# 电 子 科 技 大 学

# 实 验 报 告

**学生姓名：Lolipop　　学 号：2018091202000　　指导教师：xxx**

**实验地点：清水河科技实验大楼　　实验时间：**

## 一、实验室名称：学校实验中心软件实验室

## 二、实验项目名称：学生课程成绩查询程序

## 三、实验学时：8 学时

## 四、实验原理：

i. 链表节点的数据结构

链表（Linked list）是一种常见的基础数据结构，是一种线性表，但是并不会按线性的顺序存储数据，而是在每一个节点里存到下一个节点的指针(Pointer)。链表最明显的好处就是，常规数组排列关联项目的方式可能不同于这些数据项目在记忆体或磁盘上顺序，数据的访问往往要在不同的排列顺序中转换。而链表是一种自我指示数据类型，因为它包含指向另一个相同类型的数据的指针（链接）。链表允许插入和移除表上任意位置上的节点，但是不允许随机存取。

当需要存储具有关联的数据项时，链表是一种合乎逻辑的选择。比如，对于学生信息管理系统而言，存储的信息可能包括学生的学号(char，字符串)、学生的姓名(char，字符串) 、学生的性别(char，字符串)以及学生的专业((char，字符串)等。

链表节点的类型应该包括存储元素的数据域，一般用 data 表示，它的类型可以是前期定义的元素结构体，以及包括存储后续节点位置的指针域，此处用 next 表示。最后一个结点的 next 指针域为空。例如下面定义一个学生信息的链表节点：

```
typedef struct students
{
```

```
    char sno[MAX_STUDENTNO_LEN];
    char sname[MAX_NAME_LEN];
    char sex[MAX_SEX_LEN];
    char major[MAX_MAJORNAME_LEN];
    struct students* next;
} student;
```

ii. 链表创建方法

初始化链表时，为了之后操作的方便，首先应定义指针指向链表头结点并分配内存，例如：

```
L = (student*)malloc(sizeof(student));
L->next = NULL;
```

然后定义一个新的指针并分配内存，存储数据，并将头结点的 next 指向该指针。循环上述操作，每次都将上一结点的 next 指向该指针，代码如下：

```
s = (student*)malloc(sizeof(student));
s->data = a[i]
s->next = NULL;
p->next = s;
p = s;
```

iii. 磁盘文件的读写方法

二进制文件（.dat）的读写方法不同于文本文件（.txt），具体操作如下：1、读二进制文件的数据内容时，首先定义文件头类型指针 FILE*，三个步骤分别是：（1）打开文件 fopen 函数；（2）读文件 fread 函数；（3）关闭文件 fclose 函数；例如：

```
if ((fp = fopen(FILE_PLACE_S, "rb")) == NULL) {
    fprintf(stderr, "can't open\n");
    exit(EXIT_FAILURE);
}

while(fread(head, sizeof(student), 1, fp)){
    head = head->next;
```

```
    }

    fclose(fp);
```

2、写二进制文件的数据内容时，首先定义文件头类型指针 FILE*，三个步骤分别是：（1）打开文件 fopen 函数；（2）读文件 fwrite 函数；（3）关闭文件 fclose 函数；例如：

```
    if ((fp = fopen(FILE_PLACE_S, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while(fwrite(head, sizeof(student), 1, fp)){
        head = head->next;
     }

    fclose(fp);
```

iv. 链表的排序方法

定义三个指针* p, * pre, * q。令*pre 指向头结点，*p 指向待排序链表结点，*q 指向待排序链表结点的下一个结点。

首先将*p 指向头结点的下一个的下一个结点（第二个数据结点），然后将头结点的下一个结点（第一个数据结点）的 next 域置空，此操作将原链表分成了两个链表，此时的*p 指向第二个链表的第一个数据结点。

当*p 不指向 NULL 时，将*q 指向*p 的 next，将*p 指向的数据结点中的待排序值与*pre 指向结点的后一个数据结点的值所比较。若满足条件，则*pre 向后移动一位；若不满足条件，则将*p 指向的结构接在*pre 指向的下一个的下一个结点处。将*p 和*q 后移一位，循环上述过程。代码如下：

```
    student* p, * pre, * q;
    p = L->next->next;
    L->next->next = NULL;
    while (p != NULL) {
        q = p->next;
```

```
        pre = L;
        while (pre->next != NULL && strcmp(pre->next->sno,p->sno)<0) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }
```

v. 链表内容查询的方法

读取需查询内容并保存，对于链表的每一个数据结点，将该内容与结点的对应数据进行比较，若相同，则保存该结点，将指针指向下一节点；若不同，则将指针指向下一节点。

vi. 多个链表内容链接查询的方法

定义一个新的链表结构，用于保存多个链表的相应内容。读取需查询内容并保存，对于每个链表的每一个数据结点，将该内容与结点的对应数据进行比较，若相同，则将该结点的内容保存到新的链表中，将指针指向下一节点；若不同，则将指针指向下一节点。

vii. 栈链表结点的数据结构

栈型链表的定义与单链表的定义完全一致，如下：

```
typedef struct sstacks
{
    char sno[MAX_STUDENTNO_LEN];
    char sname[MAX_NAME_LEN];
    char sex[MAX_SEX_LEN];
    char major[MAX_MAJORNAME_LEN];
    struct sstacks *next;
} sstack;
```

