

**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**

**专业班级： 计算机科学与技术(校际交流)1601**

**学 号： U201610270**

**姓 名： 蒋苡杭**

**指导教师： 周全**

**报告日期： 2018年 3月 10 日**

**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

[1 引言 1](#_Toc508825405)

[1.1 课题背景与意义 1](#_Toc508825406)

[1.2 国内外研究现状 1](#_Toc508825407)

[1.3 课程设计的主要研究工作 1](#_Toc508825408)

[2 系统总体设计 3](#_Toc508825409)

[2.1 系统需求分析 3](#_Toc508825410)

[2.2 总体设计 3](#_Toc508825411)

[3 系统详细设计 5](#_Toc508825412)

[3.1 数据结构 5](#_Toc508825413)

[3.2 关键算法设计 7](#_Toc508825414)

[4 系统实现与测试 25](#_Toc508825415)

[4.1 系统实现 25](#_Toc508825416)

[4.2 系统测试 27](#_Toc508825417)

[5 总结与展望 44](#_Toc508825418)

[5.1 全文总结 44](#_Toc508825419)

[5.2 工作展望 44](#_Toc508825420)

[6 体会 45](#_Toc508825421)

[参考文献 46](#_Toc508825422)

[附录 47](#_Toc508825423)

# 1 引言

## 课题背景与意义

掌握图、查找表等的存储结构与基本算法，通过解决较复杂的基于图等模型的实际问题，提高学生对数据结构知识综合运用的技能与实践能力。并且可应用到对排课的合理性与科学性评价，提高教学管理水平，更好地服务教师与学生。

## 国内外研究现状

随着高校教学体制的不断变化，高校教学管理工作及信息资料的存档、查询、管理工作量日益增大和情况复杂。迫切需要有一款功能强大，操作简单，又具有人性化的教学信息管理系统，来减轻教学管理人员的工作压力。教学管理信息化是信息和网络技术在教学管理中的应用，是建立在信息技术上的，依据一定的教育思想和管理思想，对教学全过程进行有效的计划、组织、协调、控制，从而实现教育目标的一个过程。高校教学管理信息系统的处理对象是高校教学管理过程中的各种信息，所执行的是高校中的各项教学管理工作。教学管理信息化的建设是指信息系统及网络的硬件建设和相关软件的开发及应用。

和国外相比，国外政府相当重视教育领域中信息化的发展，在政府的支持下，各学校都对学校管理和教学信息进行了网络化改造，同时加强了辅助教学的建设，所有学校全部联网，并能在网上进行招生和信息的管理。西方发达国家大部分名牌高校就已经较为成功地完成了数字化校园的建设工作。随着计算机网络技术迅速的发展，国外高校在教育信息化方面也取得了进步，逐步实现了数字化的校园和信息化的教育，各高校都采用了远程教学、教师学生的互动教学、电子化辅助教学、网络共享课件和虚拟课堂等多种方式实现电子化的教学。经过多年运行和在使用过程中的不断改进，国外的这种管理模式己经处于比较完善的稳定运行阶段。

## 课程设计的主要研究工作

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。

# 系统总体设计

## 系统需求分析

本系统实现的目标是在可以对课堂、课程、教师、教室、班级这些数据对象进行增删改查的基础之上对现有排课情况进行评估，帮助排课人员更为方便合理地安排课程以及根据班级、教师、教室的情况及时做出课堂的调整，系统主要实现以下功能：

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。

在加载完数据的基础之上，用户可以对任何一个功能进行操作，不需要分辨先后顺序。

## 总体设计

本系统由四大功能模块组成：主要对象动态维护模块、主要对象查找模块、主要对象合理性分析模块、文件模块。功能模块图如下图2.1。

**2.2.1 主要对象动态维护模块功能说明**

主要对象动态维护模块的功能包括课堂、课程、教室、教师、班级的增加、删除和修改。

**2.2.2 主要对象查找模块**

主要对象查找模块功能包括教师、教室、班级、课程的查找，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，空闲教室的查找。

**2.2.3 主要对象合理性分析模块**

主要对象合理性分析模块功能包括教室利用率分析、能效分析，教师承担课堂合理性分析，班级参与课堂的合理性分析以及新老教师的特别分析。

**2.2.4 文件模块**

文件模块功能包括教师、课堂、课程、教室、班级的相关信息的文件写入和读取，根据数据对象的联系建立邻接表等。

完整的系统模块结构图如下:

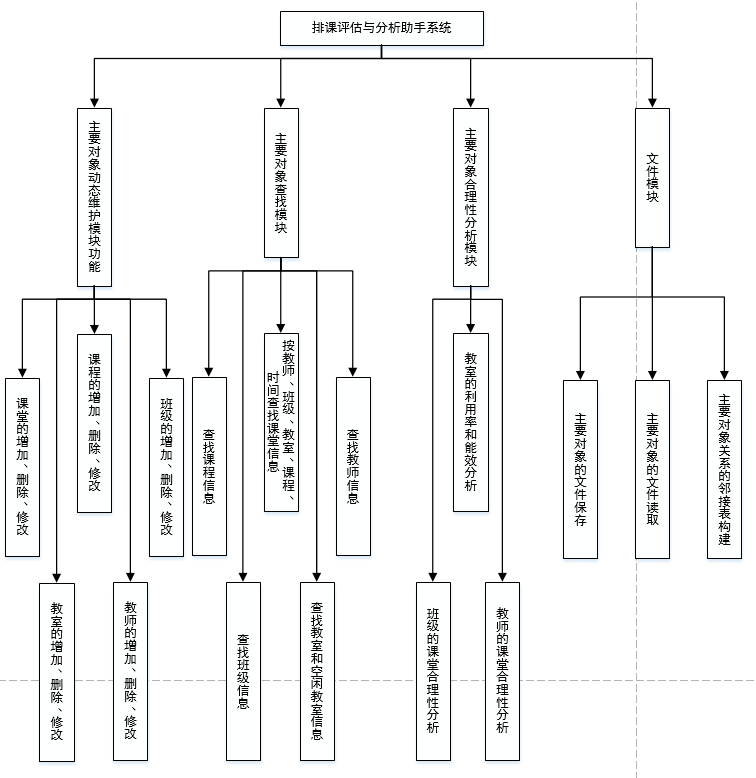


图2-1 系统模块结构图

# 系统详细设计

## 有关数据结构的定义

3.1.1 数据关系表示

按照任务要求，系统需要处理的基础信息有五种：课堂基本信息、课程基本信息、教师基本信息、教室基本信息和班级基本信息。这五种数据信息间存在这样的关联：教师基本信息与课堂基本信息存在关联，构成二分图的邻接表结构，以每一个教师结点为弧尾的弧都会指向一个课堂结点，代表该教师教授这门课程；教室基本信息与课堂基本信息存在关联，构成二分图的邻接表结构，以每一个教室结点为弧尾的弧都会指向一个课堂结点，代表该课程在该教室内上；班级基本信息与课堂基本信息存在关联，构成二分图的邻接表结构，以每一个班级结点为弧尾的弧都会指向一个课堂结点，代表该班级上这门课程。多种数据结构间的关联如图所示如图3-1所示。

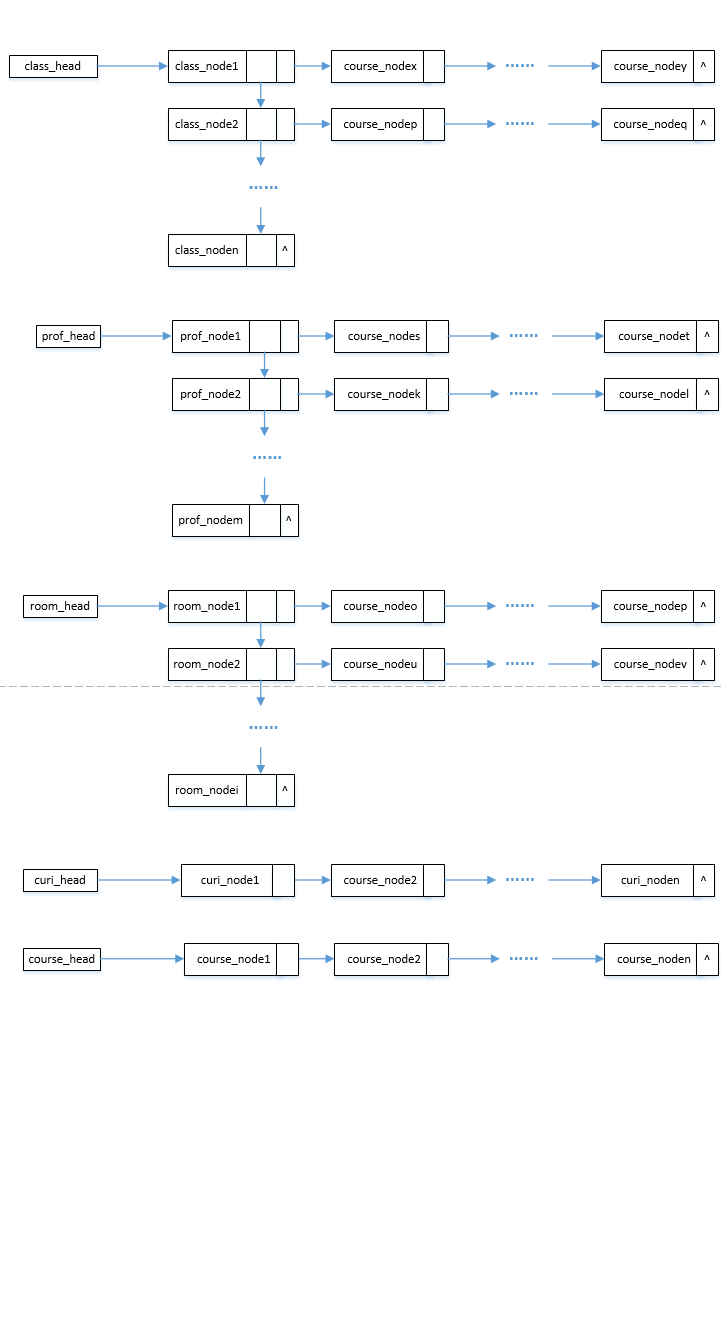


图3-1 系统模块结构图

3.1.2 数据内容描述

本系统中处理的数据类型主要包括课堂数据类型，教室数据类型，教师数据类型，课程数据类型和班级数据类型。每种数据类型的数据项和具体信息请见表3-1。

表3-1 数据类型及成员变量信息表



## 关键算法设计

本部分根据不同模块分类，进一步考虑到班级、教室、教师、课程、课堂五类数据结构之间的重复性很高，特意从每个模块中挑选出了具有代表性的班级数据类型进行算法的说明。当然，主要对象的合理性分析模块是本次课程设计的精髓所在，分析内容种类很多，具有实用价值，因此涉及这一部分的算法的函数我着重笔墨进行了详尽的分析。

3.2.1 主要对象的插入，删除，修改算法

(1)插入班级结点函数: InsertClassNode(CLASS\_NODE \* class\_hd)

**算法流程图：**

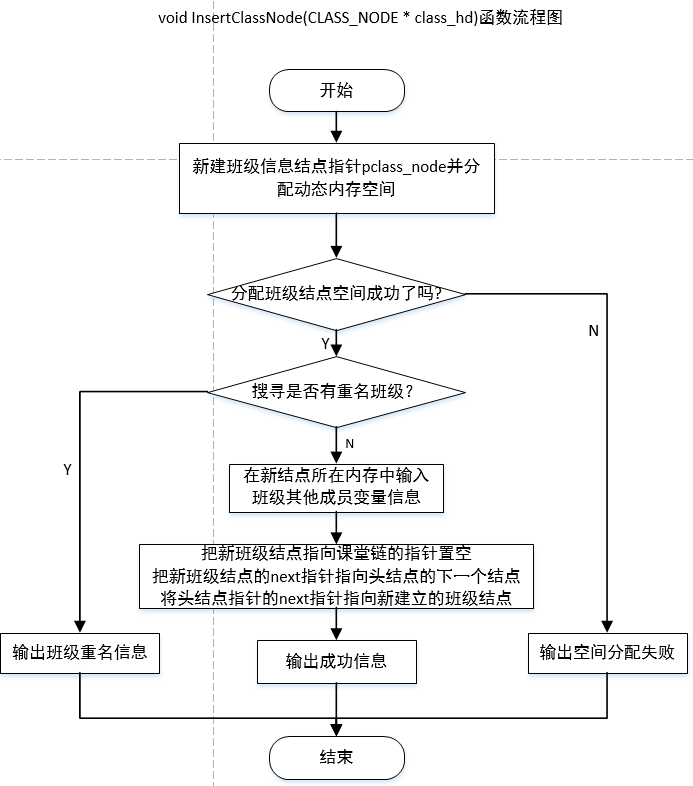


图3-2 InsertClassNode函数流程图

**时间复杂度：**算法中有一层循环，循环执行的次数i随着班级数目变化，假设其值为n，则算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为一个班级结点指针CLASS\_NODE所占的空间大小，因此空间复杂度为常量阶O(1)。

(2) 修改班级结点函数:ModiClassNode(CLASS\_NODE \* class\_hd)

**算法流程图：**

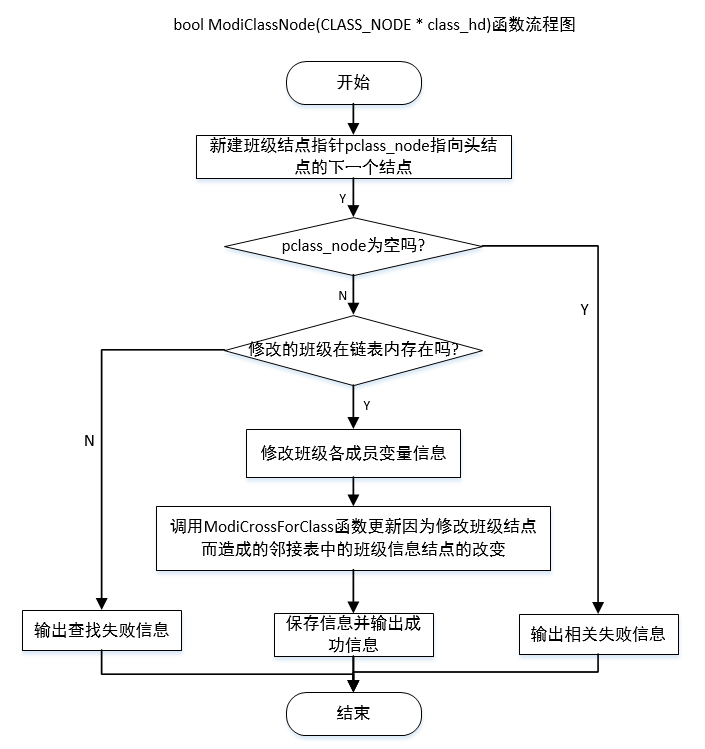


图3-3 ModiClassNode函数流程图

**时间复杂度：**算法中有一层循环，循环执行的次数i随着班级数目变化，假设其值为n，则算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为个别班级结点指针和int型计数变量所占的空间大小，因此空间复杂度为常量阶O(1)。

(3) 删除班级结点函数:DelClassNode(CLASS\_NODE \* class\_hd)

**算法思想：**首先根据需要删除的班级名称搜寻到班级链中相应的班级结点pclass\_node。然后遍历该班级链后面邻接表中的课堂结点，注意首先利用course\_temp保存正在遍历的这个结点的地址，然后将遍历指针pcourse\_node后移，方便下一个课堂结点的删除。接着，判断课堂结点的班级数是否只有一个，如果是，那么就说明这个课堂只有这一个班级，遍历搜寻到课堂链中相应的课堂结点，这个课堂结点应该调用DelWithoutId直接删除（具体描述见3.2.1第4个函数）。如果这个课堂下有多个班级，那么只需要将这个班级从课堂的班级名称中移除，同时将课堂信息中与班级有关的学生人数成员和班级个数修改即可。最后，释放course\_temp指针指向的课堂结点的空间。

**时间复杂度：**算法中有两层循环，第一层循环用于遍历并删除待删除的班级结点所在的班级-课堂邻接表中的课堂结点信息。第二层循环用于定位课堂信息单链中的课堂结点，并且修改或者删除相应结点。假设课堂的结点数目为n，那么算法的时间复杂度为O(n2)。

**空间复杂度：**算法所需要分配的空间大小为两个课堂结点指针和两个int型计数变量所占的空间大小，因此空间复杂度为常量阶O(1)。

(4) 直接删除课堂结点函数:DelWithoutId(COURSE\_NODE \*pcourse\_node)

**算法思想：**第一步，修改因为删除课堂结点引起的其他各项信息的变化。传入参数课堂信息结点指针pcourse\_node指向待删除的课堂结点，新建班级结点指针pclass\_node，调用SeekClass函数(具体描述见本部分第7个函数)使该指针指向具有相同班级名称的班级结点内存空间。现在pclass\_node指向需要修改与课堂有关信息的班级结点的内存块(每一个被删除的课堂内至少有一个班级)，调用ModiClassAfterDel函数（具体描述见3.2.1第5个函数）修改对应的班级结点。依次类推，修改教师，课程，教室结点中与课堂信息有关的部分，不重复叙述。

第二步要改变课堂单链表中待删除课堂结点的前后各课堂结点的逻辑关系。首先，调用PriorCourseElem函数（具体描述见3.2.1第6个函数），找到待删除结点前面一个前驱结点，使其的next指针“绕过”要删除的结点，拿前驱结点的next指针域指向当前结点的下一个结点，然后将遍历指针后移并且将待删除结点的空间释放。最后一步是保存相关信息。

**时间复杂度：**设课堂数目为n，教师数目为m，课程数和教师数分别为p和q，由于各数据的修改不存在嵌套关系，因此时间复杂度为O(m+n+p+q)。

**空间复杂度：**算法所需要分配的空间大小为个别结点指针，因此空间复杂度为常量阶O(1)。

(5)删除课堂结点后修改班级结点函数:ModiClassAfterDel(CLASS\_NODE \*pclass\_node, COURSE\_NODE \*pcourse\_node)

**算法流程图：**

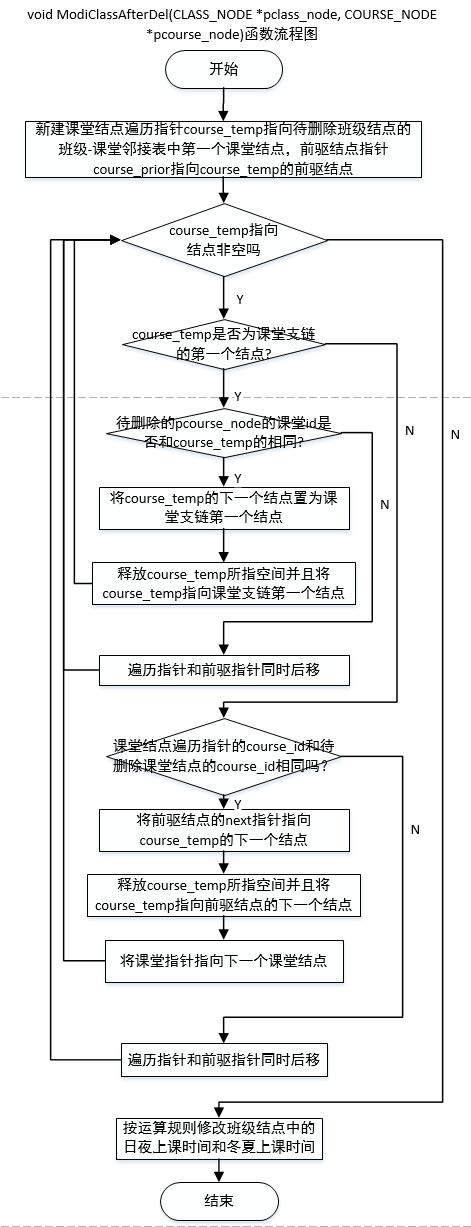


图3-4 ModiClassAfterDel函数流程图

**时间复杂度：**算法中有一层循环，用于遍历并修正包含待删除课堂结点pcourse\_node的班级结点中的信息。假设该班级后的课堂支链中的结点数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为两个课堂结点指针，分别指向前驱结点和遍历结点和两个int型计数变量所占的空间大小，因此空间复杂度为常量阶O(1)。

(6) 查找指定课堂结点的前驱的函数: PriorCourseElem(COURSE\_NODE \* L, char \*course\_id)

**算法流程图：**

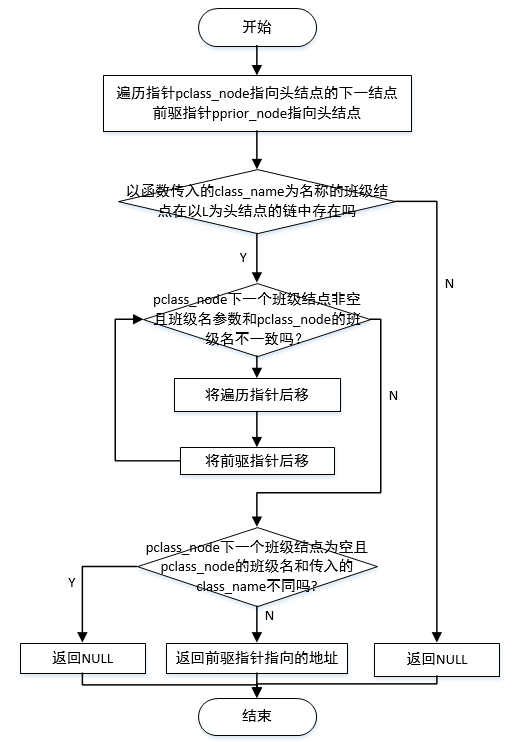


图3-5 PriorCourseElem函数流程图

**时间复杂度：**算法中有两个并列的一层循环，第一个一层循环用于查找课堂是否存在，第二个一层循环用于遍历整个课堂链，假设课堂链中的结点数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为两个课堂结点指针，分别指向前驱结点和遍历结点，因此空间复杂度为常量阶O(1)。

(7) 查找班级结点函数: SeekClass(CLASS\_NODE \*class\_hd, char \*class\_name)

**算法思想：**算法较为简单，首先将新建班级结点遍历指针pclass\_node指向班级单链头结点。当pclass\_node不为空的时候，检查正在遍历的结点的班级名称是否与传入的class\_name参数相同，如果相同的话，就将find标志变量置为1，跳出循环，返回遍历指针所指的地址。如果遍历结束后跳出循环时find依然为0，那么返回NULL。

**时间复杂度：**算法中有一层循环，假设班级数目为n，则算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为一个班级结点指针和一个int型标志所占的空间大小，因此空间复杂度为常量阶O(1)。

3.2.2 主要对象的查找算法

(1) 查找班级函数: SearchClassNode(CLASS\_NODE \* class\_hd)

**算法思想：**新建班级结点指针pclass\_node，使其指向根据用户输入的班级名称调用SeekClass函数（具体描述见3.2.1第7个函数）锁定的班级结点。如果pclass\_node为空，则说明查找班级失败，否则输出班级相应信息。

**时间复杂度：**算法中存在一个一层循环，用于查找班级是否存在，假设班级数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为一个班级结点指针，因此空间复杂度为常量阶O(1)。

(2) 按照班级查找课堂函数: SearchCourseNodeByClass(CLASS\_NODE \* class\_hd)

**算法思想：**新建班级结点指针pclass\_node，使其指向根据用户输入的班级名称调用SeekClass函数（具体描述见3.2.1第7个函数）锁定的班级结点。新建课堂结点指针pcourse\_node，使其指向pclass\_node后的第一个课堂结点(如果为空就指向NULL)。如果pclass\_node为空，那么说明该班级根本没有参与任何一个课堂，否则遍历该班级结点的课堂支链并且输出相应的信息。

**时间复杂度：**算法中存在两个并列的一层循环，第一个一层循环用于查找班级是否存在，第二个一层循环用于遍历整个课堂链并且输出相应信息，假设课堂链中的结点数目为n，班级链中的结点数目为m，那么算法的时间复杂度为O(max(m, n))。

**空间复杂度：**算法所需要分配的空间大小为一个班级结点指针和一个课堂结点指针，因此空间复杂度为常量阶O(1)。

(3)按照时间查找课堂函数: bool SearchCourseNodeByTime(COURSE\_NODE \* course\_node)

**算法流程图：**

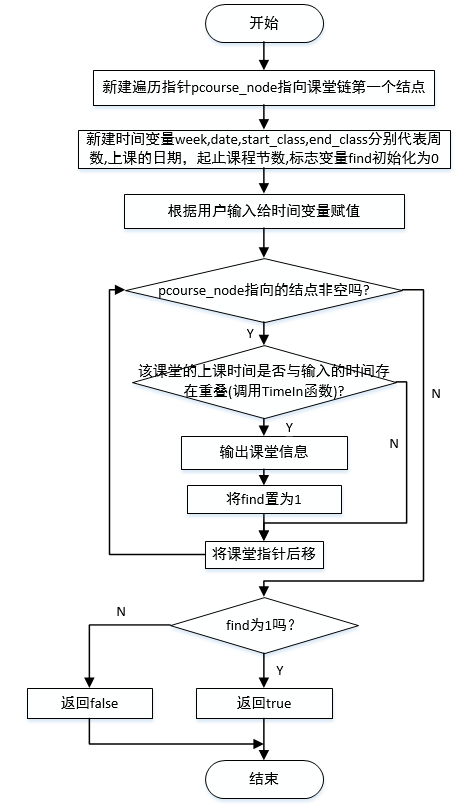


图3-6 SearchCourseNodeByTime函数流程图

**时间复杂度：**算法中有一层循环，用于检查每个课堂结点pcourse\_node的课堂时间是否与用户输入的时间重合。假设课堂的结点数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为一个课堂结点指针和一个教室结点指针以及个别用于接收输入信息的int型变量和字符型数组，因此空间复杂度为常量阶O(1)。

(4) 检查指定时间和指定课堂是否冲突的函数: TimeIn(COURSE\_NODE \* pcourse\_node, int week, int date, int start\_class, int end\_class)

**算法思想：**设置condi1和condi2两个bool变量，初始值设为false，分别代表课堂的周数是否和输入时间冲突以及课堂的日期和上课节数是否和输入时间冲突。外层循环遍历课堂的上课周期，如果其中有一个上课周期的起始周数小于等于输入的周数，并且该周期的结束周数大于等于这个输入的周数，那么将condi1置为true。

接着进入内层循环，利用内层循环遍历上课日期，如果上课日期数组中的一项和输入的日期date相同，那么继续检查这一天内课堂的起始或停止节数是否在输入的起止节数范围之内，如果二者之一是的话，就将condi2置为true。然后退出内层循环，由于内层循环每个上课日期都对应一个教室，因此，在退出内层循环时，检查condi1和condi2是否均为true，如果是的话那么就利用课程结点内存放的教室信息，调用SeekClass返回该时间段内课堂所在的教室结点指针。否则退出内层循环，继续外层循环的遍历。当遍历完所有外层循环时，退出循环，此时说明该函数未能在内层循环退出，没有冲突情况，返回NULL。

**时间复杂度：**该函数内虽然有两层循环，但是循环次数均为常量，与数据规模无关，因此时间复杂度为O(1)。

**空间复杂度：**算法所需要分配的空间大小为两个用于判定条件的bool型变量，个别用于接收输入信息的int型变量和字符型数组及指针，因此空间复杂度为常量阶O(1)。

(5) 查找空闲教室的函数: FindFreeRoom(ROOM\_NODE \*room\_hd)

**算法流程图：**

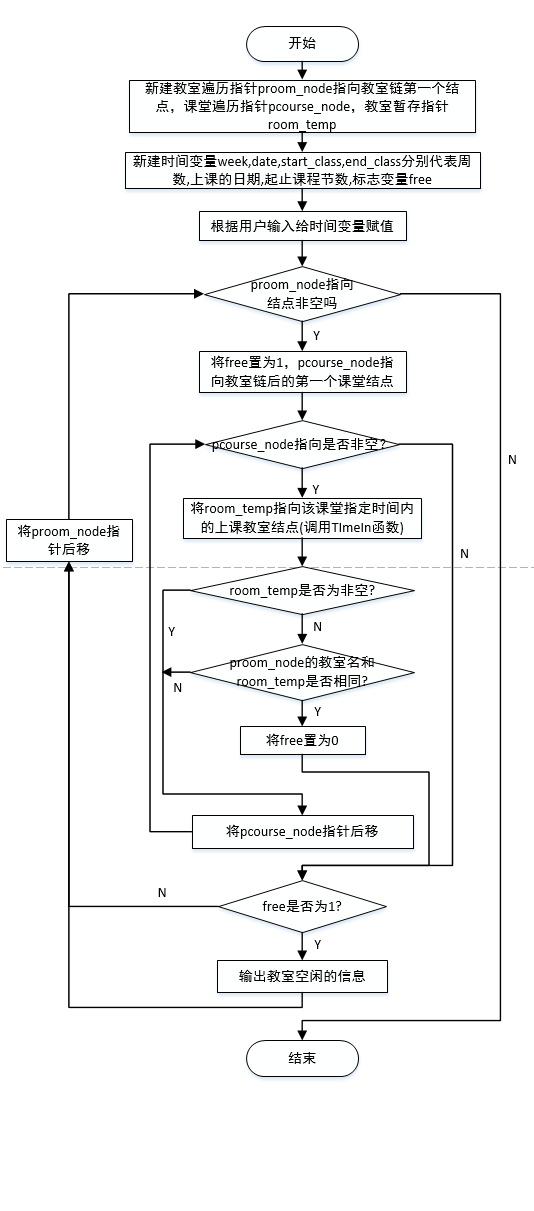


图3-7 FindFreeRoom函数流程图

**时间复杂度：**算法中有两层循环，第一层循环用于遍历教室结点所在的教室-课堂邻接表中的教室结点信息。第二层循环用于遍历该教室结点后的课堂结点信息支链中的课堂结点。假设课堂的结点数目为n，教室的结点数目为m,那么算法的时间复杂度为O(mn)。

**空间复杂度：**算法所需要分配的空间大小为个别教室结点指针和课堂结点指针以及若干int型计数变量所占的空间大小，因此空间复杂度为常量阶O(1)。

3.2.3 主要对象的合理性分析算法

(1) 教室能效分析函数: EnergyAnalysis(ROOM\_NODE \*head)

