C 语言课程设计 教学信息管理系统 大学编程作业（TUST 天津科技大学 2021 年）

一、项目简介

本教学信息管理系统 ，我使用了链表数据结构来制作，实现了简单的增删查改逻辑，实现了文件的存储，并且终端界面较为美观易用。通过这次 C 语言课程设计的实践，我巩固了数据结构的知识，熟练应用了 C 语言指针。

这个项目是我大二写的，现在回顾已经非常粗糙，分享出来一方面希望可以帮助初学者，另一方面希望能让同学们可以从目前大学中普遍毫无价值的形式主义作业中解脱出来，更加高效地学习优质计算机知识和主流编程技术，一起发扬开源精神，感受互联网技术的美好愿景。

二、项目要求

某一教学信息管理系统，有如下基础信息：

1. 教师信息（teacher.dat）包括教师号（大于 1 的正整数）、姓名、性别、学历、年龄、电话等（教师号不能重复）；
2. 课程信息（course.dat）包括课程号（大于 1 的正整数）、课程名称；
3. 教师与课程关联信息（tc.dat）包括教师与课程关联信息序号（简称：教课号，大于 1 的正整数）、开课时间（如 2021 年上半年的课程，开课时间为 202101，下半年开课的时间为 202102）、教师号、课程号；
4. 学生信息（student.dat）包括学号（大于 1 的正整数）、姓名；
5. 学生与课程关联信息（sc.dat）其信息包括学生与课程关联信息序号（简称：学课号，大于 1 的正整数）、学号、教课号；

必做要求：

1. 实现教师信息、课程信息、教师与课程关联信息、学生信息、学生与课程关联信息的录入；
2. 实现教师信息（教师号、姓名、性别、学历、年龄、电话）、课程信息（课程号、课程名称）、教师与课程关联信息（教课号、开课时间、教师号、教师名称、课程号、课程名称）、学生信息（学号、姓名）、学生与开课课程关联信息（学号、姓名、教课号、课程名称、开课时间）浏览；
3. 可实现对教师、学生与开课课程关联信息的任意查询；

选做要求：

1. 可实现对涉及的除教师号、课程号、教课号、学号、学课号外其它信息进行修改；
2. 可实现对信息进行逻辑删除（也就是对信息进行作废标记，物理上不删除）；
3. 假设老师每上一门课的基础课时量为 2，选课人数每超过 3 人（如：选课人数 <= 3 时，记课时量为 2；3 < 选课人数 <= 6 时，记课时量为 3；以此类推），则课时量加 1，根据教师号、开课时间统计并显示指定教师某学期的课时量。
4. List item

课设打分要求：

1. 须提交一个可通过编译的满足要求的 C 语言程序源文件；
2. 要有运行情况截图；
3. 如果完成必做要求的，得 80 分，否则酌情扣分；
4. 程序格式优雅，有适当得缩进、空行，标识符命名规范何理得，可酌情加分，最高 20 分。
5. 程序代码无法通过编译的，可根据错误现象酌情扣分。
6. 程序有严重问题，不具备基本输入、输出，或未完成核心算法的，最高得 50 分。