viii. 实现进栈操作方法

栈型链表的进栈操作与单链表的赋值方式类似，如下，其中 s 为链表的栈顶指针：

```
n = (sstack *)malloc(sizeof(sstack));
n->data = a[i];
n->next = s->next;
s->next = n;
```

ix. 实现出栈操作方法

首先判断栈是否为空，若为空，则返回。若不为空，则输出栈顶指针指向的内容，然后将栈顶指针指向栈链表的下一个结点，循环上述操作。对于学生信息的栈，实现代码如下：

```
if (s->next == NULL){
    printf("没有学生信息！");
    return;
}

n = s;
printf("\nsno sname sex major\n");
while ((n = n->next) != NULL) {
    printf("%-8s%-12s%-8s%-12s\n", n->sno, n->sname, n->sex, n->major);
    s->next = n->next;
    n = s;
}
```

x. 队列结点的数据结构

对于需排队的链表，建立一个新的结构体，包含队列的队首指针和队尾指针。如下：

```
typedef struct
{
    struct studentgrades* front;
    struct studentgrades* rear;
} sgqlink;
```

xi. 初始化队列的方法

即将队首指针和队尾指针置空，如下：

```
sgqlink *link;
link = (sgqlink *)malloc(sizeof(sgqlink));
link->front = link->rear = NULL;
```

xii. 实现进队列操作方法

对于所创建的新的数据结点，分情况讨论：1）当队首和队尾指针均指向 NULL 时，则将队首和队尾指针指向该结点；2）当队首和队尾指针均不指向 NULL 时，则将队尾指针的 next 指向该结点，然后将队尾指针后移（指向该结点）。最后将队尾结点的 next 置空。代码如下：

```
L = (sgrade*)malloc(sizeof(sgrade));
L->data = a[i];
if (link->front == NULL) {
    link->front = link->rear = L;
}
else {
    link->rear->next = L;
    link->rear = L;
}
link->rear->next = NULL;
```

xiii. 实现出队列操作方法

首先判断队列是否为空，若为空，则返回。若不为空，分情况讨论：1）当队首和队尾指针指向相同结点时，输出该结点的数据，并将两指针置空；2）当队首和队尾指针指向不同结点时，输出队首指针指向结点的数据，然后将队首指针向后移动一位，循环上述操作直至队首指针指向 NULL。对于学生课程考试信息的操作方法如下：

```
if (link->rear == NULL){
    printf("读取信息失败！");
    return;
}

printf("\n 学号 学生姓名 专业 序号 课程名 成绩\n") ;
if (link->front == link->rear){
```

```
        L = link->front;

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",L->sno,L->sname,L->major,L->cno,L->cname,L->score);
        link->front = link->rear = NULL;
        return;
    }

    while((L = link->front) != NULL){
        link->front = link->front->next;

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",L->sno,L->sname,L->major,L->cno,L->cname,L->score);
    }
```

## 五、实验目的：

通过本实验练习，掌握需要包括:磁盘文件的读写方法；掌握链表创建方法，掌握链表节点的数据结构定义、链表的排序方法、两个链表内容链接查询的方法；掌握创建栈、进栈和出栈的实现方法。

## 六、实验内容：

用 C++语言和单链表、栈、队列数据结构实现一个简单的学生信息管理系统，该系统具备学生信息录入、课程信息录入、成绩信息录入、学生信息排序并打印、利用栈逆序输出学生信息序列、利用队列输出学生考试成绩等功能。具体实现内容如下：

a） 系统界面控制
b） 写入学生记录、写入课程信息记录、写入学生考试成绩
c） 学号升序输出学生记录、课程号升序输出课程信息、学号和课程号升序输出学生考试成绩
d） 考试成绩降序输出所有学生考试成绩、考试成绩降序输出所选课程所有学生考试成绩、考试成绩降序输出所选课程不及格学生考试成绩
e） 输出将学号升序的学生记录逆序所生成的新链表
f） 考试成绩降序以链式队列输出所有学生考试成绩

g） 退出系统

## 七、实验器材（设备、元器件）：

PC 机一台，装有 C++语言集成开发环境

## 八、数据结构与程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <iostream>
#include <fstream>

#define MAX_STUDENTNO_LEN 15
#define MAX_NAME_LEN 20
#define MAX_SEX_LEN 10
#define MAX_COURSENO_LEN 10
#define MAX_COURSENAME_LEN 20
#define MAX_MAJORNAME_LEN 20
#define FILE_PLACE_S "student.dat"
#define FILE_PLACE_C "course.dat"
#define FILE_PLACE_G "courseGrade.dat"
#define FILE_PLACE_SG "studentGrade.dat"
#define FILE_PLACE_SGQ "studentGradeQ.dat"

//定义数据结构
typedef struct students
{
    char sno[MAX_STUDENTNO_LEN];
    char sname[MAX_NAME_LEN];
    char sex[MAX_SEX_LEN];
    char major[MAX_MAJORNAME_LEN];
    struct students* next;
} student;

typedef struct courses
{
    char cno[MAX_COURSENO_LEN];
    char cname[MAX_COURSENAME_LEN];
```