**算法流程图：**

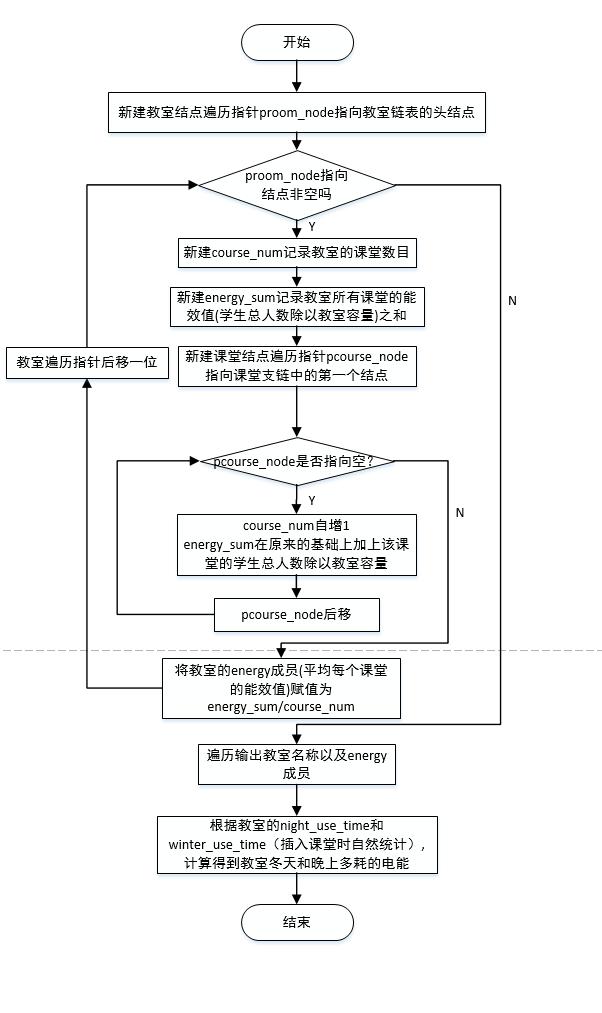


图3-8 EnergyAnalysis函数流程图

**时间复杂度：**算法中有两层循环，第一层循环用于遍历教室结点所在的班级-课堂邻接表中的课堂结点信息。第二层循环用于遍历该教室结点后的课堂结点信息单链中的课堂结点。假设教室的结点数目为n，课堂的结点数目为m,那么算法的时间复杂度为O(mn)。

**空间复杂度：**算法所需要分配的空间大小为个别教室结点指针和课堂结点指针，因此空间复杂度为常量阶O(1)。

(2)教室利用率分析函数: RoomUseRate(ROOM\_NODE \* head)

**算法思想：**遍历所有教室的结点，利用每个教室内的白天使用时间成员变量和夜晚使用时间成员变量，可以求出教室总体使用时间的成员变量，接着根据已知的一个学期的白天课时数，夜晚课时数，总课时数，可以分别求出每个教室的白天使用率，夜晚使用率和总体使用率。最后，输出每个教室的教室名和相关信息即可。

**时间复杂度：**算法中只存在一层循环，假设教室链中的结点数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为个别教室结点指针，因此空间复杂度为常量阶O(1)。

(3)教师课堂合理性函数: ProfAnalysis(PROF\_NODE \* head)

**算法流程图：**

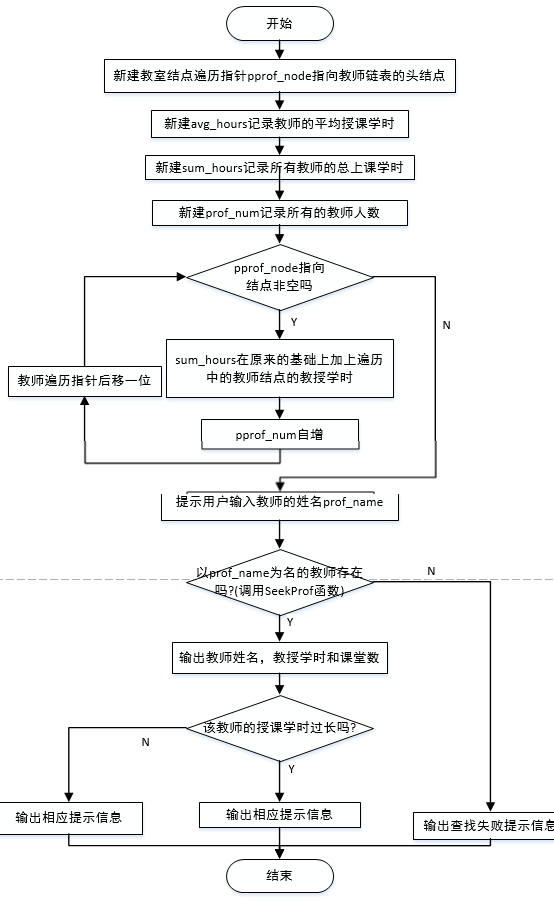


图3-9 ProfAnalysis函数流程图

**时间复杂度：**算法中有一层循环，用于计算所有教师的授课时间与教师数目。假设教师链的结点数目为n，那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为教师指针以及个别用于接收输入信息的int/float型变量和字符型数组，因此空间复杂度为常量阶O(1)。

(4)分析班级课堂合理性的函数: ClassAnalysis(CLASS\_NODE \*head)

**算法流程图：**

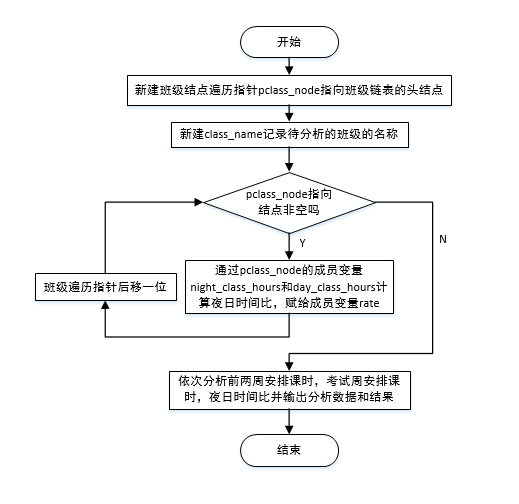


图3-10 ClassAnalysis函数流程图

分析内容有每个班级夜晚上课的时间之比，由于晚上有四节课的学时，白天有8节课的学时，因此，如果夜晚和白天上课时间的比值大于0.5，那么说明这个班夜晚上课的密度大于白天上课的密度，这是不合理的。

开学前两周的上课时间，由于开学前两周学生刚从放假的状态恢复，因此不应该安排太多的课时，否则会导致上课效率不高。

考试周的上课时间，最后两周是考试周，如果安排太多的课业，那么就没有办法复习，因此这个安排也要考虑。

**时间复杂度：**该函数内虽然有一层循环，用于遍历班级结点，计算班级的夜晚和白天的上课时间比，假设班级链中的结点个数为n，那么时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为一个班级结点指针，因此空间复杂度为常量阶O(1)。

(5) 新老教师的分析: NewAnalysis(PROF\_NODE \* pprof\_node)和OldAnalysis(PROF\_NODE \*pprof\_node)

**算法思想及流程图：**

分析的主要内容根据新老教师的不同特点分为两类，由于新教师经验不足，应该更多鼓励新教师积累经验，所以不应该让新教师承担太多学时的教学任务。老教师精力不如人，也不应该分配太多任务，而且老教师不能爬太高的楼层，这样会使得他们很吃力，因此最好让他们的课堂全部安排在三层楼及以下。新教师的分析较为简单，下面只列出老教师分析的流程图。

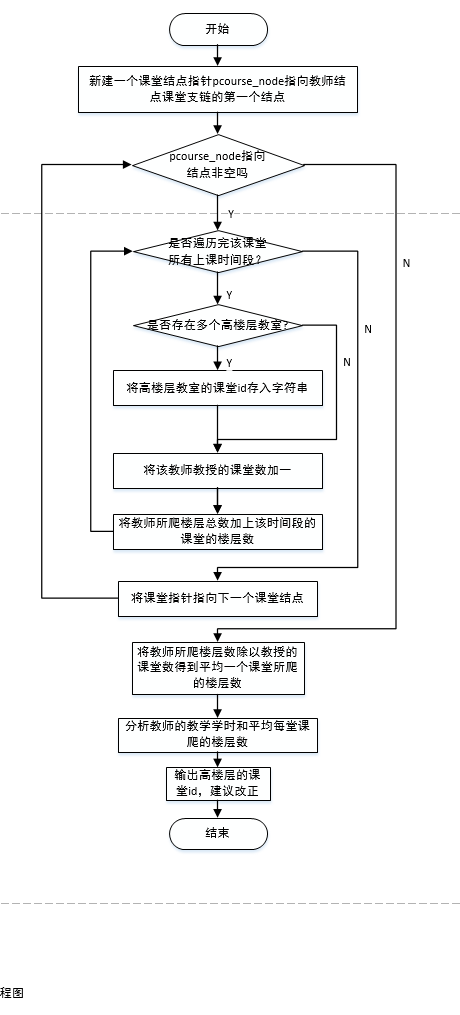


图3-11 OldAnalysis函数流程图

**时间复杂度：**算法中虽然有两层循环，第一层循环用于遍历课堂结点。第二层循环由于与时间有关，因此时间复杂度为常量。假设课堂的结点数目为n,那么算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配的空间大小为个别教师结点指针和课堂结点指针，因此空间复杂度为常量阶O(1)。

3.2.4 文件模块的算法

（1）保存班级信息：SaveClassNode(CLASS\_NODE \* class\_hd)

**算法思想:**首先检查传入的班级链头结点指针是否为空，如果为空的话，说明一个班级结点都没有，需要输出提示信息。接着，将文件指针pfout指向打开的存放班级结点信息的文件中，打开文件的方式为"w"。最后遍历每一个班级结点直到指针为NULL，将班级结点中需要存储的信息按序存入txt文件。遍历结束后，关闭文件指针。

**时间复杂度：**算法中有一层循环，假设班级的结点数目为n,那么显然算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配空间的大小为个别班级链遍历指针和文件指针的大小，因此空间复杂度为常量阶O(1)。

（2） 创建班级-课堂信息邻接表：Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head)

**算法思想：**首先新建一个课堂遍历指针用于遍历课堂单链，新建一个班级结点遍历指针用于遍历班级单链。若课堂遍历指针不为空，进入第一层循环，由于课堂结点内的成员变量包括描述一个课堂内有多少班级的int型变量，因此可以设置第二层循环，用于判断该课堂内的所有班级结点后面是不是都连接上了该课堂变量的指针。接着进入第三层循环，用于遍历班级信息链表，每一次遍历，一定会找到一个名称和第二层循环确定下的班级名称相符合的班级信息结点（如果所有的班级都已经连过了第一层循环确定的课堂结点，那么程序将提前在第二层循环跳出），只要遍历到这样的结点，那么就将第一层循环中的课堂结点信息复制到一个新结点里，再把这个新结点串到班级结点后面，接着跳出第三层循环。判断第二层循环中课堂结点后面的所有班级结点是否已经遍历完。如果已经遍历完，回到第一层循环，判断课堂单链有没有遍历完，如果遍历完，那么就跳出循环，结束函数。

**时间复杂度：**算法中有三层循环，但是第二层循环进行的次数仅仅取决于每个课堂下面有多少班级，显然一个课堂下设的班级数目为常数，因此假设班级数目为m，课堂数目为n，那么时间复杂度为O(mn)。

**空间复杂度**：算法所需要的空间主要分配在每个课堂需要根据这个课堂下面的班级数量，复制对应的课堂结点，由于每个课堂下设的班级数目为常数，因此假设班级数目为m，课堂数目为n，那么时间复杂度为O(mn)。

（3）创建班级链表：CreateClassList(CLASS\_NODE \*class\_hd)

**算法思想:** 该函数与SaveClassNode函数是对称的，因此只需要根据SaveClassNode函数的保存顺序依次读取信息即可。

**时间复杂度：**算法中有一层循环，假设班级的结点数目为n,那么显然算法的时间复杂度为O(n)。

**空间复杂度：**算法所需要分配空间的大小为个别班级链遍历指针和文件指针的大小，因此空间复杂度为常量阶O(1)。

# 系统实现与测试

## 系统实现

4.1.1 软硬件环境

本程序编译环境为windows 10，利用Code::Blocks 17.12编译。

4.1.2 函数及功能

系统中包括哪些函数，各函数的说明请见表4-1,函数间的调用关系请见图4-1。由于各数据类型的重复度过高，该图仅包括查找模块，维护模块和文件模块中的课堂类型数据和分析模块中的所有数据

表4-1 函数功能展示及说明

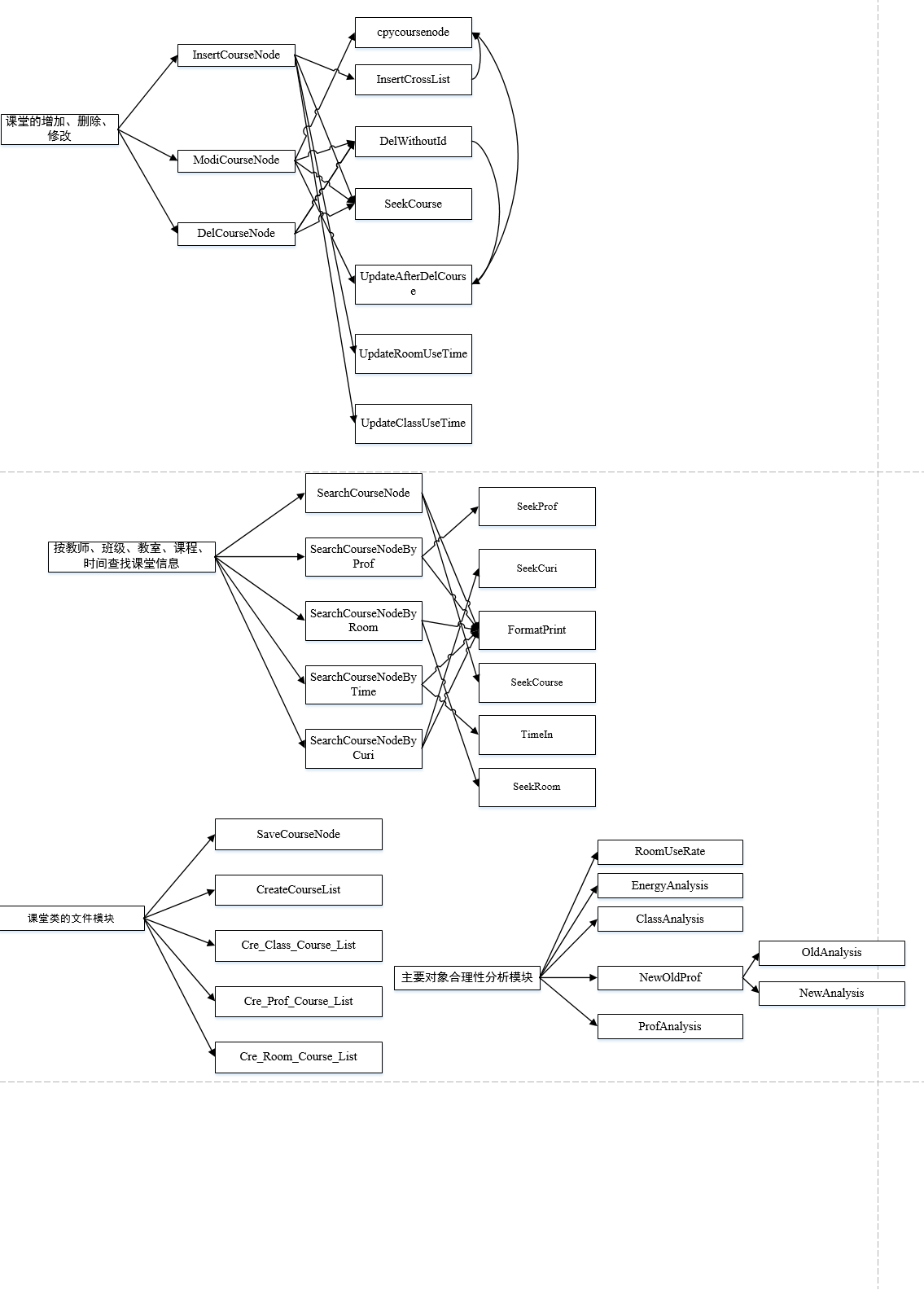


图4-1 函数调用关系图

## 系统测试

4.2.1 常用软件测试方法简述

目前常用的软件测试方法主要包括以下三类：

第一种软件测试方法**——**自动化测试

自动化测试包括回归测试和验收测试，鼓励对所有回归测试用例进行自动化测试。自动化测试就是使用自动化测试工具来进行测试，这类测试一般不需要人干预，通常在GUI、性能等测试和功能测试中用得较多。通过录制测试脚本，然后执行这个测试脚本来实现测试过程的自动化。

第二种软件测试方法**——**白盒测试

白盒测试是把测试对象看作一个打开的盒子。知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。利用白盒测试法进行动态测试时，需要测试软件产品的内部结构和处理过程，不需测试软件产品的功能。

白盒测试法的覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。其中逻辑覆盖包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖。

第三种软件测试方法**——**黑盒测试

黑盒测试又称功能测试或者数据驱动测试， 利用黑盒测试时，软件对用户来说就像一个黑盒子。根据软件的规格对软件进行的测试，这类测试不考虑软件内部的运作原理。软件测试人员以用户的角度，通过各种输入和观察软件的各种输出结果来发现软件存在的缺陷，而不关心程序具体如何实现的一种软件测试方法。

本次测试主要采用白盒测试和黑盒测试相结合的方法，在黑盒测试时又更为关注边界条件的测试。

4.2.2 模块功能及设计目标

本系统由四大功能模块组成：主要对象动态维护模块、主要对象查找模块、主要对象合理性分析模块、文件模块。

1. 主要对象动态维护模块的功能包括课堂、课程、教室、教师、班级的增加、删除和修改。设计目标是使得用户能够自行对课程表的内容进行维护，提高软件的灵活运用程度。

2. 主要对象查找模块功能包括教师、教室、班级、课程的查找，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，空闲教室的查找。设计目标是使得用户能够对主要对象进行便捷的查找工作，增加了软件的实用性。

3. 主要对象合理性分析模块功能包括教室利用率分析、能效分析，教师承担课堂合理性分析，班级参与课堂的合理性分析以及新老教师的特别分析。设计目标是帮助用户对现有课程表有个更好的认识，有助于课程表的修改。

4. 文件模块功能包括教师、课堂、课程、教室、班级的相关信息的文件写入和读取，根据数据对象的联系建立邻接表等。文件模块的设计目标是正确读取文件，保存文件，提高程序的健壮性。

4.2.3 模块功能展示及测试大纲

本次课程设计的功能展示如下图所示，由于各类型数据结构重合度较高，且技术含量较低，因此重复部分只举课堂相关的数据验证，合理性分析部分是本次课程设计的亮点，在此重点验证。功能展示见图4-2，主要功能测试大纲见表4-2，异常功能识别（健壮性）测试在4.2.5部分。

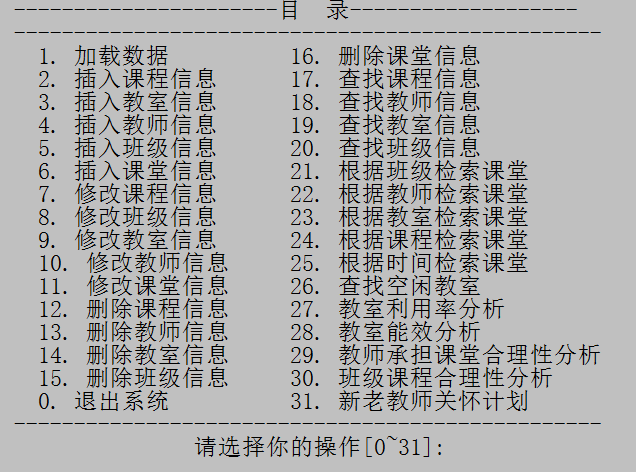


图4-2 功能展示图

表4-2 主要功能测试大纲



4.2.4 运行结果及分析

1. 测试功能及结果：

(1) 插入课堂（输入部分见图4-3，结果见图4-4）

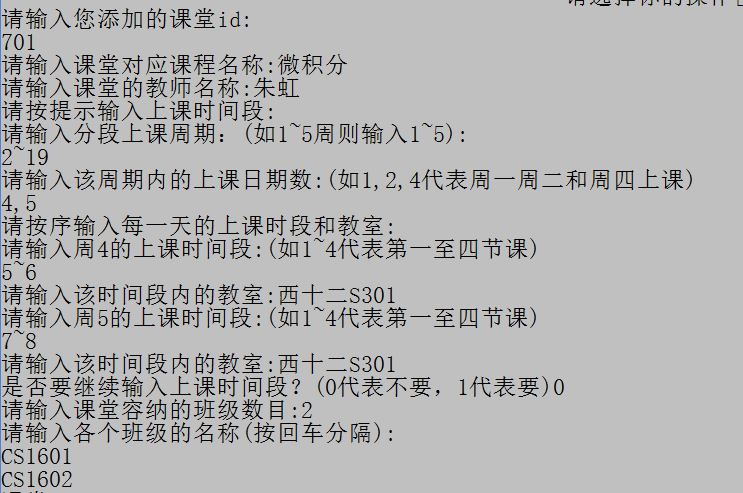


图4-3 插入课堂功能输入部分

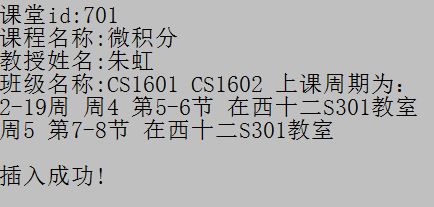


图4-4 插入课堂功能结果部分

结果分析: 根据图4-3和图4-4所示，插入课堂的信息与输入信息相符合，达到了设计目标。

2. 测试功能及结果:

(1) 修改课堂教师姓名

(2) 修改课堂701的教室为西十二S101

(3) 修改课堂701的上课班级为校交1601 CS1601（输入如图4-5，结果如图4-6）

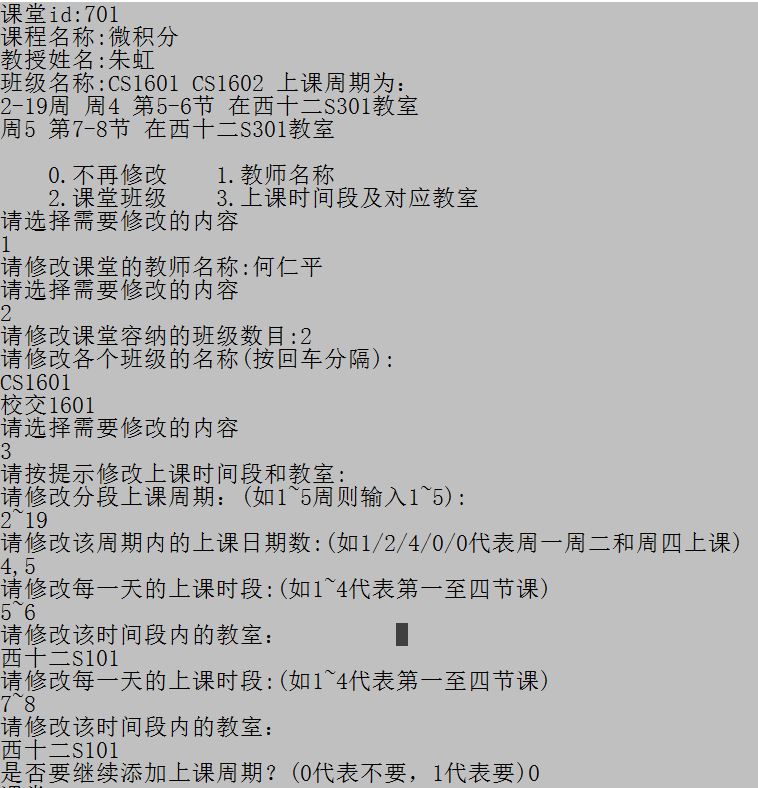


图4-5修改课堂功能输入部分

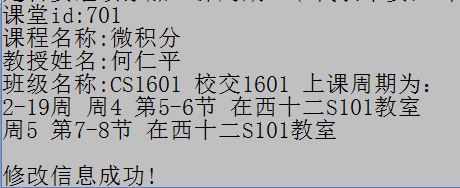


图4-6 修改课堂结果验证部分

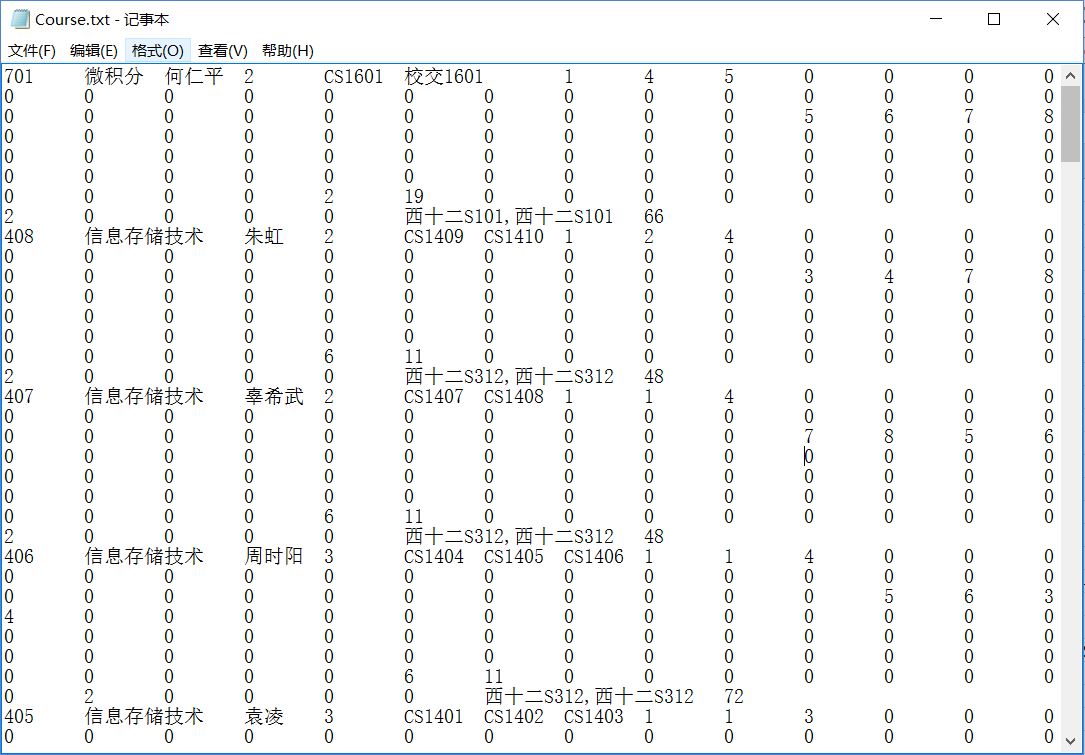


图4-7 修改课堂后文件状态图

结果分析：如图4-5和图4-6所示，对比图4-4，修改之后的信息和输入符合，并且和修改之前的区别符合预期情况，根据图4-7，文件中的701课堂已经修改完成，因此达到了设计目标。

3. 测试功能及结果:

(1) 根据班级检索课堂（输入为班级名称CS1601，结果如图4-8所示）

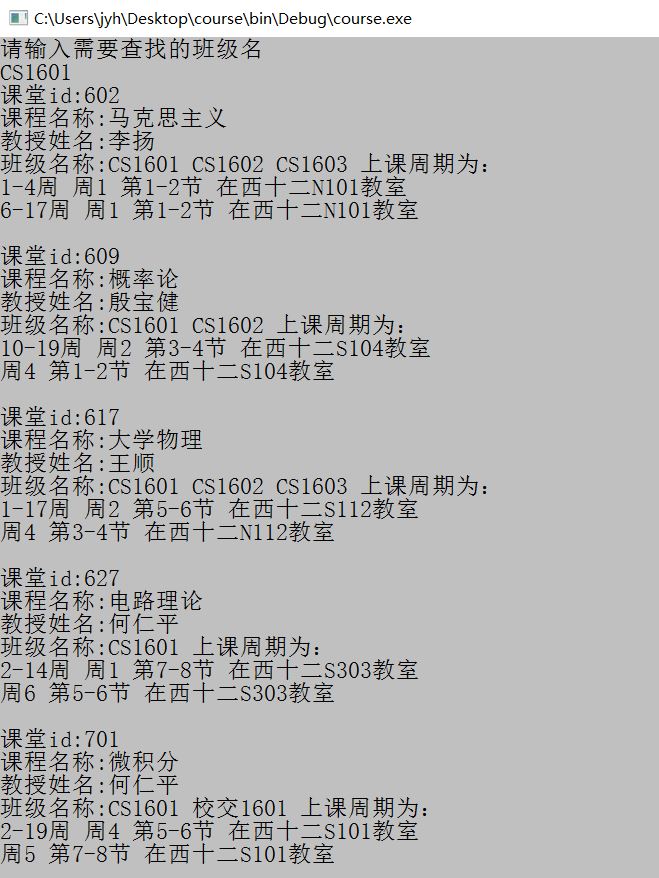


图4-8 根据班级检索课堂功能验证

结果分析：根据测试步骤1和2，此时的CS1601班应该参与课堂701，根据图4-8所示，CS1601班确实参与了701课堂，达到了设计目标。

4. 测试功能及结果:

(1) 根据教师检索课堂（结果如图4-9）

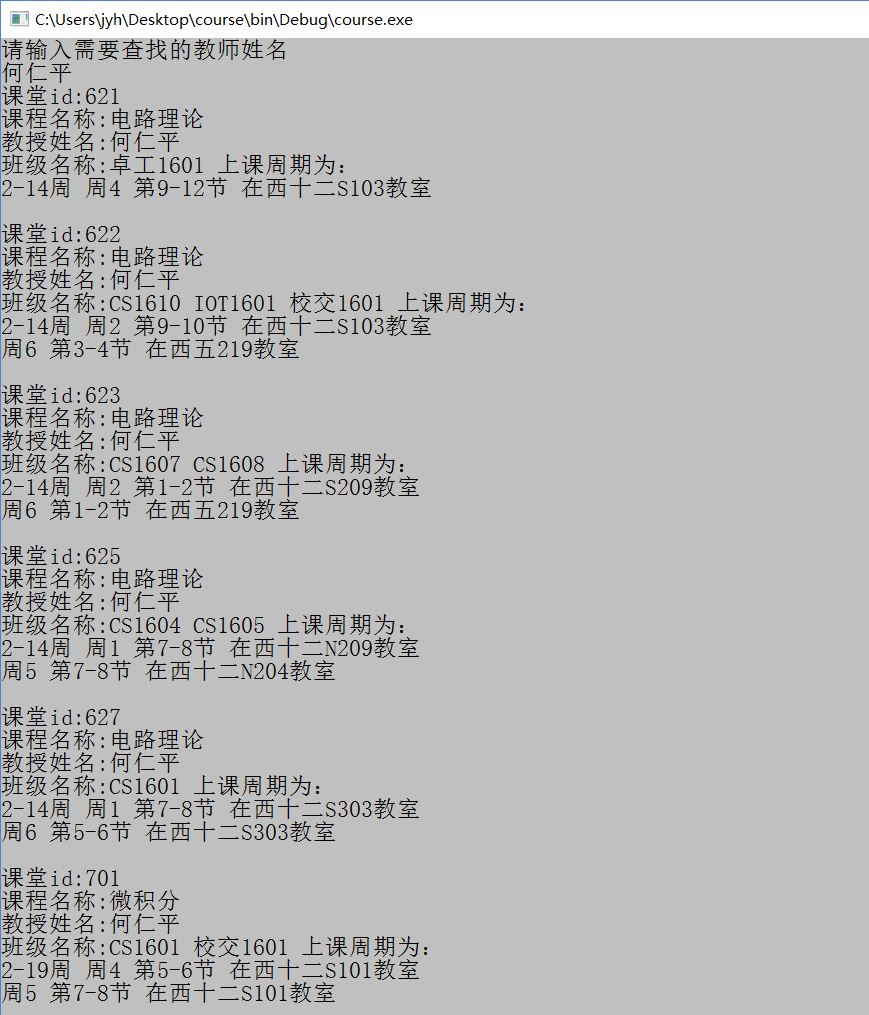


图4-9 根据教师检索课堂图

结果分析：根据测试步骤1和2，此时的何仁平应该教授课堂701，根据图498所示，何仁平确实教授了701课堂，达到了设计目标。

5. 测试功能及结果:

(1) 根据教室检索课堂（结果如图4-10）

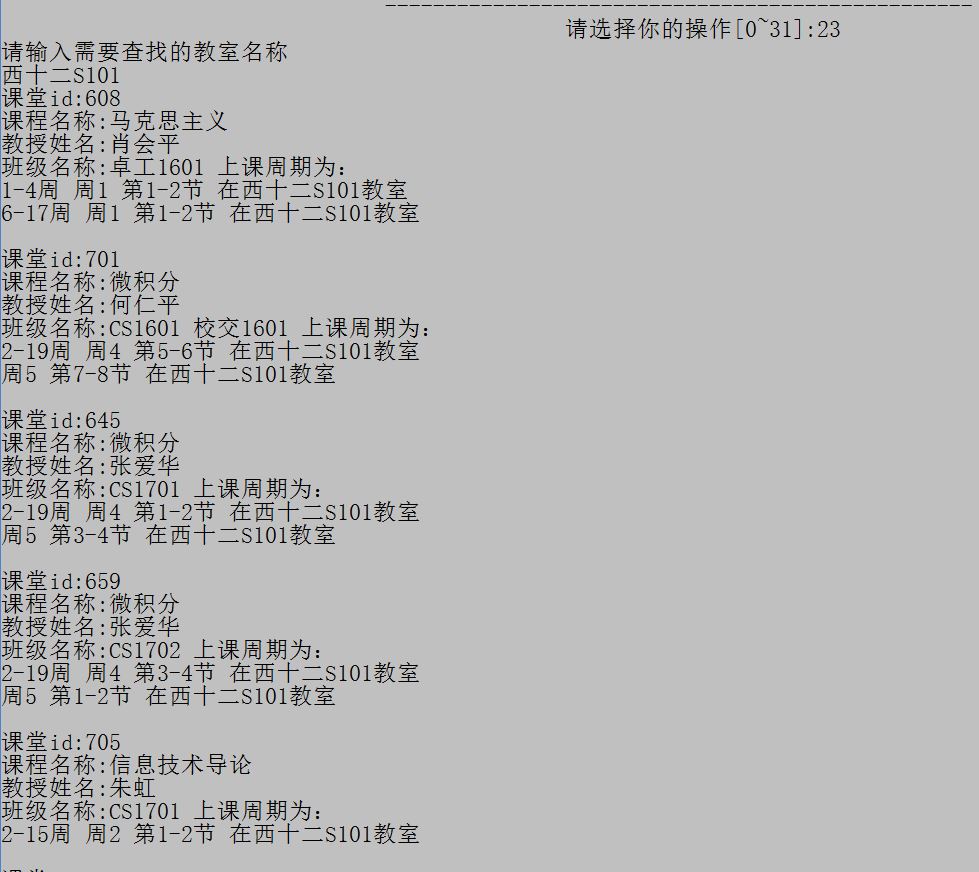


图4-10 根据教室检索课堂图

结果分析：根据测试步骤1和2，此时的西十二S101应该承担课堂701的任务，根据图4-10所示，西十二S101确实承担了701课堂的教学任务，达到了设计目标。

6. 测试功能及结果:

(1) 根据课程检索课堂（结果如图4-11所示）

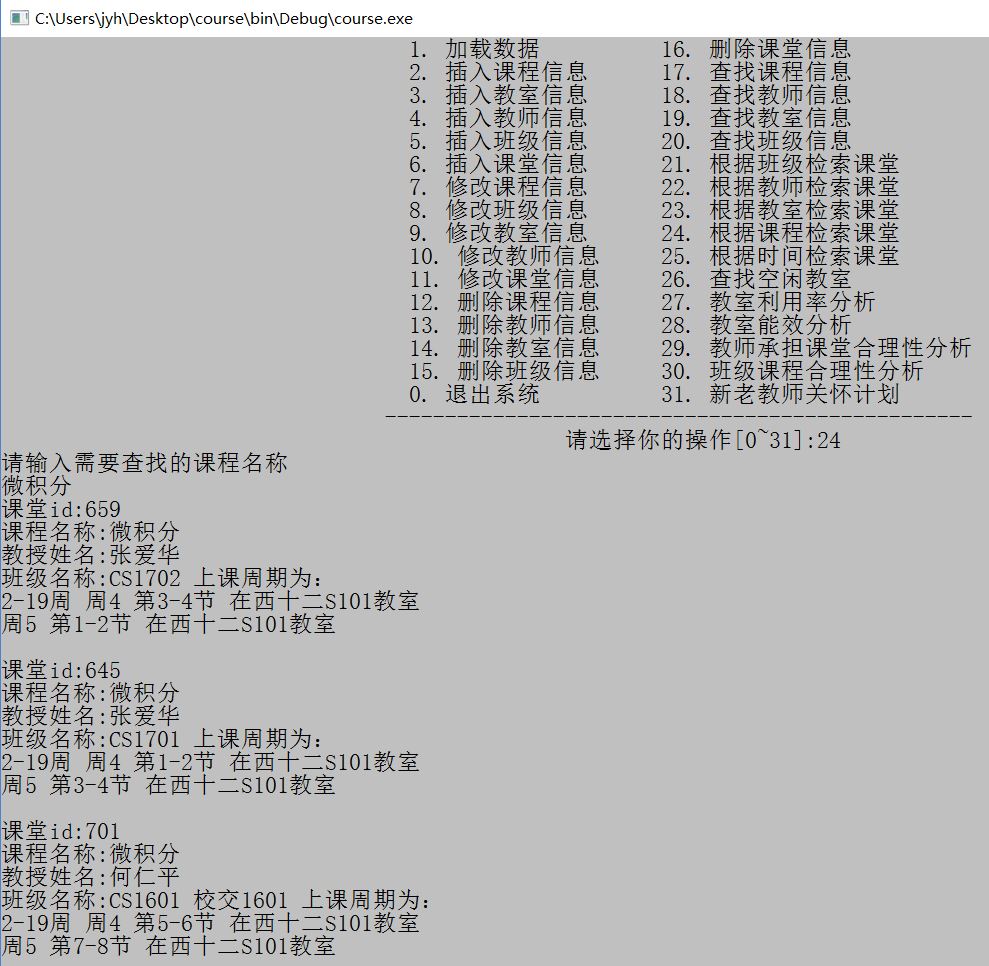


图4-11 根据课程检索课堂图

结果分析：根据测试步骤1和2，此时的微积分课程中应该包括课堂701，根据图4-11所示，微积分课程中确实包括了701课堂的教学任务，达到了设计目标。

7. 测试功能及结果:

(1) 根据时间检索课堂（结果如图4-12）

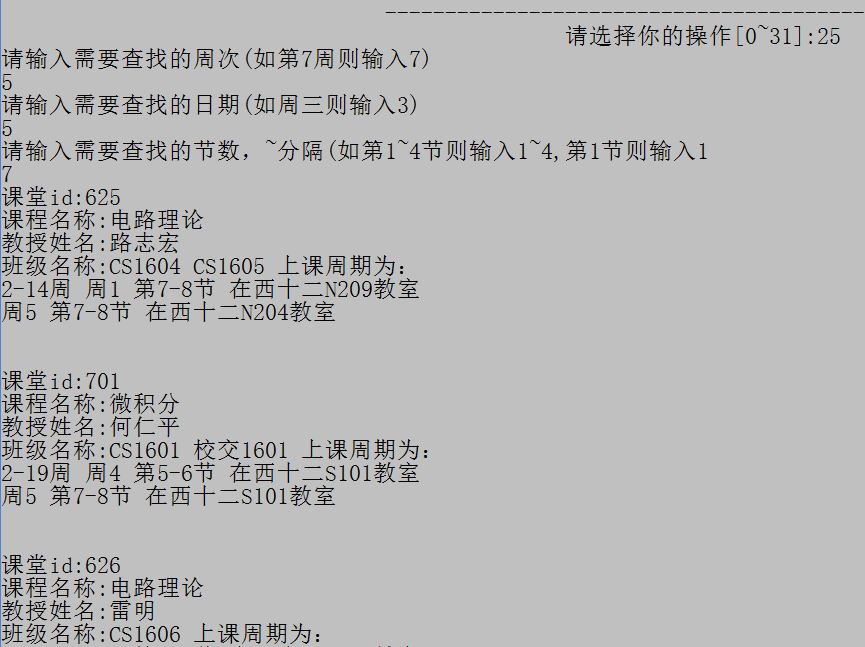


图4-12 根据时间检索课堂图

结果分析：根据测试步骤1和2，此时的输入的时间中应该包括课堂701，根据图4-11所示，该段时间的课堂中确实包括了701课堂，接着校对其余的课堂的时间，也与输入时间相符合，因此达到了设计目标。

8. 测试功能及结果:

(1) 查找空闲教室（结果如图4-13）

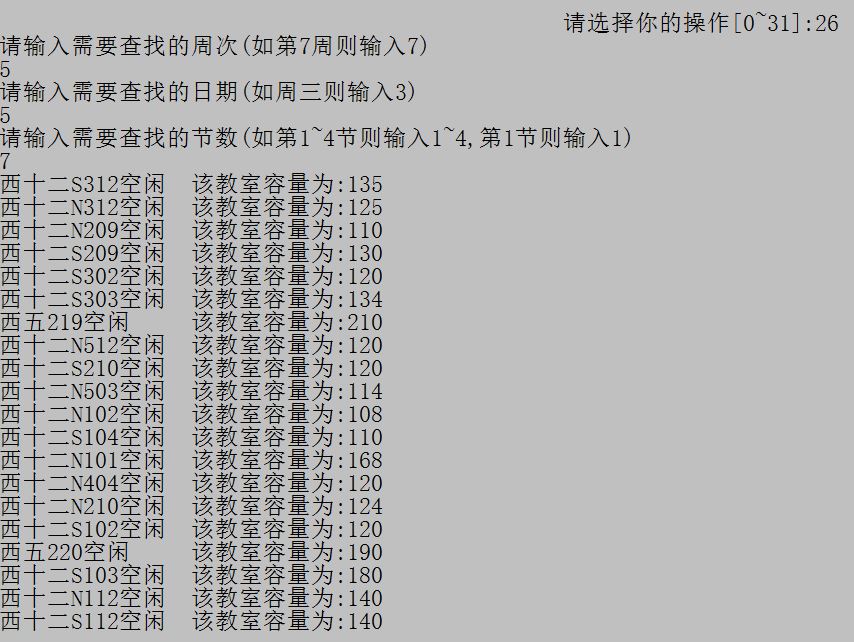


图4-13 查找空闲教室功能图

结果分析：根据测试步骤7，此时的输入的时间中西十二N204和西十二S101应该不空闲，根据图4-13所示，该段时间的空闲中确实不包括西十二N204和西十二S101，因此达到了设计目标。

9. 测试功能及结果:

(1) 教室利用率分析（结果如图4-14所示）

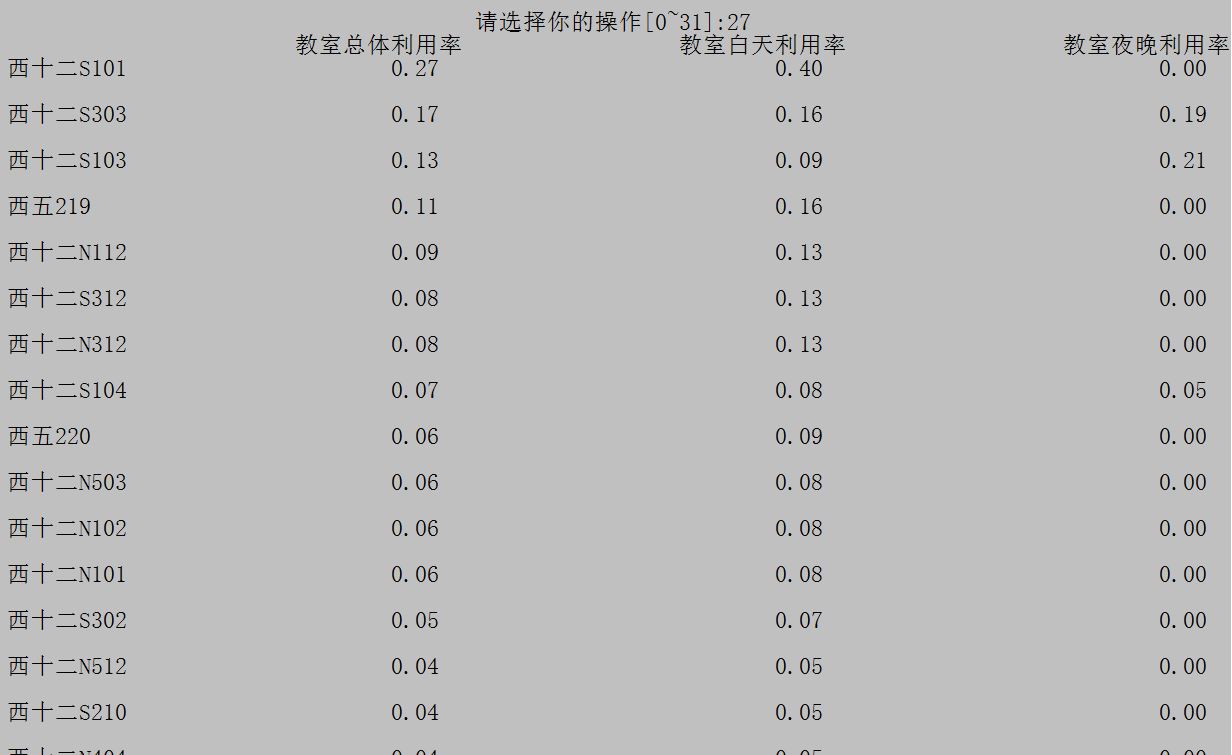


图4-14 教室利用率分析功能图

结果分析：根据测试步骤5，可以手工验证计算结果的正确性，根据图4-14所示，教室的利用率分析功能正确统计了每个教室的总体利用率，每个教室白天黑夜的分别利用率，因此达到了设计目标。

10. 测试功能及结果:

(1) 教室能效分析（结果如图4-15）

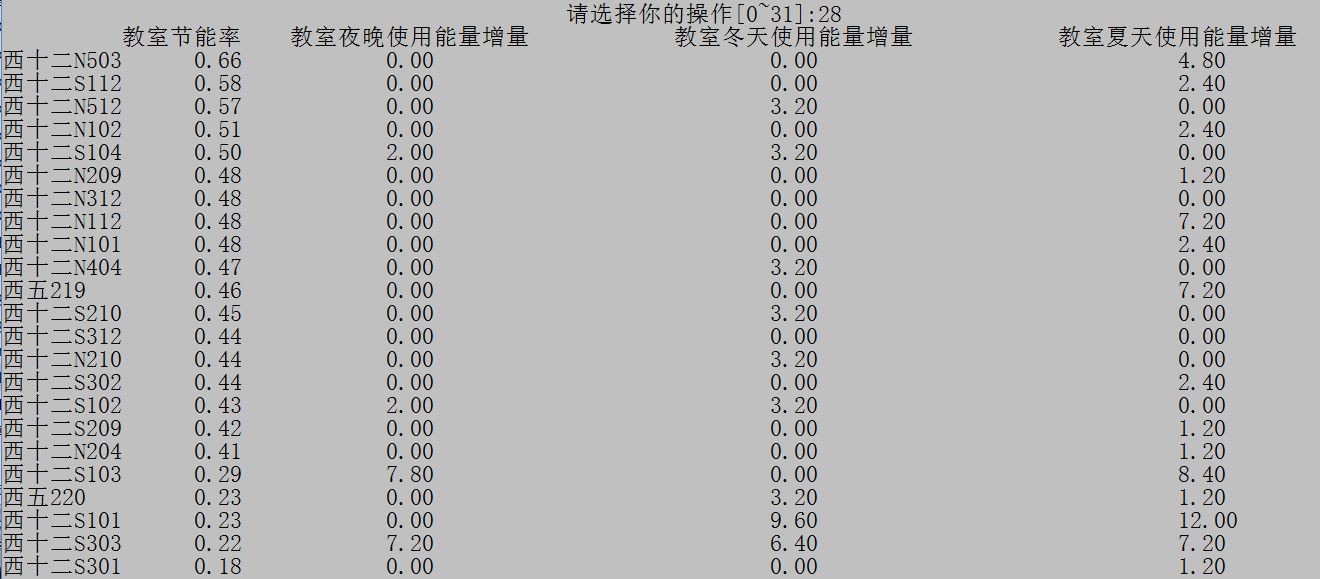


图4-15 教室能效分析功能图

结果分析：根据测试步骤5，可以手工验证计算结果的正确性，根据图4-15所示，教室的能效分析功能正确统计了每个教室的节能率，夜晚使用的电能增量，冬天使用的电能增量，夏天使用的电能增量，因此达到了设计目标。

11. 测试功能及结果:

(1) 教师承担课堂合理性分析（结果如图4-16）

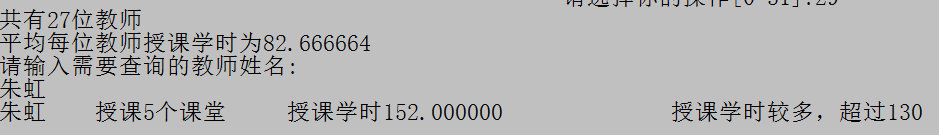


图4-16教师课堂合理性分析图

结果分析：根据测试步骤4，可以手工验证计算结果的正确性，根据图4-16所示，教师的承担课堂合理性分析主要分析了该教师的课堂数量和承担的教学学时数，在学时过多的情况下，确实会给出提示，因此达到了设计目标。

12. 测试功能及结果:

(1) 班级参与课堂合理性分析（结果如图4-17）

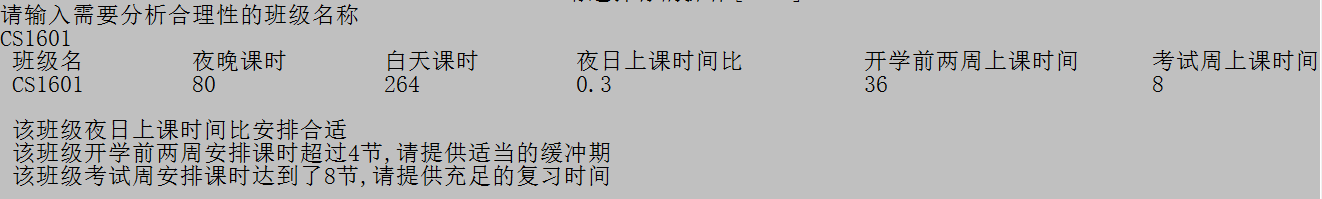


图4-17 班级参与课堂合理性分析图

结果分析：根据测试步骤3，可以手工验证计算结果的正确性，根据图4-17所示，班级的参与课堂合理性分析主要分析了该班级的夜日上课时间比和特殊时间段的上课时间，在夜日上课时间比过高或者特殊情况上课时间过多的情况下，确实会给出提示，因此达到了设计目标。

13. 测试功能及结果:

(1) 显示新老教师关怀名单供用户选择（如图4-18）：

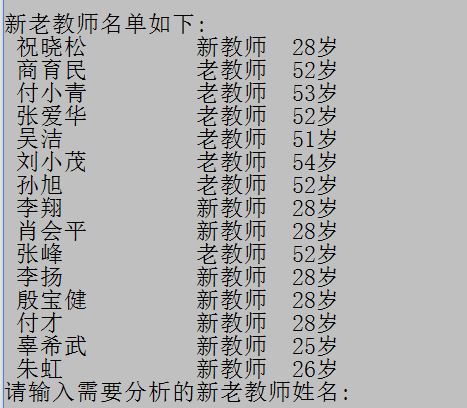


图4-18关怀教师名单图

(2) 新教师安排课堂合理性分析（结果如图4-19所示）

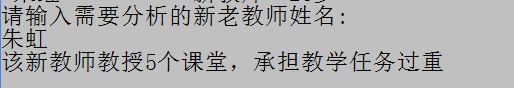


图4-19 关怀新教师功能图

(3) 老教师安排课堂合理性分析（结果如图4-20所示）

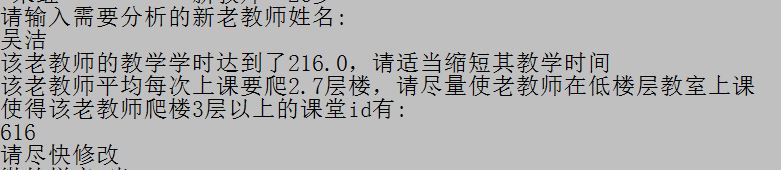


图4-20关怀老教师功能图

结果分析：根据图4-19和图4-20所示，新老教师关怀计划主要分析了新教师承担的课堂数目会不会过多，老教师承担的课堂学时是不是过多，每次上课爬的楼层是不是过高，如果安排不合适的话，确实会给出人性化的提示，因此达到了设计目标，此功能具有很强的实用性。

14. 测试功能及结果:

(1) 删除课堂信息功能（删除对象如图4-21所示，删除后文件状态如图4-22所示）

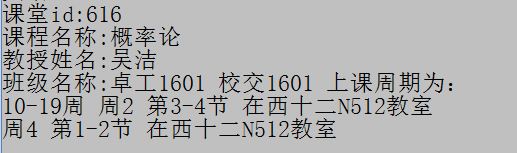


图4-21 删除课堂信息图



图4-22 删除课堂后的文件状态图

结果分析：根据文件状态，删除课堂616后，文件中便无法查找到616，因此删除成功，达到了设计目标。

4.2.5 程序健壮性分析及测试

本课程设计中功能的异常主要来自于四大方面:越界、已经存在或不存在问题以及时间冲突问题，由于数据对象较多，这四种问题可以在不同数据类型的不同功能中出现，因此只举四个典型例子供参考。

1. 越界问题：

(1) 具体描述:插入课堂，修改课堂功能中产生时间范围的越界。

(2) 健壮性分析:输入课堂的起始和终止节数一旦超过1~12的范围，就会被定义为越界，如图4-23所示。

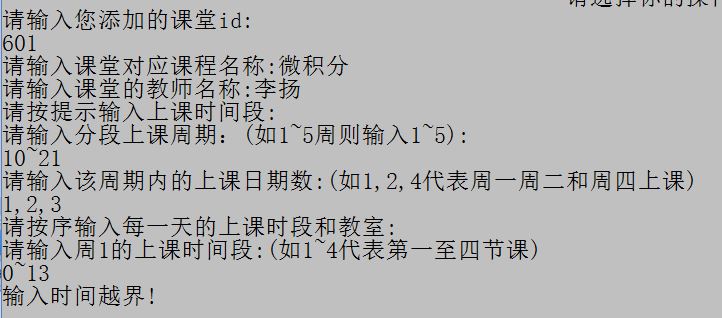


图4-23 时间越界问题

2. 已存在问题：

(1) 具体描述:插入课堂功能中产生的已经存在的教室、教师、课堂、课程、班级不能重复存在的冲突。

(2) 健壮性分析:输入的课堂id一旦存在，便不可以重复输入，如图4-24所示。

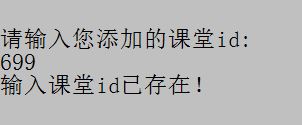


图4-24 信息已存在问题

3. 不存在问题：

(1) 具体描述:当插入某一项信息要涉及其他信息时，一定要确认这个其他信息也存在。

(2) 健壮性分析:插入课堂时输入的教师不存在，便不可以继续输入，如图4-25所示。

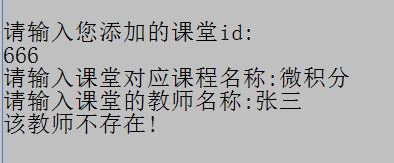


图4-25 信息不存在问题

4. 时间冲突问题：

(1) 具体描述:当插入课堂需要输入教室和教师时，很有可能他们在这个时间段内是有别的课堂的，如果有这个情况，那么就要进行提示。

(2) 健壮性分析:插入课堂时输入的时间内教室已经被占用，便不可以继续输入，如图4-26所示，根据教室检索课堂可以发现，如图4-27所示，这个时候的西十二S101教室确实是有课的。

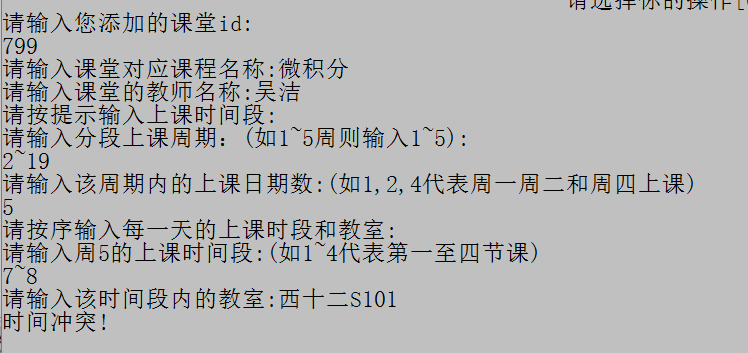


图4-26 时间冲突问题

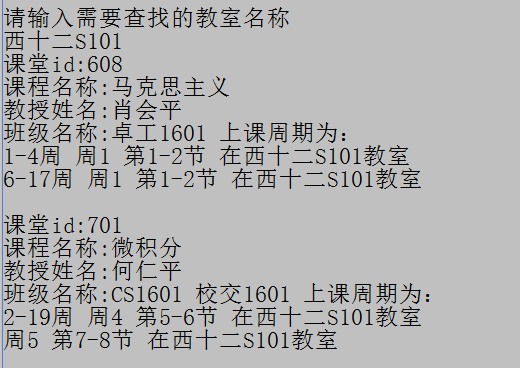


图4-27 验证时间冲突问题

# 总结与展望

## 全文总结

本次数据结构课程设计实验，本人主要做了以下工作：

1. 从文件中读取数据，并且以链表的形式构成了图的邻接表结构。
2. 完成了具有复杂联系的数据对象的增加、修改、删除工作，能够细致谨慎地处理各类数据，做到了更改一种数据，其他任何数据均正确动态更新。
3. 实现了具有很高使用价值的接地气的合理性分析功能，可以立即投入日常使用，各类功能考虑了利用率、能耗、日夜因素、冬夏因素、教师年龄因素、攀爬楼层因素，既具有环保意识，又带有人道关怀。
4. 完成了各类数据满足不同条件下的查找功能，没有查找遗漏或者查找冗余的情况出现。

## 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. 将程序界面完善得更好。
2. 优化算法，进一步降低程序复杂度。
3. 将邻接表更换为邻接矩阵、邻接多重表等图结构进行尝试。

# 体会

1. 代码命名规范性的提升与习惯的培养

这次课程设计开发难度大，所花时间长，这样的特点使我不得不改善我的编程规范性。例如，由于函数众多，各类数据结构也很多，我不可能像平时做实验一样随便起个Node,List就作为变量名，所以我尽量用英文单词组合的形式表现出函数的功能和各种数据结构成员变量的内涵。这显然是一个正确的良好的习惯，在编程的时候大大提升了我的效率，程序的可读性也进一步提升。

2. 测试函数的运用

本次课程设计的数据结构各数据类型之间关系的复杂是我从未遇到过的，总共有三个链表型邻接表和两个单链表，这也意味着每当我更新其中一个数据类型的时候，其余数据类型也需要经历更新操作。我希望每写完一个功能模块中的功能我就能迅速判断这个函数写的对不对。因此，测试函数是在大规模编程中一个很实用的手段，我写了遍历测试函数，来在我写完一个模块功能后输出结果。

**3. 大量函数的使用，减少代码段长度**

之前写实验的时候，大概整个数据结构里有多少个函数我就会写多少个函数，这次课程设计中除了根据模块定义的函数之外，我又写了很多辅助函数，在主要的模块函数中可以被调用。比如FormatPrint这个函数我就调用了很多次，TimeIn这个函数尽管值只调用了两次，但是起到了不同的作用，分别用于查找空闲教室和根据时间查找课堂。这些函数的大量使用使我的代码清晰易读，还减少了很多我自己的工作量。

**4. 尽量使程序具有实用价值**

本次课程设计，我设计了很多任务书之外的具有实用意义的功能，比如考试周，开学周排课，新老教师关怀计划等，这让我有一种成就感，我觉得自己不仅仅在完成一个任务，而是为很多的老师同学谋福利，这是我做这个感觉很开心的原因。

# 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

**main.cpp**

#include "course.h"

#include "Experiment2.h"

int main(void)

{

system("color 70");

int flag = 0;

InitaCuriList(&gp\_curi\_head); //基本ADT,初始化课程链头结点

InitaRoomList(&gp\_room\_head); //基本ADT,初始化教室链头结点

InitaClassList(&gp\_class\_head); //基本ADT,初始化班级链头结点

InitaProfList(&gp\_prof\_head); //基本ADT,初始化教师链头结点

InitaCourseList(&gp\_course\_head); //基本ADT,初始化课程链头结点

int op = 1;

while (op)

{

system("cls");

printf("\t\t\t\t----------------------目 录-------------------\n");

printf("\t\t\t\t-------------------------------------------------\n");

printf("\t\t\t 1. 加载数据 16. 删除课堂信息\n");

printf("\t\t\t 2. 插入课程信息 17. 查找课程信息\n");

printf("\t\t\t 3. 插入教室信息 18. 查找教师信息\n");

printf("\t\t\t 4. 插入教师信息 19. 查找教室信息\n");

printf("\t\t\t 5. 插入班级信息 20. 查找班级信息\n");

printf("\t\t\t 6. 插入课堂信息 21. 根据班级检索课堂\n");

printf("\t\t\t 7. 修改课程信息 22. 根据教师检索课堂\n");

printf("\t\t\t 8. 修改班级信息 23. 根据教室检索课堂\n");

printf("\t\t\t 9. 修改教室信息 24. 根据课程检索课堂\n");

printf("\t\t\t 10. 修改教师信息 25. 根据时间检索课堂\n");

printf("\t\t\t 11. 修改课堂信息 26. 查找空闲教室\n");

printf("\t\t\t 12. 删除课程信息 27. 教室利用率分析\n");

printf("\t\t\t 13. 删除教师信息 28. 教室能效分析\n");

printf("\t\t\t 14. 删除教室信息 29. 教师承担课堂合理性分析\n");

printf("\t\t\t 15. 删除班级信息 30. 班级课程合理性分析\n");

printf("\t\t\t 0. 退出系统 31. 新老教师关怀计划\n");

printf("\t\t\t\t-------------------------------------------------\n");

printf("\t\t\t\t 请选择你的操作[0~31]:");

scanf("%d", &op);

switch (op)

{

case 1:

flag = 1;