三、项目源码

程序设计思路：

1. 界面打印
2. 数据结构的设计，抽象出教学信息
3. 链表函数的设计，实现增删
4. 查改
5. 功能交互
6. 文件读写

```c
/*

教学信息管理系统_C语言课程设计

一、某一教学信息管理系统，有如下基础信息：
    1、教师信息（teacher.dat）包括教师号（大于1的正整数）、姓名、性别、学历、年龄、电话等（教师号不能重复）
    2、课程信息（course.dat）包括课程号（大于1的正整数）、课程名称；
    3、教师与课程关联信息（tc.dat）包括教师与课程关联信息序号（简称：教课号，大于1的正整数）、
       开课时间（如2021年上半年的课程，开课时间为202101，下半年开课的时间为202102）、教师号、课程号；
    4、学生信息（student.dat）包括学号（大于1的正整数）、姓名；
    5、学生与课程关联信息（sc.dat）其信息包括学生与课程关联信息序号（简称：学课号，大于1的正整数）、学号、
二、必做要求：
    1、实现教师信息、课程信息、教师与课程关联信息、学生信息、学生与课程关联信息的录入；（30分）
    2、实现教师信息（教师号、姓名、性别、学历、年龄、电话）、课程信息（课程号、课程名称）、
       教师与课程关联信息（教课号、开课时间、教师号、教师名称、课程号、课程名称）、学生信息（学号、姓名）、
       学生与开课课程关联信息（学号、姓名、教课号、课程名称、开课时间）浏览；（20分）
    3、可实现对（1）中教师、学生与开课课程关联信息的任意查询；（30分）
三、选做要求（三选一）：
    1、可实现对 二.1 中涉及的除教师号、课程号、教课号、学号、学课号外其它信息进行修改；（20分）
    2、可实现对 二.1中的信息进行逻辑删除（也就是对信息进行作废标记，物理上不删除）；（20分）
    3、假设老师每上一门课的基础课时量为2，选课人数每超过3人（如：选课人数 <= 3时，记课时量为2；3 < 选课人数
       记课时量为3；以此类推），则课时量加1，根据教师号、开课时间统计并显示指定教师某学期的课时量。（20分）
四、课设打分要求：
    1、须提交一个可通过编译的满足一、二要求的C语言程序源文件；
    2、要有运行情况截图；
    3、如果完成1、2要求的，得80分，否则根据一、二标注分值酌情扣分；
    4、程序格式优雅，有适当得缩进、空行，标识符命名规范何理得，可酌情加分，最高20分。
    5、程序代码无法通过编译的，可根据错误现象酌情扣分。
    6、程序有严重问题，不具备基本输入、输出，或未完成核心算法的，最高得50分。

*/

/*

    程序设计思路

    1.界面打印
    2.数据结构的设计，抽象出教学信息
    3.链表函数的设计，实现增删查改
    4.功能交互
    5.文件读写

*/
#define _CRT_SECURE_NO_WARNINGS
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>        //防止闪屏：system("pause");

/*1.界面打印*/
/*主菜单函数*/
void Menu();

/*显示信息菜单函数*/
void MenuDispiaysTheInformation();

/*录入信息菜单函数*/
void MenuEnterTheInformation();

/*删除信息菜单函数*/
void MenuDeleteTheInformation();

/*查找信息菜单函数*/
void MenuFindTheInformation();

/*修改信息菜单函数*/
void MenuReviseTheInformation();

/*2.数据结构的设计，抽象出教学信息*/
struct TeacherDat    //教师信息
{
    int teacher_number;    //教师号
    char teacher_name[40];        //教师姓名
    char teacher_sex[8];        //教师性别
    char teacher_academic_degree[40];    //教师学历
    char teacher_age[8];        //教师年龄
    char teacher_phone_number[40];    //教师电话号
};

struct CourseDat//课程信息
{
    int course_number;//课程号
    char course_name[40];//课程名称
};

struct TeacherAndCourseDat//教师与课程关联信息
{
    int teaching_class_number;//教课号
    char class_time[40];//开课时间
    int teacher_number;    //教师号
    int course_number;//课程号
};

struct StudentDat//学生信息
{
    int student_number;//学号
    char student_name[40];//学生姓名
};

struct StudentAndCourseDat//学生与课程关联信息
{
    int learn_class_number;//学课号
    int student_number;//学号
    int teaching_class_number;//教课号
};

/*3.链表函数的设计，实现增删查改*/
struct NodeTeacherDat//教师信息链表中的节点结构体
{
    struct TeacherDat data_teacher;//教师信息数据域
    struct NodeTeacherDat* next_teacher;//教师信息数据指针
};

struct NodeTeacherDat* List_teacher;//创建全局List_teacher指针变量给交互功能函数

/*创建教师信息链表的函数*/
struct NodeTeacherDat* CreatList_TeacherDat();

/*创建教师信息结点的函数*/
struct NodeTeacherDat* CreateNode_TeacherDat(struct TeacherDat data_teacher);

/*插入教师信息结点的函数*/
void InsertNodeByHead_TeacherDat(struct NodeTeacherDat* head_teacher, struct TeacherDat data_teach

/*指定位置教师信息删除的函数*/
void DeleteAppoinNode_TeacherDat(struct NodeTeacherDat* head_teacher, char* teacher_name);

/*查找指定位置教师信息的结点的函数*/
struct NodeTeacherDat* SearchInfoByDate_TeacherDat(struct NodeTeacherDat* head_teacher, char* teac

/*指定位置教师信息修改的函数*/
void ReviseAppoinNode_TeacherDat(struct NodeTeacherDat* head_teacher, char* teacher_name);
```

```c
/*打印教师信息链表的函数*/
void PrintList_TeacherDat(struct NodeTeacherDat* head_teacher);


struct NodeCourseDat//课程信息链表中的节点结构体
{
    struct CourseDat data_course;//课程信息数据域
    struct NodeCourseDat* next_course;//课程信息数据指针
};

struct NodeCourseDat* List_course;//创建全局List_course指针变量给交互功能函数

/*创建课程信息链表的函数*/
struct NodeCourseDat* CreatList_CourseDat();

/*创建课程信息结点的函数*/
struct NodeCourseDat* CreateNode_CourseDat(struct CourseDat data_course);

/*插入课程信息结点的函数*/
void InsertNodeByHead_CourseDat(struct NodeCourseDat* head_course, struct CourseDat data_course);

/*指定位置课程信息删除的函数*/
void DeleteAppoinNode_CourseDat(struct NodeCourseDat* head_course, int course_number);

/*查找指定位置课程信息的结点的函数*/
struct NodeCourseDat* SearchInfoByDate_CourseDat(struct NodeCourseDat* head_course, int course_num

/*指定位置课程信息修改的函数*/
void ReviseAppoinNode_CourseDat(struct NodeCourseDat* head_course, int course_number);

/*打印课程信息链表的函数*/
void PrintList_CourseDat(struct NodeCourseDat* head_course);


struct NodeTeacherAndCourseDat//教师与课程信息链表中的节点结构体
{
    struct TeacherAndCourseDat data_teacherandcourse;//教师与课程信息数据域
    struct NodeTeacherAndCourseDat* next_teacherandcourse;//教师与课程信息据指针
};

struct NodeTeacherAndCourseDat* List_teacherandcourse;//创建全局List_teacherandcourse指针变量给交互

/*创建教师与课程信息链表的函数*/
```

```
struct NodeTeacherAndCourseDat* CreatList_TeacherAndCourseDat();

/*创建教师与课程信息结点的函数*/
struct NodeTeacherAndCourseDat* CreateNode_TeacherAndCourseDat(struct TeacherAndCourseDat data_tea

/*插入教师与课程信息结点的函数*/
void InsertNodeByHead_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, s

/*指定位置教师与课程信息删除的函数*/
void DeleteAppoinNode_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, i

/*查找指定位置教师与课程信息的结点的函数*/
struct NodeTeacherAndCourseDat* SearchInfoByDate_TeacherAndCourseDat(struct NodeTeacherAndCourseDa

/*指定位置教师与课程信息修改的函数*/
void ReviseAppoinNode_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, i

/*打印教师与课程信息链表的函数*/
void PrintList_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse);


struct NodeStudentDat//学生信息链表中的节点结构体
{
    struct StudentDat data_student;//学生信息数据域
    struct NodeStudentDat* next_student;//学生信息数据指针
};

struct NodeStudentDat* List_student;//创建全局List_student指针变量给交互功能函数

/*创建学生信息链表的函数*/
struct NodeStudentDat* CreatList_StudentDat();

/*创建学生信息结点的函数*/
struct NodeStudentDat* CreateNode_StudentDat(struct StudentDat data_student);

/*插入学生信息结点的函数*/
void InsertNodeByHead_StudentDat(struct NodeStudentDat* head_student, struct StudentDat data_stude

/*指定位置学生信息删除的函数*/
void DeleteAppoinNode_StudentDat(struct NodeStudentDat* head_student, char* student_name);

/*查找指定位置学生信息的结点的函数*/
struct NodeStudentDat* SearchInfoByDate_StudentDat(struct NodeStudentDat* head_student, char* stuc
/*指定位置学生信息修改的函数*/
void ReviseAppoinNode_StudentDat(struct NodeStudentDat* head_student, char* student_name);

/*打印学生信息链表的函数*/
void PrintList_StudentDat(struct NodeStudentDat* head_student);


struct NodeStudentAndCourseDat//学生与课程信息链表中的节点结构体
{
    struct StudentAndCourseDat data_studentandcourse;//学生与课程信息数据域
    struct NodeStudentAndCourseDat* next_studentandcourse;//学生与课程信息据指针
};

struct NodeStudentAndCourseDat* List_studentandcourse;//创建全局List_studentandcourse指针变量给交互

/*创建学生与课程信息链表的函数*/
struct NodeStudentAndCourseDat* CreatList_StudentAndCourseDat();

/*创建学生与课程信息结点的函数*/
struct NodeStudentAndCourseDat* CreateNode_StudentAndCourseDat(struct StudentAndCourseDat data_stu

/*插入学生与课程信息结点的函数*/
void InsertNodeByHead_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, s

/*指定位置学生与课程信息删除的函数*/
void DeleteAppoinNode_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, i

/*查找指定位置学生与课程信息的结点的函数*/
struct NodeStudentAndCourseDat* SearchInfoByDate_StudentAndCourseDat(struct NodeStudentAndCourseDa

/*指定位置学生与课程信息修改的函数*/
void ReviseAppoinNode_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, i

/*打印学生与课程信息链表的函数*/
void PrintList_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse);

/*4.功能交互*/
/*功能交互函数*/
void KeyDown();

/*5.文件读写*/
/*文件读read操作: 链表数据的读取函数*/
void ReadInfoFromFile_TeacherDat(struct NodeTeacherDat* head_teacher, char* filename_teacher);//字
void ReadInfoFromFile_CourseDat(struct NodeCourseDat* head_course, char* filename_course);//字符串
```

```c
void ReadInfoFromFile_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, c
void ReadInfoFromFile_StudentDat(struct NodeStudentDat* head_student, char* filename_student);//字
void ReadInfoFromFile_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, c

/*文件写write操作: 链表数据的存储函数*/
void WriteInfoToFile_TeacherDat(struct NodeTeacherDat* head_teacher, char* filename_teacher); // 字
void WriteInfoToFile_CourseDat(struct NodeCourseDat* head_course, char* filename_course);//字符串常
void WriteInfoToFile_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, ch
void WriteInfoToFile_StudentDat(struct NodeStudentDat* head_student, char* filename_student);//字符
void WriteInfoToFile_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, ch

/*主函数*/
int main()
{
    /*所有操作都同步到文件*/

    List_teacher = CreatList_TeacherDat();//全局List_teacher指针变量给交互功能函数
