```c
#include <stdio.h>
#include <stdlib.h> // malloc
#include <string.h> // strcpy

typedef enum front_color_e
{
    BLACK = 30, // 黑色
    RED,        // 红色
    GREEN,      // 绿色
    YELLOW,     // 黄色
    BLUE,       // 蓝色
    PURPLE,     // 紫色
    CYAN,       // 青色
    WHITE,      // 白色
} Front_Color;  // 前景色

typedef enum back_color_e
{
    BLACK_B = 40, // 黑色
    RED_B,        // 红色
    GREEN_B,      // 绿色
    YELLOW_B,     // 黄色
    BLUE_B,       // 蓝色
    PURPLE_B,     // 紫色
    CYAN_B,       // 青色
    WHITE_B,      // 白色
} Back_Color;     // 背景色

typedef enum attr_e
{
    BOLD = 1,       // 加粗
    UNDERLINE = 4, // 下划线
    BLINK,          // 闪烁
    REVERSE = 7,    // 反显
    HIDE,           // 隐藏
} Attr;             // 文本属性

// 定义学生信息结构体,链表的节点
typedef struct stu_s
{
    int sid;
    char name[32];
    int age;

    struct stu_s *next;
} STU;

// 添加学生
STU *insert_stu(STU *head, STU *item)
{
    // 如果头指针为空,说明链表为空,直接将新建的节点作为头指针
    if (head == NULL)
    {
        head = item;
```

```
    }
    else
    {
        // 如果头指针不为空,说明链表不为空,需要找到链表的尾部,将新建的节点插入到尾部
        STU *p = head;           // 创建一个临时指针,用来保存头指针,防止头指针丢失
        while (p->next != NULL) // 当 p 的下一个节点不为空时,说明 p 不是尾部
        {
            p = p->next; // p 一直向后移动,直到找到尾部
        }
        p->next = item; // 将 item 插入到尾部
    }
    return head; // 返回头指针
}

// 删除学生
STU *delete_stu(STU *head, char sid)
{
    if (NULL == head)
    {
        printf("\033[%dm无数据,无法删除\033[0m\n", RED);
        return NULL;
    }
    if (head->sid == sid)
    {
        // *tp 为临时指针,用来保存头指针
        STU *tp = head;    // 保存头指针
        head = head->next; // 头指针指向下一个节点
        free(tp);          // 释放头指针
    }
    else
    {
        STU *tp = head;                                    // 保存头指针
        while (NULL != tp->next && sid != tp->next->sid) // 当 tp 的下一个节点不为空
时,且下一个节点的学号不等于要删除的学号时,继续向后移动指针 tp
        {
            tp = tp->next; // tp 一直向后移动,直到找到尾部
        }
        if (NULL == tp->next) // 到达了链表尾部还是没有找到要删除的内容
        {
            printf("\033[%dm未找到 sid = %d 的学生,删除失败\033[0m\n", RED, sid);
        }
        else
        {
            STU *tpx = tp->next;  // 保存要删除的节点
            tp->next = tpx->next; // 将链表位置指向要删除的节点的下一个节点
            free(tpx);            // 释放要删除的节点
            printf("\033[%dm删除 sid = %d 的学生成功\033[0m\n", BLUE, sid);
        }
    }
    return head; // 返回头指针
}

// 打印学生
void shows(STU *head)
{
    // 如果头指针为空,说明链表为空,直接返回
```

```
    if (head == NULL)
    {
        printf("\033[%dm无数据,无法打印\033[0m\n", RED);
        return;
    }
    else
    {
        // 如果头指针不为空,说明链表不为空,需要遍历链表,打印每一个节点的信息
        STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
        // 清屏
        printf("\033[2J");
        printf("--------------------\n");
        printf("\033[%dm学号\t姓名\t年龄\033[0m\n", BLUE_B);
        while (p != NULL) // 当 p 不为空时,说明 p 不是尾部,则打印 p 的信息
        {
            printf("\033[%dm%d\t%s\t%d\033[0m\n", BLUE, p->sid, p->name, p->age);
// 打印 p 的信息
            p = p->next;
 // p 一直向后移动,直到找到尾部
        }
    }
}

// 查询学生
STU *find_stu(STU *head, char sid)
{
    if (NULL == head)
    {
        printf("\033[%dm无数据,无法查询\033[0m\n", RED);
        return NULL; // 返回空指针,说明没有找到要查询的节点
    }

    STU *tp = head;                             // 保存头指针
    while (NULL != tp->next && sid != tp->sid) // 当 tp 的下一个节点不为空时,且下一个
节点的学号不等于要查询的学号时,继续向后移动指针 tp
    {
        tp = tp->next; // tp 一直向后移动,直到找到尾部
    }