```
    int classhours;
    struct courses* next;
} course;

typedef struct grades
{
    char sno[MAX_STUDENTNO_LEN];
    char cno[MAX_COURSENO_LEN];
    int score;
    struct grades* next;
} grade;

typedef struct studentgrades
{
    char sno[MAX_STUDENTNO_LEN];
    char sname[MAX_NAME_LEN];
    char major[MAX_MAJORNAME_LEN];
    char cno[MAX_COURSENO_LEN];
    char cname[MAX_COURSENAME_LEN];
    int score;
    struct studentgrades* next;
} sgrade;

typedef struct sstacks
{
    char sno[MAX_STUDENTNO_LEN];
    char sname[MAX_NAME_LEN];
    char sex[MAX_SEX_LEN];
    char major[MAX_MAJORNAME_LEN];
    struct sstacks *next;
} sstack;

typedef struct
{
    struct studentgrades* front;
    struct studentgrades* rear;
} sgqlink;

//1
void insert_s(student*& L);
void createfile_s(void);
//2
void insert_c(course*& L);
void createfile_c(void);
```

```
//3
void insert_g(grade*& L);
void createfile_g(void);
//base
void readfile_s(student*& L);
void readfile_c(course*& L);
void readfile_g(grade*& L);
void readfile_sg(sgrade*& L);
//4
void print_s(void);
//5
void print_c(void);
//6
void print_g(void);
//7
void print_all(void);
//8
void search_cgrade(void);
//9
void search_cfail(void);
//10
void print_sstack(void);
//11
void print_allqueue(void);
//else
void gettime(void);

//主函数
int main(void) {
    int choice;

    loop:
    printf("\n=============  "); gettime(); printf("  =============");
    printf("\n================================================\n");
    printf("1、写入学生记录\n");
    printf("2、写入课程信息记录\n");
    printf("3、写入学生考试成绩\n");
    printf("4、学号升序输出学生记录\n");
    printf("5、课程号升序输出课程信息\n");
    printf("6、学号和课程号升序输出学生考试成绩\n");
    printf("7、考试成绩降序输出所有学生考试成绩\n");
    printf("8、考试成绩降序输出所选课程所有学生考试成绩\n");
    printf("9、考试成绩降序输出所选课程不及格学生考试成绩\n");
    printf("10、输出将学号升序的学生记录逆序所生成的新链表\n");
```

```
    printf("11、考试成绩降序以链式队列输出所有学生考试成绩\n");
    printf("0、退出系统\n");
    printf("================================================\n");
    printf("请输入序号: ");
    scanf("%d", &choice);
    printf("================================================\n");

    switch (choice) {
    case 1: createfile_s(); goto loop;
    case 2: createfile_c(); goto loop;
    case 3: createfile_g(); goto loop;
    case 4: print_s(); goto loop;
    case 5: print_c(); goto loop;
    case 6: print_g(); goto loop;
    case 7: print_all(); goto loop;
    case 8: search_cgrade(); goto loop;
    case 9: search_cfail(); goto loop;
    case 10: print_sstack(); goto loop;
    case 11: print_allqueue(); goto loop;
    case 0: printf("成功退出! \n"); exit(1);
    }

    exit(0);
}

//else
void gettime(void){
    time_t timep;
    struct tm *p;
    time (&timep);
    p=gmtime(&timep);

    printf("%d-",1900+p->tm_year);/*获取当前年份*/
    printf("%d-",1+p->tm_mon);/*获取当前月份*/
    printf("%d   ",p->tm_mday);/*获取当前月份日数*/
    printf("%d:",8+p->tm_hour); /*获取当前时*/
    printf("%d:",p->tm_min); /*获取当前分*/
    printf("%d",p->tm_sec); /*获取当前秒*/
}

//Function 1
void createfile_s(void) {
    student* L,* head;
    char ch;
```

```
    FILE* fp;
    insert_s(L);

    printf("更改文件?('Y' to save）\n");
    scanf("%s", &ch);
    if (ch != 'Y') {
        free(L);
        printf("成功返回! ");
        return;
    }

    head = L;

    if ((fp = fopen(FILE_PLACE_S, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while(fwrite(head, sizeof(student), 1, fp)){
        head = head->next;
    }

    fclose(fp);
    printf("保存成功! \n");
    return;
}

void insert_s(student*& L) {
    student* p, * s;
    L = (student*)malloc(sizeof(student));
    char ch;
    int num=0;

    p = L;
    loop:
    s = (student*)malloc(sizeof(student));
    printf("请输入学生序号： ");
    scanf("%s", s->sno);
    printf("请输入学生姓名： ");
    scanf("%s", s->sname);
    printf("请输入学生性别： ");
    scanf("%s", s->sex);
    printf("请输入学生专业： ");
    scanf("%s", s->major);
```

```
    s->next = NULL;
    p->next = s;
    p = s;
    num++;

    printf("\n 已输入 %d 个学生的信息\n 输入'Y'继续添加学生信息，输入任意返回
\n", num);
    scanf("%s", &ch);

    while (ch == 'Y') {
        goto loop;
    }

    return;
}

//Function 2
void createfile_c(void) {
    course* L,* head;
    char ch;
    FILE* fp;
    insert_c(L);

    printf("更改文件?('Y' to save）\n");
    scanf("%s", &ch);
    if (ch != 'Y') {
        free(L);
        printf("成功返回! ");
        return;
    }

    head = L;

    if ((fp = fopen(FILE_PLACE_C, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while(fwrite(head, sizeof(course), 1, fp)){
        head = head->next;
    }

