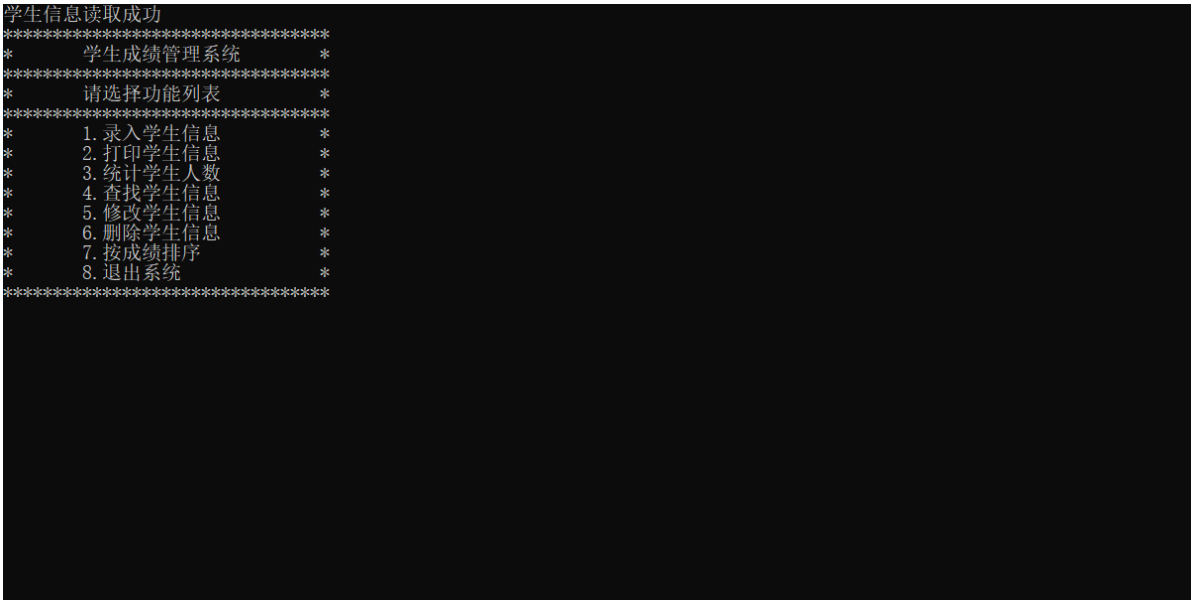
学生管理系统

第1节 实现效果

利用单链表实现学生管理系统，具体功能包括

1. 录入学生信息
2. 打印学生信息
3. 统计学生人数
4. 查找学生信息
5. 修改学生信息
6. 删除学生信息
7. 按成绩排序

同时将学生信息保存至文件中，当再次启动程序时，自动读取学生数据。



第2节. 实现过程

1. 头文件

创建`StudentManager.h`，加入以下代码

```
// 标准输入输出
#include <stdio.h>
// _getch:不需要回车，直接获取输入的字符，需要包含conio.h头文件
#include <conio.h>
// 动态申请内存
#include <stdlib.h>
```

2. 封装学生和结点数据

定义学生结构体，用于保存学生的学号、姓名、成绩信息，并使用typedef给`struct _Student`类型起别名为`Student`，方便使用。

```
typedef struct _Student
{
    int stuNum; //学号
    char name[20];//姓名
    int score; //成绩
}Student;
```

定义结点结构体，用于保存链表中的结点数据，结点需要保存学生信息及下一个结点的地址，并使用typedef给`struct _Node`类型起别名为`Node`，方便使用。

```
typedef struct _Node
{
    Student stu; //学生
    struct _Node* next; //指向下一个结点的指针
}Node;
```

3. 创建头结点

创建`StudentManager.c`，加入main函数，并创建链表的头结点，定义head头指针指向头结点

```
//导入头文件
#include "StudentManager.h"

int main() {

    //创建头结点
    Node* head = malloc(sizeof(Node));
    head->next = NULL;

    return 0;
}
```

4. 系统功能提示

定义welcome函数，用于提示用户系统功能

```
void welcome() {
    printf("**********************************\n");
    printf("*\t学生成绩管理系统\t*\n");
    printf("**********************************\n");
    printf("*\t请选择功能列表\t\t*\n");
    printf("**********************************\n");
    printf("*\t1.录入学生信息\t\t*\n");
    printf("*\t2.打印学生信息\t\t*\n");
    printf("*\t3.统计学生人数\t\t*\n");
    printf("*\t4.查找学生信息\t\t*\n");
    printf("*\t5.修改学生信息\t\t*\n");
```

```
    printf("*\t6.删除学生信息\t\t*\n");
    printf("*\t7.按成绩排序\t\t*\n");
    printf("*\t8.退出系统\t\t*\n");
    printf("**********************************\n");
}
```

