该先修课程已存在");

getchar(); ClearScreen(); return TRUE;

}

pcarc = pcarc->nextarc;

}

pcarc = (CouArc \*)malloc((sizeof(CouArc))); //插入先修课弧结点

pcarc->prename=(char \*)malloc(sizeof(cbuffer));

strcpy(pcarc->prename, cbuffer);

//新节点插入先修课链表

pcarc->nextarc = cougraph->coulist[i].archead;

cougraph->coulist[i].archead = pcarc;

cougraph->coulist[i].arcnum++;

printf("添加成功\n继续? 是--Y\t否--其他键");

j=getchar(); getchar();

}while(j=='Y' || j=='y');

break;

case 3:

if(cougraph->coulist[i].arcnum==0){

printf("该课程无先修课");

getchar(); ClearScreen(); return TRUE;

}

printf("输入待删先修课程名称:");

scanf("%s", cbuffer); getchar();

pcarc = cougraph->coulist[i].archead;

while(pcarc){ //查找该课程是否为已有的先修课

if(!strcmp(pcarc->prename, cbuffer)) break;

pcarc = pcarc->nextarc;

}

if(!pcarc){

printf("未找到相应课程");

getchar(); ClearScreen(); return TRUE;

}

if(pcarc == cougraph->coulist[i].archead){ //待删结点是头结点

cougraph->coulist[i].archead = pcarc->nextarc;

free(pcarc->prename); free(pcarc);

}

else{ //待删结点非头结点

for(qcarc=cougraph->coulist[i].archead;qcarc->nextarc!=pcarc;qcarc=qcarc->nextarc) ; //q移动至p的前驱

qcarc->nextarc = pcarc->nextarc; free(pcarc->prename); free(pcarc);

}

cougraph->coulist[i].arcnum--;

printf("删除成功");

break;

case 4:

return TRUE;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL AltGroup(void){ //pass

char cbuf3[20];

fflush(stdin);

printf("请输入待改班级名称:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("该班级不存在");

getchar(); ClearScreen(); return TRUE;

}

//选择修改方式

count = 0;

do{

if(count) printf("选项越界,重新输入");

printf("修改什么信息?\n1--班号\t2--添加已修课程\t3--修改已修课程\t4--删除已修课程\t5--人数\t6--返回菜单\n");

scanf("%hu", &hubuffer); getchar(); count++;

if(choice==6){ ClearScreen(); return TRUE; }

}while(hubuffer<1 || hubuffer>6);

switch(hubuffer){

case 1:

flag=TRUE;

do{

printf("请输入新班号:");

scanf("%s", cbuf2); getchar();

//验证新班号合法性

if((strstr(cbuf3,"15") && !strstr(cbuf2,"15")) || (strstr(cbuf3,"16") && !strstr(cbuf2,"16")) || (strstr(cbuf3,"17") && !strstr(cbuf2,"17"))){

printf("不可跨年级修改班号\n"); flag=FALSE;

}

}while(flag==FALSE);

//按班号确定验证头结点

if(strstr(cbuf2,"15")) qsch=grade[0].schead;

else if(strstr(cbuf2,"16")) qsch=grade[1].schead;

else if(strstr(cbuf2,"17")) qsch=grade[2].schead;

//验证新班号存在性

for(;qsch!=NULL;qsch=qsch->next){

if(!strcmp(qsch->classname,cbuf2)){

printf("目标班级已存在");

getchar(); ClearScreen(); return TRUE;

}

}

//修改班号

strcpy(psch->classname, cbuf2);

printf("修改成功");

break;

case 2:

printf("请输入新已修课程名称:");

scanf("%s", cbuffer); getchar();

//验证新课程存在性

for(i=0; i<psch->do\_num; i++)

if(!strcmp(psch->done[i], cbuffer)){

printf("该课程已存在");

getchar(); ClearScreen(); return TRUE;

}

//添加课程

strcpy(psch->done[psch->do\_num], cbuffer);

psch->do\_num++;

printf("添加成功");

break;

case 3:

printf("输入待改已修课程名称:");

scanf("%s", cbuffer); getchar();

//验待改课程存在性

for(i=0; i<psch->do\_num; i++)

if(!strcmp(psch->done[i], cbuffer)) break;

if(j==psch->do\_num){

printf("未找到相应课程");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入新已修课程名称:");

scanf("%s", cbuffer); getchar();

//验证新班号存课程名称合法性

for(j=0;j<psch->do\_num;j++){

if(!strcmp(psch->done[j], cbuffer)){

printf("该课程已存在");

getchar(); ClearScreen(); return TRUE;

}

}

//修改课程

strcpy(psch->done[i], cbuffer);

printf("修改成功");

break;

case 4:

if(psch->do\_num==0){

printf("无已修课程"); break;

}

printf("请输入待删已修课程:");

scanf("%c", cbuffer); getchar();

//验待删课程存在性

for(i=0; i<psch->do\_num; i++)

if(!strcmp(psch->done[i], cbuffer)) break;

if(i==psch->do\_num){

printf("未找到相应课程");

getchar(); ClearScreen(); return TRUE;

}

//删除课程

for(j=i;j<psch->do\_num-1;j++)

strcpy(psch->done[j],psch->done[j+1]);

psch->do\_num--;

if(psch->do\_num==0) printf("表中已无已修课程");

printf("删除成功"); getchar();

break;

case 5:

printf("请输入新人数:");

scanf("%hu", &(psch->clsize)); getchar();

printf("修改成功");

/\*屏蔽修改在修课程的接口

case 5:

printf("请输入新在修课程名称:");

scanf("%s", cbuffer); getchar();

for(i=0; i<psch->pr\_num; i++){

if(!strcmp(psch->progress[i], cbuffer)){

printf("该课程已存在"); return FALSE;

}

}

strcpy(psch->progress[psch->pr\_num], cbuffer);

psch->pr\_num++;

printf("添加成功");

break;

case 6:

if(psch->pr\_num==0){

printf("已无在修课程"); break;

}

printf("请输入待改在修课程名称:");

scanf("%s", cbuffer); getchar();

for(i=0; i<psch->pr\_num; i++)

if(!strcmp(psch->progress[i],cbuffer)) break;

if(i==psch->pr\_num){

printf("未找到相应课程"); return FALSE;

}

printf("请输入新在修课程名称:");

scanf("%s", cbuffer); getchar();

for(j=0;j<psch->pr\_num;j++){

if(!strcmp(psch->progress[j], cbuffer)){

printf("该课程已存在"); return FALSE;

}

}

strcpy(psch->progress[i], cbuffer);

printf("修改成功");

break;

case 7:

if(psch->pr\_num==0){

printf("无在修课程"); break;

}

printf("请输入待删在修课程名称:");

scanf("%s", cbuffer); getchar();

for(i=0; i<psch->pr\_num; i++)

if(!strcmp(psch->progress[i],cbuffer)) break;

if(i==psch->pr\_num){

printf("未找到相应课程"); return FALSE;

}

for(j=i;j<psch->pr\_num-1;j++)

strcpy(psch->progress[j], psch->progress[j+1]);

printf("删除成功");

psch->pr\_num--;

if(psch->pr\_num==0) printf("表中已无已修课程");

break;

\*/

case 6:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL SeekTeacher(void){ //pass

fflush(stdin);

if(telist->length==0){

printf("教师名单为空");

getchar(); ClearScreen(); return TRUE;

}

count=0; flag=FALSE;

do{

if(count) printf("选项越界,重新输入");

printf("查找方式:\n1--姓名\t2--年龄区间\t3--性别\t4--回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1: //按姓名查找

printf("请输入教师姓名:");

scanf("%s", cbuffer); getchar();

for(i=0;i<telist->length;i++)

if(!strcmp(telist->elem[i].tname,cbuffer)) break;

if(i==telist->length){

printf("未找到相应教师");

getchar(); ClearScreen(); return TRUE;

}

printf("性别:");

if(telist->elem[i].gender==0) printf("女");

else printf("男");

printf("年龄:%hu\n查找成功",telist->elem[i].age);

break;

case 2: //按年龄区间查找

printf("输入年龄下限:");

scanf("%hu", &buf1); getchar();

printf("输入年龄上限:");

scanf("%hu", &buf2); getchar();

for(i=0;i<telist->length;i++)

if(telist->elem[i].age>=buf1 && telist->elem[i].age<=buf2){

printf("姓名:%s\t年龄:%hu\t性别:",telist->elem[i].tname,telist->elem[i].age);

flag=TRUE;

if(telist->elem[i].gender==0) printf("女\n");

else printf("男\n");