CreateRoomList(gp\_room\_head);

printf("教室信息加载成功\n");

CreateCourseList(gp\_course\_head);

printf("课堂信息加载成功\n");

CreateCuriList(gp\_curi\_head);

printf("课程信息加载成功\n");

CreateClassList(gp\_class\_head);

printf("班级信息加载成功\n");

CreateProfList(gp\_prof\_head);

printf("教师信息加载成功\n");

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

printf("教师-课堂邻接表创建成功\n");

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

printf("班级-课堂邻接表创建成功\n");

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

printf("教室-课堂邻接表创建成功\n");

getchar();

getchar();

break;

case 2:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

InsertCuriNode(gp\_curi\_head); //插入课程结点

SaveCuriNode(gp\_curi\_head); //保存课程结点

getchar();

getchar();

break;

case 3:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

InsertRoomNode(gp\_room\_head); //插入教室结点

SaveRoomNode(gp\_room\_head); //保存教室结点

getchar();

getchar();

break;

case 4:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

InsertProfNode(gp\_prof\_head); //插入教师结点

SaveProfNode(gp\_prof\_head); //保存教师结点

getchar();

getchar();

break;

case 5:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

InsertClassNode(gp\_class\_head); //插入教室结点

SaveClassNode(gp\_class\_head); //保存班级信息

getchar();

getchar();

break;

case 6:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

InsertCourseNode(gp\_course\_head); //插入课堂信息

SaveCourseNode(gp\_course\_head); //保存课堂信息

getchar();

getchar();

break;

case 7:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ModiCuriNode(gp\_curi\_head); //修改课程信息

SaveCuriNode(gp\_curi\_head); //保存课程信息

getchar();

getchar();

break;

case 8:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ModiClassNode(gp\_class\_head); //修改班级信息

SaveClassNode(gp\_class\_head); //保存班级信息

getchar();

getchar();

break;

case 9:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ModiRoomNode(gp\_room\_head); //修改教室信息

SaveRoomNode(gp\_room\_head); //保存教室信息

getchar();

getchar();

break;

case 10:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ModiProfNode(gp\_prof\_head); //修改教师信息

SaveProfNode(gp\_prof\_head); //保存教师信息

getchar();

getchar();

break;

case 11:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

if(!ModiCourseNode(gp\_course\_head)) //如果修改失败，信息回滚

{

InfoReturn();

}

//保存其余信息

SaveClassNode(gp\_class\_head);

SaveCourseNode(gp\_course\_head);

SaveCuriNode(gp\_curi\_head);

SaveRoomNode(gp\_room\_head);

getchar();

getchar();

break;

case 12:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

DelCuriNode(gp\_curi\_head); //删除课程信息

SaveCuriNode(gp\_curi\_head);

SaveCourseNode(gp\_course\_head);

getchar();

getchar();

break;

case 13:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

DelProfNode(gp\_prof\_head); //删除教师信息

SaveProfNode(gp\_prof\_head);

getchar();

getchar();

break;

case 14:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

DelRoomNode(gp\_room\_head); //删除教室信息

SaveRoomNode(gp\_room\_head);

getchar();

getchar();

break;

case 15:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

DelClassNode(gp\_class\_head); //删除班级信息

SaveClassNode(gp\_class\_head);

SaveCourseNode(gp\_course\_head);

getchar();

getchar();

break;

case 16:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

DelCourseNode(gp\_course\_head); //删除课堂信息

SaveCourseNode(gp\_course\_head);

getchar();

getchar();

break;

case 17:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

//更新创建邻接表

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCuriNode(gp\_curi\_head); //搜索课程结点

getchar();

getchar();

break;

case 18:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchProfNode(gp\_prof\_head); //搜索教师结点

getchar();

getchar();

break;

case 19:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

//更新邻接表结点

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchRoomNode(gp\_room\_head); //搜索教室结点

getchar();

getchar();

break;

case 20:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

//更新邻接表

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchClassNode(gp\_class\_head); //搜索班级信息

getchar();

getchar();

break;

case 21:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

//更新邻接表

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCourseNodeByClass(gp\_class\_head); //搜索课堂信息

getchar();

getchar();

break;

case 22:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCourseNodeByProf(gp\_prof\_head); //通过教师查找课堂

getchar();

getchar();

break;

case 23:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCourseNodeByRoom(gp\_room\_head); //通过教室查找课堂

getchar();

getchar();

break;

case 24:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCourseNodeByCuri(gp\_curi\_head);

getchar();

getchar();

break;

case 25:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

SearchCourseNodeByTime(gp\_course\_head);

getchar();

getchar();

break;

case 26:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

FindFreeRoom(gp\_room\_head);

getchar();

getchar();

break;

case 27:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

RoomUseRate(gp\_room\_head);

getchar();

getchar();

break;

case 28:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

EnergyAnalysis(gp\_room\_head);

getchar();

getchar();

break;

case 29:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

ProfAnalysis(gp\_prof\_head);

getchar();

getchar();

break;

case 30:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

ClassAnalysis(gp\_class\_head);

getchar();

getchar();

break;

case 31:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

CreateRoomList(gp\_room\_head);

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CreateClassList(gp\_class\_head);

CreateProfList(gp\_prof\_head);

Cre\_Prof\_Course\_List(gp\_prof\_head, gp\_course\_head);

Cre\_Class\_Course\_List(gp\_class\_head, gp\_course\_head);

Cre\_Room\_Course\_List(gp\_room\_head, gp\_course\_head);

NewOldProf(gp\_prof\_head);

getchar();

getchar();

break;

case 32:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ListClassTrabverse(gp\_class\_head);

getchar();

getchar();

break;

case 33:

if(flag==0)

{

printf("尚未加载数据!");

getchar();

getchar();

break;

}

ListCourseTrabverse(gp\_course\_head);

getchar();

getchar();

break;

case 0:

Exit();

break;

}//end of switch

}//end of while

printf("欢迎下次再使用本系统！\n");

}//end of main()

/\*\*

\* 函数名称：CreateCuriList

\* 函数参数：curi\_hd 课程链头结点指针

\* 函数功能：从txt文件中读取课程链信息，创建课程链链表

\* 返回值：创建链表成功返回课程链头结点指针, 失败则返回NULL

\*\*/

CURI\_NODE\* CreateCuriList(CURI\_NODE \*curi\_hd)

{

int ret = 0;

CURI\_NODE \* pcuri\_node=NULL, \* pprior\_curi\_node=NULL; //遍历指针和前驱指针

FILE \*curi\_write = fopen("Curi.txt", "r+");

if(!curi\_write)

{

printf("打开文件失败！\n");

fclose(curi\_write);

return NULL;

}

pprior\_curi\_node = curi\_hd;

while (!feof(curi\_write))

{

pcuri\_node = (CURI\_NODE\*)malloc(sizeof(CURI\_NODE));

ret = fscanf(curi\_write, "%s", pcuri\_node->curi\_name);

if(ret == -1) break; //防止fscanf多读取一项

fscanf(curi\_write, "%f", &pcuri\_node->credit);

fscanf(curi\_write, "%f", &pcuri\_node->hours);

fscanf(curi\_write, "%d", &pcuri\_node->course\_num);

//调整链表结构

pprior\_curi\_node->next = pcuri\_node;

pprior\_curi\_node = pprior\_curi\_node->next;

pcuri\_node->next = NULL;

}

fclose(curi\_write); //关闭文件

return pcuri\_node;

}

/\*\*

\* 函数名称：ProfAnalysis

\* 函数参数：head 教师链头结点指针

\* 函数功能：分析教师课堂安排合理性

\* 返回值：无

\*\*/

void ProfAnalysis(PROF\_NODE \* head)

{

char prof\_name[12];

float avg\_hours = 0; //所有教师平均教学时长

float sum\_hours = 0; //所有教师的所有教学时长

int prof\_num = 0; //教师人数

PROF\_NODE \*pprof\_node = head->next, \*prof\_temp = NULL;

//遍历链表，统计相关信息

while(pprof\_node)

{

sum\_hours+=pprof\_node->prof\_hours;

prof\_num++; //教师人数自增

pprof\_node = pprof\_node->next;

}

avg\_hours = sum\_hours/prof\_num; //统计平均授课学时

printf("共有%d位教师\n", prof\_num); //输出教师总人数

printf("平均每位教师授课学时为%f\n", avg\_hours);

printf("请输入需要查询的教师姓名:\n");

getchar();

scanf("%s", prof\_name);

if(!(prof\_temp=SeekProf(gp\_prof\_head, prof\_name)))

{

printf("该教师不存在\n");

return;

}

//遍历并且输出信息

printf("%s\t授课%d个课堂\t授课学时%f\t", prof\_temp->prof\_name, prof\_temp->prof\_num\_course, prof\_temp->prof\_hours);

if(prof\_temp->prof\_hours>130)

printf("\t授课学时较多，超过130\n");

else

printf("\t该教师分配的任务较合理\n");

return;

}

/\*\*

\* 函数名称：ClassAnalysis

\* 函数参数：head 班级链头结点指针

\* 函数功能：分析班级课堂安排合理性

\* 返回值：无

\*\*/

void ClassAnalysis(CLASS\_NODE \*head)

{

CLASS\_NODE \*pclass\_node=head->next, \*class\_temp=NULL;

char class\_name[30];

printf("请输入需要分析合理性的班级名称\n");

scanf("%s", class\_name);

//搜寻班级结点

if(!(class\_temp=SeekClass(gp\_class\_head, class\_name)))

{

printf("该班级不存在!\n");

return;

}

printf(" 班级名\t\t夜晚课时\t白天课时\t夜日上课时间比\t\t开学前两周上课时间\t考试周上课时间\n");

//遍历计算每个班级的信息

while(pclass\_node)

{

pclass\_node->rate = pclass\_node->night\_class\_hours\*1.0/pclass\_node->day\_class\_hours;

pclass\_node = pclass\_node->next;

}

printf(" %-12s\t%-8d\t%-8d\t%-8.1f\t\t%-8d\t\t%-8d\t\t\n", class\_temp->class\_name, class\_temp->night\_class\_hours, class\_temp->day\_class\_hours, class\_temp->rate, class\_temp->summer\_class\_hours, class\_temp->winter\_class\_hours);

if(class\_temp->rate>0.5)

printf(" 该班级夜晚安排课程过多\n");

else

printf(" 该班级夜日上课时间比安排合适\n");

if(class\_temp->summer\_class\_hours>4)

{

printf(" 该班级开学前两周安排课时超过4节,请提供适当的缓冲期\n");

}

if(class\_temp->winter\_class\_hours>4)

{

printf(" 该班级考试周安排课时达到了%d节,请提供充足的复习时间\n", class\_temp->winter\_class\_hours);

}

else if(class\_temp->summer\_class\_hours<=4)

printf(" 该班级在开学阶段和期末阶段的课程安排较为合理\n");

return;

}

/\*\*

\* 函数名称：RoomUseRate

\* 函数参数：head 教室链头结点指针

\* 函数功能：分析教室利用率并排序输出

\* 返回值：无

\*\*/

void RoomUseRate(ROOM\_NODE \* head)

{

ROOM\_NODE \*proom\_node = head->next;

//遍历教室链表，统计各教室的白天夜晚和总体使用率

while(proom\_node)

{

proom\_node->day\_use\_rate = proom\_node->day\_use\_time\*1.0/(DAY\_CLASS);

proom\_node->night\_use\_rate = proom\_node->night\_use\_time\*1.0/(NIGHT\_CLASS);

proom\_node->total\_use\_rate = proom\_node->total\_use\_time\*1.0/(TOTAL\_CLASS);

proom\_node = proom\_node->next;

}

ROOM\_NODE \*tail = proom\_node;

QuickSortRate(head, tail); //根据教室总体利用率排序

//遍历链表，输出信息

proom\_node = head->next;

printf("\t\t\t\t教室总体利用率\t\t\t教室白天利用率\t\t\t教室夜晚利用率\n");

while(proom\_node)

{

printf("\t%-12s\t\t\t%-8.2f\t\t\t%-8.2f\t\t\t%-8.2f\n", proom\_node->room\_name, proom\_node->total\_use\_rate, proom\_node->day\_use\_rate, proom\_node->night\_use\_rate);

proom\_node = proom\_node->next;

}

return;

}

/\*\*

\* 函数名称：EnergyAnalysis

\* 函数参数：head 教室链头结点指针

\* 函数功能：分析教室能效情况

\* 返回值：无

\*\*/

void EnergyAnalysis(ROOM\_NODE \*head)

{

ROOM\_NODE \*proom\_node = head->next;

//统计各个教室的平均能效值

while(proom\_node)

{

int course\_num = 0; //统计某个教室的课程数量

float energy\_sum = 0; //统计某个教室各个课堂的能效值之和

COURSE\_NODE \*pcourse\_node = proom\_node->firstarc;

while(pcourse\_node)

{

course\_num++;

energy\_sum += pcourse\_node->total\_stu\_num\*1.0/proom\_node->volume;

pcourse\_node = pcourse\_node->next;

}

proom\_node->energy = energy\_sum/course\_num;

if(proom\_node->energy>1)

{

proom\_node->energy = 1;

}

proom\_node = proom\_node->next;

}

ROOM\_NODE \*tail = proom\_node;

QuickSortEnergy(head, tail); //根据能效值排序

proom\_node = head->next;

//遍历输出各教室的能效值

printf("\t 教室节能率\t教室夜晚使用能量增量\t\t教室冬天使用能量增量\t\t教室夏天使用能量增量\n");

while(proom\_node)

{

printf("%-12s\t%-8.2f\t%-8.2f\t\t\t%-8.2f\t\t\t %-8.2f\n", proom\_node->room\_name, proom\_node->energy, proom\_node->night\_use\_time\*NIGHT\_INCREMENT, proom\_node->winter\_use\_time\*WINTER\_INCREMENT, proom\_node->summer\_use\_time\*SUMMER\_INCREMENT);

proom\_node = proom\_node->next;

}

return;

}

/\*\*

\* 函数名称：QuickSortEnergy

\* 函数参数：head 教室链头结点指针

tail 教室链尾结点指针

\* 函数功能：将教室结点链根据能效值重新排序

\* 返回值：无

\*\*/

void QuickSortEnergy(ROOM\_NODE \*head, ROOM\_NODE \*tail)

{

if (head->next==tail||head->next->next==tail)

return;

ROOM\_NODE\* mid = head->next; //mid指向标准值

ROOM\_NODE\* p = head; //指向比标准值大的值

ROOM\_NODE\* q = mid; //指向比标准值小的值

float pivot = mid->energy;

ROOM\_NODE\* t=mid->next;

while (t!=tail)

{

if (t->energy>pivot) //将t所指结点串入p链中

p = p->next = t;

else

q = q->next = t; //将t所指结点串入q链中

t = t->next;

}

p->next = mid;

q->next = tail;

QuickSortEnergy(head, mid);

QuickSortEnergy(mid, tail);

}

/\*\*

\* 函数名称：QuickSortRate

\* 函数参数：head 教室链头结点指针

tail 教室链尾结点指针

\* 函数功能：将教室链结点根据利用率重新排序

\* 返回值：无

\*\*/

void QuickSortRate(ROOM\_NODE \*head, ROOM\_NODE \*tail)

{

if (head->next==tail||head->next->next==tail)

return;

ROOM\_NODE\* mid = head->next; //mid指向标准值

ROOM\_NODE\* p = head; //指向比标准值大的值

ROOM\_NODE\* q = mid; //指向比标准值小的值

float pivot = mid->total\_use\_rate;

ROOM\_NODE\* t = mid->next; //t为遍历指针

while (t!=tail)

{

if (t->total\_use\_rate>pivot)

p = p->next = t;

else

q = q->next = t;

t = t->next;

}

p->next = mid; //将两段链串起来

q->next = tail;

QuickSortRate(head, mid);

QuickSortRate(mid, tail);

}

/\*\*

\* 函数名称：FindFreeRoom

\* 函数参数：room\_hd 教室链头结点指针

\* 函数功能：根据输入的时间或者时间段查找空闲的教室

\* 返回值：无

\*\*/

void FindFreeRoom(ROOM\_NODE \*room\_hd)

{

int week, date, start\_class, end\_class,free;

ROOM\_NODE \*proom\_node = room\_hd->next, \*room\_temp;

COURSE\_NODE \* pcourse\_node;

char classes[10]; //待输入的查找节数字符串

printf("请输入需要查找的周次(如第7周则输入7)\n");

getchar();

scanf("%d", &week);

printf("请输入需要查找的日期(如周三则输入3)\n");

getchar();

scanf("%d", &date);

printf("请输入需要查找的节数(如第1~4节则输入1~4,第1节则输入1)\n");

getchar();

scanf("%s", classes);

if(!DealString1(classes, &start\_class, &end\_class))

{

printf("输入有误!\n");

return;

}

//遍历教室信息链

while(proom\_node)

{

free = 1;

pcourse\_node = proom\_node->firstarc;

//遍历每个教室后每个课堂信息结点

while(pcourse\_node)

{

//printf("%s\n", pcourse\_node->course\_id);

room\_temp = TimeIn(pcourse\_node, week, date, start\_class, end\_class); //找到特定时间段某个课堂所在的教室

if(!room\_temp);

else if(strcmp(room\_temp->room\_name,proom\_node->room\_name)==0) //如果找到的教室恰好是正在遍历的教室结点

{

free = 0;

break;

}

pcourse\_node = pcourse\_node->next;

}

if(free)

printf("%s空闲\t该教室容量为:%d\n", proom\_node->room\_name, proom\_node->volume);

proom\_node = proom\_node->next;

}

getchar();

return;

}

/\*\*

\* 函数名称：DealString1

\* 函数参数：String 待分解字符串

\* 函数功能：解析包含课堂起止节数的字符串信息

\* 返回值：解析成功返回true, 否则返回false

\*\*/

bool DealString1(char \*String, int \*start\_class, int \*end\_class)

{

char \*ad;

char \*temp = String;

int all\_digit;

\*start\_class = \*end\_class = 0;

if((ad=strstr(String,"~"))&&isdigit(\*(ad-1))&&isdigit(\*(ad+1)))

{

char \*split;

split = strtok(String, "~"); //以~为分隔符分隔字符串

\*start\_class = atoi(split);

split = strtok(NULL, "~");

\*end\_class = atoi(split);

return true;

}

while(\*temp)

{

if(!isdigit(\*temp))

return false;

temp++;

}

\*start\_class = \*end\_class = atoi(String);

return true;

}

/\*\*

\* 函数名称：CreateRoomList

\* 函数参数：room\_hd 课程链首结点指针

\* 函数功能：从txt文件中读取信息，创建教室链表

\* 返回值：创建成功返回教室链表头结点，否则返回NULL

\*\*/

ROOM\_NODE\* CreateRoomList(ROOM\_NODE \*room\_hd)

{

int ret = 0;

char room\_name[30];

ROOM\_NODE \* proom\_node = NULL, \* pprior\_room\_node = NULL;

FILE \*room\_write = fopen("Room.txt", "r+");

if(!room\_write)

{

printf("打开文件失败！\n");

fclose(room\_write);

return NULL;

}

pprior\_room\_node = room\_hd; //前驱指针指向头结点

while (!feof(room\_write))

{

ret = fscanf(room\_write, "%s", room\_name);

if(ret == -1) break; //防止fscanf多读取一项

proom\_node = (ROOM\_NODE\*)malloc(sizeof(ROOM\_NODE)); //为新的教室结点分配空间

strcpy(proom\_node->room\_name, room\_name);

fscanf(room\_write, "%d", &proom\_node->summer\_use\_time);

fscanf(room\_write, "%d", &proom\_node->winter\_use\_time);

fscanf(room\_write, "%d", &proom\_node->day\_use\_time);

fscanf(room\_write, "%d", &proom\_node->night\_use\_time);

fscanf(room\_write, "%d", &proom\_node->total\_use\_time);

fscanf(room\_write, "%d", &proom\_node->volume);

pprior\_room\_node->next = proom\_node;

pprior\_room\_node = pprior\_room\_node->next;

//printf("%s\n", proom\_node->room\_name);

proom\_node->next = NULL;

proom\_node->firstarc = NULL;

}

fclose(room\_write); //关闭文件

return proom\_node;

}

/\*\*

\* 函数名称：CreateProfList

\* 函数参数：prof\_hd 教师链首结点指针

\* 函数功能：从txt文件中读取信息，创建教师链表

\* 返回值：创建成功返回教师链表头结点，否则返回NULL

\*\*/

PROF\_NODE\* CreateProfList(PROF\_NODE \*prof\_hd)

{

int ret = 0;

char prof\_name[30];

char prof\_title[30];

PROF\_NODE \* pprof\_node = NULL, \* pprior\_prof\_node = NULL; //遍历结点指针和前驱指针

FILE \*prof\_write = fopen("Prof.txt", "r+");

if(!prof\_write)

{

printf("打开文件失败！\n");

fclose(prof\_write);

return NULL;

}

pprior\_prof\_node = prof\_hd;

while (!feof(prof\_write))

{

ret = fscanf(prof\_write, "%s", prof\_name);

if(ret == -1) break; //防止fscanf多读取一项

fscanf(prof\_write, "%s", prof\_title);

pprof\_node = (PROF\_NODE\*)malloc(sizeof(PROF\_NODE));

fscanf(prof\_write, "%f", &pprof\_node->prof\_hours);

fscanf(prof\_write, "%d", &pprof\_node->prof\_num\_course);

fscanf(prof\_write, "%d", &pprof\_node->age);

strcpy(pprof\_node->prof\_name, prof\_name);

strcpy(pprof\_node->prof\_title, prof\_title);

pprior\_prof\_node->next = pprof\_node;

pprior\_prof\_node = pprior\_prof\_node->next;

//printf("%s\n", pprof\_node->prof\_name);

pprof\_node->next = NULL;

pprof\_node->firstarc = NULL;

}

fclose(prof\_write); //关闭文件

return pprof\_node;

}

/\*\*

\* 函数名称：CreateClassList

\* 函数参数：class\_hd 班级链首结点指针

\* 函数功能：从txt文件中读取信息，创建班级链表

\* 返回值：创建成功返回班级链表头结点，否则返回NULL

\*\*/

CLASS\_NODE\* CreateClassList(CLASS\_NODE \*class\_hd)

{

int ret = 0;

char class\_name[30];

CLASS\_NODE \* pclass\_node = NULL, \* pprior\_class\_node = NULL; //遍历结点指针和前驱指针

FILE \*class\_write = fopen("Class.txt", "r+");

if(!class\_write)

{

printf("打开文件失败！\n");

fclose(class\_write);

return NULL;

}

pprior\_class\_node = class\_hd;

while (!feof(class\_write))

{

ret = fscanf(class\_write, "%s", class\_name);

if(ret == -1) break; //防止fscanf多读取一项

pclass\_node = (CLASS\_NODE\*)malloc(sizeof(CLASS\_NODE));

strcpy(pclass\_node->class\_name, class\_name);

fscanf(class\_write, "%d", &pclass\_node->class\_stu\_amount);

fscanf(class\_write, "%d", &pclass\_node->summer\_class\_hours);

fscanf(class\_write, "%d", &pclass\_node->winter\_class\_hours);

fscanf(class\_write, "%d", &pclass\_node->day\_class\_hours);

fscanf(class\_write, "%d", &pclass\_node->night\_class\_hours);

pprior\_class\_node->next = pclass\_node;

pprior\_class\_node = pprior\_class\_node->next;

//printf("%s\n", pclass\_node->class\_name);

pclass\_node->next = NULL;

pclass\_node->firstarc = NULL;

}

fclose(class\_write); //关闭文件

return pclass\_node;

}

/\*\*

\* 函数名称：InitaCuriList

\* 函数参数：线性表L的地址

\* 函数功能：新建课程链表头结点，并将L的next指针置空

\* 返回值：成功构造返回OK，否则返回ERROR

\*\*/

Status InitaCuriList(CURI\_NODE \*\*L)

{

(\*L) = (CURI\_NODE\* )malloc(sizeof(CURI\_NODE)); //为表头结点分配空间

if(!(\*L))

{

printf("创建头结点失败\n");

return ERROR;

}

//初始化表头结点信息

(\*L)->credit = -1;

(\*L)->hours = -1;

(\*L)->next = NULL;

strcpy((\*L)->curi\_name, "head");

return OK;

}

/\*\*

\* 函数名称：InsertCuriNode

\* 函数参数：课程结点头指针hd

\* 函数功能：插入课程信息结点

\* 返回值：无

\*\*/

void InsertCuriNode(CURI\_NODE \* curi\_hd)

{

CURI\_NODE \*pcuri\_node;

pcuri\_node = (CURI\_NODE \*)malloc(sizeof(CURI\_NODE)); //给待插入课程结点分配空间

if(pcuri\_node)

{

getchar();

printf("请输入您所要添加的课程的名称:");

scanf("%s", pcuri\_node->curi\_name);

}

else

{

printf("分配空间失败!");

return;

}

if(SeekCuri(curi\_hd, pcuri\_node->curi\_name)!=NULL)

{

printf("输入课程已存在!\n");

free(pcuri\_node); //释放已分配空间

return;

}

else

{

printf("请输入课程学分: ");

getchar();

scanf("%f", &pcuri\_node->credit);

printf("请输入课程学时: ");

getchar();

scanf("%f", &pcuri\_node->hours);

pcuri\_node->course\_num = 0; //课程的课堂数目置零

pcuri\_node->next = curi\_hd->next;

curi\_hd->next = pcuri\_node;

pcuri\_node->firstarc = NULL;

printf("插入成功!\n");

return;

}

}

/\*\*

\* 函数名称：ModiCuriNode

\* 函数参数：hd 课程结点头指针

\* 函数功能：修改课程信息结点

\* 返回值：修改成功则返回TRUE,失败则返回FALSE

\*\*/

bool ModiCuriNode(CURI\_NODE \* hd)

{

CURI\_NODE \*pcuri\_node;

char curi\_name[30], old\_curi\_name[30], new\_curi\_name[30]; //修改前的课程名称和修改后的课程名称

float old\_hours, new\_hours; //修改前的学时和修改后的学时

hd = hd->next;

pcuri\_node = hd;

if(pcuri\_node)

{

getchar();

printf("请输入您所要修改的课程的名称:");

scanf("%s", curi\_name);

}

else

{

printf("课程链表暂未创建!");

return false;

}

if((pcuri\_node = SeekCuri(hd, curi\_name))!=NULL)

{

old\_hours = pcuri\_node->hours;

strcpy(old\_curi\_name, pcuri\_node->curi\_name); //保存修改前的课程名称

printf("请输入新的课程名称:");

getchar();

scanf("%s", new\_curi\_name);

strcpy(pcuri\_node->curi\_name, new\_curi\_name); //修改原有课程名称

printf("请输入您所要修改的课程的学分:");

getchar();

scanf("%f", &pcuri\_node->credit);

printf("请输入您所要修改的课程的学时:");

getchar();

scanf("%f", &pcuri\_node->hours);

new\_hours = pcuri\_node->hours; //保存修改后的课程学时

ModiCuriInCourse(gp\_course\_head, old\_curi\_name, new\_curi\_name, old\_hours, new\_hours);

SaveCourseNode(gp\_course\_head); //保存课堂信息

SaveProfNode(gp\_prof\_head); //保存教师信息

printf("修改信息成功!");

return true;

}

else

{

printf("您输入的课程不存在\n");

return false;

}

}

/\*\*

\* 函数名称：ModiCuriInCourse

\* 函数参数：course\_hd 课程结点头指针

\* 函数功能：修改所有邻接表中被修改的课程信息

\* 返回值：修改成功则返回TRUE,失败则返回FALSE

\*\*/

bool ModiCuriInCourse(COURSE\_NODE \*course\_hd, char \* old\_curi\_name, char \*new\_curi\_name, float old\_hours, float new\_hours)

{

int flag = 0;

COURSE\_NODE \*temp;

PROF\_NODE \* pprof\_node = gp\_prof\_head->next;

ROOM\_NODE \* proom\_node = gp\_room\_head->next;

CLASS\_NODE \*pclass\_node = gp\_class\_head->next;

//修改教室-课堂邻接表中的课程信息

while(proom\_node)

{

temp = proom\_node->firstarc;

while(temp)

{

if(strcmp(temp->course\_curi\_name, old\_curi\_name)==0)

{

strcpy(temp->course\_curi\_name, new\_curi\_name);

}

temp = temp->next;

}

proom\_node = proom\_node->next;

}

//修改教室-班级邻接表中的课程信息

while(pclass\_node)

{

temp = pclass\_node->firstarc;

while(temp)

{

if(strcmp(temp->course\_curi\_name, old\_curi\_name)==0)

{

strcpy(temp->course\_curi\_name, new\_curi\_name);

}

temp = temp->next;

}

pclass\_node = pclass\_node->next;

}

//修改教室-教师邻接表中的课程信息

while(pprof\_node)

{

flag = 1;

temp = pprof\_node->firstarc;

while(temp)

{

if(strcmp(temp->course\_curi\_name, old\_curi\_name)==0)

{

pprof\_node->prof\_hours = pprof\_node->prof\_hours-old\_hours+new\_hours;

strcpy(temp->course\_curi\_name, new\_curi\_name);

}

temp = temp->next;

}

pprof\_node = pprof\_node->next;

}

//修改整个课堂单链表中的课程信息

temp=course\_hd->next;

while(temp)

{

if(strcmp(temp->course\_curi\_name, old\_curi\_name)==0)

{

strcpy(temp->course\_curi\_name, new\_curi\_name);

}

temp = temp->next;

}

if(flag)

return true;

else

return false;

}

/\*\*

\* 函数名称：DelCuriNode

\* 函数参数：hd 课程结点头指针

\* 函数功能：删除课程信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelCuriNode(CURI\_NODE \* hd)

{

CURI\_NODE \*pcuri\_node = hd->next;

char curi\_name[30];

if(pcuri\_node)

{

getchar();

printf("请输入您所要删除的课程的名称:");

scanf("%s", curi\_name);

}

else

{

printf("课程链表暂未创建!");

return false;

}

if((pcuri\_node = SeekCuri(hd, curi\_name))!=NULL) //搜索指定的课程结点成功

{

DelCourseAfterCuri(pcuri\_node, gp\_course\_head);

CURI\_NODE \*pprior\_node = PriorCuriElem(hd, curi\_name); //找到待删除结点的前驱

pprior\_node->next = pcuri\_node->next;

pprior\_node = pcuri\_node;

pcuri\_node = pcuri\_node->next;

free(pprior\_node);

printf("删除成功!\n");

return true;

}

else

{

printf("您输入的课程不存在\n");

return false;

}

}

/\*\*

\* 函数名称：DelCourseAfterCuri

\* 函数参数：pcuri\_node 课程信息结点

\* 函数功能：删除与删除的课程同名的课堂

\* 返回值：无

\*\*/

void DelCourseAfterCuri(CURI\_NODE \*pcuri\_node, COURSE\_NODE \*course\_hd)

{

COURSE\_NODE \*pcourse\_node = course\_hd->next, \*course\_temp = NULL;

//遍历课堂链表

while(pcourse\_node)

{

course\_temp = pcourse\_node;

pcourse\_node=pcourse\_node->next;