    List_course = CreatList_CourseDat();//全局List_course指针变量给交互功能函数
    List_teacherandcourse = CreatList_TeacherAndCourseDat();//全局List_teacherandcourse指针变量给交
    List_student = CreatList_StudentDat();//全局List_student指针变量给交互功能函数
    List_studentandcourse = CreatList_StudentAndCourseDat();//全局List_studentandcourse指针变量给交

   /*文件读read操作*/
    ReadInfoFromFile_TeacherDat(List_teacher, "teacher.dat.txt");
    ReadInfoFromFile_CourseDat(List_course, "course.dat.txt");
    ReadInfoFromFile_TeacherAndCourseDat(List_teacherandcourse, "teacherandcourse.dat.txt");
    ReadInfoFromFile_StudentDat(List_student, "student.dat.txt");
    ReadInfoFromFile_StudentAndCourseDat(List_studentandcourse, "studentandcourse.dat.txt");

    /*运行界面及功能代码*/
    while (1)    //条件为真
    {
        Menu();
        KeyDown();

        system("pause");
        system("cls");
    }

    return 0;
}

/*1.界面打印*/
/*主菜单函数*/
```

```c
void Menu()
{
    system("cls");//清屏
    system("color 0b");//修改字体颜色，0表示背景为黑色，b表示字体为淡浅绿色
    printf("欢迎使用教学信息管理系统！(^▽^ )\n\n");
    printf("-----------------------------------------\n");
    printf("|\t\t\t\t\t|\n");
    printf("|\t    【教学信息管理系统】     \t|\n");
    printf("|\t\t\t\t\t|\n");
    printf("|\t\t1.显示信息\t\t|\n");
    printf("|\t\t2.录入信息\t\t|\n");
    printf("|\t\t3.删除信息\t\t|\n");
    printf("|\t\t4.查找信息\t\t|\n");
    printf("|\t\t5.修改信息\t\t|\n");
    printf("|\t\t6.退出系统\t\t|\n");
    printf("|\t\t\t\t\t|\n");
    printf("-----------------------------------------\n\n");
    printf("请输入您要进行的操作（1-6）：\n");
}

/*显示信息菜单函数*/
void MenuDispiaysTheInformation()
{
    printf("--------------【显示信息】---------------\n\n");
    printf("\t1.显示教师信息\t\n");
    printf("\t2.显示课程信息\t\n");
    printf("\t3.显示教师与课程关联信息信息\t\n");
    printf("\t4.显示学生信息\t\n");
    printf("\t5.显示学生与开课课程关联信息\t\n");
    printf("\t6.退出\t\n");
    printf("\t\t\t\t\t\n");
    printf("-----------------------------------------\n\n");
    printf("请输入您要进行的操作（1-6）：\n");
}

/*录入信息菜单函数*/
void MenuEnterTheInformation()
{
    printf("--------------【录入信息】---------------\n\n");
    printf("\t1.录入教师信息\t\n");
    printf("\t2.录入课程信息\t\n");
    printf("\t3.录入教师与课程关联信息信息\t\n");
    printf("\t4.录入学生信息\t\n");
    printf("\t5.录入学生与开课课程关联信息\t\n");
    printf("\t6.退出\t\n");
    printf("\t\t\t\t\t\n");
    printf("-----------------------------------------\n\n");
    printf("请输入您要进行的操作（1-6）：\n");
}

/*删除信息菜单函数*/
void MenuDeleteTheInformation()
{
    printf("--------------【删除信息】---------------\n\n");
    printf("\t1.删除教师信息\t\n");
    printf("\t2.删除课程信息\t\n");
    printf("\t3.删除教师与课程关联信息信息\t\n");
    printf("\t4.删除学生信息\t\n");
    printf("\t5.删除学生与开课课程关联信息\t\n");
    printf("\t6.退出\t\n");
    printf("\t\t\t\t\t\n");
    printf("-----------------------------------------\n\n");
    printf("请输入您要进行的操作（1-6）：\n");
}

/*查找信息菜单函数*/
void MenuFindTheInformation()
{
    printf("--------------【查找信息】---------------\n\n");
    printf("\t1.查找教师信息\t\n");
    printf("\t2.查找课程信息\t\n");
    printf("\t3.查找教师与课程关联信息信息\t\n");
    printf("\t4.查找学生信息\t\n");
    printf("\t5.查找学生与开课课程关联信息\t\n");
    printf("\t6.退出\t\n");
    printf("\t\t\t\t\t\n");
    printf("-----------------------------------------\n\n");
    printf("请输入您要进行的操作（1-6）：\n");
}

/*修改信息菜单函数*/
void MenuReviseTheInformation()
{
    printf("--------------【修改信息】---------------\n\n");
    printf("\t1.修改教师信息\t\n");
    printf("\t2.修改课程信息\t\n");
    printf("\t3.修改教师与课程关联信息信息\t\n");
    printf("\t4.修改学生信息\t\n");
    printf("\t5.修改学生与开课课程关联信息\t\n");
    printf("\t6.退出\t\n");
    printf("\t\t\t\t\t\n");
    printf("----------------------------------------\n\n");
    printf("请输入您要进行的操作 (1-6) : \n");
}

/*2.数据结构的设计，抽象出教学信息*/
/*上方已完成*/

/*3.链表函数的设计，实现增删查改*/
/*创建教师信息链表的函数*/
struct NodeTeacherDat* CreatList_TeacherDat()
{
    /*结构体指针变量表示表头*/
    /*指针到变量，用动态内存申请*/
    struct NodeTeacherDat* head_teacher = (struct NodeTeacherDat*)malloc(sizeof(struct NodeTeacher

    /*表头：做差异化处理，数据域data不做初始化*/
    head_teacher->next_teacher = NULL;

    return head_teacher;
}

/*创建教师信息结点的函数*/
struct NodeTeacherDat* CreateNode_TeacherDat(struct TeacherDat data_teacher)
{
    /*结构体指针变量表示新结点*/
    /*指针到变量，用动态内存申请*/
    struct NodeTeacherDat* newnode_teacher = (struct NodeTeacherDat*)malloc(sizeof(struct NodeTeac

    newnode_teacher->data_teacher = data_teacher;//数据为形参
    newnode_teacher->next_teacher = NULL;

    return newnode_teacher;
}

/*插入教师信息结点的函数*/
void InsertNodeByHead_TeacherDat(struct NodeTeacherDat* head_teacher, struct TeacherDat data_teach
{
    struct NodeTeacherDat* newnode_teacher = CreateNode_TeacherDat(data_teacher);//调用创建结点函数

    newnode_teacher->data_teacher = data_teacher;
    /*表头法插入*/
    newnode_teacher->next_teacher = head_teacher->next_teacher;
    head_teacher->next_teacher = newnode_teacher;
}

/*指定位置教师信息删除的函数*/
void DeleteAppoinNode_TeacherDat(struct NodeTeacherDat* head_teacher, char* teacher_name)//字符串常
{
    struct NodeTeacherDat* posnode_teacher = head_teacher->next_teacher;//指定位置节点为表头下一个节
    struct NodeTeacherDat* posfrontnode_teacher = head_teacher;//指定位置前的节点为表头节点
    int choice_delete_Y_N_teacher = 0;
    int delete_Y_N_teacher = 0;

    if (posnode_teacher == NULL)
    {
        printf("数据为空，无法删除！\n");
        return;
    }

    /*char teacher_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (posnode_teacher && ((delete_Y_N_teacher = strcmp(posnode_teacher->data_teacher.teacher_
    {
        posfrontnode_teacher = posnode_teacher;
        posnode_teacher = posfrontnode_teacher->next_teacher;
        if (posnode_teacher == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法删除！\n");
            return;
        }
    }

    /*找到之后，删除指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("教师号\t    教师姓名\t教师性别\t教师学历\t教师年龄\t教师电话号\n");
    printf("%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n\n", posnode_teacher->data_teacher.teacher_number, p
    printf("是否要删除指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_delete_Y_N_teacher);
    switch (choice_delete_Y_N_teacher)
    {
    case 1:
        posfrontnode_teacher->next_teacher = posnode_teacher->next_teacher;
        free(posnode_teacher);
        printf("已删除！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消删除！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*查找指定位置教师信息的结点的函数*/
struct NodeTeacherDat* SearchInfoByDate_TeacherDat(struct NodeTeacherDat* head_teacher, char* teac
{
    struct NodeTeacherDat* pmove_teacher = head_teacher->next_teacher;//第一个节点没有初始化数据，从
    int strcmp_Y_N_teacher = 0;

    if (pmove_teacher == NULL)
    {
        printf("数据为空，无法查找！\n");
        return NULL;
    }

    /*char teacher_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (pmove_teacher && ((strcmp_Y_N_teacher = strcmp(pmove_teacher->data_teacher.teacher_name
    {
        pmove_teacher = pmove_teacher->next_teacher;
    }
    if (strcmp_Y_N_teacher != 0)
    {
```

```c
        return NULL;
    }
    else
    {
        return pmove_teacher;
    }
}

/*指定位置教师信息修改的函数*/
void ReviseAppoinNode_TeacherDat(struct NodeTeacherDat* head_teacher, char* teacher_name)//字符串常
{
    struct NodeTeacherDat* revisenode_teacher = head_teacher->next_teacher;//指定位置节点为表头下一
    int choice_revise_Y_N_teacher = 0;
    int revise_Y_N_teacher = 0;

    if (revisenode_teacher == NULL)
    {
        printf("数据为空，无法修改！\n");
        return;
    }

    /*char teacher_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (revisenode_teacher && ((revise_Y_N_teacher = strcmp(revisenode_teacher->data_teacher.te
    {
        revisenode_teacher = revisenode_teacher->next_teacher;
        if (revisenode_teacher == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法修改！\n");
            return;
        }
    }

    /*找到之后，修改指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("教师号\t    教师姓名\t教师性别\t教师学历\t教师年龄\t教师电话号\n");
    printf("%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n\n", revisenode_teacher->data_teacher.teacher_number
    printf("是否要修改指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_revise_Y_N_teacher);
    switch (choice_revise_Y_N_teacher)
    {
    case 1:
        printf("请输入要修改的教师号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_teacher->data_teacher.teacher_number);
        printf("请输入要修改的教师姓名：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacher->data_teacher.teacher_name);
        printf("请输入要修改的教师性别：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacher->data_teacher.teacher_sex);