}

if(flag==FALSE) printf("无符合要求的教师"); getchar();

break;

case 3: //按性别查找

printf("输入目标性别(男--1\t女--0):");

j=getchar(); getchar();

for(i=0;i<telist->length;i++){

if(j=='0'){

if(!telist->elem[i].gender){

printf("姓名:%s\t年龄:%hu\n",telist->elem[i].tname,telist->elem[i].age);

flag=TRUE; //flag标记输出了有效信息

} }

else{

if(telist->elem[i].gender){

printf("姓名:%s\t年龄:%hu\n",telist->elem[i].tname,telist->elem[i].age);

flag=TRUE; //flag标记输出了有效信息

} } }

if(flag==FALSE) printf("无符合要求的教师"); getchar();

case 4:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL SeekRoom(void){ //pass

fflush(stdin);

if(rolist->length==0){

printf("教室名单为空");

getchar(); ClearScreen(); return TRUE;

}

count=0;

do{

if(count) printf("选项越界,重新输入");

printf("查找方式:\n1--编号\t2--容量\t3--回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==3){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>3);

switch(choice){

case 1: //按教室编号查找

printf("请输入教室编号:");

scanf("%s", cbuffer); getchar();

for(i=0;i<rolist->length;i++)

if(!strcmp(rolist->elem[i].code,cbuffer)) break;

if(i==rolist->length){

printf("未找到相应教室");

getchar(); ClearScreen(); return TRUE;

}

printf("教室容量:%hu\n查找成功",rolist->elem[i].rsize);

break;

case 2: //按教室容量查找

flag=FALSE;

printf("输入容量下限:");

scanf("%hu", &buf1); getchar();

printf("输入容量上限:");

scanf("%hu", &buf2); getchar();

for(i=0;i<rolist->length;i++)

if(rolist->elem[i].rsize>=buf1 && rolist->elem[i].rsize<=buf2){

printf("编号:%s\t容量:%hu\n",rolist->elem[i].code,rolist->elem[i].rsize);

flag=TRUE; //flag标记输出了有效信息

}

if(flag==FALSE) printf("无符合要求的教室"); getchar();

case 3:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL SeekCourse(void){ //pass

fflush(stdin);

BOOL glag=FALSE;

if(cougraph->length==0){

printf("课程名单为空");

getchar(); ClearScreen(); return TRUE;

}

printf("请输入待查课程名称子串:");

scanf("%s", cbuffer); getchar();

for(i=0; i<cougraph->length; i++)

//模糊查找

if(strstr(cougraph->coulist[i].name,cbuffer)){

printf("课程名称:%s\n",cougraph->coulist[i].name);

flag = FALSE;

printf("先修课程:\n");

for(pcarc=cougraph->coulist[i].archead;pcarc!=NULL;pcarc=pcarc->nextarc){

printf("%s\n", pcarc->prename);

flag = TRUE; //flag标记输出了有效信息

}

if(flag==FALSE) printf("该课无先修课程\n");

flag = FALSE;

printf("后继课程:\n");

for(j=0; j<cougraph->length; j++){

if(j==i) continue;

//查找方法：寻找当前课程是否有先修课为目标课程

for(pcarc=cougraph->coulist[j].archead;pcarc!=NULL;pcarc = pcarc->nextarc)

if(!strcmp(pcarc->prename, cougraph->coulist[i].name)){

printf("%s\n", cougraph->coulist[j].name);

flag = TRUE; //flag标记输出了有效信息

}

}

if(flag==FALSE) printf("该课无后继课程\n");

glag=TRUE;

}

if(glag==FALSE) printf("未找到相应课程");

getchar(); ClearScreen();

return TRUE;

}

BOOL SeekGroup(void){ //pass

char cbuf3[20];

fflush(stdin);

if(!grade[0].schead && !grade[1].schead && !grade[2].schead){

printf("课表名单为空"); return FALSE;

}

count=0;

do{

if(count) printf("选项越界,重新输入");

printf("查找方式\n1--班号\t2--人数\t3--回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==3){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>3);

switch(choice){

case 1: //按班号查找

printf("请输入待查班级名称:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("该班级不存在");

getchar(); ClearScreen(); return TRUE;

}

//输出班级相关信息

printf("人数:%hu\n",psch->clsize);

if(psch->do\_num!=0){

printf("已修课程清单:\n");

for(j=0; j<psch->do\_num; j++)

printf("%s\n", psch->done[j]);

printf("共%hu门\n",psch->do\_num);

}

else printf("无已修课程\n");

if(psch->pr\_num!=0){

printf("在修课程清单:\n");

for(j=0; j<psch->pr\_num; j++)

printf("%s\n", psch->progress[j]);

printf("共%hu门\n",psch->pr\_num);

}

else printf("无在修课程\n");

break;

case 2: //按班级人数查找

flag=FALSE;

printf("输入人数下限:");

scanf("%hu", &buf1); getchar();

printf("输入人数上限:");

scanf("%hu", &buf2); getchar();

for(i=0;i<3;i++)

for(psch=grade[i].schead;psch!=NULL;psch=psch->next)

if(psch->clsize>=buf1 && psch->clsize<=buf2){

//输出班级相关信息

printf("班级:%s\t人数:%hu\n",psch->classname,psch->clsize);

if(psch->do\_num!=0){

printf("已修课程清单:\n");

for(j=0; j<psch->do\_num; j++)

printf("%s\n", psch->done[j]);

printf("共%hu门\n",psch->do\_num);

}

else printf("无已修课程\n");

if(psch->pr\_num!=0){

printf("在修课程清单:\n");

for(j=0; j<psch->pr\_num; j++)

printf("%s\n", psch->progress[j]);

printf("共%hu门\n",psch->pr\_num);

}

else printf("无在修课程\n");

flag=TRUE; //flag标记输出了有效信息

}

if(flag==FALSE) printf("无符合要求的班级"); getchar();

case 3:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL AddClass(void){ //pass

fflush(stdin);

Class \*ptemp=(Class \*)malloc(sizeof(Class));

//课程名写入样本结构

printf("输入课程名:");

scanf("%s", ptemp->coursename); getchar();

for(i=0;i<cougraph->length;i++)

if(!strcmp(cougraph->coulist[i].name,ptemp->coursename)) break;

if(i==cougraph->length){ //新建课程(先修课节点未添加)

AddNewCourse(ptemp->coursename);

cougraph->length++;

printf("新课程已导入\n");

}

//教室号写入样本结构

printf("输入教室号:");

scanf("%s", ptemp->roomcode); getchar(); //多个教室的处理？

for(k=0; k<rolist->length; k++)

if(!strcmp(rolist->elem[k].code,ptemp->roomcode)) break; //k存放目标教室在教室清单中的位置

if(k==rolist->length) AddNewRoom(ptemp->roomcode);//新建教室

rolist->length++;

//教师姓名写入样本结构

printf("输入教师姓名:");

scanf("%s", ptemp->tname); getchar(); //多个教师的处理？

for(l=0; l<telist->length; l++)

if(!strcmp(telist->elem[l].tname,ptemp->tname)) break; //l存放目标教师在教师清单中的位置

if(l==telist->length) AddNewTeacher(ptemp->tname);//新建教师

telist->length++;

//课堂信息存入临时结点，待确定班级后以其为模板向各个班级中复制结点

printf("输入上课星期(阿拉伯数字):");

scanf("%hu", &buf1); getchar();

TimeInput(&(ptemp->cou\_st), &(ptemp->cou\_ed), ptemp->wk\_st, ptemp->wk\_ed);

count=0;

printf("输入班级数量:");

scanf("%hu", &hubuffer); getchar();

for(i=0;i<hubuffer;i++){

pcls=(Class \*)malloc(sizeof(Class));

//复制样本结构用于接入链表

strcpy(pcls->coursename, ptemp->coursename);

strcpy(pcls->roomcode, ptemp->roomcode);

strcpy(pcls->tname, ptemp->tname);

pcls->cou\_st=ptemp->cou\_st; pcls->cou\_ed=ptemp->cou\_ed;

memcpy(pcls->wk\_st, ptemp->wk\_st, sizeof(ptemp->wk\_st));

memcpy(pcls->wk\_ed, ptemp->wk\_ed, sizeof(ptemp->wk\_ed));

//输入班级信息

printf("输入班号:");