//如果课程名和课堂名符合

if(strcmp(course\_temp->course\_curi\_name, pcuri\_node->curi\_name)==0)

{

DelWithoutId(course\_temp);

}

}

return;

}

/\*\*

\* 函数名称：SaveCuriNode

\* 函数参数：curi\_hd 课程链头结点指针

\* 函数功能：保存课程链至txt文件

\* 返回值：保存成功返回TRUE, 否则返回FALSE

\*\*/

bool SaveCuriNode(CURI\_NODE \* curi\_hd)

{

if(!curi\_hd)

{

printf("没有课程存在!\n");

return false;

}

curi\_hd = curi\_hd->next;

CURI\_NODE \*pcuri\_node;

FILE \*pfout; //文件指针

pfout = fopen(gp\_curi\_info\_filename, "w"); //只写方式打开文件指针

if(!pfout) //文件指针为空

{

printf("课程文件保存失败！\n");

fclose(pfout);

return false;

}

//遍历课程结点并保存

for (pcuri\_node = curi\_hd; pcuri\_node != NULL; pcuri\_node = pcuri\_node->next)

{

fprintf(pfout, "%s\t", pcuri\_node->curi\_name); //写入课程名称

fprintf(pfout, "%.1f\t", pcuri\_node->credit); //写入课程学分

fprintf(pfout, "%.1f\t", pcuri\_node->hours); //写入课程学时

fprintf(pfout, "%d\n", pcuri\_node->course\_num); //写入课程数目

}

fclose(pfout); //关闭文件指针

return true;

}

/\*\*

\* 函数名称：SearchCuriNode

\* 函数参数：hd 课程链头结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项，如果有就输出相应信息

\* 返回值：查中时输出课程信息, 否则返回NULL

\*\*/

bool SearchCuriNode(CURI\_NODE \* hd)

{

CURI\_NODE \*pcuri\_node;

char curi\_name[30];

printf("请输入需要查找的课程名称\n");

getchar();

scanf("%s", curi\_name);

//查找课程名的信息

pcuri\_node = SeekCuri(hd, curi\_name);

if(!pcuri\_node)

{

printf("查找课程失败\n");

return false;

}

if(pcuri\_node)

{

//输出课程的所有信息

printf("学分:%.1f\n", pcuri\_node->credit);

printf("学时:%.1f\n", pcuri\_node->hours);

printf("下设课堂数:%d\n", pcuri\_node->course\_num);

return true;

}

else if(!(hd->next))

{

printf("课程数为0！\n");

return false;

}

return false;

}

/\*\*

\* 函数名称：SeekCuri

\* 函数参数：curi\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项

\* 返回值：查中时返回结点的地址, 否则返回NULL

\*\*/

CURI\_NODE \* SeekCuri(CURI\_NODE \* hd, char \*curi\_name)

{

CURI\_NODE \*pcuri\_node;

int find = 0;

if(!hd)

{

return NULL;

}

for (pcuri\_node=hd; pcuri\_node!=NULL; pcuri\_node=pcuri\_node->next)

{

if (strcmp(curi\_name, pcuri\_node->curi\_name) == 0)

{

find = 1;

break;

}

}

if (find)

{

return pcuri\_node;

}

else

{

return NULL;

}

}

/\*\*

\* 函数名称：InitaRoomList

\* 函数参数：线性表L的地址

\* 函数功能：新建教室链表头结点，并将L的next指针置空

\* 返回值：成功构造返回OK，否则返回ERROR

\*\*/

Status InitaRoomList(ROOM\_NODE \*\*L)

{

(\*L) = (ROOM\_NODE\* )malloc(sizeof(ROOM\_NODE)); //新建表头结点

if(!(\*L))

{

printf("创建头结点失败\n");

return ERROR;

}

(\*L)->volume = -1;

(\*L)->next = NULL;

strcpy((\*L)->room\_name, "rhead");

return OK;

}

/\*\*

\* 函数名称：ModiCourseNode

\* 函数参数：course\_hd 课堂结点头指针

\* 函数功能：修改课堂信息结点, 更新其余相关链的信息

\* 返回值：修改成功则返回TRUE,失败则返回FALSE

\*\*/

bool ModiCourseNode(COURSE\_NODE \* course\_hd)

{

COURSE\_NODE \*pcourse\_node;

CLASS\_NODE \*pclass\_node;

PROF\_NODE \*pprof\_node;

CURI\_NODE \*pcuri\_node;

int i = 0, j = 0;

char course\_id[20];

char prof\_name[12], curi\_name[30];

course\_hd = course\_hd->next;

pcourse\_node = course\_hd;

COURSE\_NODE \*course\_temp = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE)); //作为修改后的结点插入

COURSE\_NODE \*course\_return = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE)); //保存原有结点信息，用于信息回滚

if(course\_temp)

{

printf("请输入您所要修改的课堂的id:");

getchar();

scanf("%s", course\_id);

}

else

{

printf("课堂链表暂未创建!");

return false;

}

if(!(pcourse\_node=SeekCourse(gp\_course\_head, course\_id)))

{

printf("该课堂尚未创建\n");

return false;

}

printf("\n");

FormatPrint(pcourse\_node);

cpycoursenode(course\_temp, pcourse\_node); //复制全部信息至course\_temp

cpycoursenode(course\_return, pcourse\_node); //复制全部信息至course\_return

DelWithoutId(pcourse\_node); //从链表中删除course\_node

if(1)

{

int op = 1;

int firsttime = 1;

printf(" 0.不再修改 1.教师名称\n");

printf(" 2.课堂班级 3.上课时间段及对应教室\n");

while(op)

{

printf("请选择需要修改的内容\n");

scanf("%d", &op);

switch(op)

{

case 1:

firsttime = 0;

printf("请修改课堂的教师名称:");

getchar();

scanf("%s", prof\_name); //暂时存储输入流,以待检查

if(!(pprof\_node = SeekProf(gp\_prof\_head, prof\_name)))

{

printf("请先创建教师的基本信息\n");

InsertCourseWithoutInput(course\_return);

return false;

}

strcpy(course\_temp->course\_prof\_name, prof\_name);

break;

case 2:

firsttime = 0;

course\_temp->total\_stu\_num = 0;

printf("请修改课堂容纳的班级数目:");

getchar();

scanf("%d", &course\_temp->course\_class\_num);

printf("请修改各个班级的名称(按回车分隔):\n");

for(i=0; i<course\_temp->course\_class\_num; i++)

{

char class\_name[30] = {'\0'};

scanf("%s", class\_name);

if(!(pclass\_node=SeekClass(gp\_class\_head, class\_name)))

{

printf("请先创建班级的基本信息!\n");

InsertCourseWithoutInput(course\_return);

return false;

}

//printf("该班级学期内白天课程共%d节，晚上课程共%d节\n", pclass\_node->day\_class\_hours, pclass\_node->night\_class\_hours);

//printf("该班级学期内夏天课程共%d节，冬天课程共%d节\n", pclass\_node->summer\_class\_hours, pclass\_node->winter\_class\_hours);

strcpy(course\_temp->course\_class\_name[i], class\_name);

}

break;

case 3:

{

firsttime = 0;

i = 0;

int firstsubstr = 1; //标记变量

ROOM\_NODE \*proom\_node;

printf("请按提示修改上课时间段和教室:\n");

while(1)

{

int day\_num = 0;

//修改上课周期

printf("请修改分段上课周期：(如1~5周则输入1~5):\n");

char classes[20] = {'\0'};

getchar();

scanf("%s", classes);

if(!DealString1(classes, &course\_temp->course\_term\_period[i][0], &course\_temp->course\_term\_period[i][1]))

{

printf("输入有误!\n");

InfoReturn();

free(course\_temp);

return false;

}

//scanf("%d~%d", &course\_temp->course\_term\_period[i][0], &course\_temp->course\_term\_period[i][1]);

printf("请修改该周期内的上课日期数:(如1/2/4/0/0代表周一周二和周四上课)\n");

char week\_date[15] = {'\0'};

getchar();

scanf("%s", week\_date);

if(!DealString3(week\_date, course\_temp->course\_week\_period[i]))

{

printf("输入有误!\n");

InfoReturn();

free(course\_temp);

return false;

}

//scanf("%d/%d/%d/%d/%d/%d", &course\_temp->course\_week\_period[i][0], &course\_temp->course\_week\_period[i][1], &course\_temp->course\_week\_period[i][2], &course\_temp->course\_week\_period[i][3], &course\_temp->course\_week\_period[i][4], &course\_temp->course\_week\_period[i][5]);

for(day\_num=0; course\_temp->course\_week\_period[i][day\_num]!=0; day\_num++) ;

//遍历上课时段

for(j=0; j<day\_num; j++)

{

printf("请修改每一天的上课时段:(如1~4代表第一至四节课)\n");

char room\_name[30];

char period[10]={'\0'};

getchar();

scanf("%s", period);

if(!DealString1(period, &course\_temp->course\_day\_period[i][j][0], &course\_temp->course\_day\_period[i][j][1]))

{

printf("输入有误!\n");

InfoReturn();

free(course\_temp);

return false;

}

//scanf("%d~%d", &course\_temp->course\_day\_period[i][j][0], &course\_temp->course\_day\_period[i][j][1]);

printf("请修改该时间段内的教室：\n");

getchar();

scanf("%s", room\_name);

if(!(proom\_node=SeekRoom(gp\_room\_head, room\_name)))

{

printf("该教室不存在");

InsertCourseWithoutInput(course\_return);

return false;

}

//逗号分隔字符串

if(firstsubstr==1)

{

strcpy(course\_temp->course\_room\_name, room\_name);

firstsubstr = 0;

}

else

{

strcat(course\_temp->course\_room\_name, ",");

strcat(course\_temp->course\_room\_name, room\_name);

}

}

course\_temp->day\_period[i] = day\_num;

i++;

printf("是否要继续添加上课周期？(0代表不要，1代表要)");

scanf("%d", &op);

if(!op)

break;

}

//将i赋值给课堂变量

course\_temp->term\_period = i;

break;

}

case 0:

if(firsttime==1)

{

//信息回滚

InsertCourseWithoutInput(course\_return);

return false;

}

else

break;

}

}

FormatPrint(course\_temp);

InsertCourseWithoutInput(course\_temp);

printf("修改信息成功!");

return true;

}

else

{

printf("您输入的课堂不存在\n");

return false;

}

}

/\*\*

\* 函数名称：DelCourseNode

\* 函数参数：hd 课程结点头指针

\* 函数功能：删除课程信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelCourseNode(COURSE\_NODE \* hd)

{

COURSE\_NODE \*pcourse\_node;

char course\_id[20];

pcourse\_node = hd->next;

if(pcourse\_node)

{

getchar();

printf("请输入您所要删除的课堂的id:");

scanf("%s", course\_id);

}

else

{

printf("课程链表暂未创建!");

return false;

}

if((pcourse\_node = SeekCourse(hd, course\_id))!=NULL)

{

//更新删除相应课堂信息后的其他信息

UpdateAfterDelCourse(pcourse\_node);

COURSE\_NODE \*pprior\_node = PriorCourseElem(hd, course\_id);

pprior\_node->next = pcourse\_node->next;

pprior\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

free(pprior\_node);

printf("删除成功!\n");

//保存其他信息

SaveClassNode(gp\_class\_head);

SaveCuriNode(gp\_curi\_head);

SaveProfNode(gp\_prof\_head);

SaveRoomNode(gp\_room\_head);

return true;

}

else

{

printf("您输入的课程不存在\n");

return false;

}

}

/\*\*

\* 函数名称：DelCourseNode

\* 函数参数：pcourse\_node 待删除的课堂结点

\* 函数功能：删除传入的课堂信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelWithoutId(COURSE\_NODE \*pcourse\_node)

{

UpdateAfterDelCourse(pcourse\_node); //首先更新删除后的连锁信息

//找到待删除结点的前驱指针

COURSE\_NODE \*pprior\_node = PriorCourseElem(gp\_course\_head, pcourse\_node->course\_id);

pprior\_node->next = pcourse\_node->next;

pprior\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

free(pprior\_node);

//保存因删除结点而导致变动的其余信息

SaveClassNode(gp\_class\_head);

SaveCuriNode(gp\_curi\_head);

SaveProfNode(gp\_prof\_head);

SaveRoomNode(gp\_room\_head);

return true;

}

/\*\*

\* 函数名称：UpdateAfterDelCourse

\* 函数参数：pcourse\_node 待删除的课堂结点

\* 函数功能：更新删除传入的课堂信息结点后的其余信息

\* 返回值：无

\*\*/

void UpdateAfterDelCourse(COURSE\_NODE \* pcourse\_node)

{

int i = 0, j = 0;

//找到每个课堂下面的班级

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

CLASS\_NODE \* pclass\_node = SeekClass(gp\_class\_head, pcourse\_node->course\_class\_name[i]);

ModiClassAfterDel(pclass\_node, pcourse\_node);

}

PROF\_NODE \* prof\_node = SeekProf(gp\_prof\_head, pcourse\_node->course\_prof\_name);

ModiProfAfterDel(prof\_node, pcourse\_node);

CURI\_NODE \* curi\_node = SeekCuri(gp\_curi\_head, pcourse\_node->course\_curi\_name);

ModiCuriAfterDel(curi\_node, pcourse\_node);

char temp[100] = {'\0'};

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

char \*split;

for(i=0; i<pcourse\_node->term\_period; i++)

{

int num\_of\_week = pcourse\_node->course\_term\_period[i][1]-pcourse\_node->course\_term\_period[i][0]+1;

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

int start\_class = pcourse\_node->course\_day\_period[i][j][0];

int end\_class = pcourse\_node->course\_day\_period[i][j][1];

//分解教室名构成的字符串

split = strtok(room\_name, ",");

ROOM\_NODE \*proom\_node = SeekRoom(gp\_room\_head, split);

room\_name = NULL;

//修改删除课堂结点后的教室信息

ModiRoomAfterDel(proom\_node, pcourse\_node, num\_of\_week, start\_class, end\_class);

}

}

return;

}

/\*\*

\* 函数名称：ModiProfAfterDel

\* 函数参数：pcourse\_node 待删除的课堂结点

pprof\_node 需要修改的教师结点

\* 函数功能：更新删除传入的课堂信息结点后的教师结点信息

\* 返回值：无

\*\*/

void ModiProfAfterDel(PROF\_NODE \* pprof\_node, COURSE\_NODE \*pcourse\_node)

{

float course\_hours = 0;

COURSE\_NODE \*course\_prior = NULL;

CURI\_NODE \* pcuri\_node = SeekCuri(gp\_curi\_head, pcourse\_node->course\_curi\_name);

course\_hours = pcuri\_node->hours;

pprof\_node->prof\_num\_course--; //教授课程减一

pprof\_node->prof\_hours-=course\_hours; //上课学时减去课程的学时

COURSE\_NODE \*course\_temp = pprof\_node->firstarc;

//遍历教师链后面的课堂链

while(course\_temp)

{

//分情况讨论

if(course\_temp==pprof\_node->firstarc) //该课堂恰好是第一个课堂结点

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0) //该课堂的名称和需要删除的课堂名称相同

{

pprof\_node->firstarc = course\_temp->next;

free(course\_temp);

course\_temp = pprof\_node->firstarc;

}

else

{

course\_temp = pprof\_node->firstarc->next; //遍历指针后移

course\_prior = pprof\_node->firstarc; //前驱指针移向教室链后的首结点

}

}

else if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_prior->next = course\_temp->next;

free(course\_temp);

course\_temp = course\_prior->next;

}

else

{

course\_temp = course\_temp->next; //遍历指针后移

course\_prior = course\_prior->next; //前驱指针后移

}

}

return;

}

void ModiCuriAfterDel(CURI\_NODE \*pcuri\_node, COURSE\_NODE \*pcourse\_node)

{

pcuri\_node->course\_num--;

return;

}

/\*\*

\* 函数名称：ModiClassAfterDel

\* 函数参数：pcourse\_node 待删除的课堂结点

pclass\_node 需要修改的班级结点

\* 函数功能：更新删除传入的课堂信息结点后的班级结点信息

\* 返回值：无

\*\*/

void ModiClassAfterDel(CLASS\_NODE \*pclass\_node, COURSE\_NODE \*pcourse\_node)

{

int i, j;

//遍历课堂的每个周期

for(i=0; i<pcourse\_node->term\_period; i++)

{

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

int start\_class = pcourse\_node->course\_day\_period[i][j][0], end\_class = pcourse\_node->course\_day\_period[i][j][1];

int num\_of\_week = pcourse\_node->course\_term\_period[i][1]-pcourse\_node->course\_term\_period[i][0]+1;

//修改每堂课对应班级的时间

if(start\_class>=9)

{

pclass\_node->night\_class\_hours -= num\_of\_week\*(end\_class-start\_class+1);

}

else if(end\_class<=8)

{

pclass\_node->day\_class\_hours -= num\_of\_week\*(end\_class-start\_class+1);

}

else

{

//修改早晚信息

pclass\_node->day\_class\_hours -= num\_of\_week\*(8-start\_class+1);

pclass\_node->night\_class\_hours -= num\_of\_week\*(end\_class-9+1);

}

if(pcourse\_node->course\_term\_period[i][1]>=18)

{

int num\_of\_winter = pcourse\_node->course\_term\_period[i][1]-18+1;

pclass\_node->winter\_class\_hours-=num\_of\_winter\*(end\_class-start\_class+1);

}

if(pcourse\_node->course\_term\_period[i][0]<=2)

{

int num\_of\_summer = 2-pcourse\_node->course\_term\_period[i][0]+1;

pclass\_node->summer\_class\_hours -= num\_of\_summer\*(end\_class-start\_class+1);

}

}

}

//前驱结点指针和遍历结点指针

COURSE\_NODE \*course\_temp = pclass\_node->firstarc, \*course\_prior = NULL;

//遍历每个班级后面连接的所有课堂信息结点

while(course\_temp)

{

//根据课堂结点在邻接表中的位置分情况讨论

if(course\_temp==pclass\_node->firstarc)

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0) //如果遍历到的课堂结点恰好是要删除的那个

{

pclass\_node->firstarc = course\_temp->next;

free(course\_temp);

course\_temp = pclass\_node->firstarc;

}

else

{

course\_temp = pclass\_node->firstarc->next;

course\_prior = pclass\_node->firstarc;

}

}

else if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_prior->next = course\_temp->next;

free(course\_temp);

course\_temp = course\_prior->next;

}

else

{

course\_temp = course\_temp->next; //遍历课堂结点指针后移

course\_prior = course\_prior->next; //前驱课堂结点指针后移

}

}

return;

}

/\*\*

\* 函数名称：ModiRoomAfterDel

\* 函数参数：pcourse\_node 待删除的课堂结点

proom\_node 需要修改的教室结点

\* 函数功能：更新删除传入的课堂信息结点后的班级结点信息

\* 返回值：无

\*\*/

void ModiRoomAfterDel(ROOM\_NODE \*proom\_node, COURSE\_NODE \*pcourse\_node,int num\_of\_week, int start\_class, int end\_class)

{

int i;

for(i=0; i<pcourse\_node->term\_period; i++) //遍历待删除课堂结点的时间分段

{

if(pcourse\_node->course\_term\_period[i][1]>=18)

{

proom\_node->winter\_use\_time-=(pcourse\_node->course\_term\_period[i][1]-18+1)\*(end\_class-start\_class+1);

}

if(pcourse\_node->course\_term\_period[i][0]<=2)

{

proom\_node->summer\_use\_time-=(2-pcourse\_node->course\_term\_period[i][0]+1)\*(end\_class-start\_class+1);

}

}

//修改每个教室的白天黑夜利用时间

if(start\_class>=9)

{

proom\_node->night\_use\_time -= num\_of\_week\*(end\_class-start\_class+1);

}

else if(end\_class<=8)

{

proom\_node->day\_use\_time -= num\_of\_week\*(end\_class-start\_class+1);

}

else

{

proom\_node->day\_use\_time -= num\_of\_week\*(8-start\_class+1);

proom\_node->night\_use\_time -= num\_of\_week\*(end\_class-9+1);

}

//修改教室结点的总体利用时间

proom\_node->total\_use\_time -= num\_of\_week\*(end\_class-start\_class+1);

COURSE\_NODE \*course\_temp = proom\_node->firstarc, \*course\_prior = NULL;

//遍历课堂结点信息链

while(course\_temp)

{

//分情况讨论

if(course\_temp==proom\_node->firstarc)

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

proom\_node->firstarc = course\_temp->next;

free(course\_temp);

course\_temp = proom\_node->firstarc;

}

else

{

course\_temp = proom\_node->firstarc->next;

course\_prior = proom\_node->firstarc;

}

}

else if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_prior->next = course\_temp->next;

free(course\_temp);

course\_temp = course\_prior->next;

}

else

{

course\_temp = course\_temp->next;

course\_prior = course\_prior->next;

}

}

return;

}

/\*\*

\* 函数名称：ModiRoomNode

\* 函数参数：room\_hd 教室结点头指针

\* 函数功能：修改教室信息结点

\* 返回值：修改成功则返回TRUE,失败则返回FALSE

\*\*/

bool ModiRoomNode(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \*proom\_node;

char room\_name[30];

room\_hd = room\_hd->next;

proom\_node = room\_hd;

if(proom\_node)

{

printf("请输入您所要修改的教室的名称:");

getchar();

scanf("%s", room\_name);

}

else

{

printf("课程链表暂未创建!");

return false;

}

if((proom\_node = SeekRoom(room\_hd, room\_name))!=NULL) //输入的教室不存在

{

printf("请输入您所要修改的教室的容量:");

getchar();

scanf("%d", &proom\_node->volume); //修改教室容量

printf("修改信息成功!");

return true;

}

else

{

printf("您输入的教室不存在\n");

return false;

}

}

/\*\*

\* 函数名称：InsertRoomNode

\* 函数参数：room\_hd 教师链表头指针指针

\* 函数功能：插入教师信息结点

\* 返回值：无

\*\*/

void InsertRoomNode(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \*proom\_node;

proom\_node = (ROOM\_NODE \*)malloc(sizeof(ROOM\_NODE)); //给待插入课程结点分配空间

if(proom\_node)

{

getchar();

printf("请输入您所要添加的教室的名称:");

scanf("%s", proom\_node->room\_name);

}

else

{

printf("分配空间失败!");

return;

}

if(SeekRoom(room\_hd, proom\_node->room\_name)!=NULL) //查找是否有重名的教室

{

printf("输入教室已存在!\n");

free(proom\_node); //释放已经申请的空间

return;

}

else

{

printf("请输入教室容量: ");

getchar();

scanf("%d", &proom\_node->volume);

//初始化教室相关信息

proom\_node->day\_use\_time = 0;

proom\_node->night\_use\_time = 0;

proom\_node->total\_use\_time = 0;

proom\_node->winter\_use\_time = 0;

proom\_node->summer\_use\_time = 0;

//将新结点从链表头部插入

proom\_node->next = room\_hd->next;

room\_hd->next = proom\_node;

proom\_node->firstarc = NULL;

printf("插入成功!\n");

return;

}

}

/\*\*

\* 函数名称：DelRoomNode

\* 函数参数：hd 教室结点头指针

\* 函数功能：删除教室信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelRoomNode(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \*proom\_node;

char room\_name[30];

proom\_node = room\_hd->next;

if(proom\_node)

{

getchar();

printf("请输入您所要删除的教室的名称:");

scanf("%s", room\_name);

}

else

{

printf("课程链表暂未创建!");

return false;

}

if((proom\_node=SeekRoom(room\_hd, room\_name))!=NULL) //找到指定教室名称的教室结点

{

if(proom\_node->total\_use\_time>0)

{

printf("该教室仍然承担教学任务!\n");

return false;

}

ROOM\_NODE \*pprior\_node = PriorRoomElem(room\_hd, room\_name); //找到待删除节点的前驱结点

pprior\_node->next = proom\_node->next;

pprior\_node = proom\_node;

proom\_node = proom\_node->next;

free(pprior\_node);

printf("删除成功!\n");

return true;

}

else

{

printf("您输入的教室不存在\n");

return false;

}

}

/\*\*

\* 函数名称：SeekCuri

\* 函数参数：curi\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项

\* 返回值：查中时输出课程信息, 否则返回NULL

\*\*/

bool SearchRoomNode(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \*proom\_node;

char room\_name[30];

printf("请输入需要查找的教室名称\n");

getchar();

scanf("%s", room\_name);

proom\_node = SeekRoom(room\_hd, room\_name);

if(!proom\_node)

{

printf("查找教室失败\n");

return false;

}

if(proom\_node) //教室结点被顺利查找

{

printf("教室容量:%d\n", proom\_node->volume);

printf("夏季使用时长:%d节课\n", proom\_node->summer\_use\_time);

printf("冬季使用时长:%d节课\n", proom\_node->winter\_use\_time);

printf("夜晚使用时长:%d节课\n", proom\_node->night\_use\_time);

return true;

}

else if(!(room\_hd->next))

{

printf("教室数为0！\n");

return false;

}

return false;

}

/\*\*

\* 函数名称：SeekCuri

\* 函数参数：curi\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项

\* 返回值：查中时返回结点的地址, 否则返回NULL

\*\*/

ROOM\_NODE \* SeekRoom(ROOM\_NODE \* room\_hd, char \*room\_name)

{

ROOM\_NODE \*proom\_node;

int find = 0;

if(!room\_hd)

{

return NULL;

}

for (proom\_node=room\_hd; proom\_node!=NULL; proom\_node=proom\_node->next)

{

if (strcmp(room\_name, proom\_node->room\_name) == 0)

{

find = 1;

break;

}

}

if (find)

{

return proom\_node;

}

else

{

return NULL;

}

}

/\*\*

\* 函数名称：PriorCuriElem

\* 函数参数：线性表L，待查找前驱结点的课程名称

\* 函数功能：查找指定课程结点的前驱结点

\* 返回值：成功则返回指向前驱结点的指针，否则返回NULL

\*\*/

CURI\_NODE\* PriorCuriElem(CURI\_NODE \* L, char \*curi\_name)

{

CURI\_NODE\* pcuri\_node = L->next; //遍历指针

CURI\_NODE\* pprior\_node = L; //前驱指针

if(SeekCuri(L, curi\_name)==NULL)

{

return NULL;

}

while (pcuri\_node->next && strcmp(pcuri\_node->curi\_name, curi\_name)) //后继不为空并且遍历指针指向元素非查找元素

{

pcuri\_node = pcuri\_node->next;

pprior\_node = pprior\_node->next;

}

if((!pcuri\_node->next)&&strcmp(pcuri\_node->curi\_name, curi\_name)) //后继为空且遍历指针指向元素还是不是查找元素

{

return NULL;

}

else

{

return pprior\_node; //返回指向前驱结点的指针

}

}

/\*\*

\* 函数名称：PriorRoomElem

\* 函数参数：线性表L，待查找前驱结点的教室名称

\* 函数功能：查找指定教室结点的前驱结点

\* 返回值：成功则返回指向前驱结点的指针，否则返回NULL

\*\*/

ROOM\_NODE\* PriorRoomElem(ROOM\_NODE \* L, char \*room\_name)

{

ROOM\_NODE\* proom\_node = L->next; //遍历指针

ROOM\_NODE\* pprior\_node = L; //前驱指针

if(SeekRoom(L, room\_name)==NULL)

{

return NULL;

}

while (proom\_node->next && strcmp(proom\_node->room\_name, room\_name)) //后继不为空并且遍历指针指向元素非查找元素

{

proom\_node = proom\_node->next;

pprior\_node = pprior\_node->next;

}

if((!proom\_node->next)&&strcmp(proom\_node->room\_name, room\_name)) //后继为空且遍历指针指向元素还是不是查找元素

{

return NULL;

}

else

{

return pprior\_node; //返回指向前驱结点的指针

}

}

/\*\*

\* 函数名称：SaveRoomNode

\* 函数参数：room\_hd 教室链首结点指针

\* 函数功能：保存课程链至txt文件

\* 返回值：保存成功返回TRUE, 否则返回FALSE

\*\*/

bool SaveRoomNode(ROOM\_NODE \* room\_hd)

{

if(!room\_hd)

{

printf("没有教室存在!\n");

return false;

}

room\_hd = room\_hd->next;

ROOM\_NODE \*proom\_node;

FILE \*pfout; //文件指针

pfout = fopen(gp\_room\_info\_filename, "w");

if(!pfout) //文件指针为空

{

printf("课程文件保存失败！\n");

return false;

}

//遍历课程结点并保存

for (proom\_node = room\_hd; proom\_node != NULL; proom\_node = proom\_node->next)

{

fprintf(pfout, "%s\t", proom\_node->room\_name); //写入教室名称

fprintf(pfout, "%d\t", proom\_node->summer\_use\_time); //写入教室夏天使用时长

fprintf(pfout, "%d\t", proom\_node->winter\_use\_time); //写入教室冬天使用时长

fprintf(pfout, "%d\t", proom\_node->day\_use\_time); //写入教室白天使用时长

fprintf(pfout, "%d\t", proom\_node->night\_use\_time); //写入教室夜晚使用时长

fprintf(pfout, "%d\t", proom\_node->total\_use\_time); //写入教室总体使用时长

fprintf(pfout, "%d\n", proom\_node->volume); //写入课程容量

}

fclose(pfout); //关闭文件指针

return true;

}

/\*\*

\* 函数名称：ListCuriTrabverse

\* 函数参数：课程单链表头结点指针

\* 函数功能：依次显示线性表中的每个元素。

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListCuriTrabverse(CURI\_NODE \* curi\_hd)

{

curi\_hd = curi\_hd->next;

CURI\_NODE \* pcuri\_node = curi\_hd;

if(!pcuri\_node)

{

return ERROR;

}

printf("\t\t\t课程名称\t\t课程学分\t\t课程学时\t\t课堂数量\n");

while (pcuri\_node)

{

printf("\t\t%-16s\t\t%-8.1f\t\t%-8.1f\t\t%-8d\n", pcuri\_node->curi\_name, pcuri\_node->credit, pcuri\_node->hours, pcuri\_node->course\_num);

pcuri\_node = pcuri\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListRoomTrabverse

\* 函数参数：教室单链表头结点指针

\* 函数功能：依次显示线性表中的每个元素。

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListRoomTrabverse(ROOM\_NODE \* room\_hd)

{

room\_hd = room\_hd->next;

ROOM\_NODE \* proom\_node = room\_hd;

if(!proom\_node)

{

return ERROR;

}

while (proom\_node) //遍历教室结点链并输出相应信息

{

printf("教室名称为%s 教室容量为%d人\n", proom\_node->room\_name, proom\_node->volume);

printf("白天使用时长%d节课 夜晚使用时长%d节课\n", proom\_node->day\_use\_time, proom\_node->night\_use\_time);

printf("夏天使用时长%d节课 冬天使用时长%d节课\n", proom\_node->summer\_use\_time, proom\_node->winter\_use\_time);

proom\_node = proom\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListClassTrabverse

\* 函数参数：班级单链表头结点指针

\* 函数功能：依次显示线性表中的每个元素

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListClassTrabverse(CLASS\_NODE \* class\_hd)

{

class\_hd = class\_hd->next;

CLASS\_NODE \* pclass\_node = class\_hd;

if(!pclass\_node)

{

return ERROR;