在头文件中声明函数，并在main函数中调用，后面定义的其他函数，也应该先在头文件中声明，再调用：

StudentManager.h

```
//欢迎信息
void welcome();
```

main函数

```
//创建头结点
Node* head = malloc(sizeof(Node));
head->next = NULL;

welcome();
```

5. 根据用户输入选择功能

通过`_getch()`函数获取用户输入的单个字符，此函数不需要回车，并通过switch进行判断

```
welcome();
char c = _getch();
switch (c)
{
    case '1':  //录入学生信息
        break;
    case '2':  //打印学生信息
        break;
    case '3':  //统计学生人数
        break;
    case '4':  //查找学生信息
        break;
    case '5':  //修改学生信息
        break;
    case '6':  //删除学生信息
        break;
    case '7':  //按成绩排序
        break;
    case '8':  //退出系统
        break;
    default:
        break;
}
```

6. 录入学生信息

定义`inputStudent`函数，实现新增学生信息的功能，在main函数中调用。

```
void inputStudent(Node* head) {
    //定义指针指向头结点，用于遍历链表
    Node* move = head;
    while (move->next != NULL) {
        move = move->next;
    }
    //创建结点
    Node* fresh = malloc(sizeof(Node));
    fresh->next = NULL;
    //输入用户信息
    printf("请输入学生的学号、姓名、成绩: ");
    scanf("%d%s%d", &fresh->stu.stuNum, fresh->stu.name, &fresh->stu.score);
    //将新创建的结点，添加到链表的尾部
    move->next = fresh;
}
```

在main函数中调用

```
...
case '1':  //录入学生信息
        inputStudent(head);
        break;
...
```

7. 实现循环录入

添加循环，将`welcome`函数及switch判断都加入到循环体，保持程序一直运行，实现循环录入学生信息功能

```
int main() {

    //创建头结点
    Node* head = malloc(sizeof(Node));
    head->next = NULL;

    while (1) {
        welcome();
        /*
        等待从键盘接收一个字符，输入字符后，回车才能录入，注意：回车符号会被存储至缓存区，下次循环导致获取到回车
        //char c = getchar();
        //getchar(); 吸收回车
        getch:不需要回车，直接获取输入的字符，需要包含conio.h头文件，2022之前的版本需要使用_getch()
        */
        char c = _getch();
        switch (c)
        {
```

```
            case '1':  //录入学生信息
                inputStudent(head);
                break;
            case '2':  //打印学生信息
                break;
            case '3':  //统计学生人数
                break;
            case '4':  //查找学生信息
                break;
            case '5':  //修改学生信息
                break;
            case '6':  //删除学生信息
                break;
            case '7':  //按成绩排序
                break;
            case '8':  //退出系统
                break;
            default:
                break;
        }
    }
    return 0;
}
```

注意，如果使用的是`getchar`函数或`scanf`函数，会存在缓冲区的问题，每次输入数据后，回车都会被保存到缓冲区，导致下次出现问题，需要再使用一次getchar吸收回车。

`_getch`函数不需要回车就可以直接读取到数据，不存在此问题。

8. 程序暂停和清空控制台

每次录入学生信息后，进入到下一次循环，会重新打印欢迎信息，可以先使程序暂停，等待用户下一步指示

```
system("pause");//按下任意键后，会继续执行程序
```

当用户按下任意键后，可以先清空控制台，防止重复出现欢迎信息

```
system("cls");
```

两者合用，加入到`inputStudent`函数中

```
void inputStudent(Node* head) {
    //定义指针指向头结点，用于遍历链表
    Node* move = head;
    ...
    ...
    system("pause");
    system("cls");
}
```

9. 打印学生信息

定义`printStudent`函数，实现打印所有学生信息的功能

```
void printStudent(Node* head) {
    Node* move = head->next;
    while (move != NULL) {
        printf("学号:%d 姓名:%s 成绩:%d\n",move->stu.stuNum,move->stu.name,move-
>stu.score);
        move = move->next;
    }
    system("pause");
    system("cls");
}
```

在main函数中调用

```
...
case '2':  //打印学生信息
            printStudent(head);
            break;
...
```

10. 统计学生人数

定义`countStudent`函数，统计学生总人数

```
void countStudent(Node* head) {
    int count = 0;
    Node* move = head->next;
    while (move != NULL) {
        count++;
        move = move->next;
    }
    printf("学生总人数为:%d\n",count);
    system("pause");
    //清屏
    system("cls");
}
```

在main函数中调用

```
case '3':  //统计学生人数
            countStudent(head);
            break;
```