//按班号确定查找头结点

scanf("%s", cbuffer); getchar();

if(strstr(cbuffer,"15")){

//查找目标班级的存在性

for(psch=grade[0].schead; psch!=NULL; psch=psch->next)

if(!strcmp(psch->classname,cbuffer)) break;

if(psch==NULL){ //在课表结构中插入新班级

psch=(Schedule \*)malloc(sizeof(Schedule));

strcpy(psch->classname, cbuffer);

psch->do\_num=psch->pr\_num=0;

printf("输入人数:");

scanf("%hu",&(psch->clsize)); getchar();

psch->next=grade[0].schead;

grade[0].schead=psch;

for(j=0;j<6;j++){ //时间占位及单周课表结构初始化

psch->week[j].classhead=NULL; psch->week[j].cl\_num=0;

psch->week[j].ochead.head=NULL; psch->week[j].ochead.len=0;

}

grade[0].gr\_num++;

}

}

else if(strstr(cbuffer,"16")){

//查找目标班级的存在性

for(psch=grade[1].schead; psch!=NULL; psch=psch->next)

if(!strcmp(psch->classname,cbuffer)) break;

if(psch==NULL){ //在课表结构中插入新班级

psch=(Schedule \*)malloc(sizeof(Schedule));

strcpy(psch->classname, cbuffer);

psch->do\_num=psch->pr\_num=0;

psch->next=grade[1].schead;

grade[1].schead=psch;

for(j=0;j<6;j++){ //时间占位及单周课表结构初始化

psch->week[j].classhead=NULL; psch->week[j].cl\_num=0;

psch->week[j].ochead.head=NULL; psch->week[j].ochead.len=0;

}

grade[1].gr\_num++;

}

}

//查找目标班级的存在性

else if(strstr(cbuffer,"17")){

for(psch=grade[2].schead; psch!=NULL; psch=psch->next)

if(!strcmp(psch->classname,cbuffer)) break;

if(psch==NULL){ //在课表结构中插入新班级

psch=(Schedule \*)malloc(sizeof(Schedule));

strcpy(psch->classname, cbuffer);

psch->do\_num=psch->pr\_num=0;

psch->next=grade[2].schead; grade[2].schead=psch;

for(j=0;j<6;j++){ //时间占位及单周课表结构初始化

psch->week[j].classhead=NULL; psch->week[j].cl\_num=0;

psch->week[j].ochead.head=NULL; psch->week[j].ochead.len=0;

}

grade[2].gr\_num++;

}

}

else{ //截住乱码

printf("班号输入错误");

getchar(); ClearScreen(); return TRUE;

}

for(j=0;j<psch->pr\_num;j++) //查找在修课单中是否有目标课程

if(!strcmp(psch->progress[j],ptemp->coursename)) break;

if(j==psch->pr\_num){ //若无则添加该课

strcpy(psch->progress[j],ptemp->coursename);

psch->pr\_num++;

}

//插入课程节点

//特殊情况：当前本日无课或新课为一天中最先上的课

if(psch->week[buf1-1].classhead==NULL || psch->week[buf1-1].classhead->cou\_st > pcls->cou\_st || psch->week[buf1-1].classhead->wk\_st[0]>pcls->wk\_st[0]){

pcls->next=psch->week[buf1-1].classhead;

psch->week[buf1-1].classhead=pcls;

}

else{ //在课程链表中部插入

for(qcls=psch->week[buf1-1].classhead; qcls->next!=NULL; qcls=qcls->next)

if(qcls->next->cou\_st>pcls->cou\_st || (qcls->next->cou\_st==pcls->cou\_st && qcls->next->wk\_st[0]>pcls->wk\_st[0])){

pcls->next=qcls->next; qcls->next=pcls; break;

}

if(qcls->next==NULL){ //在课程链表尾部插入

pcls->next=qcls->next; qcls->next=pcls;

}

}

psch->week[buf1-1].cl\_num++;

//在班级下属数组中对相关时间段占位

AddTimeNode(pcls->wk\_st, pcls->wk\_ed, pcls->cou\_st, pcls->cou\_ed, &(psch->week[buf1-1].ochead));

}

//在相应教师结点下插入时间占位节点

AddTimeNode(ptemp->wk\_st, ptemp->wk\_ed, ptemp->cou\_st, ptemp->cou\_ed, &(telist->elem[l].ochead[buf1-1]));

//在相应教室结点下插入时间占位节点

AddTimeNode(ptemp->wk\_st, ptemp->wk\_ed, ptemp->cou\_st, ptemp->cou\_ed, &(rolist->elem[k].ochead[buf1-1]));

printf("输入完毕");

getchar(); ClearScreen();

return TRUE;

}

BOOL DelClass(void){ //pass

fflush(stdin);

char cbuf3[20];

unsigned short buf3[5], buf4[5], buf5=0, buf6=0;

printf("输入待删课堂课程名:");

scanf("%s", cbuffer); getchar();

printf("输入待删课堂班级数目:");

scanf("%hu", &hubuffer); getchar();

printf("输入上课日期(阿拉伯数字)");

scanf("%hu", &buf1); getchar();

for(i=0;i<hubuffer;i++){

count=0;

printf("输入班号:");

psch=ClassInput(cbuf3);

scanf("%s", cbuf3); getchar();

if(psch==NULL){

printf("未找到该班级,继续操作下一班级"); getchar(); continue;

}

//删除目标课程

for(pcls=psch->week[buf1-1].classhead;pcls!=NULL;pcls=pcls->next)

if(!strcmp(pcls->coursename, cbuffer)) break;

if(pcls==NULL){

printf("未在该日期下找到相应课程\n"); break;

}

//复制结点信息

strcpy(cbuf1, pcls->roomcode); strcpy(cbuf2, pcls->tname);

memcpy(buf3, pcls->wk\_st, sizeof(buf3)); memcpy(buf4, pcls->wk\_ed, sizeof(buf4));

buf5=pcls->cou\_st; buf6=pcls->cou\_ed;

//在班级下属数组中删除相关时间段

DelTimeNode(pcls->wk\_st,pcls->wk\_ed,pcls->cou\_st,pcls->cou\_ed,&(psch->week[buf1-1].ochead));

DelClassNode(pcls, psch, buf1-1); //删除课程结点

}

//在教室结构中删除占位时间结点

for(i=0;i<rolist->length;i++)

if(!strcmp(rolist->elem[i].code, cbuf1)) break;

if(i!=rolist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(rolist->elem[i].ochead[buf1-1]));

//在教师结构中删除占位时间

for(i=0;i<telist->length;i++)

if(!strcmp(telist->elem[i].tname, cbuf2)) break;

if(i!=telist->length)

DelTimeNode(buf3,buf4,buf5,buf6,&(telist->elem[i].ochead[buf1-1]));

printf("删除成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL AltClass(void){ //pass

fflush(stdin);

unsigned short buf3[5], buf4[5], buf5, buf6, buf7, buf8[5], buf9[5];

//buf1-buf4为新信息，buf6-buf9为旧信息

char cbuf3[20];

Class \*pcls=NULL;

Schedule \*psch=NULL;

printf("输入待改课堂课程名:");

scanf("%s", cbuffer); getchar();

printf("输入待改课堂班级数目:");

scanf("%hu", &hubuffer); getchar();

printf("输入上课日(阿拉伯数字)");

scanf("%hu", &buf5); getchar(); //以上信息用于在课表中定位课堂

count=0;

do{

if(count) printf("选项越界,重新输入");

printf("修改什么信息?\n1--教室\t2--任课教师\t3--上课时间\t4--回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1: //改教室

printf("输入新教室:");

scanf("%s",cbuf2); getchar();

for(l=0;l<rolist->length;l++)

if(!strcmp(rolist->elem[l].code,cbuf2)) break;

if(l==rolist->length) AddNewRoom(cbuf2);//改成新教室且越界时线性表扩容

rolist->length++;

//l记录查找表中新教室位置

break;

case 2: //改教师

printf("输入新教师姓名:");

scanf("%s",cbuf2); getchar();

for(l=0; l<telist->length; l++)

if(!strcmp(telist->elem[l].tname,cbuf2)) break;

if(l==telist->length) AddNewTeacher(cbuf2);//新建教师

telist->length++;

//l记录查找表中新教师位置

break;

case 3: //改时间

TimeInput(&buf1, &buf2, buf3, buf4);

break;

case 4:

return TRUE;

}

for(i=0;i<hubuffer;i++){

printf("输入班号:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("未找到%s班,继续查找下个班级\n",cbuf3); continue;

}

//检查输入课堂存在性

for(pcls=psch->week[buf5-1].classhead; pcls!=NULL; pcls=pcls->next)

if(!strcmp(pcls->coursename, cbuffer)) break;

if(pcls==NULL){

printf("未找到目标课堂,继续查找下个班级\n"); continue;