}

while (pclass\_node) //遍历班级信息并输出

{

printf("%s %d人\n", pclass\_node->class\_name, pclass\_node->class\_stu\_amount);

printf("该班级学期内白天课程共%d节，晚上课程共%d节\n", pclass\_node->day\_class\_hours, pclass\_node->night\_class\_hours);

printf("该班级学期内夏天课程共%d节，冬天课程共%d节\n", pclass\_node->summer\_class\_hours, pclass\_node->winter\_class\_hours);

pclass\_node = pclass\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListProfTrabverse

\* 函数参数：教师单链表头结点指针

\* 函数功能：依次显示线性表中的每个元素

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListProfTrabverse(PROF\_NODE \* prof\_hd)

{

prof\_hd = prof\_hd->next;

PROF\_NODE \* pprof\_node = prof\_hd;

if(!pprof\_node)

{

return ERROR;

}

while (pprof\_node) //遍历教师信息并且输出

{

printf("%s %s\n", pprof\_node->prof\_name, pprof\_node->prof\_title);

printf("一学期承担%d个课堂，%.1f学时的教学任务\n", pprof\_node->prof\_num\_course, pprof\_node->prof\_hours);

pprof\_node = pprof\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListCourseTrabverse

\* 函数参数：课堂单链表头结点指针

\* 函数功能：依次显示线性表中的每个元素

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListCourseTrabverse(COURSE\_NODE \* course\_hd)

{

course\_hd = course\_hd->next;

COURSE\_NODE \* pcourse\_node = course\_hd;

if(!pcourse\_node)

{

return ERROR;

}

while (pcourse\_node) //遍历课堂信息并且输出

{

FormatPrint(pcourse\_node);

printf("该课堂的总人数为：%d\n\n", pcourse\_node->total\_stu\_num);

pcourse\_node = pcourse\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListClassCourseTrabverse

\* 函数参数：班级单链表头结点指针

\* 函数功能：依次显示班级-课堂邻接表中的班级-课堂关系

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListClassCourseTrabverse(CLASS\_NODE \* class\_hd)

{

class\_hd = class\_hd->next;

CLASS\_NODE \* pclass\_node = class\_hd;

COURSE\_NODE \* pcourse\_node;

if(!pclass\_node)

{

return ERROR;

}

while (pclass\_node) //遍历班级信息结点

{

pcourse\_node = pclass\_node->firstarc;

printf("%s %d:\n", pclass\_node->class\_name, pclass\_node->class\_stu\_amount);

//遍历到某个班级结点的时候输出邻接表中该结点后的所有课堂信息

while(pcourse\_node)

{

printf("%s:", pcourse\_node->course\_id);

printf("%s\n", pcourse\_node->course\_prof\_name);

pcourse\_node = pcourse\_node->next;

}

pclass\_node = pclass\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListProfCourseTrabverse

\* 函数参数：教师单链表头结点指针

\* 函数功能：依次显示教师-课堂邻接表中的教师-课堂关系

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListProfCourseTrabverse(PROF\_NODE \* prof\_hd)

{

prof\_hd = prof\_hd->next;

PROF\_NODE \* pprof\_node = prof\_hd;

COURSE\_NODE \* pcourse\_node;

if(!pprof\_node)

{

return ERROR;

}

while (pprof\_node) //遍历教师信息结点

{

pcourse\_node = pprof\_node->firstarc;

printf("%s %s:\n", pprof\_node->prof\_name, pprof\_node->prof\_title);

//遍历到某个教师结点的时候输出邻接表中该结点后的所有课堂信息

while(pcourse\_node)

{

printf("%s:", pcourse\_node->course\_id);

printf("%s\n", pcourse\_node->course\_prof\_name);

pcourse\_node = pcourse\_node->next;

}

pprof\_node = pprof\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：ListRoomCourseTrabverse

\* 函数参数：教室单链表头结点指针

\* 函数功能：依次显示教室-课堂邻接表中的教室-课堂关系

\* 返回值：成功遍历返回OK，否则返回ERROR

\*\*/

Status ListRoomCourseTrabverse(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \* proom\_node = room\_hd->next;

COURSE\_NODE \* pcourse\_node;

if(!proom\_node)

{

return ERROR;

}

printf("教室名\t\t授课教师\t\t课堂人数\t\t课堂id\n");

while (proom\_node)

{

pcourse\_node = proom\_node->firstarc;

//遍历到某个教室结点的时候输出邻接表中该结点后的所有课堂信息

while(pcourse\_node)

{

printf("%s\t\t", proom\_node->room\_name);

printf("%s\t\t", pcourse\_node->course\_prof\_name);

printf("%d\t\t", pcourse\_node->total\_stu\_num);

printf("%s\t\t\n", pcourse\_node->course\_id);

pcourse\_node = pcourse\_node->next;

}

proom\_node = proom\_node->next;

}

return OK;

}

/\*\*

\* 函数名称：Cre\_Room\_Course\_List

\* 函数参数：class\_hd 班级链首结点指针

course\_class\_head 用于创建班级-课堂有向图的课堂链

\* 函数功能：创建班级-课堂有向图

\* 返回值：保存成功返回指向图中第一个班级结点的指针, 否则返回NULL

\*\*/

ROOM\_NODE \* Cre\_Room\_Course\_List(ROOM\_NODE \* room\_hd, COURSE\_NODE \* gp\_course\_head)

{

COURSE\_NODE \*pcourse\_node = gp\_course\_head->next;

COURSE\_NODE \*qcourse\_node, \*qtemp = NULL;

ROOM\_NODE \*proom\_node = NULL;

while(pcourse\_node) //遍历课堂结点信息链

{

qcourse\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

proom\_node = room\_hd->next;

while(proom\_node) //遍历教室结点信息链

{

if(strstr(qcourse\_node->course\_room\_name, proom\_node->room\_name)) //如果教室名称在课堂的教室名字符串里面

{

qtemp = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

cpycoursenode(qtemp, qcourse\_node);

qtemp->next = proom\_node->firstarc;

proom\_node->firstarc = qtemp;

}

proom\_node = proom\_node->next;

}

}

return room\_hd;

}

/\*\*

\* 函数名称：cpycoursenode

\* 函数参数：a 信息复制的目标课堂结点

b 复制信息的来源

\* 函数功能：将b中的信息全部复制到a中

\* 返回值：复制成功返回true, 否则返回false

\*\*/

bool cpycoursenode(COURSE\_NODE \*a, COURSE\_NODE \*b)

{

int i,j;

strcpy(a->course\_id, b->course\_id);

strcpy(a->course\_prof\_name, b->course\_prof\_name);

a->course\_class\_num = b->course\_class\_num;

strcpy(a->course\_curi\_name, b->course\_curi\_name);

strcpy(a->course\_room\_name, b->course\_room\_name);

a->total\_stu\_num = b->total\_stu\_num;

a->term\_period = b->term\_period;

a->next = NULL;

//复制周期内天数

for(i=0; i<b->term\_period; i++)

{

a->day\_period[i] = b->day\_period[i];

}

//依次复制班级名称

for(i=0; i<b->course\_class\_num; i++)

{

strcpy(a->course\_class\_name[i], b->course\_class\_name[i]);

}

//依次复制上课周次

for(i=0; i<b->term\_period; i++)

{

for(j=0; j<b->day\_period[i]; j++)

{

a->course\_week\_period[i][j]=b->course\_week\_period[i][j];

}

}

//依次复制上课节次

for(i=0; i<b->term\_period; i++)

{

for(j=0; j<b->day\_period[i]; j++)

{

a->course\_day\_period[i][j][0]=b->course\_day\_period[i][j][0];

a->course\_day\_period[i][j][1]=b->course\_day\_period[i][j][1];

}

}

for(i=0; i<b->term\_period; i++)

{

a->course\_term\_period[i][0]=b->course\_term\_period[i][0];

a->course\_term\_period[i][1]=b->course\_term\_period[i][1];

}

return true;

}

/\*\*

\* 函数名称：Cre\_Class\_Course\_List

\* 函数参数：class\_hd 班级链首结点指针

course\_class\_head 用于创建班级-课堂有向图的课堂链

\* 函数功能：创建班级-课堂有向图

\* 返回值：保存成功返回指向图中第一个班级结点的指针, 否则返回NULL

\*\*/

CLASS\_NODE\* Cre\_Class\_Course\_List(CLASS\_NODE \*class\_hd, COURSE\_NODE \*gp\_course\_head)

{

COURSE\_NODE \*pcourse\_node = gp\_course\_head->next;

COURSE\_NODE \*qcourse\_node = NULL, \*qtemp = NULL;

CLASS\_NODE \*pclass\_node = NULL;

int i;

while(pcourse\_node) //遍历课堂信息链

{

qcourse\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

for(i=0; i<qcourse\_node->course\_class\_num; i++)

{

pclass\_node = class\_hd->next;

while(pclass\_node) //遍历班级信息链

{

if(strcmp(qcourse\_node->course\_class\_name[i], pclass\_node->class\_name)==0)

{

qtemp = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

cpycoursenode(qtemp, qcourse\_node);

qtemp->next = pclass\_node->firstarc;

pclass\_node->firstarc = qtemp;

break;

}

else

{

pclass\_node = pclass\_node->next;

}

}

}

}

return class\_hd;

}

/\*\*

\* 函数名称：Cre\_Prof\_Course\_List

\* 函数参数：prof\_hd 班级链首结点指针

cprof\_courese\_hd 用于创建教师-课堂有向图的课堂链

\* 函数功能：创建教师-课堂有向图

\* 返回值：保存成功返回指向图中第一个教师结点的指针, 否则返回NULL

\*\*/

PROF\_NODE\* Cre\_Prof\_Course\_List(PROF\_NODE \*prof\_hd, COURSE\_NODE \*cprof\_courese\_hd)

{

COURSE\_NODE \*pcourse\_node = cprof\_courese\_hd->next;

COURSE\_NODE \*qcourse\_node = NULL, \*qtemp = NULL;

PROF\_NODE \*pprof\_node = NULL;

while(pcourse\_node) //遍历课堂信息链

{

qcourse\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

pprof\_node = prof\_hd->next;

while(pprof\_node) //遍历教师信息链

{

if(strcmp(qcourse\_node->course\_prof\_name, pprof\_node->prof\_name)==0) //课堂的教师姓名ihe教师姓名相同

{

qtemp = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

cpycoursenode(qtemp, qcourse\_node);

qtemp->next = pprof\_node->firstarc;

pprof\_node->firstarc = qtemp;

}

pprof\_node = pprof\_node->next;

}

}

return prof\_hd;

}

/\*\*

\* 函数名称：InitiaProfList

\* 函数参数：线性表L的地址

\* 函数功能：新建教师表头结点，并将next指针置空

\* 返回值：成功构造返回OK，否则返回ERROR

\*\*/

Status InitaProfList(PROF\_NODE \*\*L)

{

(\*L) = (PROF\_NODE\* )malloc(sizeof(PROF\_NODE)); //新建表头结点

if(!(\*L))

{

printf("创建头结点失败\n");

return ERROR;

}

(\*L)->next = NULL;

(\*L)->firstarc = NULL;

strcpy((\*L)->prof\_name, "phead");

strcpy((\*L)->prof\_title, "phead");

return OK;

}

/\*\*

\* 函数名称：ModiProfNode

\* 函数参数：prof\_hd 教师信息链头结点指针

\* 函数功能：修改教师信息结点

\* 返回值：修改成功则返回true,失败则返回false

\*\*/

bool ModiProfNode(PROF\_NODE \* prof\_hd)

{

PROF\_NODE \*pprof\_node;

char prof\_name[12]; //需要修改的教师结点

prof\_hd = prof\_hd->next;

pprof\_node = prof\_hd;

if(pprof\_node)

{

printf("请输入您所要修改的教师的姓名:");

getchar();

scanf("%s", prof\_name);

}

else

{

printf("教师链表暂未创建!");

return false;

}

if((pprof\_node = SeekProf(prof\_hd, prof\_name))!=NULL) //查找到教师姓名

{

printf("请输入您所要修改的教师的职位:");

getchar();

scanf("%s", pprof\_node->prof\_title);

printf("修改信息成功!");

return true;

}

else

{

printf("您输入的教师不存在\n");

return false;

}

}

/\*\*

\* 函数名称：InsertProfNode

\* 函数参数：prof\_hd 教师信息链头结点指针

\* 函数功能：插入教师信息结点

\* 返回值：无

\*\*/

void InsertProfNode(PROF\_NODE \* prof\_hd)

{

PROF\_NODE \*pprof\_node;

pprof\_node = (PROF\_NODE \*)malloc(sizeof(PROF\_NODE));

if(pprof\_node)

{

getchar();

printf("请输入您所要添加的教师的姓名:");

scanf("%s", pprof\_node->prof\_name);

}

else

{

printf("分配空间失败!");

return;

}

if(SeekProf(prof\_hd, pprof\_node->prof\_name)!=NULL)

{

printf("输入教师已存在!\n");

free(pprof\_node); //释放已分配的空间

return;

}

else

{

printf("请输入教师职位: ");

getchar();

scanf("%s", pprof\_node->prof\_title);

printf("请输入教师年龄：");

getchar();

scanf("%d", &pprof\_node->age);

//初始化教师结点的统计信息

pprof\_node->prof\_hours = 0;

pprof\_node->prof\_num\_course = 0;

pprof\_node->next = prof\_hd->next;

prof\_hd->next = pprof\_node;

pprof\_node->firstarc = NULL;

pprof\_node->prof\_hours = 0;

printf("插入成功!\n");

return;

}

}

/\*\*

\* 函数名称：DelProfNode

\* 函数参数：hd 课程结点头指针

\* 函数功能：删除课程信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelProfNode(PROF\_NODE \* prof\_hd)

{

PROF\_NODE \*pprof\_node;

char prof\_name[30];

pprof\_node = prof\_hd->next;

if(pprof\_node)

{

getchar();

printf("请输入您所要删除的教师的名称:");

scanf("%s", prof\_name);

}

else

{

printf("教师链表暂未创建!");

return false;

}

if((pprof\_node = SeekProf(prof\_hd, prof\_name))!=NULL) //找到对应教师名称的结点

{

if(pprof\_node->prof\_num\_course>0)

{

printf("该教师尚在教学!\n");

return false;

}

PROF\_NODE \*pprior\_node = PriorProfElem(prof\_hd, prof\_name); //找到教师结点的前驱结点

pprior\_node->next = pprof\_node->next;

pprior\_node = pprof\_node;

pprof\_node = pprof\_node->next;

free(pprior\_node);

printf("删除成功!\n");

return true;

}

else

{

printf("您输入的教师不存在\n");

return false;

}

}

/\*\*

\* 函数名称：SeekCuri

\* 函数参数：curi\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项

\* 返回值：查中时输出课程信息, 否则返回NULL

\*\*/

bool SearchProfNode(PROF\_NODE \* prof\_hd)

{

PROF\_NODE \*pprof\_node;

char prof\_name[30];

printf("请输入需要查找的教师姓名\n");

getchar();

scanf("%s", prof\_name);

pprof\_node = SeekProf(prof\_hd, prof\_name);

if(!pprof\_node)

{

printf("查找教师失败\n");

return false;

}

if(pprof\_node) //遍历教师结点并输出信息

{

printf("教师职称:%s\n", pprof\_node->prof\_title);

printf("本学期授课总学时:%.1f\n", pprof\_node->prof\_hours);

printf("本学期授课课堂数:%d\n", pprof\_node->prof\_num\_course);

return true;

}

else if(!(prof\_hd->next))

{

printf("教师数为0！\n");

return false;

}

return false;

}

/\*\*

\* 函数名称：SeekCuri

\* 函数参数：curi\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与curi\_name重名的选项

\* 返回值：查中时返回结点的地址, 否则返回NULL

\*\*/

PROF\_NODE \* SeekProf(PROF\_NODE \* prof\_hd, char \*prof\_name)

{

PROF\_NODE \*pprof\_node;

int find = 0;

if(!prof\_hd)

{

return NULL;

}

for (pprof\_node=prof\_hd; pprof\_node!=NULL; pprof\_node=pprof\_node->next)

{

if (strcmp(prof\_name, pprof\_node->prof\_name) == 0) //比对教师信息是否相同

{

find = 1;

break;

}

}

if (find)

{

return pprof\_node;

}

else

{

return NULL;

}

}

/\*\*

\* 函数名称：PriorProfElem

\* 函数参数：线性表L，待查找前驱结点的教师名称

\* 函数功能：查找指定教师结点的前驱结点

\* 返回值：成功则返回指向前驱结点的指针，否则返回NULL

\*\*/

PROF\_NODE\* PriorProfElem(PROF\_NODE \* L, char \*prof\_name)

{

PROF\_NODE\* pprof\_node = L->next; //遍历指针

PROF\_NODE\* pprior\_node = L; //前驱指针

if(SeekProf(L, prof\_name)==NULL)

{

return NULL;

}

while (pprof\_node->next && strcmp(pprof\_node->prof\_name, prof\_name)) //后继不为空并且遍历指针指向元素非查找元素

{

pprof\_node = pprof\_node->next;

pprior\_node = pprior\_node->next;

}

if((!pprof\_node->next)&&strcmp(pprof\_node->prof\_name, prof\_name)) //后继为空且遍历指针指向元素还是不是查找元素

{

return NULL;

}

else

{

return pprior\_node; //返回指向前驱结点的指针

}

}

/\*\*

\* 函数名称：SaveCuriNode

\* 函数参数：hd 课程链首结点指针

\* 函数功能：保存课程链至txt文件

\* 返回值：保存成功返回TRUE, 否则返回FALSE

\*\*/

bool SaveProfNode(PROF\_NODE \* prof\_hd)

{

if(!prof\_hd)

{

printf("没有教师存在!\n");

return false;

}

prof\_hd = prof\_hd->next;

PROF\_NODE \*pprof\_node;

FILE \*pfout; //文件指针

pfout = fopen(gp\_prof\_info\_filename, "w");

if(!pfout) //文件指针为空

{

printf("教师文件保存失败！\n");

fclose(pfout);

return false;

}

//遍历教师结点链

for (pprof\_node = prof\_hd; pprof\_node != NULL; pprof\_node = pprof\_node->next)

{

/\*保存教师信息\*/

fprintf(pfout, "%s\t", pprof\_node->prof\_name);

fprintf(pfout, "%s\t", pprof\_node->prof\_title);

fprintf(pfout, "%f\t", pprof\_node->prof\_hours);

fprintf(pfout, "%d\t", pprof\_node->prof\_num\_course);

fprintf(pfout, "%d\n", pprof\_node->age);

}

fclose(pfout); //文件指针关闭

return true;

}

/\*\*

\* 函数名称：InitiaClassList

\* 函数参数：线性表L的地址

\* 函数功能：新建一个表头结点，并将next指针置空

\* 返回值：成功构造返回OK，否则返回ERROR

\*\*/

Status InitaClassList(CLASS\_NODE \*\*L)

{

(\*L) = (CLASS\_NODE\* )malloc(sizeof(CLASS\_NODE)); //新建表头结点

if(!(\*L))

{

printf("创建头结点失败\n");

return ERROR;

}

(\*L)->next = NULL;

(\*L)->class\_stu\_amount = -1;

strcpy((\*L)->class\_name, "chead");

return OK;

}

/\*\*

\* 函数名称：ModiClassNode

\* 函数参数：class\_hd 教师信息链头结点指针

\* 函数功能：修改教师信息结点

\* 返回值：修改成功则返回true,失败则返回false

\*\*/

bool ModiClassNode(CLASS\_NODE \* class\_hd)

{

CLASS\_NODE \*pclass\_node;

char class\_name[30];

class\_hd = class\_hd->next;

pclass\_node = class\_hd;

int old\_stu\_amount = 0; //修改之前的学生数量

if(pclass\_node)

{

printf("请输入您所要修改的班级的名称:");

getchar();

scanf("%s", class\_name);

}

else

{

printf("班级链表暂未创建!");

return false;

}

if((pclass\_node = SeekClass(class\_hd, class\_name))!=NULL) //如果输入的班级存在

{

old\_stu\_amount = pclass\_node->class\_stu\_amount;

printf("请输入您所要修改的班级的人数:");

getchar();

scanf("%d", &pclass\_node->class\_stu\_amount); //修改班级人数

if(ModiClassInCourse(gp\_course\_head, pclass\_node, old\_stu\_amount)) //修改邻接表中的课程信息

{

printf("修改信息成功!");

return true;

}

}

else

{

printf("您输入的班级不存在\n");

return false;

}

return false;

}

/\*\*

\* 函数名称：ModiClassInCourse

\* 函数参数：course\_hd 课堂信息链头结点指针

pclass\_node 需要修改的班级信息结点

old\_stu\_amount 修改前的班级学生数目

\* 函数功能：修改教师信息结点

\* 返回值：修改成功则返回true,失败则返回false

\*\*/

bool ModiClassInCourse(COURSE\_NODE \*course\_hd, CLASS\_NODE \*pclass\_node, int old\_stu\_amount)

{

int flag = 0, i = 0;

COURSE\_NODE \* pcourse = course\_hd->next;

//遍历每个课堂结点信息

while(pcourse)

{

for(i=0; i<pcourse->course\_class\_num; i++)

{

if(strcmp(pcourse->course\_class\_name[i], pclass\_node->class\_name)==0)

{

flag = 1;

pcourse->total\_stu\_num = pcourse->total\_stu\_num+pclass\_node->class\_stu\_amount-old\_stu\_amount;

//修改邻接表中的课堂结点信息

ModiCrossForClass(pcourse);

}

}

pcourse = pcourse->next;

}

SaveCourseNode(gp\_course\_head); //保存课堂信息

if(flag)

return true;

else

return false;

}

/\*\*

\* 函数名称：ModiCrossForClass

\* 函数参数：pcourse\_node 课堂信息链头结点指针

\* 函数功能：修改邻接表中的课堂信息结点

\* 返回值：无

\*\*/

void ModiCrossForClass(COURSE\_NODE \*pcourse\_node)

{

COURSE\_NODE \*course\_temp;

int i;

PROF\_NODE \*pprof\_node=SeekProf(gp\_prof\_head, pcourse\_node->course\_prof\_name);

CLASS\_NODE \*pclass\_node;

ROOM\_NODE \*proom\_node = gp\_room\_head->next;

course\_temp = pprof\_node->firstarc;

while(course\_temp) //更新教师结点后的课堂链的学生信息

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_temp->total\_stu\_num = pcourse\_node->total\_stu\_num;

}

course\_temp = course\_temp->next;

}

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

pclass\_node = SeekClass(gp\_class\_head, pcourse\_node->course\_class\_name[i]); //搜寻班级结点

course\_temp = pclass\_node->firstarc;

while(course\_temp) //更新班级结点后的课堂链的学生信息

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_temp->total\_stu\_num = pcourse\_node->total\_stu\_num;

}

course\_temp = course\_temp->next;

}

}

while(proom\_node)

{

course\_temp = proom\_node->firstarc;

while(course\_temp) //更新教室结点后的课堂链的学生信息

{

if(strcmp(course\_temp->course\_id, pcourse\_node->course\_id)==0)

{

course\_temp->total\_stu\_num = pcourse\_node->total\_stu\_num;

}

course\_temp = course\_temp->next;

}

proom\_node = proom\_node->next;

}

return;

}

/\*\*

\* 函数名称：InsertClassNode

\* 函数参数：class\_hd 班级信息链头结点

\* 函数功能：插入班级信息结点

\* 返回值：无

\*\*/

void InsertClassNode(CLASS\_NODE \* class\_hd)

{

CLASS\_NODE \*pclass\_node;

pclass\_node = (CLASS\_NODE \*)malloc(sizeof(CLASS\_NODE)); //给待插入课程结点分配空间

if(pclass\_node)

{

getchar();

printf("请输入您所要添加的班级的名称:");

scanf("%s", pclass\_node->class\_name);

}

else

{

printf("分配空间失败!");

return;

}

if(SeekClass(class\_hd, pclass\_node->class\_name)!=NULL) //检查班级重名情况

{

printf("输入班级名已存在!\n");

free(pclass\_node);

return;

}

else

{

printf("请输入班级学生人数: ");

getchar();

scanf("%d", &pclass\_node->class\_stu\_amount);

//初始化班级结点相关信息

pclass\_node->night\_class\_hours = 0;

pclass\_node->day\_class\_hours = 0;

pclass\_node->next = class\_hd->next;

class\_hd->next = pclass\_node;

pclass\_node->firstarc = NULL;

printf("插入成功!\n");

return;

}

}

/\*\*

\* 函数名称：DelClassNode

\* 函数参数：class\_hd 班级信息链头结点

\* 函数功能：删除班级信息结点

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

bool DelClassNode(CLASS\_NODE \* class\_hd)

{

CLASS\_NODE \*pclass\_node;

char class\_name[30];

pclass\_node = class\_hd->next;

//输入需要删除的班级名称

if(pclass\_node)

{

getchar();

printf("请输入您所要删除的班级的名称:");

scanf("%s", class\_name);

}

else

{

printf("班级链表暂未创建!");

return false;

}

//搜寻指定名称的班级结点

if((pclass\_node = SeekClass(class\_hd, class\_name))!=NULL) //找到待删除的班级结点

{

DelCourseAfterC(pclass\_node, gp\_course\_head); //修改删除班级结点后的其他信息

//找到待删除结点的前驱结点

CLASS\_NODE \*pprior\_node = PriorClassElem(class\_hd, class\_name);

pprior\_node->next = pclass\_node->next;

pprior\_node = pclass\_node;

pclass\_node = pclass\_node->next;

free(pprior\_node);

printf("删除成功!\n");

return true;

}

else

{

printf("您输入的班级不存在\n");

return false;

}

}

/\*\*

\* 函数名称：DelCourseAfterC

\* 函数参数：pclass\_node 班级信息结点

course\_hd 课堂信息链头结点

\* 函数功能：删除班级信息结点后进行其余结点信息的修改

\* 返回值：删除成功则返回TRUE,失败则返回FALSE

\*\*/

void DelCourseAfterC(CLASS\_NODE \*pclass\_node, COURSE\_NODE \*course\_hd)

{

int i ,j;

COURSE\_NODE \*pcourse\_node = pclass\_node->firstarc, \*qcourse\_node = course\_hd->next, \*course\_temp = NULL;

//遍历课堂结点信息

while(pcourse\_node)

{

course\_temp = pcourse\_node; //保存待删除或修改的结点的地址

pcourse\_node = pcourse\_node->next; //遍历指针后移

//只有一个班级的情况

if(course\_temp->course\_class\_num==1)

{

//直接把那个课堂给删了

qcourse\_node = SeekCourse(course\_hd, course\_temp->course\_id);

DelWithoutId(qcourse\_node);

}

else

{

qcourse\_node = SeekCourse(course\_hd, course\_temp->course\_id);

//课堂内的学生总人数减去班级人数

qcourse\_node->total\_stu\_num-=pclass\_node->class\_stu\_amount;

//修改每个课堂结点的班级数目和班级名称

for(i=0;i<qcourse\_node->course\_class\_num;i++)

{

if(strcmp(qcourse\_node->course\_class\_name[i], pclass\_node->class\_name)==0)

{

for(j=i;j<qcourse\_node->course\_class\_num;j++)

{

strcpy(qcourse\_node->course\_class\_name[j],qcourse\_node->course\_class\_name[j+1]);

}

break;

}

}

qcourse\_node->course\_class\_num--; //删除一个班级，必须减一

}

free(course\_temp);

}

return;

}

/\*\*

\* 函数名称：SearchClassNode

\* 函数参数：class\_hd 课程链首结点指针

\* 函数功能：根据输入信息查找相应的班级结点

\* 返回值：查中时输出班级信息, 否则返回NULL

\*\*/

bool SearchClassNode(CLASS\_NODE \* class\_hd)

{

CLASS\_NODE \*pclass\_node;

char class\_name[30];

printf("请输入需要查找的班级名称:");

getchar();

scanf("%s", class\_name);

pclass\_node = SeekClass(class\_hd, class\_name);

if(!pclass\_node)

{

printf("查找班级失败\n");

return false;

}

if(pclass\_node) //查找班级结点成功

{

printf("班级学生数目:%d\n", pclass\_node->class\_stu\_amount);

printf("考试周上课节数:%d节课\n", pclass\_node->winter\_class\_hours);

printf("开学前两周上课节数:%d节课\n", pclass\_node->summer\_class\_hours);

printf("夜晚上课节数:%d节课\n", pclass\_node->night\_class\_hours);

return true;

}

else if(!(class\_hd->next))

{

printf("班级数为0！\n");

return false;

}

return false;

}

/\*\*

\* 函数名称：SeekClass

\* 函数参数：class\_name 待比对的班级名称,

class\_hd 班级链首结点指针

\* 函数功能：查找班级信息链中是否有与class\_name重名的结点

\* 返回值：查中时返回结点的地址, 否则返回NULL

\*\*/

CLASS\_NODE \* SeekClass(CLASS\_NODE \*class\_hd, char \*class\_name)

{

CLASS\_NODE \*pclass\_node;

int find = 0;

if(!class\_hd)

{

return NULL;

}

for (pclass\_node=class\_hd; pclass\_node!=NULL; pclass\_node=pclass\_node->next)

{

if (strcmp(class\_name, pclass\_node->class\_name) == 0)

{

find = 1;

break;

}

}

if (find)

{

return pclass\_node;

}

else

{

return NULL;

}

}

/\*\*

\* 函数名称：PriorClassElem

\* 函数参数：线性表L，待查找前驱结点的班级名称

\* 函数功能：查找指定班级结点的前驱结点

\* 返回值：成功则返回指向前驱结点的指针，否则返回NULL

\*\*/

CLASS\_NODE\* PriorClassElem(CLASS\_NODE \* L, char \*class\_name)

{

CLASS\_NODE\* pclass\_node = L->next; //遍历指针

CLASS\_NODE\* pprior\_node = L; //前驱指针

if(SeekClass(L, class\_name)==NULL)

{

return NULL;

}

while (pclass\_node->next && strcmp(pclass\_node->class\_name, class\_name)) //后继不为空并且遍历指针指向元素非查找元素

{

pclass\_node = pclass\_node->next;

pprior\_node = pprior\_node->next;

}

if((!pclass\_node->next)&&strcmp(pclass\_node->class\_name, class\_name)) //后继为空且遍历指针指向元素还是不是查找元素

{

return NULL;

}

else

{

return pprior\_node; //返回指向前驱结点的指针

}

}

/\*\*

\* 函数名称：SaveClassNode

\* 函数参数：class\_hd 课程链头结点指针

\* 函数功能：保存班级链至txt文件

\* 返回值：保存成功返回TRUE, 否则返回FALSE

\*\*/

bool SaveClassNode(CLASS\_NODE \* class\_hd)

{

if(!class\_hd)

{

printf("没有班级存在!\n");

return false;

}

class\_hd = class\_hd->next;

CLASS\_NODE \*pclass\_node;

FILE \*pfout; //文件指针

pfout = fopen(gp\_class\_info\_filename, "w");

if(!pfout) //文件指针为空

{

printf("课程文件保存失败！\n");

fclose(pfout);

return false;