    fclose(fp);
    printf("保存成功! \n");
```

```
    return;
}

void insert_c(course*& L) {
    course* p, * s;
    L = (course*)malloc(sizeof(course));
    char ch;
    int num=0;

    p = L;
    loop:
    s = (course*)malloc(sizeof(course));
    printf("请输入课程序号： ");
    scanf("%s", s->cno);
    printf("请输入课程名称： ");
    scanf("%s", s->cname);
    printf("请输入课程学时： ");
    scanf("%d", &s->classhours);
    s->next = NULL;
    p->next = s;
    p = s;
    num++;

    printf("\n 已输入 %d 个课程的信息\n 输入'Y'继续添加课程信息，输入任意返回
\n", num);

    scanf("%s", &ch);
    while (ch == 'Y') {
        goto loop;
    }

    free(s);
    return;
}

//Function 3
void createfile_g(void) {
    grade* L,* head;
    char ch;
    FILE* fp;
    insert_g(L);

    printf("更改文件?('Y' to save）\n");
    scanf("%s", &ch);
```

```
    if (ch != 'Y') {
        free(L);
        printf("成功返回! ");
        return;
    }

    head = L;

    if ((fp = fopen(FILE_PLACE_G, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while(fwrite(head, sizeof(grade), 1, fp)){
        head = head->next;
    }

    fclose(fp);
    printf("保存成功! \n");
    return;
}

void insert_g(grade*& L) {
    grade* p, * s;
    L = (grade*)malloc(sizeof(grade));
    char ch;
    int num=0;

    p = L;
    loop:
    s = (grade*)malloc(sizeof(grade));
    printf("请输入学生序号： ");
    scanf("%s", s->sno);
    printf("请输入课程序号： ");
    scanf("%s", s->cno);
    printf("请输入学生分数： ");
    scanf("%d", &s->score);
    s->next = NULL;
    p->next = s;
    p = s;
    num++;

    printf("\n 已输入 %d 个学生成绩信息\n 输入'Y'继续添加成绩信息，输入任意返回
\n", num);
```

```
    scanf("%s", &ch);
    while (ch == 'Y') {
        goto loop;
    }

    free(s);
    return;
}

//base
void readfile_s(student*& L) {
    FILE* fp;
    student *p,*pNext,*pre;
    L = (student*)malloc(sizeof(student));
    p = L;

    if ((fp = fopen(FILE_PLACE_S, "rb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while (fread(p, sizeof(student), 1, fp)) {
        pNext = (student *)malloc(sizeof(student));
        p->next = pNext;
        pre = p;
        p = pNext;
    }

    pre->next = NULL;
    free(pNext);
    return;
}

void readfile_c(course*& L) {
    FILE* fp;
    course *p,*pNext,*pre;
    L = (course*)malloc(sizeof(course));
    p = L;

    if ((fp = fopen(FILE_PLACE_C, "rb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }
```

```
    while (fread(p, sizeof(course), 1, fp)) {
        pNext = (course *)malloc(sizeof(course));
        p->next = pNext;
        pre = p;
        p = pNext;
    }

    pre->next = NULL;
    free(pNext);
    return;
}

void readfile_g(grade*& L) {
    FILE* fp;
    grade *p,*pNext,*pre;
    L = (grade*)malloc(sizeof(grade));
    p = L;

    if ((fp = fopen(FILE_PLACE_G, "rb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while (fread(p, sizeof(grade), 1, fp)) {
        pNext = (grade *)malloc(sizeof(grade));
        p->next = pNext;
        pre = p;
        p = pNext;
    }

    pre->next = NULL;
    free(pNext);
    return;
}

void readfile_sg(sgrade*& L){
    FILE* fp;
    sgrade *p,*pNext,*pre;
    L = (sgrade*)malloc(sizeof(sgrade));
    p = L;

    if ((fp = fopen(FILE_PLACE_SG, "rb")) == NULL) {
        fprintf(stderr, "can't open\n");
```

```
        exit(EXIT_FAILURE);
    }

    while (fread(p, sizeof(sgrade), 1, fp)) {
        pNext = (sgrade *)malloc(sizeof(sgrade));
        p->next = pNext;
        pre = p;
        p = pNext;
    }

    pre->next = NULL;
    free(pNext);
    return;
}

//Function 4
void print_s(void) {
    student* L;
    readfile_s(L);

    student* p, * pre, * q;
    p = L->next->next;
    L->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = L;
        while (pre->next != NULL && strcmp(pre->next->sno,p->sno)<0) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    p = L;
    printf("\nsno     sname       sex       major\n");
    while ((p = p->next) != NULL) {
        printf("%-8s%-12s%-10s%-10s\n",    p->sno,    p->sname,    p->sex,
p->major);
    }

    return;
}
```

```
//Function 5
void print_c(void) {
    course* L;
    readfile_c(L);

    course* p, * pre, * q;
    p = L->next->next;
    L->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = L;
        while (pre->next != NULL && strcmp(pre->next->cno,p->cno)<0) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    p = L;
    printf("\ncno     cname       classhour\n");
    while ((p = p->next) != NULL) {
        printf("%-8s%-12s%-10d\n", p->cno, p->cname, p->classhours);
    }
    return;
}

//Function 6
void print_g(void) {
    grade* L;
    readfile_g(L);

    grade* p, * pre, * q;