}

buf6=pcls->cou\_st; buf7=pcls->cou\_ed;

memcpy(buf8, pcls->wk\_st, sizeof(pcls->wk\_st));

memcpy(buf9, pcls->wk\_ed, sizeof(pcls->wk\_ed)); //原上课时段放入缓存

switch(choice){

case 1: //修改教室占用情况

strcpy(cbuf3, pcls->roomcode); //cbuf3为旧信息，cbuf2为新信息

strcpy(pcls->roomcode,cbuf2); break;

case 2: //修改教师占用情况

strcpy(cbuf3, pcls->tname);

strcpy(pcls->tname,cbuf2); break;

case 3: //修改时间占用情况

//删除目标课堂占用的原时段

DelTimeNode(pcls->wk\_st,pcls->wk\_ed,pcls->cou\_st,pcls->cou\_ed,&(psch->week[buf5-1].ochead));

strcpy(cbuf2, pcls->roomcode); strcpy(cbuf3, pcls->tname);

//两个缓冲串用于在对应的教室及教师结构下修改时间占位

pcls->cou\_st=buf1; pcls->cou\_ed=buf2;

memcpy(pcls->wk\_st, buf3, sizeof(buf3));

memcpy(pcls->wk\_ed, buf4, sizeof(buf4)); //在目标课堂中更改上课时段

AddTimeNode(pcls->wk\_st, pcls->wk\_ed, pcls->cou\_st, pcls->cou\_ed, &(psch->week[buf5-1].ochead));

}

}

flag=FALSE;

switch(choice){

case 1: //修改教室占用情况

for(j=0;j<rolist->length;j++)

if(!strcmp(rolist->elem[j].code, cbuf3)) break;

//删除目标课堂对原教室的占用

if(j!=rolist->length)

DelTimeNode(buf8,buf9,buf6,buf7,&(rolist->elem[j].ochead[buf5-1]));

//添加目标课堂对新教室的占用

AddTimeNode(buf8, buf9, buf6, buf7, &(rolist->elem[l].ochead[buf5-1]));

break;

case 2: //修改教师占用情况

for(j=0;j<telist->length;j++)

if(!strcmp(telist->elem[j].tname, cbuf3)) break;

//删除目标课堂对原教师的占用

if(j!=telist->length)

DelTimeNode(buf8,buf9,buf6,buf7,&(telist->elem[j].ochead[buf5-1]));

//添加目标课堂对新教师的占用

AddTimeNode(buf8, buf9, buf6, buf7, &(telist->elem[l].ochead[buf5-1]));

break;

case 3:

for(l=0;l<rolist->length;l++)

if(!strcmp(rolist->elem[l].code, cbuf2)) break;

//删除目标教室的原占用时间

if(l!=rolist->length)

DelTimeNode(buf8,buf9,buf6,buf7,&(rolist->elem[l].ochead[buf5-1]));

//添加目标教室的新占用时间

AddTimeNode(buf3, buf4, buf1, buf2, &(rolist->elem[l].ochead[buf5-1]));

for(l=0;l<telist->length;l++)

if(!strcmp(telist->elem[l].tname, cbuf3)) break;

//删除目标教师的原占用时间

if(l!=telist->length)

DelTimeNode(buf8,buf9,buf6,buf7,&(telist->elem[l].ochead[buf5-1]));

//添加目标教师的新占用时间

AddTimeNode(buf3, buf4, buf1, buf2, &(telist->elem[l].ochead[buf5-1]));

break;

}

printf("修改成功");

getchar(); ClearScreen();

return TRUE;

}

BOOL SeekClass(void){ //pass

fflush(stdin);

char cbuf3[20];

unsigned short buf8;

BOOL glag;

count=0;

do{

if(count++) printf("选项越界,重新输入");

printf("按何种方式检索?\n1--任课教师\t2--班级\t\t3--教室+课程+时间\t4--时间\t\t5--返回主菜单\n");

scanf("%hu", &hubuffer); getchar();

if(hubuffer==5){ ClearScreen(); return TRUE; }

}while(hubuffer<1 || hubuffer>5);

flag=FALSE;

switch(hubuffer){

case 1:

printf("输入任课教师:");

scanf("%s", cbuffer); getchar();

for(i=0;i<3;i++){ //三个年级

for(psch=grade[i].schead; psch!=NULL; psch=psch->next)

for(j=0;j<6;j++) //每星期有课的六天

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next)

//找到相应课程

if(!strcmp(pcls->tname, cbuffer)){

printf("课程:%s\t班级:%s\t上课教室:%s\n上课时间:", pcls->coursename,psch->classname,pcls->roomcode);

k=0; flag=TRUE; //flag标记输出了有效信息

while(pcls->wk\_st[k]&&k<5){

if(k) putchar(',');

printf("%hu-%hu",pcls->wk\_st[k],pcls->wk\_ed[k]); k++;

}

printf("周,周%hu,%hu-%hu节\n",j+1,pcls->cou\_st,pcls->cou\_ed);

}

}

if(flag==FALSE) printf("未找到相关课堂");

break;

case 2:

printf("输入班号:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("未找到该班级");

getchar(); ClearScreen(); return TRUE;

}

for(i=0;i<6;i++)

for(pcls=psch->week[i].classhead;pcls!=NULL;pcls=pcls->next){

printf("课程:%s\t任课教师:%s\t上课教室:%s\n上课时间:", pcls->coursename,pcls->tname,pcls->roomcode);

j=0; flag=TRUE; //flag标记输出了有效信息

while(pcls->wk\_st[j]&&j<5){

if(j) putchar(',');

printf("%hu-%hu",pcls->wk\_st[j],pcls->wk\_ed[j]); j++;

}

printf("周,周%hu,%hu-%hu节\n",i+1,pcls->cou\_st,pcls->cou\_ed);

}

if(flag==FALSE) printf("未找到相关课堂");

break;

case 3:

printf("输入待查课堂课程名:");

scanf("%s", cbuffer); getchar();

printf("输入上课教室:");

scanf("%s",cbuf1); getchar();

printf("输入上课星期:");

scanf("%hu", &buf5); getchar();

TimeInput(&buf1, &buf2, buf3, buf4);

for(i=0;i<3;i++){

for(psch=grade[i].schead; psch!=NULL; psch=psch->next)

for(pcls=psch->week[buf5-1].classhead;pcls!=NULL;pcls=pcls->next){

hubuffer=0;//检查上课时间对应关系

hubuffer=(pcls->cou\_st-buf1)+(pcls->cou\_ed-buf2);

count=0;

while((pcls->wk\_st[count]||buf3[count]||pcls->wk\_ed[count]||buf4[count])&&count<5){ //上课周数数组元素有效

hubuffer=(pcls->wk\_st[count]-buf3[count])+(pcls->wk\_ed[count]-buf4[count]);

count++;

}

//全部信息核查完毕，且完全重合

if(!strcmp(pcls->coursename, cbuffer)&&!strcmp(pcls->roomcode,cbuf1)&&!hubuffer){

printf("班级:%s\t任课教师:%s\n", psch->classname,pcls->tname);

flag=TRUE; //flag标记输出了有效信息

}

}

}

if(flag==FALSE) printf("未找到相关课堂");

break;

case 4:

printf("输入开始节数:");

scanf("%hu", &buf1); getchar();

printf("输入结束节数:");

scanf("%hu", &buf2); getchar();

printf("输入目标星期:");

scanf("%hu", &buf8); getchar();

printf("输入开始周数:");

scanf("%hu", &buf5); getchar();

printf("输入结束周数:");

scanf("%hu", &buf6); getchar();

flag=FALSE;

for(i=0;i<3;i++){

for(psch=grade[i].schead; psch!=NULL; psch=psch->next)

for(pcls=psch->week[buf8-1].classhead;pcls!=NULL;pcls=pcls->next){

glag=FALSE;

hubuffer=count=0;//检查上课时间对应关系

if(pcls->cou\_st>=buf1&&pcls->cou\_ed<=buf2) hubuffer=1;

//目标班级在指定时间范围内有课

while(pcls->wk\_st[count]||pcls->wk\_ed[count]){

if(buf5 >= pcls->wk\_st[count] && buf6<= pcls->wk\_ed[count]){

glag=TRUE; break; //flag标记输出了有效信息

}

count++;

}

//时间参数满足要求

if(hubuffer&&glag){

printf("课程:%s\t教室:%s\t班级:%s\t任课教师:%s\n", pcls->coursename,pcls->roomcode,psch->classname,pcls->tname);

flag=TRUE; //flag标记输出了有效信息

} }

}

if(flag==FALSE) printf("未找到相关课堂");

case 5:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL RoomUsage(void){

fflush(stdin);

unsigned short buf7, buf8;