        printf("请输入要修改的教师学历：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacher->data_teacher.teacher_academic_degree);
        printf("请输入要修改的教师年龄：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacher->data_teacher.teacher_age);
        printf("请输入要修改的教师电话号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacher->data_teacher.teacher_phone_number);
        printf("已修改！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消修改！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*打印教师信息链表的函数*/
void PrintList_TeacherDat(struct NodeTeacherDat* head_teacher)
{
    struct NodeTeacherDat* pmove_teacher = head_teacher->next_teacher;//第一个节点没有初始化数据，从

    printf("教师号\t    教师姓名\t教师性别\t教师学历\t教师年龄\t教师电话号\n");
    while (pmove_teacher) //条件为真
    {
        printf("%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n", pmove_teacher->data_teacher.teacher_number, p
        pmove_teacher = pmove_teacher->next_teacher;
    }
    printf("\n\n");
}


/*创建课程信息链表的函数*/
struct NodeCourseDat* CreatList_CourseDat()
{
    /*结构体指针变量表示表头*/
    /*指针到变量，用动态内存申请*/
    struct NodeCourseDat* head_course = (struct NodeCourseDat*)malloc(sizeof(struct NodeCourseDat)

    /*表头：做差异化处理，数据域data不做初始化*/
    head_course->next_course = NULL;

    return head_course;
}

/*创建课程信息结点的函数*/
struct NodeCourseDat* CreateNode_CourseDat(struct CourseDat data_course)
{
    /*结构体指针变量表示新结点*/
    /*指针到变量，用动态内存申请*/
    struct NodeCourseDat* newnode_course = (struct NodeCourseDat*)malloc(sizeof(struct NodeCourseD

    newnode_course->data_course = data_course;//数据为形参
    newnode_course->next_course = NULL;

    return newnode_course;
}

/*插入课程信息结点的函数*/
void InsertNodeByHead_CourseDat(struct NodeCourseDat* head_course, struct CourseDat data_course)
{
    struct NodeCourseDat* newnode_course = CreateNode_CourseDat(data_course);//调用创建结点函数来创

    newnode_course->data_course = data_course;

    /*表头法插入*/
    newnode_course->next_course = head_course->next_course;
    head_course->next_course = newnode_course;
}
```

```c
/*指定位置课程信息删除的函数*/
void DeleteAppoinNode_CourseDat(struct NodeCourseDat* head_course, int course_number)
{
    struct NodeCourseDat* posnode_course = head_course->next_course;//指定位置节点为表头下一个节点
    struct NodeCourseDat* posfrontnode_course = head_course;//指定位置前的节点为表头节点
    int choice_delete_Y_N_course = 0;
    int delete_Y_N_course = 0;

    if (posnode_course == NULL)
    {
        printf("数据为空，无法删除！\n");
        return;
    }

    while (posnode_course && (delete_Y_N_course = (posnode_course->data_course.course_number != co
    {
        posfrontnode_course = posnode_course;
        posnode_course = posfrontnode_course->next_course;
        if (posnode_course == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法删除！\n");
            return;
        }
    }

    /*找到之后，删除指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("课程号\t    课程名称\n");
    printf("%d\t\t%s\n\n", posnode_course->data_course.course_number, posnode_course->data_course.
    printf("是否要删除指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_delete_Y_N_course);
    switch (choice_delete_Y_N_course)
    {
    case 1:
        posfrontnode_course->next_course = posnode_course->next_course;
        free(posnode_course);
        printf("已删除！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消删除！\n\n");
        system("pause");
```

```c
            break;
        default:
            printf("选择错误，请重新输入！\n\n");
            system("pause");
            break;
        }
}

/*查找指定位置课程信息的结点的函数*/
struct NodeCourseDat* SearchInfoByDate_CourseDat(struct NodeCourseDat* head_course, int course_num
{
    struct NodeCourseDat* pmove_course = head_course->next_course;//第一个节点没有初始化数据，从第二
    int compare_Y_N_course = 0;

    if (pmove_course == NULL)
    {
        printf("数据为空，无法查找！\n");
        return NULL;
    }

    while (pmove_course && (compare_Y_N_course = (pmove_course->data_course.course_number != cours
    {
        pmove_course = pmove_course->next_course;
    }
    if (compare_Y_N_course != 0)
    {
        return NULL;
    }
    else
    {
        return pmove_course;
    }
}

/*指定位置课程信息修改的函数*/
void ReviseAppoinNode_CourseDat(struct NodeCourseDat* head_course, int course_number)
{
    struct NodeCourseDat* revisenode_course = head_course->next_course;//指定位置节点为表头下一个节
    int choice_revise_Y_N_course = 0;
    int revise_Y_N_course = 0;

    if (revisenode_course == NULL)
    {
        printf("数据为空，无法修改！\n");
        return;
    }

    while (revisenode_course && (revise_Y_N_course = (revisenode_course->data_course.course_number
    {
        revisenode_course = revisenode_course->next_course;
        if (revisenode_course == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法修改！\n");
            return;
        }
    }

    /*找到之后，修改指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("课程号\t    课程名称\n");
    printf("%d\t\t%s\n\n", revisenode_course->data_course.course_number, revisenode_course->data_c
    printf("是否要修改指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_revise_Y_N_course);
    switch (choice_revise_Y_N_course)
    {
    case 1:
        printf("请输入要修改的课程号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_course->data_course.course_number);
        printf("请输入要修改的课程名称：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_course->data_course.course_name);
        printf("已修改！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消修改！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*打印课程信息链表的函数*/
```

```c
void PrintList_CourseDat(struct NodeCourseDat* head_course)
{
    struct NodeCourseDat* pmove_course = head_course->next_course;//第一个节点没有初始化数据，从第二个

    printf("课程号\t    课程名称\n");
    while (pmove_course) //条件为真
    {
        printf("%d\t\t%s\n", pmove_course->data_course.course_number, pmove_course->data_course.co
        pmove_course = pmove_course->next_course;
    }
    printf("\n\n");
}


/*创建教师与课程信息链表的函数*/
struct NodeTeacherAndCourseDat* CreatList_TeacherAndCourseDat()
{
    /*结构体指针变量表示表头*/
    /*指针到变量，用动态内存申请*/
    struct NodeTeacherAndCourseDat* head_teacherandcourse = (struct NodeTeacherAndCourseDat*)mallo

    /*表头：做差异化处理，数据域data不做初始化*/
    head_teacherandcourse->next_teacherandcourse = NULL;

    return head_teacherandcourse;
}

/*创建教师与课程信息结点的函数*/
struct NodeTeacherAndCourseDat* CreateNode_TeacherAndCourseDat(struct TeacherAndCourseDat data_tea
{
    /*结构体指针变量表示新结点*/
    /*指针到变量，用动态内存申请*/
    struct NodeTeacherAndCourseDat* newnode_teacherandcourse = (struct NodeTeacherAndCourseDat*)ma

    newnode_teacherandcourse->data_teacherandcourse = data_teacherandcourse;//数据为形参
    newnode_teacherandcourse->next_teacherandcourse = NULL;

    return newnode_teacherandcourse;
}

/*插入教师与课程信息结点的函数*/
void InsertNodeByHead_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, s
{
    struct NodeTeacherAndCourseDat* newnode_teacherandcourse = CreateNode_TeacherAndCourseDat(data
    newnode_teacherandcourse->data_teacherandcourse = data_teacherandcourse;

    /*表头法插入*/
    newnode_teacherandcourse->next_teacherandcourse = head_teacherandcourse->next_teacherandcourse
    head_teacherandcourse->next_teacherandcourse = newnode_teacherandcourse;
}

/*指定位置教师与课程信息删除的函数*/
void DeleteAppoinNode_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, i
{
    struct NodeTeacherAndCourseDat* posnode_teacherandcourse = head_teacherandcourse->next_teacher
    struct NodeTeacherAndCourseDat* posfrontnode_teacherandcourse = head_teacherandcourse;//指定位
    int choice_delete_Y_N_teacherandcourse = 0;
    int delete_Y_N_teacherandcourse = 0;

    if (posnode_teacherandcourse == NULL)
    {
        printf("数据为空，无法删除！\n");
        return;
    }

    while (posnode_teacherandcourse && (delete_Y_N_teacherandcourse = (posnode_teacherandcourse->d
    {
        posfrontnode_teacherandcourse = posnode_teacherandcourse;
        posnode_teacherandcourse = posfrontnode_teacherandcourse->next_teacherandcourse;
        if (posnode_teacherandcourse == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法删除！\n");
            return;
        }
    }

    /*找到之后，删除指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("教课号\t    开课时间\t教师号\t课程号\n");
    printf("%d\t\t%s\t\t%d\t\t%d\n\n", posnode_teacherandcourse->data_teacherandcourse.teaching_cl
    printf("是否要删除指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_delete_Y_N_teacherandcourse);