11. 查找学生信息

定义`findStudent`函数，用于根据学号查找学生信息

```
void findStudent(Node* head) {
    int stuNum;
```

```
    printf("请输入要查找的学生的学号: ");
    scanf("%d", &stuNum);

    Node* move = head->next;
    while (move != NULL) {
        if (move->stu.stuNum = stuNum) {
            printf("学号:%d 姓名:%s 成绩:%d\n", move->stu.stuNum, move->stu.name,
move->stu.score);
            system("pause");
            system("cls");
            return;
        }
        move = move->next;
    }
    printf("未查找到学生信息\n");
    system("pause");
    system("cls");
}
```

在main函数中调用

```
case '4':  //查找学生信息
    findStudent(head);
```

12. 学生信息持久化

定义`saveStudent`函数，将链表数据保存至文件中

```
void saveStudent(Node* head) {
    //打开文件
    FILE* file = fopen("./stu.info", "w");
    if (file == NULL) {
        printf("打开文件失败\n");
        return;
    }
    Node* move = head->next;
    while (move != NULL) {
        //将结构体写入文件
        if (fwrite(&move->stu, sizeof(Student), 1, file) != 1) {
            printf("保存%s出现错误\n", move->stu.name);
        }
        move = move->next;
    }
    //关闭文件
    fclose(file);
}
```

在`inputStudent`函数中调用`saveStudent`，实现信息的保存

```
void inputStudent(Node* head) {
    //定义指针指向头结点，用于遍历链表
    Node* move = head;
    ...
    //将学生信息保存至文件
    saveStudent(head);
    system("pause");
    system("cls");
}
```

定义`loadStudent`函数实现学生信息的读取

```
void loadStudent(Node* head) {
    //打开文件
    FILE* file = fopen("./stu.info", "r");
    if (!file) {
        printf("未找到学生文件,跳过读取\n");
        return;
    }
    //创建一个结点
    Node* fresh = malloc(sizeof(Node));
    fresh->next = NULL;
    Node* move = head;
    while (fread(&fresh->stu, sizeof(Student), 1, file) == 1) {
        move->next = fresh;
        move = fresh;
        fresh = malloc(sizeof(Node));
        fresh->next = NULL;
    }
    free(fresh);  //最后多定义一个fresh，要将它释放掉
    //关闭文件
    fclose(file);
    printf("读取成功\n");
}
```

在main函数中调用`loadStudent`，每次程序启动时，先读取之前存储的学生信息

```
int main() {

    //创建头结点
    Node* head = malloc(sizeof(Node));
    head->next = NULL;
    //读取学生信息
    loadStudent(head);
    ...
    while (1) {
```

13. 修改学生信息

定义`modifyStudent`函数，用于根据学生学号修改学生信息

```
void modifyStudent(Node* head) {
    printf("请输入需要修改的学生学号 ");
```

```
    int stuNum = 0;
    scanf("%d", &stuNum);
    Node* move = head;
    while (move != NULL) {
        if (move->stu.stuNum == stuNum) {
            printf("请输入学生姓名，成绩\n");
            scanf("%s%d",move->stu.name,&move->stu.score);
            printf("修改学生信息成功\n");
            //不再循环
            break;
        }
        move = move->next;
    }
    //如果循环完毕，也没找到学生
    if (move == NULL) {
        printf("未找到学生信息\n");
    }
    //同步到文件
    saveStudent(head);
    system("pause");
    system("cls");
}
```

在main函数中调用

```
case '5':  //修改学生信息
    modifyStudent(head);
    break;
```

13. 删除学生信息

定义`deleteStudent`函数，用于根据学号删除学生

```
void deleteStudent(Node* head) {
    printf("请输入要删除的学生学号 ");
    int stuNum = 0;
    scanf("%d", &stuNum);

    Node* move = head;
    while (move->next != NULL) {
        if (stuNum == move->next->stu.stuNum) {
            Node* temp = move->next;
            move->next = move->next->next;  //删除结点只需要一句
            free(temp);       //最后记得将删除的动态空间释放掉
            temp = NULL;  //释放后随即指向NULL
            //同步到文件
            saveStudent(head);
            printf("删除学生成功\n");
            break;
        }
        move = move->next;
    }
    if (move->next == NULL) {
        printf("未找到学生信息\n");
```

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

14. 按成绩排序

定义`sortStudent`函数，使用冒泡排序，按成绩从小到大进行排序

```
void sortStudent(Node* head) {
    Node* save = NULL;
    Node* move = NULL;
    for (Node* turn = head->next; turn->next != NULL; turn = turn->next) {
        for (move = head->next; move->next != save; move = move->next) {
            if (move->stu.score > move->next->stu.score) {
                Student temp = move->stu;
                move->stu = move->next->stu;
                move->next->stu = temp;
            }
        }
        save = move;
    }
    printStudent(head);
}
```

在main函数中调用

```
case '7':  //按成绩排序
    sortStudent(head);
```

15. 退出系统

```
case '8':  //退出系统
    system("cls");
    printf("Bye Bye!\n");
    exit(0);
```