    //双升序排列
    p = L->next->next;
    L->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = L;
        while (pre->next != NULL && strcmp(pre->next->cno,p->cno) <= 0) {
            if(strcmp(pre->next->sno,p->sno)        <        0        ||
strcmp(pre->next->cno,p->cno) < 0 ){
                pre = pre->next;
```

```
            }
            else break;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    //打印信息
    p = L;
    printf("\nsno     cno     score\n");
    while((p = p->next) != NULL){
        printf("%-8s%-8s%-8d\n", p->sno, p->cno, p->score);
    }

    return;
}

//Function 7
void print_all(void){
    student *Ls,*Lsp;
    course *Lc,*Lcp;
    grade *Lg;
    readfile_s(Ls);
    readfile_c(Lc);
    readfile_g(Lg);

    grade* p, * pre, * q;

    //按照成绩降序排列
    p = Lg->next->next;
    Lg->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = Lg;
        while (pre->next != NULL && pre->next->score > p->score) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    sgrade *L,*Lsg,*m,*head;
```

```
L = (sgrade*)malloc(sizeof(sgrade));
Lsg = (sgrade*)malloc(sizeof(sgrade));
m = L;

//遍历学生成绩链表
p = Lg;
while((p = p->next) != NULL){
    Lsp = Ls->next;
    Lcp = Lc->next;

    //寻找对应结点
    while (strcmp(p->sno,Lsp->sno) != 0){
        Lsp = Lsp->next;
        if (Lsp == NULL){
            printf("学生信息不存在! \n");
            return ;
        }
    }
    while (strcmp(p->cno,Lcp->cno) != 0){
        Lcp = Lcp->next;
        if (Lcp == NULL){
            printf("课程信息不存在! \n");
            return ;
        }
    }

    //赋值
    Lsg->score = p->score;
    strcpy(Lsg->sno,p->sno);
    strcpy(Lsg->sname,Lsp->sname);
    strcpy(Lsg->major,Lsp->major);
    strcpy(Lsg->cno,p->cno);
    strcpy(Lsg->cname,Lcp->cname);

    //组合链表
    m->next = Lsg;
    m = Lsg;
    Lsg = (sgrade*)malloc(sizeof(sgrade));
}
m->next = NULL;
free(Lsg);

//打印信息
m = L;
```

```
    printf("\n 学号     学生姓名     专业       序号     课程名       成绩\n") ;
    while((m = m->next) != NULL){

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",m->sno,m->sname,m->major,m->c
no,m->cname,m->score);
    }

    //存储到文件中
    FILE* fp;
    head = L;

    if ((fp = fopen(FILE_PLACE_SG, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    while(fwrite(head, sizeof(sgrade), 1, fp)){
        head = head->next;
    }

    fclose(fp);

    return;
}

//Function 8
void search_cgrade(void){
    sgrade *L,*Lcg,*p,*q,*pre;
    readfile_sg(L);

    char ch[MAX_COURSENO_LEN];
    printf("请输入查找的课程序号：  ");
    scanf("%s",ch);

    //创建链表
    Lcg = (sgrade*)malloc(sizeof(sgrade));
    q = Lcg;
    p = L;
    while ((p = p->next) != NULL){
        if (strcmp(p->cno,ch) == 0){
            q->next = p;
            q = q->next;
        }
    }
```

```
    q->next = NULL;

    //验证链表是否为空
    if (Lcg->next == NULL) {
        printf("未找到该课程数据！\n");
        return;
    }

    //按照成绩降序排列
    p = Lcg->next->next;
    Lcg->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = Lcg;
        while (pre->next != NULL && pre->next->score > p->score) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    //打印信息
    q = Lcg;
    printf("\n 学号     学生姓名     专业       序号     课程名       成绩\n") ;
    while((q = q->next) != NULL){

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",q->sno,q->sname,q->major,q->c
no,q->cname,q->score);
    }

    return;
}

//Function 9
void search_cfail(void) {
    sgrade *L,*Lcg,*p,*q,*pre;
    readfile_sg(L);

    char ch[MAX_COURSENO_LEN];
    printf("请输入查找的课程序号：  ");
    scanf("%s",ch);

    //创建成绩小于 60 分的链表
```

```
    Lcg = (sgrade*)malloc(sizeof(sgrade));
    q = Lcg;
    p = L;
    while ((p = p->next) != NULL){
        if (strcmp(p->cno,ch) == 0 && p->score < 60){
            q->next = p;
            q = q->next;
        }
    }
    q->next = NULL;

    //验证链表是否为空
    if (Lcg->next == NULL) {
        printf("未找到该课程数据或所有同学均及格! \n");
        return;
    }

    //按照成绩降序排列
    p = Lcg->next->next;
    Lcg->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = Lcg;
        while (pre->next != NULL && pre->next->score > p->score) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    //打印信息
    q = Lcg;
    printf("\n 学号     学生姓名     专业       序号     课程名       成绩\n") ;
    while((q = q->next) != NULL){

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",q->sno,q->sname,q->major,q->c
no,q->cname,q->score);
    }

    return;
}

//Function 10
```

```
void print_sstack(void) {
    student* L;
    readfile_s(L);