    switch (choice_delete_Y_N_teacherandcourse)
    {
    case 1:
        posfrontnode_teacherandcourse->next_teacherandcourse = posnode_teacherandcourse->next_teac
        free(posnode_teacherandcourse);
        printf("已删除! \n\n");
        system("pause");
        break;
    case 2:
        printf("已取消删除! \n\n");
        system("pause");
        break;
    default:
        printf("选择错误, 请重新输入! \n\n");
        system("pause");
        break;
    }
}

/*查找指定位置教师与课程信息的结点的函数*/
struct NodeTeacherAndCourseDat* SearchInfoByDate_TeacherAndCourseDat(struct NodeTeacherAndCourseDa
{
    struct NodeTeacherAndCourseDat* pmove_teacherandcourse = head_teacherandcourse->next_teacherar
    int compare_Y_N_teacherandcourse = 0;

    if (pmove_teacherandcourse == NULL)
    {
        printf("数据为空, 无法查找! \n");
        return NULL;
    }

    while (pmove_teacherandcourse && (compare_Y_N_teacherandcourse = (pmove_teacherandcourse->data
    {
        pmove_teacherandcourse = pmove_teacherandcourse->next_teacherandcourse;
    }
    if (compare_Y_N_teacherandcourse != 0)
    {
        return NULL;
    }
    else
    {
        return pmove_teacherandcourse;
    }
}

/*指定位置教师与课程信息修改的函数*/
void ReviseAppoinNode_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, i
{
```

```c
    struct NodeTeacherAndCourseDat* revisenode_teacherandcourse = head_teacherandcourse->next_teac
    int choice_revise_Y_N_teacherandcourse = 0;
    int revise_Y_N_teacherandcourse = 0;

    if (revisenode_teacherandcourse == NULL)
    {
        printf("数据为空，无法修改！\n");
        return;
    }

    while (revisenode_teacherandcourse && (revise_Y_N_teacherandcourse = (revisenode_teacherandcou
    {
        revisenode_teacherandcourse = revisenode_teacherandcourse->next_teacherandcourse;
        if (revisenode_teacherandcourse == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法修改！\n");
            return;
        }
    }

    /*找到之后，修改指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("教课号\t    开课时间\t教师号\t课程号\n");
    printf("%d\t\t%s\t\t%d\t\t%d\n\n", revisenode_teacherandcourse->data_teacherandcourse.teaching
    printf("是否要修改指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_revise_Y_N_teacherandcourse);
    switch (choice_revise_Y_N_teacherandcourse)
    {
    case 1:
        printf("请输入要修改的教课号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_teacherandcourse->data_teacherandcourse.teaching_class_number);
        printf("请输入要修改的开课时间：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_teacherandcourse->data_teacherandcourse.class_time);
        printf("请输入要修改的教师号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_teacherandcourse->data_teacherandcourse.teacher_number);
        printf("请输入要修改的课程号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_teacherandcourse->data_teacherandcourse.course_number);
        printf("已修改！\n\n");
        system("pause");
```

```
            break;
        case 2:
            printf("已取消修改！\n\n");
            system("pause");
            break;
        default:
            printf("选择错误，请重新输入！\n\n");
            system("pause");
            break;
        }
}

/*打印教师与课程信息链表的函数*/
void PrintList_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse)
{
    struct NodeTeacherAndCourseDat* pmove_teacherandcourse = head_teacherandcourse->next_teacherar

    printf("教课号\t    开课时间\t教师号\t\t课程号\n");
    while (pmove_teacherandcourse) //条件为真
    {
        printf("%d\t\t%s\t\t%d\t\t%d\n", pmove_teacherandcourse->data_teacherandcourse.teaching_cl
        pmove_teacherandcourse = pmove_teacherandcourse->next_teacherandcourse;
    }
    printf("\n\n");
}


/*创建学生信息链表的函数*/
struct NodeStudentDat* CreatList_StudentDat()
{
    /*结构体指针变量表示表头*/
    /*指针到变量，用动态内存申请*/
    struct NodeStudentDat* head_student = (struct NodeStudentDat*)malloc(sizeof(struct NodeStudent

    /*表头：做差异化处理，数据域data不做初始化*/
    head_student->next_student = NULL;

    return head_student;
}

/*创建学生信息结点的函数*/
struct NodeStudentDat* CreateNode_StudentDat(struct StudentDat data_student)
{
    /*结构体指针变量表示新结点*/
    /*指针到变量，用动态内存申请*/
    struct NodeStudentDat* newnode_student = (struct NodeStudentDat*)malloc(sizeof(struct NodeStuc

    newnode_student->data_student = data_student;//数据为形参
    newnode_student->next_student = NULL;

    return newnode_student;
}

/*插入学生信息结点的函数*/
void InsertNodeByHead_StudentDat(struct NodeStudentDat* head_student, struct StudentDat data_stude
{
    struct NodeStudentDat* newnode_student = CreateNode_StudentDat(data_student);//调用创建结点函数

    newnode_student->data_student = data_student;

    /*表头法插入*/
    newnode_student->next_student = head_student->next_student;
    head_student->next_student = newnode_student;
}

/*指定位置学生信息删除的函数*/
void DeleteAppoinNode_StudentDat(struct NodeStudentDat* head_student, char* student_name)//字符串常
{
    struct NodeStudentDat* posnode_student = head_student->next_student;//指定位置节点为表头下一个节
    struct NodeStudentDat* posfrontnode_student = head_student;//指定位置前的节点为表头节点
    int choice_delete_Y_N_student = 0;
    int delete_Y_N_student = 0;

    if (posnode_student == NULL)
    {
        printf("数据为空，无法删除！\n");
        return;
    }

    /*char student_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (posnode_student && ((delete_Y_N_student = strcmp(posnode_student->data_student.student_
    {
        posfrontnode_student = posnode_student;
        posnode_student = posfrontnode_student->next_student;
        if (posnode_student == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法删除！\n");
            return;
        }
    }

    /*找到之后，删除指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("学号\t    学生姓名\n");
    printf("%d\t\t%s\n\n", posnode_student->data_student.student_name, posnode_student->data_stude
    printf("是否要删除指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_delete_Y_N_student);
    switch (choice_delete_Y_N_student)
    {
    case 1:
        posfrontnode_student->next_student = posnode_student->next_student;
        free(posnode_student);
        printf("已删除！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消删除！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*查找指定位置学生信息的结点的函数*/
struct NodeStudentDat* SearchInfoByDate_StudentDat(struct NodeStudentDat* head_student, char* stud
{
    struct NodeStudentDat* pmove_student = head_student->next_student;//第一个节点没有初始化数据，从
    int strcmp_Y_N_student = 0;

    if (pmove_student == NULL)
    {
        printf("数据为空，无法查找！\n");
        return NULL;
    }

    /*char student_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (pmove_student && ((strcmp_Y_N_student = strcmp(pmove_student->data_student.student_name
    {
        pmove_student = pmove_student->next_student;
    }
    if (strcmp_Y_N_student != 0)
    {
        return NULL;
    }
    else
    {
        return pmove_student;
    }
}

/*指定位置学生信息修改的函数*/
void ReviseAppoinNode_StudentDat(struct NodeStudentDat* head_student, char* student_name)//字符串常
{
    struct NodeStudentDat* revisenode_student = head_student->next_student;//指定位置节点为表头下一个
    int choice_revise_Y_N_student = 0;
    int revise_Y_N_student = 0;

    if (revisenode_student == NULL)
    {
        printf("数据为空，无法修改！\n");
        return;
    }

    /*char student_name[40]是字符串，用strcmp函数比较字符串*/
    /*
    两个字符串自左向右开始相比（按 ASCII 值大小相比较）
    如果 str1 < str2 返回一个小于0的数
    如果 str1 = str2 返回0
    如果 str1 < str2 返回一个大于0的数
    */
    while (revisenode_student && ((revise_Y_N_student = strcmp(revisenode_student->data_student.st
    {
        revisenode_student = revisenode_student->next_student;
        if (revisenode_student == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法修改！\n");
            return;
        }
    }

    /*找到之后，修改指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("学号\t    学生姓名\n");
    printf("%d\t\t%s\n\n", revisenode_student->data_student.student_number, revisenode_student->da
    printf("是否要修改指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_revise_Y_N_student);
    switch (choice_revise_Y_N_student)
    {
    case 1:
        printf("请输入要修改的学号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_student->data_student.student_number);
        printf("请输入要修改的学生姓名：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%s", revisenode_student->data_student.student_name);
        printf("已修改！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消修改！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*打印学生信息链表的函数*/