}

//遍历班级结点并保存

for (pclass\_node = class\_hd; pclass\_node != NULL; pclass\_node = pclass\_node->next)

{

/\*保存班级信息\*/

fprintf(pfout, "%s\t", pclass\_node->class\_name);

fprintf(pfout, "%d\t", pclass\_node->class\_stu\_amount);

fprintf(pfout, "%d\t", pclass\_node->summer\_class\_hours);

fprintf(pfout, "%d\t", pclass\_node->winter\_class\_hours);

fprintf(pfout, "%d\t", pclass\_node->day\_class\_hours);

fprintf(pfout, "%d\n", pclass\_node->night\_class\_hours);

}

fclose(pfout); //关闭文件

return true;

}

/\*\*

\* 函数名称：InitaCourseList

\* 函数参数：线性表L的地址

\* 函数功能：新建课堂链表头结点，并将L的next指针置空

\* 返回值：成功构造返回OK，否则返回ERROR

\*\*/

Status InitaCourseList(COURSE\_NODE \*\*L)

{

(\*L) = (COURSE\_NODE\* )malloc(sizeof(COURSE\_NODE)); //新建表头结点

if(!(\*L))

{

printf("创建头结点失败\n");

return ERROR;

}

(\*L)->term\_period = -1;

(\*L)->next = NULL;

return OK;

}

/\*\*

\* 函数名称：Initarray

\* 函数参数：pcourse\_node 指定课堂结点的指针

\* 函数功能：初始化课堂结点的信息

\* 返回值：无

\*\*/

void Initarray(COURSE\_NODE \*pcourse\_node)

{

int i, j;

for(i=0;i<5;i++)

{

pcourse\_node->day\_period[i]=0; //初始化一周的上课天数

}

for(i=0;i<5;i++)

{

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

{

pcourse\_node->course\_week\_period[i][j]=0; //初始化每周的上课日期

}

}

for(i=0;i<5;i++)

{

for(j=0;j<5;j++)

{

pcourse\_node->course\_day\_period[i][j][0]=0; //初始化每周的上课日期

pcourse\_node->course\_day\_period[i][j][1]=0;

}

}

for(i=0;i<5;i++)

{

pcourse\_node->course\_term\_period[i][0]=0; //初始化上课节次

pcourse\_node->course\_term\_period[i][1]=0;

}

return;

}

/\*\*

\* 函数名称：InsertCourseNode

\* 函数参数：course\_hd 课堂链头结点指针

\* 函数功能：插入课堂结点至课堂链表

\* 返回值：保存成功返回链尾结点指针, 否则返回NULL

\*\*/

void InsertCourseNode(COURSE\_NODE \* course\_hd)

{

COURSE\_NODE \*pcourse\_node;

CLASS\_NODE \*pclass\_node;

ROOM\_NODE \*proom\_node;

CURI\_NODE \*pcuri\_node;

PROF\_NODE \*pprof\_node;

pcourse\_node = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

pclass\_node = (CLASS\_NODE \*)malloc(sizeof(CLASS\_NODE));

//初始化课堂结点的统计信息，

pcourse\_node->term\_period = 0;

pcourse\_node->total\_stu\_num = 0;

int i = 0, j =0, op = 0;

char prof\_name[12], curi\_name[30];

int num\_of\_week, start\_class, end\_class;

if(pcourse\_node)

{

getchar();

printf("请输入您添加的课堂id:\n");

scanf("%s", pcourse\_node->course\_id);

}

else

{

printf("分配空间失败!");

return;

}

//初始化其他信息

Initarray(pcourse\_node);

if(SeekCourse(course\_hd, pcourse\_node->course\_id)!=NULL) //防止课堂id重合

{

printf("输入课堂id已存在！\n");

free(pcourse\_node);

return;

}

else

{

printf("请输入课堂对应课程名称:");

getchar();

scanf("%s", curi\_name); //暂时存储输入流,以待检查

if(!(pcuri\_node=SeekCuri(gp\_curi\_head, curi\_name)))

{

printf("该课程不存在!\n");

free(pcourse\_node);

return;

}

strcpy(pcourse\_node->course\_curi\_name, curi\_name);

printf("请输入课堂的教师名称:");

getchar();

scanf("%s", prof\_name); //暂时存储输入流,以待检查

if(!(pprof\_node=SeekProf(gp\_prof\_head, prof\_name)))

{

printf("该教师不存在!\n");

free(pcourse\_node);

return;

}

strcpy(pcourse\_node->course\_prof\_name, prof\_name);

i = 0;

int firstsubstr = 1;

printf("请按提示输入上课时间段:\n");

while(1)

{

int day\_num = 0;

printf("请输入分段上课周期：(如1~5周则输入1~5):\n");

char classes[20] = {'\0'};

getchar();

scanf("%s", classes);

if(!DealString1(classes, &pcourse\_node->course\_term\_period[i][0], &pcourse\_node->course\_term\_period[i][1]))

{

printf("输入有误!\n");

InfoReturn();

free(pcourse\_node);

return;

}

//scanf("%s", &pcourse\_node->course\_term\_period[i][0], &pcourse\_node->course\_term\_period[i][1]);

num\_of\_week=pcourse\_node->course\_term\_period[i][1]-pcourse\_node->course\_term\_period[i][0]+1; //计算上课周期中的周数

printf("请输入该周期内的上课日期数:(如1,2,4代表周一周二和周四上课)\n");

char week\_date[15] = {'\0'};

getchar();

scanf("%s", week\_date);

if(!DealString3(week\_date, pcourse\_node->course\_week\_period[i]))

{

printf("输入有误!\n");

InfoReturn();

free(pcourse\_node);

return;

}

//getchar();

//scanf("%d/%d/%d/%d/%d/%d", &pcourse\_node->course\_week\_period[i][0], &pcourse\_node->course\_week\_period[i][1], &pcourse\_node->course\_week\_period[i][2], &pcourse\_node->course\_week\_period[i][3], &pcourse\_node->course\_week\_period[i][4], &pcourse\_node->course\_week\_period[i][5]);

//计算周期中一周内的上课天数

for(day\_num=0; pcourse\_node->course\_week\_period[i][day\_num]!=0; day\_num++) ;

printf("请按序输入每一天的上课时段和教室:\n");

pcourse\_node->term\_period++;

pcourse\_node->day\_period[i] = day\_num; //第i-1个上课周期中每周上课的天数

for(j=0; j<day\_num; j++)

{

int date = pcourse\_node->course\_week\_period[i][j];

printf("请输入周%d的上课时间段:(如1~4代表第一至四节课)\n", date);

char room\_name[30] = {'\0'};

char period[10]={'\0'};

getchar();

scanf("%s", period);

if(!DealString1(period, &pcourse\_node->course\_day\_period[i][j][0], &pcourse\_node->course\_day\_period[i][j][1]))

{

printf("输入有误!\n");

InfoReturn();

free(pcourse\_node);

return;

}

if(pcourse\_node->course\_day\_period[i][j][0]<1||pcourse\_node->course\_day\_period[i][j][0]>12

||pcourse\_node->course\_day\_period[i][j][1]<1||pcourse\_node->course\_day\_period[i][j][1]>12)

{

printf("输入时间越界!\n");

InfoReturn();

free(pcourse\_node);

return;

}

start\_class = pcourse\_node->course\_day\_period[i][j][0];

end\_class = pcourse\_node->course\_day\_period[i][j][1];

printf("请输入该时间段内的教室:");

getchar();

scanf("%s", room\_name);

//检查输入的教室字符串

if(!(proom\_node=SeekRoom(gp\_room\_head, room\_name)))

{

printf("插入失败，请先创建教室的基本信息");

InfoReturn(); //信息回滚,使得教室，班级等信息回归插入之前的状态

free(pcourse\_node);

return;

}

COURSE\_NODE \*course\_temp = course\_hd->next;

while(course\_temp)

{

if(ContraRoom(proom\_node, pcourse\_node, course\_temp))

{

printf("时间冲突!\n");

InfoReturn(); //信息回滚,使得教室，班级等信息回归插入之前的状态

free(pcourse\_node);

return;

}

course\_temp = course\_temp->next;

}

//逗号分隔字符串

if(firstsubstr==1)

{

strcpy(pcourse\_node->course\_room\_name, room\_name);

firstsubstr=0;

}

else

{

strcat(pcourse\_node->course\_room\_name, ",");

strcat(pcourse\_node->course\_room\_name, room\_name);

}

//更新因为课堂结点而造成的教室信息改变

UpdateRoomUseTime(proom\_node, num\_of\_week, start\_class, end\_class);

if(pcourse\_node->course\_term\_period[i][1]>=18)

{

proom\_node->winter\_use\_time+=(pcourse\_node->course\_term\_period[i][1]-18+1)\*(end\_class-start\_class+1);

}

if(pcourse\_node->course\_term\_period[i][0]<=2)

{

proom\_node->summer\_use\_time+=(2-pcourse\_node->course\_term\_period[i][0]+1)\*(end\_class-start\_class+1);

}

}

i++;

printf("是否要继续输入上课时间段？(0代表不要，1代表要)");

getchar();

scanf("%d", &op);

if(!op)

break;

}

pcourse\_node->term\_period = i; //课堂上课周期数目

printf("请输入课堂容纳的班级数目:");

getchar();

scanf("%d", &pcourse\_node->course\_class\_num);

printf("请输入各个班级的名称(按回车分隔):\n");

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

char class\_name[30] = {'\0'};

getchar();

scanf("%s", class\_name);

if(!(pclass\_node=SeekClass(gp\_class\_head, class\_name)))

{

printf("请先创建班级的基本信息!\n");

InfoReturn(); //信息回滚,使得教室，班级等信息回归插入之前的状态

free(pcourse\_node);

return;

}

//修改班级上课时间等信息

UpdateClassUseTime(pclass\_node, pcourse\_node);

strcpy(pcourse\_node->course\_class\_name[i], class\_name);

pcourse\_node->total\_stu\_num+=pclass\_node->class\_stu\_amount; //计算课堂总人数

}

FormatPrint(pcourse\_node);

pcourse\_node->next = course\_hd->next;

course\_hd->next = pcourse\_node;

//因课堂的修改而引起的其余各种结点的信息修改

pcuri\_node->course\_num++; //修改一门课程下的课堂数目

pprof\_node->prof\_hours+=pcuri\_node->hours; //插入课堂后，修改相应教师的工作时长

pprof\_node->prof\_num\_course++; //修改教师的教学课堂数目

printf("插入成功!\n");

InsertCrossList(pcourse\_node); //将新增的课堂结点加入到邻接表中

SaveClassNode(gp\_class\_head); //保存班级信息

SaveRoomNode(gp\_room\_head); //保存教室信息

SaveCuriNode(gp\_curi\_head); //保存课程信息

SaveProfNode(gp\_prof\_head); //保存教师信息

return;

}

}

/\*\*

\* 函数名称：FormatPrint

\* 函数参数：pcourse\_node 课堂链结点指针

\* 函数功能：格式化输出课堂信息

\* 返回值：无

\*\*/

void FormatPrint(COURSE\_NODE \*pcourse\_node)

{

int i;

char temp[100] = {'\0'}; //如果不使用temp,那么pcourse\_node->course\_room\_name将被更改

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

char \*split;

printf("课堂id:%s\n", pcourse\_node->course\_id);

printf("课程名称:%s\n", pcourse\_node->course\_curi\_name);

printf("教授姓名:%s\n", pcourse\_node->course\_prof\_name);

printf("班级名称:");

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

printf("%s ", pcourse\_node->course\_class\_name[i]);

}

printf("上课周期为：\n");

for (i=0; i<pcourse\_node->term\_period; i++)

{

int j = 0;

if(pcourse\_node->course\_term\_period[i][0]!=pcourse\_node->course\_term\_period[i][1])

{

printf("%d-%d", pcourse\_node->course\_term\_period[i][0], pcourse\_node->course\_term\_period[i][1]);

printf("周 ");

while((j<pcourse\_node->day\_period[i]))

{

split = strtok(room\_name, ",");

printf("周");

printf("%d ", pcourse\_node->course\_week\_period[i][j]);

printf("第");

printf("%d-%d节 ", pcourse\_node->course\_day\_period[i][j][0], pcourse\_node->course\_day\_period[i][j][1]);

printf("在%s教室", split);

printf("\n");

room\_name = NULL;

j++;

}

}

else

{

printf("%d", pcourse\_node->course\_term\_period[i][0]);

printf("周 ");

while((j<pcourse\_node->day\_period[i]))

{

split = strtok(room\_name, ",");

printf("周");

printf("%d ", pcourse\_node->course\_week\_period[i][j]);

printf("第");

printf("%d-%d节 ", pcourse\_node->course\_day\_period[i][j][0], pcourse\_node->course\_day\_period[i][j][1]);

printf("在%s教室", split);

printf("\n");

room\_name = NULL;

j++;

}

}

}

printf("\n");

return;

}

/\*\*

\* 函数名称：InsertCourseWithoutInput

\* 函数参数：pcourse\_node 待插入课堂链结点指针

\* 函数功能：插入课堂结点

\* 返回值：无

\*\*/

void InsertCourseWithoutInput(COURSE\_NODE \*pcourse\_node)

{

int i,j,num\_of\_week,start\_class,end\_class;

PROF\_NODE \*pprof\_node = SeekProf(gp\_prof\_head, pcourse\_node->course\_prof\_name);

CURI\_NODE \*pcuri\_node = SeekCuri(gp\_curi\_head, pcourse\_node->course\_curi\_name);

COURSE\_NODE \*course\_hd = gp\_course\_head;

COURSE\_NODE \*course\_temp = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

CLASS\_NODE \*pclass\_node = (CLASS\_NODE \*)malloc(sizeof(CLASS\_NODE));

cpycoursenode(course\_temp, pcourse\_node); //复制pcourse\_node的所有信息

//修改课程信息

pcuri\_node->course\_num++;

pprof\_node->prof\_hours+=pcuri\_node->hours;

pprof\_node->prof\_num\_course++; //修改教师的执教课堂数目

char temp[100] = {'\0'};

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

char \*split;

for(i=0;i<pcourse\_node->term\_period;i++)

{

num\_of\_week = pcourse\_node->course\_term\_period[i][1]-pcourse\_node->course\_term\_period[i][0]+1;

for(j=0;j<pcourse\_node->day\_period[i];j++)

{

start\_class = pcourse\_node->course\_day\_period[i][j][0];

end\_class = pcourse\_node->course\_day\_period[i][j][1];

split = strtok(room\_name, ",");

ROOM\_NODE \*proom\_node = SeekRoom(gp\_room\_head, split);

room\_name = NULL;

//修改教室信息

UpdateRoomUseTime(proom\_node, num\_of\_week, start\_class, end\_class);

if(pcourse\_node->course\_term\_period[i][1]>=18)

{

proom\_node->winter\_use\_time+=(pcourse\_node->course\_term\_period[i][1]-18+1)\*(end\_class-start\_class+1);

}

if(pcourse\_node->course\_term\_period[i][0]<=2)

{

proom\_node->summer\_use\_time+=(2-pcourse\_node->course\_term\_period[i][0]+1)\*(end\_class-start\_class+1);

}

}

}

pcourse\_node->total\_stu\_num = 0;

for(i=0; i<pcourse\_node->course\_class\_num; i++) //遍历并且修改班级信息

{

pclass\_node = SeekClass(gp\_class\_head, pcourse\_node->course\_class\_name[i]);

UpdateClassUseTime(pclass\_node, pcourse\_node);

pcourse\_node->total\_stu\_num+=pclass\_node->class\_stu\_amount;

}

pcourse\_node->next = course\_hd->next;

course\_hd->next = pcourse\_node;

InsertCrossList(pcourse\_node); //插入到每个邻接表中

//保存修改的信息

SaveClassNode(gp\_class\_head);

SaveRoomNode(gp\_room\_head);

SaveCuriNode(gp\_curi\_head);

SaveProfNode(gp\_prof\_head);

return;

}

/\*\*

\* 函数名称：InsertCrossList

\* 函数参数：pcourse\_node 等待插入到邻接表中的课堂结点指针

\* 函数功能：将课堂结点插入到各邻接表中

\* 返回值：无

\*\*/

void InsertCrossList(COURSE\_NODE \*pcourse\_node)

{

int i,j;

//插入教师-课堂邻接表

PROF\_NODE \*pprof\_node = SeekProf(gp\_prof\_head, pcourse\_node->course\_prof\_name); //定位到教授课堂的教师的结点

COURSE\_NODE \*course\_prof = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

cpycoursenode(course\_prof, pcourse\_node); //将课堂结点除了指针域外的所有信息复制到待插入结点

course\_prof->next = pprof\_node->firstarc; //修改指针域

pprof\_node->firstarc = course\_prof;

//插入班级-课堂邻接表

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

COURSE\_NODE \*course\_class = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

CLASS\_NODE \*pclass\_node = SeekClass(gp\_class\_head, pcourse\_node->course\_class\_name[i]);

cpycoursenode(course\_class, pcourse\_node); //将课堂结点除了指针域外的所有信息复制到待插入结点

//修改指针域

course\_class->next = pclass\_node->firstarc;

pclass\_node->firstarc = course\_class;

}

//插入教室-课堂邻接表

char \*split;

char temp[100] = {'\0'};

char check[100] = {'\0'};

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

for(i=0; i<pcourse\_node->term\_period; i++)

{

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

split = strtok(room\_name, ",");

if(strstr(check,split))

continue;

strcat(check ,split);

COURSE\_NODE \*course\_room = (COURSE\_NODE \*)malloc(sizeof(COURSE\_NODE));

ROOM\_NODE \*proom\_node = SeekRoom(gp\_room\_head, split);

cpycoursenode(course\_room, pcourse\_node);

//修改指针域

course\_room->next = proom\_node->firstarc;

proom\_node->firstarc = course\_room;

room\_name = NULL;

}

}

return;

}

/\*\*

\* 函数名称：UpdateRoomUseTime

\* 函数参数：proom\_node 等待更新的教室结点指针

num\_of\_week 在该教室的课堂持续周数

start\_class 在该教室的课堂开始上课的节数

end\_class 在该教室的课堂下课的节数

\* 函数功能：插入课堂结点后更新教室信息结点的白天黑夜和总的使用时长

\* 返回值：无

\*\*/

void UpdateRoomUseTime(ROOM\_NODE \* proom\_node, int num\_of\_week, int start\_class, int end\_class)

{

if(start\_class>=9)

{

proom\_node->night\_use\_time += num\_of\_week\*(end\_class-start\_class+1);

}

else if(end\_class<=8)

{

proom\_node->day\_use\_time += num\_of\_week\*(end\_class-start\_class+1);

}

else

{

proom\_node->day\_use\_time += num\_of\_week\*(8-start\_class+1);

proom\_node->night\_use\_time += num\_of\_week\*(end\_class-9+1);

}

proom\_node->total\_use\_time += num\_of\_week\*(end\_class-start\_class+1);

//printf("\n名称：%s白天%d黑夜%d\n", proom\_node->room\_name, proom\_node->day\_use\_time, proom\_node->night\_use\_time);

return;

}

/\*\*

\* 函数名称：UpdateClassUseTime

\* 函数参数：pclass\_node 等待更新的班级结点指针

pcourse\_node 新插入的课堂结点指针

\* 函数功能：插入课堂结点后更新班级信息结点的内容

\* 返回值：无

\*\*/

void UpdateClassUseTime(CLASS\_NODE \* pclass\_node, COURSE\_NODE \* pcourse\_node)

{

int i,j;

//遍历班级某课堂的上课时间并统计早晚，冬夏的时间变化

for(i=0; i<pcourse\_node->term\_period; i++)

{

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

int start\_class = pcourse\_node->course\_day\_period[i][j][0], end\_class = pcourse\_node->course\_day\_period[i][j][1];

int num\_of\_week = pcourse\_node->course\_term\_period[i][1]-pcourse\_node->course\_term\_period[i][0]+1;

if(start\_class>=9)

{

pclass\_node->night\_class\_hours += num\_of\_week\*(end\_class-start\_class+1);

}

else if(end\_class<=8)

{

pclass\_node->day\_class\_hours += num\_of\_week\*(end\_class-start\_class+1);

}

else

{

pclass\_node->day\_class\_hours += num\_of\_week\*(8-start\_class+1);

pclass\_node->night\_class\_hours += num\_of\_week\*(end\_class-9+1);

}

if(pcourse\_node->course\_term\_period[i][1]>=18)

{

int num\_of\_winter = pcourse\_node->course\_term\_period[i][1]-18+1;

pclass\_node->winter\_class\_hours+=num\_of\_winter\*(end\_class-start\_class+1);

}

if(pcourse\_node->course\_term\_period[i][0]<=2)

{

int num\_of\_summer = 2-pcourse\_node->course\_term\_period[i][0]+1;

pclass\_node->summer\_class\_hours += num\_of\_summer\*(end\_class-start\_class+1);

}

//printf("\n名称：%s白天%d黑夜%d\n", pclass\_node->class\_name, pclass\_node->day\_class\_hours, pclass\_node->night\_class\_hours);

}

}

return;

}

/\*\*

\* 函数名称：SeekCourse

\* 函数参数：course\_name 待比对的课程名称,

hd 课程链首结点指针

\* 函数功能：查找课程信息链中是否有与course\_name重名的选项

\* 返回值：查中时返回结点的地址, 否则返回NULL

\*\*/

COURSE\_NODE \* SeekCourse(COURSE\_NODE \* hd, char \*course\_id)

{

COURSE\_NODE \*pcourse\_node;

int find = 0;

if(!hd)

{

return NULL;

}

for (pcourse\_node=hd; pcourse\_node!=NULL; pcourse\_node=pcourse\_node->next)

{

if (strcmp(course\_id, pcourse\_node->course\_id) == 0)

{

find = 1;

break;

}

}

if (find)

{

return pcourse\_node;

}

else

{

return NULL;

}

}

/\*\*

\* 函数名称：PriorCourseElem

\* 函数参数：线性表L，待查找前驱结点的课堂id

\* 函数功能：查找指定课堂结点的前驱结点

\* 返回值：成功则返回指向前驱课堂结点的指针，否则返回NULL

\*\*/

COURSE\_NODE\* PriorCourseElem(COURSE\_NODE \* L, char \*course\_id)

{

COURSE\_NODE\* pcourse\_node = L->next; //遍历指针

COURSE\_NODE\* pprior\_node = L; //前驱指针

if(SeekCourse(L, course\_id)==NULL)

{

return NULL;

}

while (pcourse\_node->next && strcmp(pcourse\_node->course\_id, course\_id)) //后继不为空并且遍历指针指向元素非查找元素

{

pcourse\_node = pcourse\_node->next;

pprior\_node = pprior\_node->next;

}

if((!pcourse\_node->next)&&strcmp(pcourse\_node->course\_id, course\_id)) //后继为空且遍历指针指向元素还是不是查找元素

{

return NULL;

}

else

{

return pprior\_node; //返回指向前驱结点的指针

}

}

/\*\*

\* 函数名称：SaveCourseNode

\* 函数参数：course\_hd 课堂链头结点指针

\* 函数功能：保存课堂链至txt文件

\* 返回值：保存成功返回TRUE, 否则返回FALSE

\*\*/

bool SaveCourseNode(COURSE\_NODE \* course\_hd)

{

int i=0, j=0, k=0;

if(!course\_hd)

{

printf("没有课堂存在!\n");

return false;

}

course\_hd = course\_hd->next;

COURSE\_NODE \*pcourse\_node;

FILE \*pfout; //文件指针

pfout = fopen(gp\_course\_info\_filename, "w");

if(!pfout)

{

printf("课堂文件保存失败！\n");

fclose(pfout); //文件指针关闭

return false;

}

/\*遍历课堂信息链\*/

for (pcourse\_node = course\_hd; pcourse\_node != NULL; pcourse\_node = pcourse\_node->next)

{

/\*保存课堂信息\*/

fprintf(pfout, "%s\t", pcourse\_node->course\_id);

fprintf(pfout, "%s\t", pcourse\_node->course\_curi\_name);

fprintf(pfout, "%s\t", pcourse\_node->course\_prof\_name);

fprintf(pfout, "%d\t", pcourse\_node->course\_class\_num);

/\*遍历课堂信息链中的班级名称链\*/

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

fprintf(pfout, "%s\t", pcourse\_node->course\_class\_name[i]);

}

fprintf(pfout, "%d\t", pcourse\_node->term\_period);

for(i=0; i<5; i++)

{

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

{

fprintf(pfout,"%d\t", pcourse\_node->course\_week\_period[i][j]);

}

}

/\*遍历保存课堂信息链中上课起止节数\*/

for(i=0; i<5; i++)

{

for(j=0; j<5; j++)

{

for(k=0; k<2; k++)

fprintf(pfout,"%d\t", pcourse\_node->course\_day\_period[i][j][k]);

}

}

for(i=0; i<5; i++)

{

for(j=0; j<2; j++)

{

fprintf(pfout, "%d\t", pcourse\_node->course\_term\_period[i][j]);

}

}

for(i=0; i<5; i++)

{

fprintf(pfout, "%d\t", pcourse\_node->day\_period[i]);

}

fprintf(pfout, "%s\t", pcourse\_node->course\_room\_name);

fprintf(pfout, "%d\t", pcourse\_node->total\_stu\_num);

fprintf(pfout, "\n");

}

fclose(pfout); //文件指针关闭

return true;

}

/\*\*

\* 函数名称：CreateCourseList

\* 函数参数：course\_hd 课堂链头结点指针

\* 函数功能：保存课程链至txt文件

\* 返回值：保存成功返回链尾结点指针, 否则返回NULL

\*\*/

COURSE\_NODE\* CreateCourseList(COURSE\_NODE \*course\_hd)

{

int ret = 0, i = 0, j = 0, k = 0;

COURSE\_NODE \* pcourse\_node = NULL, \* pprior\_course\_node = NULL;

FILE \*course\_write = fopen("Course.txt", "r+");

if(!course\_write)

{

printf("打开文件失败！\n");

fclose(course\_write); //关闭文件

return NULL;

}

pprior\_course\_node = course\_hd;

while (!feof(course\_write)) //扫描文件至文件尾部

{

pcourse\_node = (COURSE\_NODE\*)malloc(sizeof(COURSE\_NODE));

ret = fscanf(course\_write, "%s", pcourse\_node->course\_id);

if(ret == -1) break;

fscanf(course\_write, "%s", pcourse\_node->course\_curi\_name);

fscanf(course\_write, "%s", pcourse\_node->course\_prof\_name);

fscanf(course\_write, "%d", &pcourse\_node->course\_class\_num);

for(i=0; i<pcourse\_node->course\_class\_num; i++)

{

fscanf(course\_write, "%s", pcourse\_node->course\_class\_name[i]);

}

fscanf(course\_write, "%d", &pcourse\_node->term\_period);

for(i=0; i<5; i++)

{

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

{

fscanf(course\_write,"%d", &pcourse\_node->course\_week\_period[i][j]);

}

}

for(i=0; i<5; i++)

{

for(j=0; j<5; j++)

{

for(k=0; k<2; k++)

fscanf(course\_write,"%d", &pcourse\_node->course\_day\_period[i][j][k]);

}

}

for(i=0; i<5; i++)

{

for(j=0; j<2; j++)

{

fscanf(course\_write, "%d", &pcourse\_node->course\_term\_period[i][j]);

}

}

for(i=0; i<5; i++)

{

fscanf(course\_write, "%d", &pcourse\_node->day\_period[i]);

}

fscanf(course\_write, "%s", pcourse\_node->course\_room\_name);

fscanf(course\_write, "%d", &pcourse\_node->total\_stu\_num);

pprior\_course\_node->next = pcourse\_node;

pprior\_course\_node = pprior\_course\_node->next;

pcourse\_node->next = NULL;

}

fclose(course\_write); //关闭文件

return pcourse\_node;

}

/\*\*

\* 函数名称：SearchCourseNodeByProf

\* 函数参数：prof\_hd 教师链表头结点指针

\* 函数功能：根据教师查找课堂

\* 返回值：查找成功返回true,查找失败返回false

\*\*/

bool SearchCourseNodeByProf(PROF\_NODE \* prof\_hd)

{

PROF\_NODE \*pprof\_node;

char prof\_name[30];

printf("请输入需要查找的教师姓名\n");

getchar();

scanf("%s", prof\_name);

pprof\_node = SeekProf(prof\_hd, prof\_name); //查找教师信息

if(!pprof\_node) //教师信息指针不为空

{

printf("查找教师失败\n");

return false;

}

COURSE\_NODE \* pcourse\_node = pprof\_node->firstarc;

if(!pcourse\_node) //如果教师结点后面一个课堂结点都没有

{

printf("该教师不承担授课任务\n");

return false;

}

while(pcourse\_node)

{

FormatPrint(pcourse\_node);

pcourse\_node = pcourse\_node->next;

}

return true;

}

/\*\*

\* 函数名称：SearchCourseNodeByRoom

\* 函数参数：room\_hd 教室结点链头指针

\* 函数功能：根据教室输出这个教室里的课堂信息

\* 返回值：查找成功返回true,否则返回false

\*\*/

bool SearchCourseNodeByRoom(ROOM\_NODE \* room\_hd)

{

ROOM\_NODE \*proom\_node;

char room\_name[30];

printf("请输入需要查找的教室名称\n");

getchar();

scanf("%s", room\_name);

proom\_node = SeekRoom(room\_hd, room\_name);

if(!proom\_node)

{

printf("查找教室失败\n");

return false;

}

COURSE\_NODE \* pcourse\_node = proom\_node->firstarc;

if(!pcourse\_node)

{

printf("该教室中没有课堂\n");

return false;

}

while(pcourse\_node) //遍历课堂信息并且输出

{

//printf("课堂id为%s\n课堂总人数为:%d\n", pcourse\_node->course\_id, pcourse\_node->total\_stu\_num);

FormatPrint(pcourse\_node);

pcourse\_node = pcourse\_node->next;

}

return true;

}

/\*\*

\* 函数名称：SearchCourseNodeByCuri

\* 函数参数：curi\_hd 课程信息头结点指针,

\* 函数功能：根据课程名查找课堂信息

\* 返回值：查中时输出课堂信息,否则返回NULL

\*\*/

bool SearchCourseNodeByCuri(CURI\_NODE \* curi\_hd)

{

CreateCourseList(gp\_course\_head);

CreateCuriList(gp\_curi\_head);

CURI\_NODE \*pcuri\_node;

char curi\_name[30];

int find = 0;

printf("请输入需要查找的课程名称\n");

getchar();

scanf("%s", curi\_name);

pcuri\_node = SeekCuri(curi\_hd, curi\_name); //检查课程是否存在

if(!pcuri\_node) //课程结点为空

{

printf("查找课程失败\n");

return false;

}

COURSE\_NODE \* pcourse\_node = gp\_course\_head;

while(pcourse\_node) //遍历课堂结点

{

if(strcmp(pcourse\_node->course\_curi\_name, curi\_name)==0)

{

find = 1;

//printf("课堂id为%s\n课堂总人数为:%d\n", pcourse\_node->course\_id, pcourse\_node->total\_stu\_num);

FormatPrint(pcourse\_node);

}

pcourse\_node = pcourse\_node->next;

}

if(find)

return true;

else

{

printf("查找课程失败!");

return false;