BOOL glag;

count=0;

do{

if(count) printf("请重新输入\n");

printf("选择分析方式:\n1--空闲教室查找\t2--教室利用率\t3--教室使用合理性\t4--返回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1:

printf("输入目标周:");

scanf("%hu", &hubuffer); getchar();

printf("输入目标星期:");

scanf("%hu", &buf5); getchar();

printf("输入起始节:");

scanf("%hu", &buf1); getchar();

printf("输入终止节:");

scanf("%hu", &buf2); getchar();

glag=FALSE;

count=0;

for(i=0;i<rolist->length;i++){

flag=FALSE;

for(pocc=rolist->elem[i].ochead[buf5-1].head;pocc!=NULL;pocc=pocc->next){

if(pocc->week==hubuffer && pocc->time>=buf1 && pocc->time<=buf2){

flag=TRUE; break;

}

}

if(flag==FALSE){

printf("%s\t", rolist->elem[i].code); glag=TRUE;

count++;

if(!(count%5)) putchar('\n');

}

}

if(glag==FALSE) printf("该时间段内无满足要求的教室");

break;

case 2:

printf("输入目标周区间(只有一周则始末周相同)\n起始周:");

scanf("%hu", &buf1); getchar();

printf("结束周:");

scanf("%hu", &buf2); getchar();

printf("输入目标日区间(只有一天则始末节相同)\n起始日:");

scanf("%hu", &buf7); getchar();

printf("结束日:");

scanf("%hu", &buf8); getchar();

printf("输入目标节区间(只有一周则始末节相同)\n起始节:");

scanf("%hu", &buf5); getchar();

printf("结束节:");

scanf("%hu", &buf6); getchar();

count=0;

for(i=0;i<rolist->length;i++)

for(j=buf7-1; j<buf8; j++)

for(pocc=rolist->elem[i].ochead[j].head;pocc!=NULL;pocc=pocc->next)

if(pocc->week>=buf1 && pocc->week<=buf2 && pocc->time>=buf5 && pocc->time<=buf6) count++;

printf("目标时间段内教室利用率为%f", count\*1.0/(rolist->length\*(buf2-buf1+1)\*(buf8-buf7+1)\*(buf6-buf5+1)));

break;

case 3:

buf6=0;

printf("输入上课教室:");

scanf("%s",cbuf1); getchar();

for(i=0;i<rolist->length;i++)

if(!strcmp(cbuf1,rolist->elem[i].code)) break;

if(i==rolist->length){

printf("未找到目标教室");

getchar(); ClearScreen(); return TRUE;

}

printf("当前教室最大容量为%hu\n",rolist->elem[i].rsize);

printf("输入待查课堂课程名:");

scanf("%s", cbuffer); getchar();

printf("输入上课星期:");

scanf("%hu", &buf5); getchar();

TimeInput(&buf1, &buf2, buf3, buf4);

flag=FALSE;

for(j=0;j<3;j++){

for(psch=grade[j].schead; psch!=NULL; psch=psch->next)

for(pcls=psch->week[buf5-1].classhead;pcls!=NULL;pcls=pcls->next){

hubuffer=0;//检查上课时间对应关系

hubuffer=(pcls->cou\_st-buf1)+(pcls->cou\_ed-buf2);

count=0;

while((pcls->wk\_st[count]||buf3[count]||pcls->wk\_ed[count]||buf4[count])&&count<5){

hubuffer=(pcls->wk\_st[count]-buf3[count])+(pcls->wk\_ed[count]-buf4[count]);

count++;

}

if(!strcmp(pcls->coursename, cbuffer)&&!strcmp(pcls->roomcode,cbuf1)&&!hubuffer){

printf("班级:%s\t人数%hu\n", psch->classname,psch->clsize);

buf6+=psch->clsize;

flag=TRUE;

}

}

}

if(flag==FALSE){

printf("未找到相关课堂"); break;

}

else{

if(buf6 > rolist->elem[i].rsize)

printf("课堂超员,需更换更大的教室\n");

else if(buf6/rolist->elem[i].rsize<=1 && buf6\*1.0/rolist->elem[i].rsize>=0.6)

printf("座位紧缺,需更换更大的教室\n");

else if(buf6\*1.0/rolist->elem[i].rsize<=0.5)

printf("空余座位过多,请考虑合并课堂或搬至较小的教室\n");

else

printf("教室合适\n");

}

case 4:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL ClassAnalysis(void){

fflush(stdin);

unsigned short occupy[22][6];

char cbuf3[20];

printf("输入待分析班级:");

psch=ClassInput(cbuf3);

if(psch==NULL){

printf("未找到该班级");

getchar(); ClearScreen(); return TRUE;

}

count=0;

do{

if(count) printf("请重新输入\n");

printf("选择分析方式:\n1--课程在每周内按早晚排布的合理性\t2--课程在每周内按日排布的合理性\n3--修课合理性\t4--返回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1: //课程在每天内排布的合理性

for(j=1;j<18;j++){

flag=TRUE;

buf1=buf2=buf5=0;

printf("第%d周",(int)j);

for(i=0;i<6;i++)

for(pocc=psch->week[i].ochead.head;pocc!=NULL;pocc=pocc->next)

if(pocc->week==j){

if(pocc->time>=1 && pocc->time<=4) buf1++;

else if(pocc->time>=5 && pocc->time<=8) buf2++;

else if(pocc->time>=9 && pocc->time<=12) buf5++;

}

if(buf5>buf1/2 || buf5>buf2/2){

printf("晚课比重过大\n"); flag=FALSE;

}

if(flag==TRUE) printf("课程在每一天内排布较合理\n");

}

break;

case 2:

printf("输入本学期有效星期数:");

scanf("%hu", &hubuffer); getchar();

memset(occupy,0,sizeof(occupy));

for(j=0;j<6;j++)

for(pocc=psch->week[j].ochead.head;pocc!=NULL;pocc=pocc->next)

occupy[pocc->week-1][j]++;

for(i=0;i<hubuffer;i++){

flag=TRUE;

printf("第%d周",(int)i+1);

for(j=0;j<5;j++)

for(k=j;k<5;k++)

if(occupy[i][j]-occupy[i][k]>=6){

printf("周%d与周%d课程分布过于悬殊\n",j+1,k+1); flag=FALSE;

}

if(flag==TRUE) printf("课程分布较合理\n");

}

buf1=buf2=buf5=buf6=0;

for(i=1;i<6;i++){

buf1+=occupy[0][i]; buf2+=occupy[1][i];

buf5+=occupy[hubuffer-2][i]; buf6+=occupy[hubuffer-1][i];

}

if(buf1>=20) printf("第1周课安排过多\n");

if(buf2>=22) printf("第2周课安排过多\n");

if(buf5>=10) printf("第%hu周课安排过多\n",hubuffer-1);

if(buf6>=4) printf("第%hu周课安排过多\n",hubuffer);

break;

case 3:

for(i=0;i<psch->pr\_num;i++){

flag=FALSE;

for(j=0;j<cougraph->length;j++)

if(!strcmp(psch->progress[i],cougraph->coulist[j].name)) break;

if(j==cougraph->length){

printf("课程%s未被录入\n",cougraph->coulist[j].name); return FALSE;

}

for(pcarc=cougraph->coulist[j].archead;pcarc!=NULL;pcarc=pcarc->nextarc){

for(k=0;k<psch->do\_num;k++)

if(!strcmp(psch->done[k],pcarc->prename)) break;

if(k==psch->do\_num){

printf("%s尚未修，本学期不应修%s\n",pcarc->prename,psch->progress[i]);

flag=TRUE;

}

}

if(flag==FALSE) printf("课程%s可在本学期修\n",cougraph->coulist[j].name);

putchar('\n');

}

case 4:

break;

}

//课程在每周内排布的合理性

getchar(); ClearScreen();

return TRUE;

}

BOOL TeacherAnalysis(void){

fflush(stdin);

unsigned short occupy[22];

printf("输入待分析教师:");

scanf("%s",cbuffer); getchar();

for(i=0;i<telist->length;i++)

if(!strcmp(telist->elem[i].tname,cbuffer)) break;

if(i==telist->length){

printf("未找到目标教师");

getchar(); ClearScreen(); return TRUE;