void PrintList_StudentDat(struct NodeStudentDat* head_student)
{
    struct NodeStudentDat* pmove_student = head_student->next_student;//第一个节点没有初始化数据，从
    printf("学号\t    学生姓名\n");
    while (pmove_student) //条件为真
    {
        printf("%d\t\t%s\n", pmove_student->data_student.student_number, pmove_student->data_stude
        pmove_student = pmove_student->next_student;
    }
    printf("\n\n");
}


/*创建学生与课程信息链表的函数*/
struct NodeStudentAndCourseDat* CreatList_StudentAndCourseDat()
{
    /*结构体指针变量表示表头*/
    /*指针到变量，用动态内存申请*/
    struct NodeStudentAndCourseDat* head_studentandcourse = (struct NodeStudentAndCourseDat*)mallo

    /*表头：做差异化处理，数据域data不做初始化*/
    head_studentandcourse->next_studentandcourse = NULL;

    return head_studentandcourse;
}

/*创建学生与课程信息结点的函数*/
struct NodeStudentAndCourseDat* CreateNode_StudentAndCourseDat(struct StudentAndCourseDat data_stu
{
    /*结构体指针变量表示新结点*/
    /*指针到变量，用动态内存申请*/
    struct NodeStudentAndCourseDat* newnode_studentandcourse = (struct NodeStudentAndCourseDat*)ma

    newnode_studentandcourse->data_studentandcourse = data_studentandcourse;//数据为形参
    newnode_studentandcourse->next_studentandcourse = NULL;

    return newnode_studentandcourse;
}

/*插入学生与课程信息结点的函数*/
void InsertNodeByHead_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, s
{
    struct NodeStudentAndCourseDat* newnode_studentandcourse = CreateNode_StudentAndCourseDat(data

    newnode_studentandcourse->data_studentandcourse = data_studentandcourse;
    /*表头法插入*/
    newnode_studentandcourse->next_studentandcourse = head_studentandcourse->next_studentandcourse
    head_studentandcourse->next_studentandcourse = newnode_studentandcourse;
}

/*指定位置学生与课程信息删除的函数*/
void DeleteAppoinNode_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, i
{
    struct NodeStudentAndCourseDat* posnode_studentandcourse = head_studentandcourse->next_student
    struct NodeStudentAndCourseDat* posfrontnode_studentandcourse = head_studentandcourse;//指定位
    int choice_delete_Y_N_studentandcourse = 0;
    int delete_Y_N_studentandcourse = 0;

    if (posnode_studentandcourse == NULL)
    {
        printf("数据为空，无法删除！\n");
        return;
    }

    while (posnode_studentandcourse && (delete_Y_N_studentandcourse = (posnode_studentandcourse->d
    {
        posfrontnode_studentandcourse = posnode_studentandcourse;
        posnode_studentandcourse = posfrontnode_studentandcourse->next_studentandcourse;
        if (posnode_studentandcourse == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法删除！\n");
            return;
        }
    }

    /*找到之后，删除指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("学课号\t    学号\t\t教课号\n");
    printf("%d\t\t%d\t\t%d\n\n", posnode_studentandcourse->data_studentandcourse.learn_class_numbe
    printf("是否要删除指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_delete_Y_N_studentandcourse);
    switch (choice_delete_Y_N_studentandcourse)
    {
    case 1:
        posfrontnode_studentandcourse->next_studentandcourse = posnode_studentandcourse->next_stud
        free(posnode_studentandcourse);
        printf("已删除！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消删除! \n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入! \n\n");
        system("pause");
        break;
    }
}

/*查找指定位置学生与课程信息的结点的函数*/
struct NodeStudentAndCourseDat* SearchInfoByDate_StudentAndCourseDat(struct NodeStudentAndCourseDa
{
    struct NodeStudentAndCourseDat* pmove_studentandcourse = head_studentandcourse->next_studentar
    int compare_Y_N_studentandcourse = 0;

    if (pmove_studentandcourse == NULL)
    {
        printf("数据为空，无法查找! \n");
        return NULL;
    }

    while (pmove_studentandcourse && (compare_Y_N_studentandcourse = (pmove_studentandcourse->data
    {
        pmove_studentandcourse = pmove_studentandcourse->next_studentandcourse;
    }
    if (compare_Y_N_studentandcourse != 0)
    {
        return NULL;
    }
    else
    {
        return pmove_studentandcourse;
    }
}

/*指定位置学生与课程信息修改的函数*/
void ReviseAppoinNode_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, i
{
    struct NodeStudentAndCourseDat* revisenode_studentandcourse = head_studentandcourse->next_stuc
    int choice_revise_Y_N_studentandcourse = 0;
    int revise_Y_N_studentandcourse = 0;
    if (revisenode_studentandcourse == NULL)
    {
        printf("数据为空，无法修改！\n");
        return;
    }

    while (revisenode_studentandcourse && (revise_Y_N_studentandcourse = (revisenode_studentandcou
    {
        revisenode_studentandcourse = revisenode_studentandcourse->next_studentandcourse;
        if (revisenode_studentandcourse == NULL)
        {
            printf("查找完毕，但未找到指定位置，无法修改！\n");
            return;
        }
    }

    /*找到之后，修改指定位置节点*/
    printf("查找完毕，指定信息为：\n");
    printf("学课号\t    学号\t\t教课号\n");
    printf("%d\t\t%d\t\t%d\n\n", revisenode_studentandcourse->data_studentandcourse.learn_class_nu
    printf("是否要修改指定信息？\n1.是\t2.否\n");
    printf("请输入您要进行的操作（1-2）：\n");
    scanf_s("%d", &choice_revise_Y_N_studentandcourse);
    switch (choice_revise_Y_N_studentandcourse)
    {
    case 1:
        printf("请输入要修改的学课号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_studentandcourse->data_studentandcourse.learn_class_number);
        printf("请输入要修改的学号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_studentandcourse->data_studentandcourse.student_number);
        printf("请输入要修改的教课号：");
        fflush(stdin);//清空缓冲区，防止字符串输入出问题
        scanf("%d", &revisenode_studentandcourse->data_studentandcourse.teaching_class_number);
        printf("已修改！\n\n");
        system("pause");
        break;
    case 2:
        printf("已取消修改！\n\n");
        system("pause");
        break;
    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");
        break;
    }
}

/*打印学生与课程信息链表的函数*/
void PrintList_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse)
{
    struct NodeStudentAndCourseDat* pmove_studentandcourse = head_studentandcourse->next_studentar

    printf("学课号\t    学号\t\t教课号\n");
    while (pmove_studentandcourse) //条件为真
    {
        printf("%d\t\t%d\t\t%d\n", pmove_studentandcourse->data_studentandcourse.learn_class_numbe
        pmove_studentandcourse = pmove_studentandcourse->next_studentandcourse;
    }
    printf("\n\n");
}

/*4.功能交互*/
/*功能交互函数*/
void KeyDown()
{
    /*所有操作都同步到文件*/

    int choice = 0;//序号选择
    int choice_1 = 0;
    int choice_2 = 0;
    int choice_3 = 0;
    int choice_4 = 0;
    int choice_5 = 0;

    struct TeacherDat data_teacher;
    struct NodeTeacherDat* pmove_teacher = NULL;
    struct CourseDat data_course;
    struct NodeCourseDat* pmove_course = NULL;
    struct TeacherAndCourseDat data_teacherandcourse;
    struct NodeTeacherAndCourseDat* pmove_teacherandcourse = NULL;
    struct StudentDat data_student;
    struct NodeStudentDat* pmove_student = NULL;
    struct StudentAndCourseDat data_studentandcourse;
    struct NodeStudentAndCourseDat* pmove_studentandcourse = NULL;
    scanf_s("%d", &choice);
    printf("\n");
    switch (choice)
    {
    case 1://显示信息
        MenuDispiaysTheInformation();
        scanf_s("%d", &choice_1);
        switch (choice_1)
        {
        case 1:
            PrintList_TeacherDat(List_teacher);
            break;
        case 2:
            PrintList_CourseDat(List_course);
            break;
        case 3:
            PrintList_TeacherAndCourseDat(List_teacherandcourse);
            break;
        case 4:
            PrintList_StudentDat(List_student);
            break;
        case 5:
            PrintList_StudentAndCourseDat(List_studentandcourse);
        case 6:
            printf("已退出! \n\n");
            system("pause");
            break;
        default:
            printf("选择错误, 请重新输入! \n\n");
            system("pause");
            break;
        }

        break;
    case 2://录入信息
        MenuEnterTheInformation();
        scanf_s("%d", &choice_2);
        switch (choice_2)
        {
        case 1:
            printf("请输入要录入的教师号: ");    //插入链表
            fflush(stdin);//清空缓冲区, 防止字符串输入出问题
            scanf("%d", &data_teacher.teacher_number);
            printf("请输入要录入的教师姓名: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_name);
            printf("请输入要录入的教师性别: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_sex);