    //学号升序排列链表
    student* p, * pre, * q;
    p = L->next->next;
    L->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = L;
        while (pre->next != NULL && strcmp(pre->next->sno,p->sno)<0) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    //将链表内容写入栈
    sstack* s,* n;
    s = (sstack *)malloc(sizeof(sstack));
    s->next = NULL;

    p = L;
    while((p = p->next) != NULL){
        n = (sstack *)malloc(sizeof(sstack));

        strcpy(n->sno,p->sno);
        strcpy(n->sname,p->sname);
        strcpy(n->sex,p->sex);
        strcpy(n->major,p->major);

        n->next = s->next;
        s->next = n;
    }

    //出栈并输出逆序信息
    if (s->next == NULL){
        printf("没有学生信息! ");
        return;
    }

    n = s;
```

```
    printf("\nsno     sname       sex     major\n");
    while ((n = n->next) != NULL) {
        printf("%-8s%-12s%-8s%-12s\n",    n->sno,    n->sname,    n->sex,
n->major);
        s->next = n->next;
        n = s;
    }

    return;
}

//Function 11
void print_allqueue(void) {
    student *Ls,*Lsp;
    course *Lc,*Lcp;
    grade *Lg;
    readfile_s(Ls);
    readfile_c(Lc);
    readfile_g(Lg);

    grade* p, * pre, * q;

    //按照成绩降序排列
    p = Lg->next->next;
    Lg->next->next = NULL;
    while (p != NULL) {
        q = p->next;
        pre = Lg;
        while (pre->next != NULL && pre->next->score > p->score) {
            pre = pre->next;
        }
        p->next = pre->next;
        pre->next = p;
        p = q;
    }

    //初始化队列指针
    sgqlink *link;
    link = (sgqlink *)malloc(sizeof(sgqlink));
    link->front = link->rear = NULL;

    //初始化目标链表
    sgrade *L;
    L = (sgrade*)malloc(sizeof(sgrade));
```

```
//遍历学生成绩链表
p = Lg;
while((p = p->next) != NULL){
    Lsp = Ls->next;
    Lcp = Lc->next;

    //寻找对应结点
    while (strcmp(p->sno,Lsp->sno) != 0){
        Lsp = Lsp->next;
        if (Lsp == NULL){
            printf("学生信息不存在! \n");
            return ;
        }
    }
    while (strcmp(p->cno,Lcp->cno) != 0){
        Lcp = Lcp->next;
        if (Lcp == NULL){
            printf("课程信息不存在! \n");
            return ;
        }
    }

    //赋值
    L->score = p->score;
    strcpy(L->sno,p->sno);
    strcpy(L->sname,Lsp->sname);
    strcpy(L->major,Lsp->major);
    strcpy(L->cno,p->cno);
    strcpy(L->cname,Lcp->cname);

    //组合链表
    if (link->front == NULL) {
        link->front = link->rear = L;
    }
    else {
        link->rear->next = L;
        link->rear = L;
    }
    L = (sgrade*)malloc(sizeof(sgrade));
}
link->rear->next = NULL;

//存储到文件中
```

```
    FILE* fp;

    if ((fp = fopen(FILE_PLACE_SGQ, "wb")) == NULL) {
        fprintf(stderr, "can't open\n");
        exit(EXIT_FAILURE);
    }

    L = link->front;

    while(fwrite(L, sizeof(sgrade), 1, fp)){
        L = L->next;
    }

    fclose(fp);

    //打印信息
    if (link->rear == NULL){
        printf("读取信息失败！");
        return;
    }

    printf("\n 学号     学生姓名     专业       序号     课程名       成绩\n") ;
    if (link->front == link->rear){
        L = link->front;

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",L->sno,L->sname,L->major,L->c
no,L->cname,L->score);
        link->front = link->rear = NULL;
        return;
    }

    while((L = link->front) != NULL){
        link->front = link->front->next;

    printf("%-8s%-12s%-10s%-8s%-12s%-8d\n",L->sno,L->sname,L->major,L->c
no,L->cname,L->score);
    }