}

count=0;

do{

if(count) printf("选项越界,重新输入");

printf("选择分析方式:\n1--教学任务按日分配\t2--教学任务分配量\t3--楼层安排合理性\t4回主菜单\n");

scanf("%hu", &choice); getchar(); count++;

if(choice==4){ ClearScreen(); return TRUE; }

}while(choice<1 || choice>4);

switch(choice){

case 1:

for(j=1;j<18;j++){

flag=TRUE;

printf("第%d周",(int)j);

buf1=buf2=buf5=0;

for(k=0;k<6;k++)

for(pocc=telist->elem[i].ochead[k].head;pocc!=NULL;pocc=pocc->next)

if(pocc->week==j){

if(pocc->time>=1 && pocc->time<=4) buf1++;

else if(pocc->time>=5 && pocc->time<=8) buf2++;

else if(pocc->time>=9 && pocc->time<=12) buf5++;

}

if(buf5>buf1/2 || buf5>buf2/2){ //在有课的基础上判断

printf("晚课比重过大\n"); flag=FALSE;

}

if(flag==TRUE) printf("课程在每一天内排布较合理\n");

}

break;

case 2:

printf("输入本学期有效星期数:");

scanf("%hu", &hubuffer); getchar();

memset(occupy,0,sizeof(occupy));

for(j=0;j<6;j++)

for(pocc=psch->week[j].ochead.head;pocc!=NULL;pocc=pocc->next)

occupy[pocc->week-1]++;

flag=FALSE;

for(j=0;j<hubuffer;j++)

//退休年龄:男-55 女-60

if((telist->elem[i].gender==0 && telist->elem[i].age<50) || (telist->elem[i].gender==1 && telist->elem[i].age<55)){

if(occupy[j]>14){

printf("第%hu周担负课堂过多\n",j+1); flag=TRUE;

}

}

else{

if(occupy[j]>10){

printf("第%hu周担负课堂过多\n",j+1); flag=TRUE;

}

}

if(flag==FALSE) printf("该教师教学任务分配较为适宜");

break;

case 3:

if((telist->elem[i].gender==0 && telist->elem[i].age<55) || (telist->elem[i].gender==1 && telist->elem[i].age<60)){

printf("该教师无特别楼层要求\n"); break;

}

else{

flag=FALSE;

for(j=0;j<3;j++){

for(psch=grade[j].schead;psch!=NULL;psch=psch->next){

for(k=0;k<6;k++){

for(pcls=psch->week[k].classhead;pcls!=NULL;pcls=pcls->next){

if(!strstr(pcls->tname,telist->elem[i].tname)) continue;

if(strstr(pcls->roomcode,"4") || strstr(pcls->roomcode,"5")){

printf("%s所在教室%s的楼层过高\n",pcls->coursename,pcls->roomcode);

flag=TRUE;

} } } } }

if(flag==FALSE) printf("该教师上课的楼层较为合理\n");

}

case 4:

break;

}

getchar(); ClearScreen();

return TRUE;

}

BOOL FileOut(void){

fflush(stdin);

FILE \*out;

time\_t lt;

char \*filename, \*pc;

const char \*appe=".txt";

lt=time(NULL);

filename=ctime(&lt);

filename[strlen(filename)-1]='\0';

strcat(filename,appe);

pc=&filename[0];

while(\*pc){

if((\*pc==':' ) || (\*pc==' ')){ \*pc='\_'; }

pc++;

}

if((out=fopen(filename,"w"))==NULL){

printf("打开失败！\n");

getchar(); ClearScreen(); return TRUE;

}

//教师清单

fprintf(out, "%hu ", telist->length);

for(i=0;i<telist->length;i++){

fprintf(out, "%s ", telist->elem[i].tname);

fprintf(out, "%hu ", telist->elem[i].gender);

fprintf(out, "%hu ", telist->elem[i].age);

for(j=0;j<6;j++){

fprintf(out, "%hu ", telist->elem[i].ochead[j].len);

for(pocc=telist->elem[i].ochead[j].head;pocc!=NULL;pocc=pocc->next){

fprintf(out, "%hu ", pocc->time);

fprintf(out, "%hu ", pocc->week);

}

}

}

//教室清单

fprintf(out, "%hu ", rolist->length);

for(i=0;i<rolist->length;i++){

fprintf(out, "%s ", rolist->elem[i].code);

fprintf(out, "%hu ", rolist->elem[i].rsize);

for(j=0;j<6;j++){

fprintf(out, "%hu ", rolist->elem[i].ochead[j].len);

for(pocc=rolist->elem[i].ochead[j].head;pocc!=NULL;pocc=pocc->next){

fprintf(out, "%hu ", pocc->time);

fprintf(out, "%hu ", pocc->week);

}

}

}

//课程清单

fprintf(out, "%hu ", cougraph->length);

for(i=0;i<cougraph->length;i++){

fprintf(out, "%s ",cougraph->coulist[i].name);

fprintf(out, "%hu " ,cougraph->coulist[i].arcnum);

for(pcarc=cougraph->coulist[i].archead;pcarc!=NULL;pcarc=pcarc->nextarc)

fprintf(out, "%s ",pcarc->prename);

}

//班级+课表清单

for(i=0;i<3;i++){

fprintf(out, "%hu ",grade[i].gr\_num);

for(psch=grade[i].schead;psch!=NULL;psch=psch->next){

fprintf(out, "%s ",psch->classname);

fprintf(out, "%hu ",psch->clsize);

fprintf(out, "%hu ",psch->do\_num);

fprintf(out, "%hu ",psch->pr\_num);

if(psch->do\_num)

for(j=0;j<psch->do\_num;j++)

fprintf(out, "%s ",psch->done[j]);

if(psch->pr\_num)

for(j=0;j<psch->pr\_num;j++)

fprintf(out, "%s ",psch->progress[j]);

for(j=0;j<6;j++){

fprintf(out, "%hu ",psch->week[j].cl\_num);

for(pcls=psch->week[j].classhead;pcls!=NULL;pcls=pcls->next){

fprintf(out, "%s ",pcls->coursename);

fprintf(out, "%s ",pcls->roomcode);

fprintf(out, "%s ",pcls->tname);

fprintf(out, "%hu ",pcls->cou\_st);

fprintf(out, "%hu ",pcls->cou\_ed);

for(k=0;k<5;k++){

fprintf(out, "%hu ",pcls->wk\_st[k]);

fprintf(out, "%hu ",pcls->wk\_ed[k]);

}

}

fprintf(out, "%hu ",psch->week[j].ochead.len);

for(pocc=psch->week[j].ochead.head;pocc!=NULL;pocc=pocc->next){

fprintf(out, "%hu ", pocc->time);

fprintf(out, "%hu ", pocc->week);

}

}

}

}

fclose(out);

printf("备份成功！按任意键继续...\n");

getchar(); ClearScreen();

return TRUE;

}

BOOL FileIn(void){

fflush(stdin);

FILE \*in;

printf("导入的数据将覆盖系统当前数据！继续——Y 返回主菜单——其它任意键\n");

fflush(stdin); j=getchar(); getchar();

if(j!='y'&& j!='Y'){

printf("数据导入取消");

getchar(); ClearScreen(); return TRUE;

}

char input[80], \*filename;

printf("请输入您要恢复的数据所在的文件名:");

fflush(stdin); scanf("%s",input); getchar();

filename=input;

//int handle,find;

if ((in = fopen(filename, "r+")) == NULL)

{

printf("您输入的备份文件不存在,系统将保留当前数据");

getchar(); ClearScreen(); return TRUE;

}

ClearMemory();

Initial();

fscanf(in, "%hu", &(telist->length));

if(telist->length >= telist->listsize)

buf1 = (telist->length-telist->listsize)/LISTINCREMENT+1;

while(telist->length >= telist->listsize){

Teacher \*te\_newbase = (Teacher \*)realloc(telist->elem, (telist->listsize+buf1\*LISTINCREMENT)\* sizeof(Teacher));

if(!te\_newbase){

printf("内存分配失败"); return FALSE;

}

telist->elem = te\_newbase;

telist->listsize += buf1\*LISTINCREMENT;

}

for(i=0;i<telist->length;i++){

fscanf(in, "%s", telist->elem[i].tname);

fscanf(in, "%hu ", &(telist->elem[i].gender));

fscanf(in, "%hu ", &(telist->elem[i].age));

for(j=0;j<6;j++){

fscanf(in, "%hu", &(telist->elem[i].ochead[j].len));

telist->elem[i].ochead[j].head=NULL;

for(k=0;k<telist->elem[i].ochead[j].len;k++){

pocc = (Occupy \*)malloc(sizeof(Occupy));

fscanf(in, "%hu", &(pocc->time));

fscanf(in, "%hu", &(pocc->week));

pocc->next=telist->elem[i].ochead[j].head;

telist->elem[i].ochead[j].head=pocc;