            printf("请输入要录入的教师学历: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_academic_degree);
            printf("请输入要录入的教师年龄: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_age);
            printf("请输入要录入的教师电话号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_phone_number);

            InsertNodeByHead_TeacherDat(List_teacher, data_teacher);
            break;
        case 2:
            printf("请输入要录入的课程号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_course.course_number);
            printf("请输入要录入的课程名称: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_course.course_name);
            InsertNodeByHead_CourseDat(List_course, data_course);
            break;
        case 3:
            printf("请输入要录入的教课号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_teacherandcourse.teaching_class_number);
            printf("请输入要录入的开课时间: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacherandcourse.class_time);
            printf("请输入要录入的教师号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_teacherandcourse.teacher_number);
            printf("请输入要录入的课程号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_teacherandcourse.course_number);
            InsertNodeByHead_TeacherAndCourseDat(List_teacherandcourse, data_teacherandcourse);
            break;
        case 4:
            printf("请输入要录入的学号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_student.student_number);
            printf("请输入要录入的学生姓名: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_student.student_name);
            InsertNodeByHead_StudentDat(List_student, data_student);
            break;
        case 5:
            printf("请输入要录入的学课号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_studentandcourse.learn_class_number);
            printf("请输入要录入的学号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_studentandcourse.student_number);
            printf("请输入要录入的教课号: ");    //插入链表
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_studentandcourse.teaching_class_number);
            InsertNodeByHead_StudentAndCourseDat(List_studentandcourse, data_studentandcourse);
            break;
        case 6:
            printf("已退出! \n\n");
            system("pause");
            break;
        default:
            printf("选择错误，请重新输入! \n\n");
            system("pause");
            break;
        }

        break;
    case 3://删除信息
        MenuDeleteTheInformation();
        scanf_s("%d", &choice_3);
        switch (choice_3)
        {
        case 1:
            printf("请输入要删除的教师姓名: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_name);
            DeleteAppoinNode_TeacherDat(List_teacher, data_teacher.teacher_name);
            break;
        case 2:
            printf("请输入要删除的课程号: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_course.course_number);
```

```
                DeleteAppoinNode_CourseDat(List_course, data_course.course_number);
                break;
            case 3:
                printf("请输入要删除的教课号: ");
                fflush(stdin);//清空缓冲区，防止字符串输入出问题
                scanf("%d", &data_teacherandcourse.teaching_class_number);
                DeleteAppoinNode_TeacherAndCourseDat(List_teacherandcourse, data_teacherandcourse.teac
                break;
            case 4:
                printf("请输入要删除的学生姓名: ");
                fflush(stdin);//清空缓冲区，防止字符串输入出问题
                scanf("%s", data_student.student_name);
                DeleteAppoinNode_StudentDat(List_student, data_student.student_name);
                break;
            case 5:
                printf("请输入要删除的学课号: ");
                fflush(stdin);//清空缓冲区，防止字符串输入出问题
                scanf("%d", &data_studentandcourse.learn_class_number);
                DeleteAppoinNode_StudentAndCourseDat(List_studentandcourse, data_studentandcourse.lear
                break;
            case 6:
                printf("已退出! \n\n");
                system("pause");
                break;
            default:
                printf("选择错误，请重新输入! \n\n");
                system("pause");
                break;
            }

            break;
        case 4://查找信息
            MenuFindTheInformation();
            scanf_s("%d", &choice_4);
            switch (choice_4)
            {
            case 1:
                printf("请输入要查找的教师姓名: ");
                fflush(stdin);//清空缓冲区，防止字符串输入出问题
                scanf("%s", data_teacher.teacher_name);
                pmove_teacher = SearchInfoByDate_TeacherDat(List_teacher, data_teacher.teacher_name);
                if (pmove_teacher == NULL)
                {
                    printf("未找到相关信息! \n");
                    system("pause");
                }
                else
                {
                    printf("教师号\t    教师姓名\t教师性别\t教师学历\t教师年龄\t教师电话号\n");
                    printf("%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n", pmove_teacher->data_teacher.teacher_r
                }
                break;
            case 2:
                printf("请输入要查找的课程号: ");
                fflush(stdin);//清空缓冲区, 防止字符串输入出问题
                scanf("%d", &data_course.course_number);
                pmove_course = SearchInfoByDate_CourseDat(List_course, data_course.course_number);
                if (pmove_course == NULL)
                {
                    printf("未找到相关信息! \n");
                    system("pause");
                }
                else
                {
                    printf("课程号\t    课程名称\n");
                    printf("%d\t\t%s\n", pmove_course->data_course.course_number, pmove_course->data_c
                }
                break;
            case 3:
                printf("请输入要查找的教课号: ");
                fflush(stdin);//清空缓冲区, 防止字符串输入出问题
                scanf("%d", &data_teacherandcourse.teaching_class_number);
                pmove_teacherandcourse = SearchInfoByDate_TeacherAndCourseDat(List_teacherandcourse, d
                if (pmove_teacherandcourse == NULL)
                {
                    printf("未找到相关信息! \n");
                    system("pause");
                }
                else
                {
                    printf("教课号\t    开课时间\t教师号\t课程号\n");
                    printf("%d\t\t%s\t\t%d\t\t%d\n\n", pmove_teacherandcourse->data_teacherandcourse.t
                }
                break;
            case 4:
                printf("请输入要查找的学生姓名: ");
                fflush(stdin);//清空缓冲区, 防止字符串输入出问题
                scanf("%s", data_student.student_name);
```

```
                pmove_student = SearchInfoByDate_StudentDat(List_student, data_student.student_name);
                if (pmove_student == NULL)
                {
                    printf("未找到相关信息！\n");
                    system("pause");
                }
                else
                {
                    printf("学号\t    学生姓名\n");
                    printf("%d\t\t%s\n", pmove_student->data_student.student_number, pmove_student->da
                }
                break;
            case 5:
                printf("请输入要查找的学课号：");
                fflush(stdin);//清空缓冲区，防止字符串输入出问题
                scanf("%d", &data_studentandcourse.learn_class_number);
                pmove_studentandcourse = SearchInfoByDate_StudentAndCourseDat(List_studentandcourse, c
                if (pmove_studentandcourse == NULL)
                {
                    printf("未找到相关信息！\n");
                    system("pause");
                }
                else
                {
                    printf("学课号\t    学号\t\t教课号\n");
                    printf("%d\t\t%d\t\t%d\n\n", pmove_studentandcourse->data_studentandcourse.learn_c
                }
                break;
            case 6:
                printf("已退出！\n\n");
                system("pause");
                break;
            default:
                printf("选择错误，请重新输入！\n\n");
                system("pause");
                break;
            }

            break;
        case 5://修改信息
            MenuReviseTheInformation();
            scanf_s("%d", &choice_5);
            switch (choice_5)
            {
```

```
        case 1:
            printf("请输入要修改的教师姓名: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_teacher.teacher_name);
            ReviseAppoinNode_TeacherDat(List_teacher, data_teacher.teacher_name);
            break;
        case 2:
            printf("请输入要修改的课程号: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_course.course_number);