    return;
}
```

## 九、程序运行结果：

a) **系统界面控制：**实现一个数字选项式的启动界面，其中包含写入学生记录、写入课程信息记录、写入学生考试成绩、学号升序输出学生记录、课程号升序输出课程信息、学号和课程号升序输出学生考试成绩、考试成绩降序输出所有学生考试成绩、考试成绩降序输出所选课程所有学生考试成绩、考试成绩降序输出所选课程不及格学生考试成绩、输出将学号升序的学生记录逆序所生成的新链表、考试成绩降序以链式队列输出所有学生考试成绩、退出系统这 12 个选项。并且这些功能可以循环调用。如图 9-1.

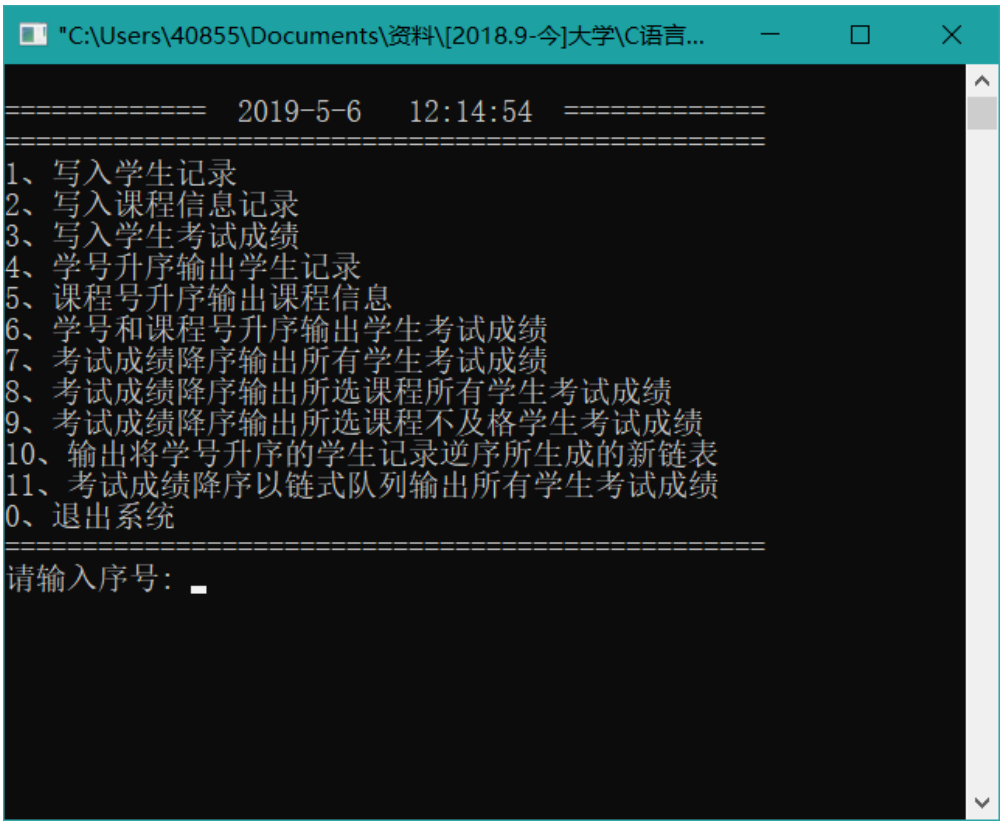


图 9-1

b) **写入学生记录、写入课程信息记录、写入学生考试成绩：**完成信息的录入，根据提示接受用户的输入的各项信息，然后保存到文件之中。上述三个功能的实现大同小异，以图 9-2 为例，完成写入学生的信息。

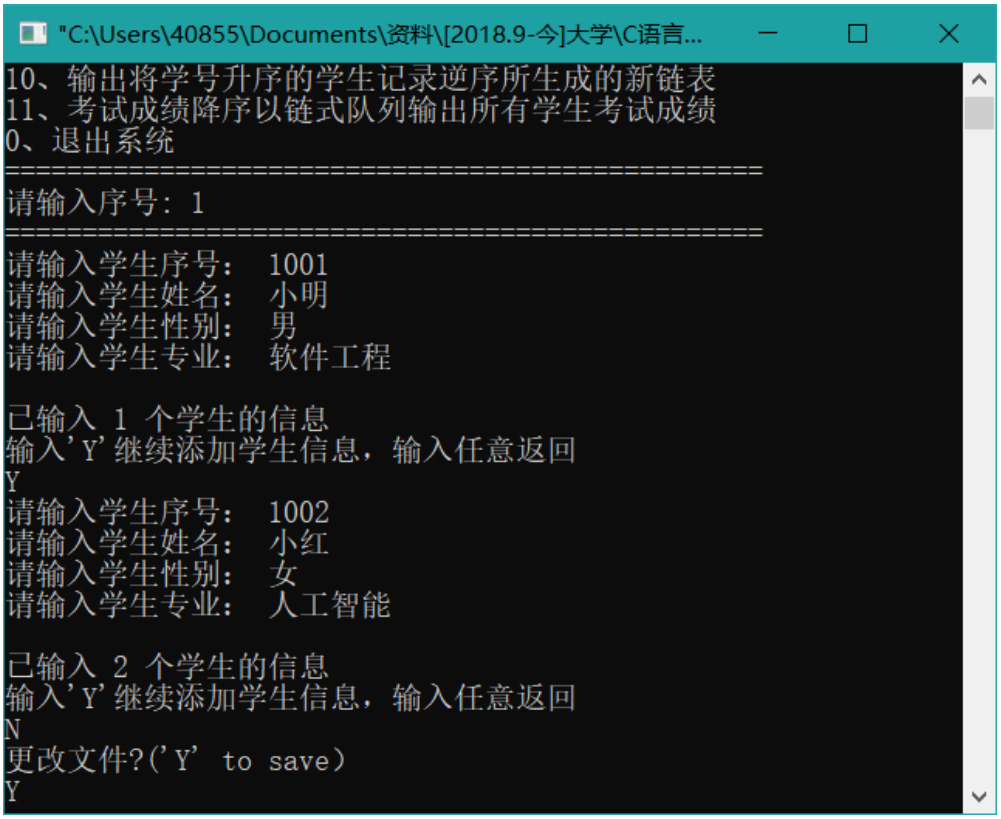


图 9-2

c） **学号升序输出学生记录、课程号升序输出课程信息、学号和课程号升序输出学生考试成绩：**从文件中分别读取并创建链表，依据要求对链表进行排序并输出，三个功能的排列的算法思路相同。对于学号和课程号的双升序排序的实现使用了双重判断，并将其保存到文件中。以图 9-3 为例，完成学号和课程号升序输出学生考试成绩。

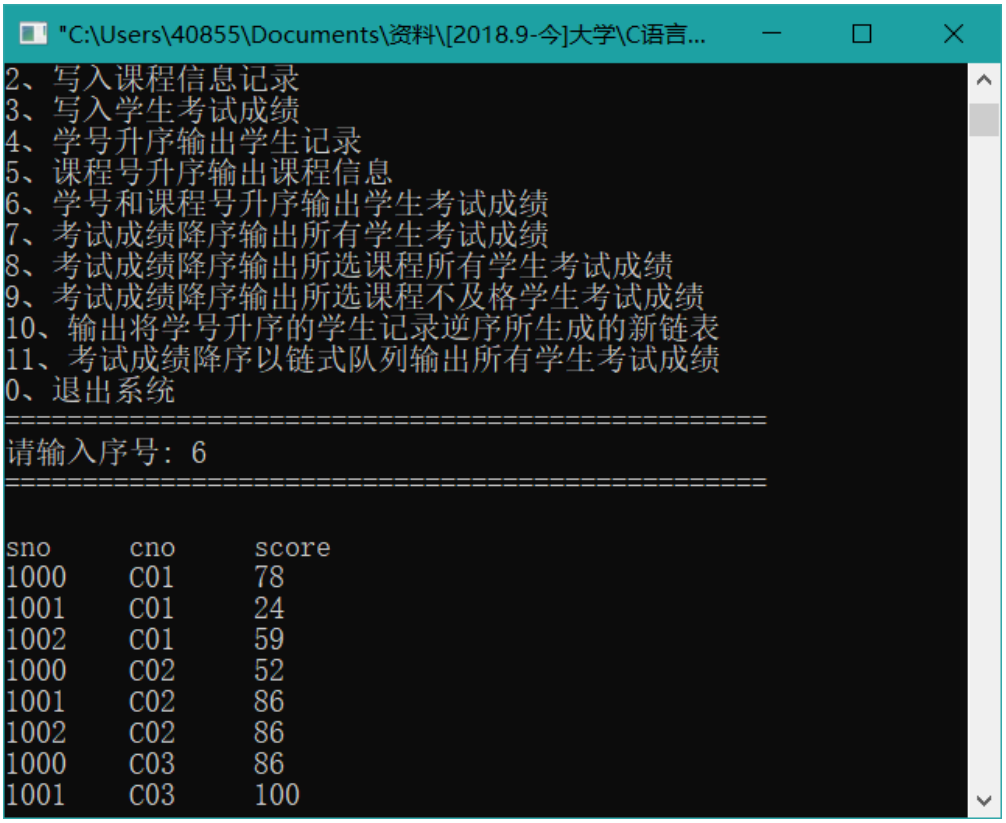


图 9-3

d) **考试成绩降序输出所有学生考试成绩、考试成绩降序输出所选课程所有学生考试成绩、考试成绩降序输出所选课程不及格学生考试成绩：**从文件中分别读取并创建链表，将相关联的数据赋值到新的数据结点并创建新的链表，依据要求对链表排序并输出。三个功能的实现思路相同，以图 9-4 为例，完成考试成绩降序输出所选课程所有学生考试成绩.