}

}

}

fscanf(in, "%hu", &(rolist->length));

if(rolist->length >= rolist->listsize)

buf1 = (rolist->length-rolist->listsize)/LISTINCREMENT+1;

while(rolist->length >= rolist->listsize){

Room \*ro\_newbase = (Room\*)realloc(rolist->elem, (rolist->listsize+buf1\*LISTINCREMENT)\* sizeof(Room));

if(!ro\_newbase){

printf("内存分配失败"); return FALSE;

}

rolist->elem = ro\_newbase;

rolist->listsize += buf1\*LISTINCREMENT;

}

for(i=0;i<rolist->length;i++){

fscanf(in, "%s", rolist->elem[i].code);

fscanf(in, "%hu ", &(rolist->elem[i].rsize));

for(j=0;j<6;j++){

fscanf(in, "%hu", &(rolist->elem[i].ochead[j].len));

rolist->elem[i].ochead[j].head=NULL;

for(k=0;k<rolist->elem[i].ochead[j].len;k++){

pocc = (Occupy \*)malloc(sizeof(Occupy));

fscanf(in, "%hu", &(pocc->time));

fscanf(in, "%hu", &(pocc->week));

pocc->next=rolist->elem[i].ochead[j].head; //节点插入顺序反了？

rolist->elem[i].ochead[j].head=pocc;

}

}

}

fscanf(in, "%hu", &(cougraph->length));

if(cougraph->length >= cougraph->listsize)

buf1 = (cougraph->length-cougraph->listsize)/LISTINCREMENT+1;

while(cougraph->length >= cougraph->listsize){

CouHead \*cou\_newbase = (CouHead\*)realloc(cougraph->coulist, (cougraph->listsize+buf1\*LISTINCREMENT)\* sizeof(CouHead));

if(!cou\_newbase){

printf("内存分配失败"); return FALSE;

}

cougraph->coulist = cou\_newbase;

cougraph->listsize += buf1\*LISTINCREMENT;

}

for(i=0;i<cougraph->length;i++){

fscanf(in, "%s", cougraph->coulist[i].name);

fscanf(in, "%hu", &(cougraph->coulist[i].arcnum));

cougraph->coulist[i].archead=NULL;

for(j=0;j<cougraph->coulist[i].arcnum;j++){

pcarc=(CouArc \*)malloc(sizeof(CouArc));

pcarc->prename=(char \*)malloc(sizeof(cbuffer));

fscanf(in, "%s", pcarc->prename);

pcarc->nextarc=cougraph->coulist[i].archead;

cougraph->coulist[i].archead=pcarc;

}

}

for(i=0;i<3;i++){

fscanf(in, "%hu", &(grade[i].gr\_num));

grade[i].schead=NULL;

for(j=0;j<grade[i].gr\_num;j++){

psch=(Schedule \*)malloc(sizeof(Schedule));

fscanf(in, "%s", psch->classname);

fscanf(in, "%hu ",&(psch->clsize));

fscanf(in, "%hu", &(psch->do\_num));

fscanf(in, "%hu", &(psch->pr\_num));

if(psch->do\_num)

for(k=0;k<psch->do\_num;k++)

fscanf(in, "%s", psch->done[k]);

if(psch->pr\_num)

for(k=0;k<psch->pr\_num;k++)

fscanf(in, "%s", psch->progress[k]);

for(k=0;k<6;k++){

fscanf(in, "%hu", &(psch->week[k].cl\_num));

psch->week[k].classhead=NULL;

for(l=0;l<psch->week[k].cl\_num;l++){

pcls=(Class \*)malloc(sizeof(Class));

fscanf(in, "%s", pcls->coursename);

fscanf(in, "%s", pcls->roomcode);

fscanf(in, "%s", pcls->tname);

fscanf(in, "%hu", &(pcls->cou\_st));

fscanf(in, "%hu", &(pcls->cou\_ed));

for(m=0;m<5;m++){

fscanf(in, "%hu", &(pcls->wk\_st[m]));

fscanf(in, "%hu", &(pcls->wk\_ed[m]));

}

pcls->next=NULL;

if(psch->week[k].classhead==NULL)

psch->week[k].classhead=pcls;

else{

for(qcls=psch->week[k].classhead;qcls->next!=NULL;qcls=qcls->next) ;

qcls->next=pcls;

}

}

fscanf(in, "%hu", &(psch->week[k].ochead.len));

psch->week[k].ochead.head=NULL;

for(l=0;l<psch->week[k].ochead.len;l++){

pocc = (Occupy \*)malloc(sizeof(Occupy));

fscanf(in, "%hu", &(pocc->time));

fscanf(in, "%hu", &(pocc->week));

pocc->next=psch->week[k].ochead.head;

psch->week[k].ochead.head=pocc;

}

}

psch->next=grade[i].schead;

grade[i].schead=psch;

}

}

fclose(in);

printf("数据导入成功,按任意键继续...\n");

getchar(); ClearScreen();

return TRUE;

}

/\*\*

\* 函数名称: MaintainTeacherlInfo

\* 函数功能: 教师信息维护目录

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL MaintainTeacherInfo(void){

BOOL bRet = TRUE;

do{

printf("您想进行何种操作？\n");

printf("录入 press1\n修改 press2\n删除 press3\n返回主菜单 press4\n");

fflush(stdin); k=getchar(); getchar();

if(k=='4') { ClearScreen(); return TRUE; }

}while(k<'1'||k>'4');

if(k=='1') bRet = AddTeacher();

else if(k=='2') bRet = AltTeacher();

else if(k=='3') bRet = DelTeacher();

return bRet;

}

/\*\*

\* 函数名称: MaintainRoomInfo

\* 函数功能: 教室信息维护目录

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL MaintainRoomInfo(void){

BOOL bRet = TRUE;

do{

printf("您想进行何种操作？\n");

printf("录入 press1\n修改 press2\n删除 press3\n返回主菜单 press4\n");

fflush(stdin); k=getchar(); getchar();

if(k=='4') { ClearScreen(); return TRUE; }

}while(k<'1'||k>'4');

if(k=='1'){ bRet = AddRoom(); }

else if(k=='2'){ bRet = AltRoom(); }

else if(k=='3'){ bRet = DelRoom(); }

return bRet;

}

/\*\*

\* 函数名称: MaintainGroupInfo

\* 函数功能: 班级信息维护目录

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL MaintainGroupInfo(void){

BOOL bRet = TRUE;

do{

printf("您想进行何种操作？\n");

printf("录入 press1\n修改 press2\n删除 press3\n返回主菜单 press4\n");

fflush(stdin); k=getchar(); getchar();

if(k=='4') { ClearScreen(); return TRUE; }

}while(k<'1'||k>'4');

if(k=='1'){ bRet = AddGroup(); }

else if(k=='2'){ bRet = AltGroup(); }

else if(k=='3'){ bRet = DelGroup(); }

return bRet;

}

/\*\*

\* 函数名称: MaintainCourseInfo

\* 函数功能: 课程信息维护目录

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL MaintainCourseInfo(void){

BOOL bRet = TRUE;

do{

printf("您想进行何种操作？\n");

printf("录入 press1\n修改 press2\n删除 press3\n返回主菜单 press4\n");

fflush(stdin); k=getchar(); getchar();

if(k=='4') { ClearScreen(); return TRUE; }

}while(k<'1'||k>'4');

if(k=='1'){ bRet = AddCourse(); }

else if(k=='2'){ bRet = AltCourse(); }

else if(k=='3'){ bRet = DelCourse(); }

return bRet;

}

/\*\*

\* 函数名称: MaintainClassInfo

\* 函数功能: 课程信息维护目录

\* 输入参数: 无

\* 输出参数: 无

\* 返 回 值: BOOL类型, 总是为TRUE

\*

\* 调用说明:

\*/

BOOL MaintainClassInfo(void){

BOOL bRet = TRUE;

do{

printf("您想进行何种操作？\n");

printf("录入 press1\n修改 press2\n删除 press3\n返回主菜单 press4\n");

fflush(stdin); k=getchar(); getchar();

if(k=='4') { ClearScreen(); return TRUE; }

}while(k<'1'||k>'4');

if(k=='1'){ bRet = AddClass(); }

else if(k=='2'){ bRet = AltClass(); }

else if(k=='3'){ bRet = DelClass(); }

return bRet;

}

BOOL Add(void){

fflush(stdin);

for(i=0;i<telist->length;i++){

printf("教师姓名:%s\n输入性别(男--1\t女--0):",telist->elem[i].tname);

scanf("%hu",&telist->elem[i].gender); getchar();

printf("输入年龄:");

scanf("%hu",&telist->elem[i].age); getchar();