}

}

/\*\*

\* 函数名称：SearchCourseNodeByTime

\* 函数参数：course\_node 课堂结点指针

\* 函数功能：根据输入的时间查找课堂

\* 返回值：查中时输出课堂信息, 否则返回NULL

\*\*/

bool SearchCourseNodeByTime(COURSE\_NODE \* course\_node)

{

CreateCourseList(gp\_course\_head);

COURSE\_NODE \*pcourse\_node = gp\_course\_head->next;

ROOM\_NODE \*proom\_node;

int week, date, start\_class = 0, end\_class = 0;

char temp;

int find = 0;

printf("请输入需要查找的周次(如第7周则输入7)\n");

getchar();

scanf("%d", &week);

printf("请输入需要查找的日期(如周三则输入3)\n");

getchar();

scanf("%d", &date);

printf("请输入需要查找的节数，~分隔(如第1~4节则输入1~4,第1节则输入1\n");

getchar();

scanf("%d%c", &start\_class, &temp);

if(temp=='~')

{

scanf("%d", &end\_class);

}

else

end\_class = start\_class;

while(pcourse\_node)

{

if((proom\_node=TimeIn(pcourse\_node, week, date, start\_class, end\_class)))

{

FormatPrint(pcourse\_node);

//printf("%s ", pcourse\_node->course\_id);

printf("\n");

//printf("该时间段在%s\n", proom\_node->room\_name);

find = 1;

}

pcourse\_node = pcourse\_node->next;

}

if(find)

return true;

else

{

printf("查找课程失败!");

return false;

}

}

/\*\*

\* 函数名称：TimeIn

\* 函数参数：pcourse\_node 课堂结点指针

week 课堂所在的周数

date 课堂一周在周几

start\_class

\* 函数功能：从txt文件中读取课程链信息，创建课程链链表

\* 返回值：如果课堂时间和输入的时间冲突,则返回课堂的教室结点指针，否则返回NULL

\*\*/

ROOM\_NODE\* TimeIn(COURSE\_NODE \* pcourse\_node, int week, int date, int start\_class, int end\_class)

{

int i, j;

bool condi1 = false, condi2 = false; //如果冲突，需要满足两个条件

char temp[100] = {'\0'}; //如果不使用temp承接,那么pcourse\_node->course\_room\_name将被更改

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

char \*split;

for(i=0; i<pcourse\_node->term\_period; i++)

{

if(pcourse\_node->course\_term\_period[i][0]<=week

&& week<=pcourse\_node->course\_term\_period[i][1]) //输入的周次是否在课堂的开课周期里面

condi1 = true;

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

split = strtok(room\_name, ",");

ROOM\_NODE \*proom\_node = SeekRoom(gp\_room\_head, split);

room\_name = NULL;

//检查课堂的上课日期是否和date相同

if(pcourse\_node->course\_week\_period[i][j]==date)

{

//检查课堂的起止节数是否在范围内

if((start\_class>=pcourse\_node->course\_day\_period[i][j][0]&&start\_class<=pcourse\_node->course\_day\_period[i][j][1])

||(end\_class>=pcourse\_node->course\_day\_period[i][j][0]&&end\_class<=pcourse\_node->course\_day\_period[i][j][1]))

condi2 = true;

}

//两个条件都满足，返回此时课堂所在的教室信息

if(condi1&&condi2)

return proom\_node;

}

}

return NULL; //搜索失败返回NULL

}

/\*\*

\* 函数名称：SearchCourseNodeByClass

\* 函数参数：class\_hd 班级链首结点指针

\* 函数功能：根据输入的班级查找相应的课堂

\* 返回值：查中时输出课堂信息, 否则返回NULL

\*\*/

bool SearchCourseNodeByClass(CLASS\_NODE \* class\_hd)

{

CLASS\_NODE \*pclass\_node;

char class\_name[30];

printf("请输入需要查找的班级名\n");

getchar();

scanf("%s", class\_name);

pclass\_node = SeekClass(class\_hd, class\_name); //查找对应的班级结点

if(!pclass\_node)

{

printf("查找班级失败\n");

return false;

}

COURSE\_NODE \*pcourse\_node = pclass\_node->firstarc;

if(!pcourse\_node)

{

printf("该班级未参与课堂\n");

return false;

}

while(pcourse\_node)

{

//printf("课堂id为%s\n课堂总人数为:%d\n", pcourse\_node->course\_id, pcourse\_node->total\_stu\_num);

FormatPrint(pcourse\_node);

pcourse\_node = pcourse\_node->next;

}

return true;

}

/\*\*

\* 函数名称：Exit

\* 函数参数：无

\* 函数功能：释放内存，退出系统

\* 返回值：无

\*\*/

void Exit(void)

{

COURSE\_NODE \* pcourse\_node = gp\_course\_head, \*qcourse\_node = NULL;

CLASS\_NODE \* pclass\_node = gp\_class\_head, \*qclass\_node = NULL;

ROOM\_NODE \* proom\_node = gp\_room\_head, \*qroom\_node = NULL;

PROF\_NODE \* pprof\_node = gp\_prof\_head, \*qprof\_node = NULL;

CURI\_NODE \* pcuri\_node = gp\_curi\_head, \*qcuri\_node = NULL;

while(pcourse\_node)

{

qcourse\_node = pcourse\_node;

pcourse\_node = pcourse\_node->next;

free(qcourse\_node);

qcourse\_node = NULL;

}

//printf("课堂内存释放完毕\n");

while(pcuri\_node)

{

qcuri\_node = pcuri\_node;

pcuri\_node = pcuri\_node->next;

free(qcuri\_node);

qcuri\_node = NULL;

}

//printf("课程内存释放完毕\n");

while(pclass\_node)

{

COURSE\_NODE \* qqcourse\_node = NULL;

qclass\_node = pclass\_node;

if(pclass\_node->next)

return;

pclass\_node = pclass\_node->next;

qcourse\_node = pclass\_node->firstarc;

while(qcourse\_node)

{

qqcourse\_node = qcourse\_node;

qcourse\_node = qcourse\_node->next;

free(qqcourse\_node);

qqcourse\_node = NULL;

}

free(qclass\_node);

qclass\_node = NULL;

}

//printf("班级内存释放完毕\n");

//遍历教师信息链表

while(pprof\_node)

{

COURSE\_NODE \* qqcourse\_node = NULL;

qprof\_node = pprof\_node;

if(pprof\_node->next)

return;

pprof\_node = pprof\_node->next;

qcourse\_node = pprof\_node->firstarc;

//删除教师链表后面的课堂链结点

while(qcourse\_node)

{

qqcourse\_node = qcourse\_node;

qcourse\_node = qcourse\_node->next;

free(qqcourse\_node);

qqcourse\_node = NULL;

}

free(qprof\_node);

qprof\_node = NULL;

}

//遍历教室信息链表

while(proom\_node)

{

COURSE\_NODE \* qqcourse\_node = NULL;

qroom\_node = proom\_node;

if(proom\_node->next)

return;

proom\_node = proom\_node->next;

qcourse\_node = pclass\_node->firstarc;

while(qcourse\_node)

{

qqcourse\_node = qcourse\_node;

qcourse\_node = qcourse\_node->next;

free(qqcourse\_node);

}

free(qroom\_node);

qroom\_node->next = NULL;

}

//printf("内存释放完毕\n");

getchar();

return;

}

/\*\*

\* 函数名称：InfoReturn

\* 函数参数：无

\* 函数功能：信息回滚

\* 返回值：无

\*\*/

void InfoReturn(void)

{

CreateClassList(gp\_class\_head);

CreateCuriList(gp\_curi\_head);

CreateProfList(gp\_prof\_head);

CreateRoomList(gp\_room\_head);

}

/\*\*

\* 函数名称：NewOldProf

\* 函数参数：prof\_hd 教师头结点指针

\* 函数功能：输出新老教师名单供用户选择

\* 返回值：无

\*\*/

void NewOldProf(PROF\_NODE \*prof\_hd)

{

char prof\_name[12];

printf("新老教师名单如下:\n");

PROF\_NODE \*pprof\_node = prof\_hd->next;

while(pprof\_node)

{

if(pprof\_node->age<31)

{

printf(" %-12s\t新教师\t%d岁\n", pprof\_node->prof\_name, pprof\_node->age);

}

else if(pprof\_node->age>50)

{

printf(" %-12s\t老教师\t%d岁\n", pprof\_node->prof\_name, pprof\_node->age);

}

pprof\_node=pprof\_node->next;

}

printf("请输入需要分析的新老教师姓名:\n");

getchar();

scanf("%s", prof\_name);

if(!(pprof\_node = SeekProf(prof\_hd, prof\_name)))

{

printf("该教师不存在\n");

return;

}

if(pprof\_node->age<51&&pprof\_node->age>30) //教师为中年教师

{

printf("该教师是中年教师\n");

return;

}

if(pprof\_node->age>50)

{

OldAnalysis(pprof\_node);

}

else

{

NewAnalysis(pprof\_node);

}

return ;

}

/\*\*

\* 函数名称：NewAnalysis

\* 函数参数：pprof\_node 新教师结点指针

\* 函数功能：分析新教师的安排课堂情况

\* 返回值：无

\*\*/

void NewAnalysis(PROF\_NODE \* pprof\_node)

{

if(pprof\_node->prof\_num\_course>3) //新教师教授课堂大于三个

{

printf("该新教师教授%d个课堂，承担教学任务过重\n", pprof\_node->prof\_num\_course);

return;

}

else

{

printf("该新教师教授%d个课堂，承担教学任务合理\n", pprof\_node->prof\_num\_course);

return;

}

}

/\*\*

\* 函数名称：OldAnalysis

\* 函数参数：pprof\_node 老教师结点指针

\* 函数功能：分析老教师的教学学时,爬楼层数

\* 返回值：无

\*\*/

void OldAnalysis(PROF\_NODE \*pprof\_node)

{

int floor\_sum = 0; //爬的楼层总数

int class\_time = 0; //课堂数目

float avg\_floor = 0; //平均爬的楼层数

int existhigh = 0; //标志变量，判断高楼层教室是否存在

char highfloor[100] = {'\0'}; //保存分析所得的高楼层教室名称

int i,j;

COURSE\_NODE \*pcourse\_node = pprof\_node->firstarc; //pcourse\_node指向分析的老教师的课堂支链

//遍历课堂链表

while(pcourse\_node)

{

char temp[100] = {'\0'}; //保存每个课堂的教室名称字符串

strcpy(temp, pcourse\_node->course\_room\_name);

char \*room\_name = temp;

char \*split;

int firsttime = 0; //标志变量

for(i=0; i<pcourse\_node->term\_period; i++)

{

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

class\_time++;

split = strtok(room\_name, ",");

if(firsttime==0&&DealString2(split)>2)

{

existhigh = 1;

firsttime = 1;

strcat(highfloor, pcourse\_node->course\_id);

strcat(highfloor, " ");

}

floor\_sum+=DealString2(split);

room\_name = NULL;

}

}

pcourse\_node = pcourse\_node->next;

}

avg\_floor = floor\_sum\*1.0/class\_time; //计算老教师平均每个课堂要上几层楼

if(pprof\_node->prof\_hours>60)

{

printf("该老教师的教学学时达到了%.1f，请适当缩短其教学时间\n", pprof\_node->prof\_hours);

}

//输出分析信息

if(existhigh)

{

printf("该老教师平均每次上课要爬%.1f层楼，请尽量使老教师在低楼层教室上课\n", avg\_floor);

printf("使得该老教师爬楼3层以上的课堂id有:\n");

printf("%s\n", highfloor);

printf("请尽快修改");

}

return;

}

/\*\*

\* 函数名称：DealString2

\* 函数参数：split 原来课堂结点中的教室信息字符串

\* 函数功能：辅助函数，用于处理字符串

\* 返回值：每个教室所在的楼层数

\*\*/

int DealString2(char \*split)

{

while(\*split)

{

split++; //字符串指针后移

if((\*split)<='9'&&(\*split)>='1')

return ((\*split)-'1');

}

return -1;

}

/\*\*

\* 函数名称：DealString3

\* 函数参数：String 上课日期字符串

a 存放上课日期的数组

\* 函数功能：辅助函数，用于处理字符串

\* 返回值：如果输入格式正确返回true，否则返回false

\*\*/

bool DealString3(char \*String, int \*a)

{

int i = 0;

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

a[i]=0;

int shouldbenum = 1;

int dot\_num = 0;

char \*temp = String;

//判断输入是否符合规范

while (\*temp)

{

if(shouldbenum==1&&(!isdigit(\*temp)))

return false;

if(((\*temp)!=',')&&(!isdigit(\*temp)))

return false;

if((\*temp)==',')

{

dot\_num++;

shouldbenum = 1;

}

if(isdigit(\*temp))

{

shouldbenum = 0;

}

temp++;

}

int count = dot\_num+1;

temp = String;

char \*split;

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

{

split = strtok(temp, ",");

a[i] = atoi(split);

temp = NULL;

}

return true;

}

/\*\*

\* 函数名称：ContraRoom

\* 函数参数：proom\_node 教室结点

pcourse\_node 课堂结点

\* 函数功能：判断教室和课堂是否产生时间冲突

\* 返回值：如果产生冲突则返回true，否则返回false

\*\*/

bool ContraRoom(ROOM\_NODE \*proom\_node, COURSE\_NODE \*pcourse\_node, COURSE\_NODE \*course\_temp)

{

//char temp[100] = {'\0'};

//strcpy(temp, pcourse\_node->course\_room\_name);

//char \*room\_name = temp;

//char \*split;

int i,j;

for(i=0; i<pcourse\_node->term\_period; i++)

{

for(j=0; j<pcourse\_node->day\_period[i]; j++)

{

int start\_class = pcourse\_node->course\_day\_period[i][j][0];

int end\_class = pcourse\_node->course\_day\_period[i][j][1];

int start\_week = pcourse\_node->course\_term\_period[i][0];

int end\_week = pcourse\_node->course\_term\_period[i][1];

int date = pcourse\_node->course\_week\_period[i][j];

int week;

//分解教室名构成的字符串

//split = strtok(room\_name, ",");

//ROOM\_NODE \*room\_temp = SeekRoom(gp\_room\_head, split);

//room\_name = NULL;

int k;

for(k=start\_week;k<=end\_week;k++)

{

week = k;

if(proom\_node==TimeIn(course\_temp, week, date, start\_class, end\_class))

return true;

}

}

}

return false;

}

**course.h 头文件**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <io.h>

#include <fcntl.h>

#include <sys\stat.h>

#include <ctype.h>

#include <time.h>

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#include <cassert>

using namespace std;

#define DAY\_CLASS 760

#define NIGHT\_CLASS 380

#define TOTAL\_CLASS 1140

#define WINTER\_INCREMENT 0.4

#define SUMMER\_INCREMENT 0.6

#define NIGHT\_INCREMENT 0.1

#ifndef COURSE\_H\_\_

#define COURSE\_H\_\_

/\*\*

\*课堂信息链结点结构

\*/

typedef struct course\_node {

char course\_id[20]; /\*\*< 课堂id\*/

char course\_curi\_name[30]; /\*\*< 课堂对应课程名称\*/

char course\_prof\_name[14]; /\*\*< 课堂教师名称\*/

int course\_class\_num; /\*\*< 课堂下的班级数量\*/

char course\_class\_name[6][20]; /\*\*< 课堂下各个班级的名称\*/

char course\_room\_name[100]; /\*\*< 课堂下教室名称组成的字符串\*/

int total\_stu\_num; /\*\*< 课堂下学生总数\*/

int term\_period; /\*\*< 一学期内该课堂分几段上课周期\*/

int day\_period[5] = {0}; /\*\*< 每一段上课周期内的天数\*/

int course\_week\_period[5][6] = {{0}}; /\*\*< 周几上课\*/

int course\_day\_period[5][5][2] = {{{0}}}; /\*\*< 每天上课的时间段\*/

int course\_term\_period[5][2] = {{0}}; /\*\*< 每个上课周期内的起止周数\*/

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

} COURSE\_NODE;

/\*\*

\*课程信息链结点结构

\*/

typedef struct curi\_node{

char curi\_name[30]; /\*\*< 课程名称\*/

float credit; /\*\*< 学分\*/

float hours; /\*\*< 学时\*/

int course\_num = 0; /\*\*< 课堂数目\*/

struct course\_node \* firstarc; /\*\*< 指向第一个课堂结点\*/

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

} CURI\_NODE;

/\*\*

\*教师信息链结点结构

\*/

typedef struct prof\_node {

char prof\_name[12]; /\*\*< 教师姓名\*/

char prof\_title[12]; /\*\*< 教师头衔\*/

struct course\_node \* firstarc; /\*\*< 指向第一个课堂结点\*/

float prof\_hours = 0; /\*\*< 教师一学期教学时长\*/

int age = 0; /\*\*< 教师年龄\*/

int prof\_num\_course = 0; /\*\*< 教师带的课堂数\*/

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

} PROF\_NODE;

/\*\*

\*教室信息链结点结构

\*/

typedef struct room\_node {

int volume; /\*\*< 教室容量\*/

char room\_name[30]; /\*\*< 教室名称\*/

int day\_use\_time = 0; /\*\*< 教室白天使用时长\*/

int night\_use\_time = 0; /\*\*< 教室夜晚使用时长\*/

int total\_use\_time = 0; /\*\*< 教室总体使用时长\*/

float energy; /\*\*< 教室节能效率\*/

float day\_use\_rate; /\*\*< 教室白天使用率\*/

float night\_use\_rate; /\*\*< 教室夜晚使用率\*/

float total\_use\_rate; /\*\*< 教室总体利用率\*/

int winter\_use\_time = 0; /\*\*< 教室冬天使用时长\*/

int summer\_use\_time = 0; /\*\*< 教室夏天使用时长\*/

struct course\_node \*firstarc; /\*\*< 指向第一个课堂结点指针\*/

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

} ROOM\_NODE;

/\*\*

班级信息链结点结构

\*/

typedef struct class\_node {

int class\_stu\_amount; /\*\*< 班级学生人数\*/

char class\_name[30]; /\*\*< 班级名称\*/

float rate; /\*\*< 班级夜日上课时间比\*/

int day\_class\_hours = 0; /\*\*< 班级白天上课节数\*/

int night\_class\_hours = 0; /\*\*< 班级夜晚上课节数\*/

int winter\_class\_hours = 0; /\*\*< 班级冬天上课节数\*/

int summer\_class\_hours = 0; /\*\*< 班级夏天上课节数\*/

struct class\_node \* next; /\*\*< 指向下一个结点的指针\*/

struct course\_node \*firstarc; /\*\*< 指向第一个课堂结点指针\*/

} CLASS\_NODE;

CURI\_NODE \*gp\_curi\_head = NULL; /\*课程信息链头指针\*/

COURSE\_NODE \*gp\_course\_head = NULL; /\*课堂信息链头指针\*/

CLASS\_NODE \*gp\_class\_head = NULL; /\*班级信息链头指针\*/

ROOM\_NODE \*gp\_room\_head = NULL; /\*教室信息链头指针\*/

PROF\_NODE \*gp\_prof\_head = NULL; /\*教师信息链头指针\*/

char const\*gp\_curi\_info\_filename = "Curi.txt";

char const\*gp\_prof\_info\_filename = "Prof.txt";

char const\*gp\_class\_info\_filename = "Class.txt";

char const\*gp\_room\_info\_filename = "Room.txt";

char const\*gp\_course\_info\_filename = "Course.txt";

void InsertCuriNode(CURI\_NODE \* gp\_curi\_head); /\*插入课程信息\*/

void InsertRoomNode(ROOM\_NODE \* gp\_room\_head); /\*插入教室信息\*/

void InsertClassNode(CLASS\_NODE \* gp\_class\_head); /\*插入班级信息\*/

void InsertProfNode(PROF\_NODE \* gp\_prof\_head); /\*插入教师信息\*/

void InsertCourseNode(COURSE\_NODE \* gp\_course\_head);/\*插入课堂信息\*/

bool SaveCuriNode(CURI\_NODE \* gp\_curi\_head); /\*保存课程信息\*/

bool SaveRoomNode(ROOM\_NODE \* gp\_room\_head); /\*保存教室信息\*/

bool SaveClassNode(CLASS\_NODE \* gp\_class\_head); /\*保存班级信息\*/

bool SaveProfNode(PROF\_NODE \* gp\_prof\_head); /\*保存教师信息\*/

bool SaveCourseNode(COURSE\_NODE \* gp\_course\_head); /\*保存课堂信息\*/

bool ModiCuriNode(CURI\_NODE \* gp\_curi\_head); /\*修改课程信息\*/

bool ModiRoomNode(ROOM\_NODE \* gp\_room\_head); /\*修改教室信息\*/

bool ModiClassNode(CLASS\_NODE \* gp\_class\_head); /\*修改班级信息\*/

bool ModiProfNode(PROF\_NODE \* gp\_prof\_head); /\*修改教师信息\*/

bool ModiCourseNode(COURSE\_NODE \* gp\_course\_head); /\*修改课堂信息\*/

bool DelCuriNode(CURI\_NODE \* gp\_curi\_head); /\*删除课程信息\*/

bool DelRoomNode(ROOM\_NODE \* gp\_room\_head); /\*删除教室信息\*/

bool DelClassNode(CLASS\_NODE \* gp\_class\_head); /\*删除班级信息\*/

bool DelProfNode(PROF\_NODE \* gp\_prof\_head); /\*删除教师信息\*/

bool DelCourseNode(COURSE\_NODE \* gp\_course\_head); /\*删除课堂信息\*/

void UpdateAfterDelCourse(COURSE\_NODE \* pcourse\_node); /\*删除课堂后更新其他结点的信息\*/

void ModiProfAfterDel(PROF\_NODE \*pprof\_node, COURSE\_NODE \*pcourse\_node); /\*删除课堂后修改教师信息\*/

void ModiCuriAfterDel(CURI\_NODE \*pcuri\_node, COURSE\_NODE \*pcourse\_node); /\*删除课堂后修改课程信息\*/

void ModiClassAfterDel(CLASS\_NODE \*pclass\_node, COURSE\_NODE \*pcourse\_node); /\*删除课堂后修改班级信息\*/

void ModiRoomAfterDel(ROOM\_NODE \*proom\_node, COURSE\_NODE \*pcourse\_node, int num\_of\_week, int start\_class, int end\_class); /\*删除课堂后修改教室信息\*/

bool SearchCuriNode(CURI\_NODE \* gp\_curi\_head); /\*搜索课程信息\*/

bool SearchRoomNode(ROOM\_NODE \* gp\_room\_head); /\*搜索教室信息\*/

bool SearchClassNode(CLASS\_NODE \* gp\_class\_head); /\*搜索班级信息\*/

bool SearchProfNode(PROF\_NODE \* gp\_prof\_head); /\*搜索教师信息\*/

bool SearchCourseNode(COURSE\_NODE \* gp\_course\_head); /\*搜索课堂信息\*/

bool SearchCourseNodeByClass(CLASS\_NODE \* class\_hd); /\*通过班级搜索课堂信息\*/

bool SearchCourseNodeByProf(PROF\_NODE \* prof\_hd); /\*通过教师搜索课堂信息\*/

bool SearchCourseNodeByRoom(ROOM\_NODE \* room\_hd); /\*通过教室搜索课堂信息\*/

bool SearchCourseNodeByTime(COURSE\_NODE \* course\_node); /\*通过时间搜索课堂信息\*/

bool SearchCourseNodeByCuri(CURI\_NODE \* curi\_hd); /\*通过课程搜索课堂信息\*/

CURI\_NODE\* CreateCuriList(CURI\_NODE \*curi\_hd); /\*创建课程信息链表\*/

ROOM\_NODE \* CreateRoomList(ROOM\_NODE \*room\_hd); /\*创建教室信息链表\*/

CLASS\_NODE \* CreateClassList(CLASS\_NODE \*class\_hd); /\*创建班级信息链表\*/

PROF\_NODE \* CreateProfList(PROF\_NODE \* gp\_prof\_head); /\*创建教师信息链表\*/

COURSE\_NODE \*CreateCourseList(COURSE\_NODE \* gp\_course\_head); /\*创建课堂信息链表\*/

CURI\_NODE \* SeekCuri(CURI\_NODE \* gp\_curi\_head, char \*curi\_name); /\*查找课程信息\*/

ROOM\_NODE \* SeekRoom(ROOM\_NODE \* gp\_room\_head, char \*room\_name); /\*查找教室信息\*/

CLASS\_NODE \* SeekClass(CLASS\_NODE \* gp\_class\_head, char \*class\_name); /\*查找班级信息\*/

PROF\_NODE \* SeekProf(PROF\_NODE \* gp\_prof\_head, char \*prof\_name); /\*查找教师信息\*/

COURSE\_NODE \* SeekCourse(COURSE\_NODE \* gp\_curi\_head, char \*course\_id); /\*查找课堂信息\*/

ROOM\_NODE\* TimeIn(COURSE\_NODE \* pcourse\_node, int week, int date, int start\_class, int end\_class);/\*根据课堂和时间查找教室\*/

void FindFreeRoom(ROOM\_NODE \*room\_hd); /\*查找空闲教室\*/

void ProfAnalysis(PROF\_NODE \* head); /\*分析教师的课堂安排合理性\*/

void ClassAnalysis(CLASS\_NODE \*head); /\*分析班级的课堂安排合理性\*/

void EnergyAnalysis(ROOM\_NODE \*head); /\*分析教室的能效\*/

void OldAnalysis(PROF\_NODE \*pprof\_node); /\*分析老教师的课堂安排合理性\*/

void NewAnalysis(PROF\_NODE \* pprof\_node); /\*分析新教师的课堂安排合理性\*/

void NewOldProf(PROF\_NODE\* gp\_prof\_head); /\*分析新老教师安排课堂合理性\*/

void RoomUseRate(ROOM\_NODE \* head); /\*分析教室利用率\*/

void DelCourseAfterC(CLASS\_NODE \*pclass\_node, COURSE\_NODE \*course\_hd);

void DelCourseAfterCuri(CURI\_NODE \*pcuri\_node, COURSE\_NODE \*course\_hd); /\*删除课程后删除相关课堂结点\*/

CLASS\_NODE \* Cre\_Class\_Course\_List(CLASS\_NODE \* gp\_class\_head, COURSE\_NODE \* course\_class\_head); /\*创建班级-课堂邻接表\*/

PROF\_NODE \* Cre\_Prof\_Course\_List(PROF\_NODE \* gp\_prof\_head, COURSE\_NODE \* cprof\_course\_head); /\*创建教师-课堂邻接表\*/

ROOM\_NODE \* Cre\_Room\_Course\_List(ROOM\_NODE \* gp\_room\_head, COURSE\_NODE \* course\_room\_head); /\*创建教室-课堂邻接表\*/

void InsertCrossList(COURSE\_NODE \*pcourse\_node); /\*插入课堂结点时修改邻接表\*/

void ModiCrossForClass(COURSE\_NODE \*pcourse\_node); /\*修改班级结点后修改邻接表\*/

void UpdateRoomUseTime(ROOM\_NODE \* proom\_node, int num\_of\_week, int start\_class, int end\_class); /\*更新课堂后会更新教室相关信息\*/

void UpdateClassUseTime(CLASS\_NODE \* pclass\_node, COURSE\_NODE \* pcourse\_node); /\*更新课堂后会更新班级相关信息\*/

bool ModiCuriInCourse(COURSE\_NODE \*course\_hd, char \* old\_curi\_name, char \* new\_curi\_name, float old\_hours, float new\_hours); /\*修改因课程信息修改而需要变动的课堂信息\*/

bool ModiClassInCourse(COURSE\_NODE \*course\_hd, CLASS\_NODE \*pclass\_node, int old\_stu\_amount); /\*修改因班级信息修改而需要变动的课堂信息\*/

bool ModiCourseAfterDelC(CLASS\_NODE \*pclass\_node); /\*删除班级结点后修改课堂信息\*/

bool DelWithoutId(COURSE\_NODE \*pcourse\_node); /\*删除传入的课堂结点\*/

void InsertCourseWithoutInput(COURSE\_NODE \*pcourse\_node); /\*插入传入的课堂结点\*/

bool ContraRoom(ROOM\_NODE \*proom\_node, COURSE\_NODE \*pcourse\_node, COURSE\_NODE \*course\_temp); /\*判断教室和课堂的时间是否冲突\*/

//极为次要的辅助函数

bool cpycoursenode(COURSE\_NODE \* a, COURSE\_NODE \*b); /\*将b课堂结点除指针域外的所有信息复制到结点a\*/

void FormatPrint(COURSE\_NODE \*pcourse\_node); /\*格式化输出一个课堂的信息\*/

void InfoReturn(void);

void QuickSortRate(ROOM\_NODE \*head, ROOM\_NODE \*tail);

void QuickSortEnergy(ROOM\_NODE \*head, ROOM\_NODE \*tail);

void Initarray(COURSE\_NODE \*pcourse\_node);

bool DealString1(char \*String, int \*start\_class, int \*end\_class);

int DealString2(char \*split);

void Exit(void);

bool DealString3(char \*String, int \*a);

#endif

**Experiment2.h** **头文件**

#define OK 1

#define TRUE 1

#define FALSE 0

#define ERROR -2

#include "course.h"

typedef int Status;

Status InitaCuriList(CURI\_NODE \*\*L); // 初始化课程信息链表

Status InitaRoomList(ROOM\_NODE \*\*L); // 初始化教室信息链表

Status InitaProfList(PROF\_NODE \*\*L); // 初始化教师信息链表

Status InitaClassList(CLASS\_NODE \*\*L); // 初始化班级信息链表

Status InitaCourseList(COURSE\_NODE \*\*L); // 初始化课堂信息链表

Status ListCuriTrabverse(CURI\_NODE \* curi\_hd); //遍历课程信息单链表

Status ListRoomTrabverse(ROOM\_NODE \* room\_hd); //遍历教室信息单链表

Status ListProfTrabverse(PROF\_NODE \* prof\_hd); //遍历教师信息单链表

Status ListClassTrabverse(CLASS\_NODE \* class\_hd); //遍历班级信息单链表

Status ListCourseTrabverse(COURSE\_NODE \* course\_head); //遍历课堂信息单链表

Status ListClassCourseTrabverse(CLASS\_NODE \* class\_head); //遍历班级-课堂邻接表

Status ListProfCourseTrabverse(PROF\_NODE \* prof\_head); //遍历教师-课堂邻接表

Status ListRoomCourseTrabverse(ROOM\_NODE \* room\_head); //遍历教室-课堂邻接表

CURI\_NODE\* PriorCuriElem(CURI\_NODE \* L, char \*curi\_name); //求指定课程结点前驱结点

ROOM\_NODE\* PriorRoomElem(ROOM\_NODE \* L, char \*room\_name); //求指定教室结点前驱结点

COURSE\_NODE\* PriorCourseElem(COURSE\_NODE \* L, char \*course\_id); //求指定课堂结点前驱结点

PROF\_NODE\* PriorProfElem(PROF\_NODE \* L, char \*prof\_name); //求指定教师结点前驱结点

CLASS\_NODE\* PriorClassElem(CLASS\_NODE \* L, char \*class\_name); //求指定班级结点前驱结点