            ReviseAppoinNode_CourseDat(List_course, data_course.course_number);
            break;
        case 3:
            printf("请输入要修改的教课号: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_teacherandcourse.teaching_class_number);
            ReviseAppoinNode_TeacherAndCourseDat(List_teacherandcourse, data_teacherandcourse.teac
            break;
        case 4:
            printf("请输入要修改的学生姓名: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%s", data_student.student_name);
            ReviseAppoinNode_StudentDat(List_student, data_student.student_name);
            break;
        case 5:
            printf("请输入要修改的学课号: ");
            fflush(stdin);//清空缓冲区，防止字符串输入出问题
            scanf("%d", &data_studentandcourse.learn_class_number);
            ReviseAppoinNode_StudentAndCourseDat(List_studentandcourse, data_studentandcourse.lear
            break;
        case 6:
            printf("已退出！\n\n");
            system("pause");
            break;
        default:
            printf("选择错误，请重新输入！\n\n");
            system("pause");
            break;
        }

        break;
    case 6://退出系统
        printf("已退出！\n\n");
        system("pause");
        exit(0);

        break;

    default:
        printf("选择错误，请重新输入！\n\n");
        system("pause");

        break;
    }

    /*文件写write操作*/
    WriteInfoToFile_TeacherDat(List_teacher, "teacher.dat.txt");
    WriteInfoToFile_CourseDat(List_course, "course.dat.txt");
    WriteInfoToFile_TeacherAndCourseDat(List_teacherandcourse, "teacherandcourse.dat.txt");
    WriteInfoToFile_StudentDat(List_student, "student.dat.txt");
    WriteInfoToFile_StudentAndCourseDat(List_studentandcourse, "studentandcourse.dat.txt");

}

/*5.文件读写*/
/*文件读read操作：链表数据的读取函数*/
void ReadInfoFromFile_TeacherDat(struct NodeTeacherDat* head_teacher, char* filename_teacher)//字符
{
    /*打开文件*/
    FILE* fp_teacher = NULL;//定义文件指针
    struct TeacherDat data_teacher;

    fp_teacher = fopen(filename_teacher, "r");//r表示以读的方式打开文件
    if (fp_teacher == NULL)
    {
        fp_teacher = fopen(filename_teacher, "w+");// w+表示以创建新文件的方式打开文件
    }

    /*读取文件*/
    while (fscanf(fp_teacher, "%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n", &data_teacher.teacher_number,
    {
        InsertNodeByHead_TeacherDat(head_teacher, data_teacher);
    }

    /*关闭文件*/
    fclose(fp_teacher);
}
```

```
void ReadInfoFromFile_CourseDat(struct NodeCourseDat* head_course, char* filename_course)//字符串常
{
    /*打开文件*/
    FILE* fp_course = NULL;//定义文件指针
    struct CourseDat data_course;

    fp_course = fopen(filename_course, "r");//r表示以读的方式打开文件
    if (fp_course == NULL)
    {
        fp_course = fopen(filename_course, "w+");// w+表示以创建新文件的方式打开文件
    }

    /*读取文件*/
    while (fscanf(fp_course, "%d\t\t%s\n", &data_course.course_number, data_course.course_name) !=
    {
        InsertNodeByHead_CourseDat(head_course, data_course);
    }

    /*关闭文件*/
    fclose(fp_course);
}

void ReadInfoFromFile_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, c
{
    /*打开文件*/
    FILE* fp_teacherandcourse = NULL;//定义文件指针
    struct TeacherAndCourseDat data_teacherandcourse;

    fp_teacherandcourse = fopen(filename_teacherandcourse, "r");//r表示以读的方式打开文件
    if (fp_teacherandcourse == NULL)
    {
        fp_teacherandcourse = fopen(filename_teacherandcourse, "w+");// w+表示以创建新文件的方式打开
    }

    /*读取文件*/
    while (fscanf(fp_teacherandcourse, "%d\t\t%s\t\t%d\t\t%d\n", &data_teacherandcourse.teaching_c
    {
        InsertNodeByHead_TeacherAndCourseDat(head_teacherandcourse, data_teacherandcourse);
    }

    /*关闭文件*/
    fclose(fp_teacherandcourse);
}
```

```
void ReadInfoFromFile_StudentDat(struct NodeStudentDat* head_student, char* filename_student)//字
{
    /*打开文件*/
    FILE* fp_student = NULL;//定义文件指针
    struct StudentDat data_student;

    fp_student = fopen(filename_student, "r");//r表示以读的方式打开文件
    if (fp_student == NULL)
    {
        fp_student = fopen(filename_student, "w+");// w+表示以创建新文件的方式打开文件
    }

    /*读取文件*/
    while (fscanf(fp_student, "%d\t\t%s\n", &data_student.student_number, data_student.student_nam
    {
        InsertNodeByHead_StudentDat(head_student, data_student);
    }

    /*关闭文件*/
    fclose(fp_student);
}

void ReadInfoFromFile_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, c
{
    /*打开文件*/
    FILE* fp_studentandcourse = NULL;//定义文件指针
    struct StudentAndCourseDat data_studentandcourse;

    fp_studentandcourse = fopen(filename_studentandcourse, "r");//r表示以读的方式打开文件
    if (fp_studentandcourse == NULL)
    {
        fp_studentandcourse = fopen(filename_studentandcourse, "w+");// w+表示以创建新文件的方式打开
    }

    /*读取文件*/
    while (fscanf(fp_studentandcourse, "%d\t\t%d\t\t%d\n", &data_studentandcourse.learn_class_numb
    {
        InsertNodeByHead_StudentAndCourseDat(head_studentandcourse, data_studentandcourse);
    }

    /*关闭文件*/
    fclose(fp_studentandcourse);
}
```

```
/*文件写write操作: 链表数据的存储函数*/
void WriteInfoToFile_TeacherDat(struct NodeTeacherDat* head_teacher, char* filename_teacher) // 字
{
    /*打开文件*/
    FILE* fp_teacher = NULL;//定义文件指针;
    struct NodeTeacherDat* pmove_teacher = head_teacher->next_teacher;

    fp_teacher = fopen(filename_teacher, "w");//w表示以写的方式打开文件
    if (fp_teacher == NULL)
    {
        printf("文件打开失败! ");
        return;
    }

    /*写入文件*/
    while (pmove_teacher)
    {
        fprintf(fp_teacher, "%d\t\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n", pmove_teacher->data_teacher.teac
        pmove_teacher = pmove_teacher->next_teacher;
    }

    /*关闭文件*/
    fclose(fp_teacher);
}

void WriteInfoToFile_CourseDat(struct NodeCourseDat* head_course, char* filename_course)//字符串常
{
    /*打开文件*/
    FILE* fp_course = NULL;//定义文件指针;
    struct NodeCourseDat* pmove_course = head_course->next_course;

    fp_course = fopen(filename_course, "w");//w表示以写的方式打开文件
    if (fp_course == NULL)
    {
        printf("文件打开失败! ");
        return;
    }

    /*写入文件*/
    while (pmove_course)
    {
        fprintf(fp_course, "%d\t\t%s\n", pmove_course->data_course.course_number, pmove_course->da
        pmove_course = pmove_course->next_course;
    }
    /*关闭文件*/
    fclose(fp_course);
}

void WriteInfoToFile_TeacherAndCourseDat(struct NodeTeacherAndCourseDat* head_teacherandcourse, ch
{
    /*打开文件*/
    FILE* fp_teacherandcourse = NULL;//定义文件指针;
    struct NodeTeacherAndCourseDat* pmove_teacherandcourse = head_teacherandcourse->next_teacherar

    fp_teacherandcourse = fopen(filename_teacherandcourse, "w");//w表示以写的方式打开文件
    if (fp_teacherandcourse == NULL)
    {
        printf("文件打开失败! ");
        return;
    }

    /*写入文件*/
    while (pmove_teacherandcourse)
    {
        fprintf(fp_teacherandcourse, "%d\t\t%s\t\t%d\t\t%d\n", pmove_teacherandcourse->data_teache
        pmove_teacherandcourse = pmove_teacherandcourse->next_teacherandcourse;
    }

    /*关闭文件*/
    fclose(fp_teacherandcourse);
}

void WriteInfoToFile_StudentDat(struct NodeStudentDat* head_student, char* filename_student)//字符
{
    /*打开文件*/
    FILE* fp_student = NULL;//定义文件指针;
    struct NodeStudentDat* pmove_student = head_student->next_student;

    fp_student = fopen(filename_student, "w");//w表示以写的方式打开文件
    if (fp_student == NULL)
    {
        printf("文件打开失败! ");
        return;
    }

    /*写入文件*/
    while (pmove_student)
```

```c
    {
        fprintf(fp_student, "%d\t\t%s\n", pmove_student->data_student.student_number, pmove_studen
        pmove_student = pmove_student->next_student;
    }

    /*关闭文件*/
    fclose(fp_student);
}
void WriteInfoToFile_StudentAndCourseDat(struct NodeStudentAndCourseDat* head_studentandcourse, ch
{
    /*打开文件*/
    FILE* fp_studentandcourse = NULL;//定义文件指针;
    struct NodeStudentAndCourseDat* pmove_studentandcourse = head_studentandcourse->next_studentan

    fp_studentandcourse = fopen(filename_studentandcourse, "w");//w表示以写的方式打开文件
    if (fp_studentandcourse == NULL)
    {
        printf("文件打开失败! ");
        return;
    }

    /*写入文件*/
    while (pmove_studentandcourse)
    {
        fprintf(fp_studentandcourse, "%d\t\t%d\t\t%d\n", pmove_studentandcourse->data_studentandco
        pmove_studentandcourse = pmove_studentandcourse->next_studentandcourse;
    }

    /*关闭文件*/
    fclose(fp_studentandcourse);
}
```

四、交流学习

互联网开源精神需要大家一起互相交流学习，互相支持奉献。欢迎大家与我友好交流。

加我 QQ 好友获取所有项目源码和项目文档，感谢大家的支持!