```
// removed
```

图 9-4

e) **输出将学号升序的学生记录逆序所生成的新链表：**利用栈链表实现逆序生成功能“学号升序输出学生记录”生成的链表并打印，如图 9-5 所示：

```
// removed
```

图 9-5

f) **考试成绩降序以链式队列输出所有学生考试成绩：**利用队列实现功能“学号和课程号升序输出学生考试成绩”并将其保存到文件中，如图 9-6 所

示：

// removed

图 9-6

g） **退出系统：**字面意思，如图 9-7 所示：

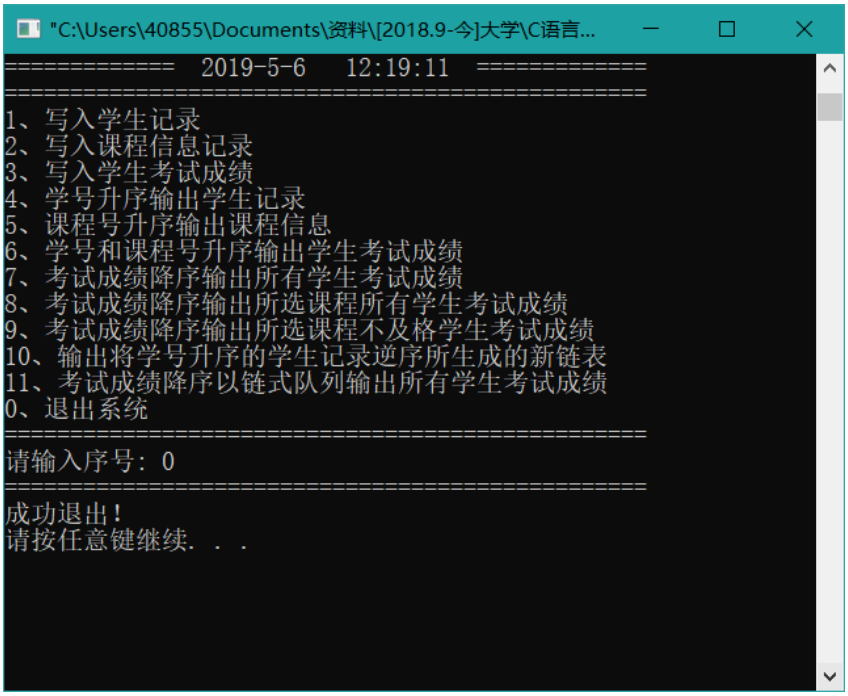


图 9-7