}

for(i=0;i<rolist->length;i++){

printf("教室编号:%s\n输入容量:",rolist->elem[i].code);

scanf("%hu",&rolist->elem[i].rsize); getchar();

}

for(i=0;i<3;i++){

for(psch=grade[i].schead;psch!=NULL;psch=psch->next){

printf("班号:%s\n输入人数:",psch->classname);

scanf("%hu",&psch->clsize); getchar();

}

}

return TRUE;

}

BOOL HelpTopic(void)

{

ShellExecuteA(NULL,"open","NOTEPAD.EXE","help.txt",NULL,SW\_SHOWNORMAL);

printf("按任意键继续...\n"); getch(); ClearScreen();

return TRUE;

}

BOOL AboutDorm(void)

{

char \*plabel\_name[] = {"系统版本：V1.0",

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

"All Rights Reserved",

"确认"

};

ShowModule(plabel\_name, 4);

return TRUE;

}

BOOL DelTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead){

count=0;

while(wk\_st[count] && count<5){ //按各段上课周数遍历

for(p=wk\_st[count]; p<=wk\_ed[count]; p++){ //按上课周遍历

for(n=cou\_st; n<=cou\_ed; n++){ //按上课节数遍历

for(pocc=ochead->head;pocc!=NULL;pocc=pocc->next){

if(pocc->week==p && pocc->time==n){ //上课时间吻合

if(pocc==ochead->head){ //时间占位节点为头结点

ochead->head=pocc->next;

free(pocc); pocc=ochead->head;

if(ochead->head==NULL){

ochead->len=0; break;

} }

else{ //时间占位结点非头结点

for(qocc=ochead->head;qocc->next!=pocc;qocc=qocc->next) ;

qocc->next = pocc->next; free(pocc); pocc=qocc;

}

ochead->len--; //时间占位结点数自减

break;

} } } }

count++;

}

return TRUE;

}

BOOL AddTimeNode(unsigned short wk\_st[], unsigned short wk\_ed[], unsigned short cou\_st, unsigned short cou\_ed, Ochead \*ochead){

unsigned short dwk, dtm;

count=0;

while(wk\_st[count]&&count<5){

for(p=wk\_st[count]; p<=wk\_ed[count]; p++)

for(n=cou\_st;n<=cou\_ed; n++){

//按先周数后节数的升序插入时间占位节点

pocc=(Occupy \*)malloc(sizeof(Occupy));

pocc->week=p; pocc->time=n;

if(ochead->len==0){ //无结点时插在第一位

pocc->next=ochead->head; ochead->head=pocc;

}

else{

dwk=pocc->week - ochead->head->week;

dtm=pocc->time - ochead->head->time;

if(dwk<0 || (dwk>=0 && dtm<0)){ //新结点值最小

pocc->next=ochead->head;

ochead->head=pocc;

}

else{ //在时间占位链表中部插入

for(qocc=ochead->head;qocc->next!=NULL;qocc=qocc->next){

dwk=pocc->week-qocc->next->week;

dtm=pocc->time-qocc->next->time;

if(dwk<0 || (dwk>=0 && dtm<0)){

pocc->next=qocc->next; qocc->next=pocc;

} }

if(qocc->next==NULL){ //在时间占位链表尾部插入

pocc->next=qocc->next; qocc->next=pocc;

} } }

ochead->len++;

}

count++;

}

return TRUE;

}

Schedule \*ClassInput(char cbuf[]){

fflush(stdin);

Schedule \*ps=NULL;

scanf("%s", cbuf); getchar();

//按班号确定查找头结点

if(strstr(cbuf,"15")) ps=grade[0].schead;

else if(strstr(cbuf,"16")) ps=grade[1].schead;

else if(strstr(cbuf,"17")) ps=grade[2].schead;

else{

printf("班号输入错误");

getchar(); ClearScreen(); return NULL;

}

//检查班号的存在性

for(;ps!=NULL;ps=ps->next)

if(!strcmp(ps->classname, cbuf)) break;

return ps;

}

BOOL AddNewRoom(char cbuf[]){

if(rolist->length >= rolist->listsize){

Room \*ro\_newbase = (Room \*)realloc(rolist->elem, (rolist->listsize+LISTINCREMENT)\* sizeof(Room));

if(!ro\_newbase){

printf("内存分配失败,返回主菜单");

getchar(); ClearScreen(); return TRUE;

}

rolist->elem = ro\_newbase;

rolist->listsize += LISTINCREMENT;

}

//添加新教室

strcpy(rolist->elem[rolist->length].code, cbuf);

//时间结构初始化

for(i=0;i<6;i++){

rolist->elem[rolist->length].ochead[i].head=NULL;

rolist->elem[rolist->length].ochead[i].len=0;

}

printf("新教室已导入\n");

return TRUE;

}

BOOL AddNewTeacher(char cbuf[]){

if(telist->length >= telist->listsize){

Teacher \*te\_newbase = (Teacher \*)realloc(telist->elem, (telist->listsize+LISTINCREMENT)\* sizeof(Teacher));

if(!te\_newbase){

printf("内存分配失败,返回主菜单");

getchar(); ClearScreen(); return TRUE;

}

telist->elem = te\_newbase;

telist->listsize += LISTINCREMENT;

}

//添加新教师

strcpy(telist->elem[telist->length].tname, cbuf);

//时间结构初始化

for(i=0;i<6;i++){

telist->elem[telist->length].ochead[i].head=NULL;

telist->elem[telist->length].ochead[i].len=0;

}

printf("新教师已导入\n");

return TRUE;

}

BOOL AddNewCourse(char cbuf[]){

if(cougraph->length >= cougraph->listsize){ //查找表扩容

CouHead \*cou\_newbase = (CouHead \*)realloc(cougraph->coulist, (cougraph->listsize+LISTINCREMENT)\* sizeof(CouHead));

if(!cou\_newbase){

printf("内存分配失败,将返回主菜单");

getchar(); ClearScreen(); return TRUE;

}

cougraph->coulist = cou\_newbase;

cougraph->listsize += LISTINCREMENT;

}

//节点初始化

strcpy(cougraph->coulist[cougraph->length].name,cbuf);

cougraph->coulist[cougraph->length].arcnum=0;

cougraph->coulist[cougraph->length].archead=NULL;

return TRUE;

}

BOOL SeekRemClass(Schedule \*ps, char cbuf[]){

Class \*qc;

for(p=0;p<ps->pr\_num;p++)

if(!strcmp(ps->progress[p],cbuf)) break; //在班级在修课程中寻找被删课程

flag=FALSE;

if(p!=ps->pr\_num){

for(n=0;n<6;n++) //查找此课程是否还有课堂

for(qc=ps->week[n].classhead;qc!=NULL;qc=qc->next){

if(flag==TRUE) break;

if(!strcmp(qc->coursename,ps->progress[p])){

flag=TRUE; break;

}

}

if(flag==FALSE){ //无课堂时删除该在修课程

for(n=p;n<ps->pr\_num-1;n++) strcpy(ps->progress[n],ps->progress[n+1]); //在班级节点中删除相应的在修课

ps->pr\_num--;

}

}

return TRUE;

}

BOOL DelClassNode(Class \*pc, Schedule \*ps, unsigned short hu){\

Class \*qc;

if(pc==ps->week[hu].classhead){ //课程链下头结点

ps->week[hu].classhead=pc->next;

free(pc); pc=ps->week[j].classhead;

if(ps->week[hu].classhead==NULL){

ps->week[hu].cl\_num=0; return TRUE;

} }

else{ //课程链下非头结点

for(qc=ps->week[hu].classhead;qc->next!=pc;qc=qc->next) ; //找到目标结点前驱

qc->next=pc->next; free(pc); pc=qc;

}

psch->week[hu].cl\_num--;

return TRUE;

}

BOOL TimeInput(unsigned short \*cou\_st, unsigned short \*cou\_ed, unsigned short wk\_st[], unsigned short wk\_ed[]){

fflush(stdin);

memset(wk\_st,0,5);

memset(wk\_ed,0,5);

printf("输入开始课时:");

scanf("%hu", cou\_st); getchar();

printf("输入结束课时:");

scanf("%hu", cou\_ed); getchar();

printf("分段输入上课周数:\n");

count = 0;

do{

printf("起始周:");

scanf("%hu", &wk\_st[count]); getchar();

printf("结束周:");

scanf("%hu", &wk\_ed[count]); getchar();

printf("继续? 是--Y\t否--其他键\n");

scanf("%c", &j); getchar(); count++;

}while(j=='Y' || j=='y');

return TRUE;

}