    if (sid == tp->sid)
        return tp; // 返回当前节点的指针,即为要查询的节点
    return NULL;   // 返回空指针,说明没有找到要查询的节点
}

// 修改学生
STU *update_stu(STU *head, char sid)
{
    if (NULL == head)
    {
        printf("\033[%dm无数据,无法修改\033[0m\n", RED);
        return NULL;
    }

    STU *tp = find_stu(head, sid); // 查找要修改的节点
    if (tp == NULL)
    {
        printf("\033[%dm未找到 sid = %d 的学生\033[0m\n", RED, sid);
```

```
    }
    else
    {
        // 临时变量,用来保存要修改的学生信息
        char msid;
        char mname[32];
        int mage;

        // 输入要修改的内容
        printf("输入要修改的学生的 学号 姓名 年龄: ");
        scanf("%hhd %s %d", &msid, mname, &mage);
        tp->sid = msid;
        strcpy(tp->name, mname);
        tp->age = mage;
        // 修改成功
        printf("\033[%dm修改 sid = %d 的学生成功\033[0m\n", BLUE, sid);
    }
    return head; // 返回头指针
}

// 排序学生
STU *sort_stu(STU *head)
{
    if (NULL == head)
    {
        printf("\033[%dm无数据,无法排序\033[0m\n", RED);
        return NULL;
    }
    // 选择排序方式: 1) 按照sid 2)按照age
    int choose; // 选择排序方式
    printf("选择排序方式: 1) 按照sid 2)按照age: ");
    scanf("%d", &choose);
    if (1 == choose) // 按照sid排序
    {
        // 选择排序方式: 1) 升序 2)降序
        int choose2; // 选择排序方式
        printf("选择排序方式: 1) 升序 2)降序: ");
        scanf("%d", &choose2);
        if (1 == choose2) // 升序
        {
            // 按照sid升序排序
            STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
            while (p != NULL)
            {
                STU *q = p->next; // 创建一个临时指针,用来保存头指针,防止头指针丢失
                while (q != NULL)
                {
                    if (p->sid > q->sid)
                    {
                        // 交换 p 和 q 的数据
                        STU *tp = (STU *)malloc(sizeof(STU));
                        tp->sid = p->sid;
                        strcpy(tp->name, p->name);
                        tp->age = p->age;

                        p->sid = q->sid;
```

```
                    strcpy(p->name, q->name);
                    p->age = q->age;

                    q->sid = tp->sid;
                    strcpy(q->name, tp->name);
                    q->age = tp->age;
                }
                q = q->next;
            }
            p = p->next;
        }
    }
    else if (2 == choose2) // 降序
    {
        STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
        while (p != NULL)
        {
            STU *q = p->next; // 创建一个临时指针,用来保存头指针,防止头指针丢失
            while (q != NULL)
            {
                if (p->sid < q->sid)
                {
                    STU *tp = (STU *)malloc(sizeof(STU));
                    tp->sid = p->sid;
                    strcpy(tp->name, p->name);
                    tp->age = p->age;

                    p->sid = q->sid;
                    strcpy(p->name, q->name);
                    p->age = q->age;

                    q->sid = tp->sid;
                    strcpy(q->name, tp->name);
                    q->age = tp->age;
                }
                q = q->next;
            }
            p = p->next;
        }
    }
    else
    {
        // 选择错误
        printf("\033[%dm选择错误\033[0m\n", RED);
    }
}
else if (2 == choose) // 按照age排序
{
    // 选择排序方式: 1) 升序 2)降序
    int choose2; // 选择排序方式
    printf("选择排序方式: 1) 升序 2)降序: ");
    scanf("%d", &choose2);
    if (1 == choose2) // 升序
    {
        STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
        while (NULL != p)
```

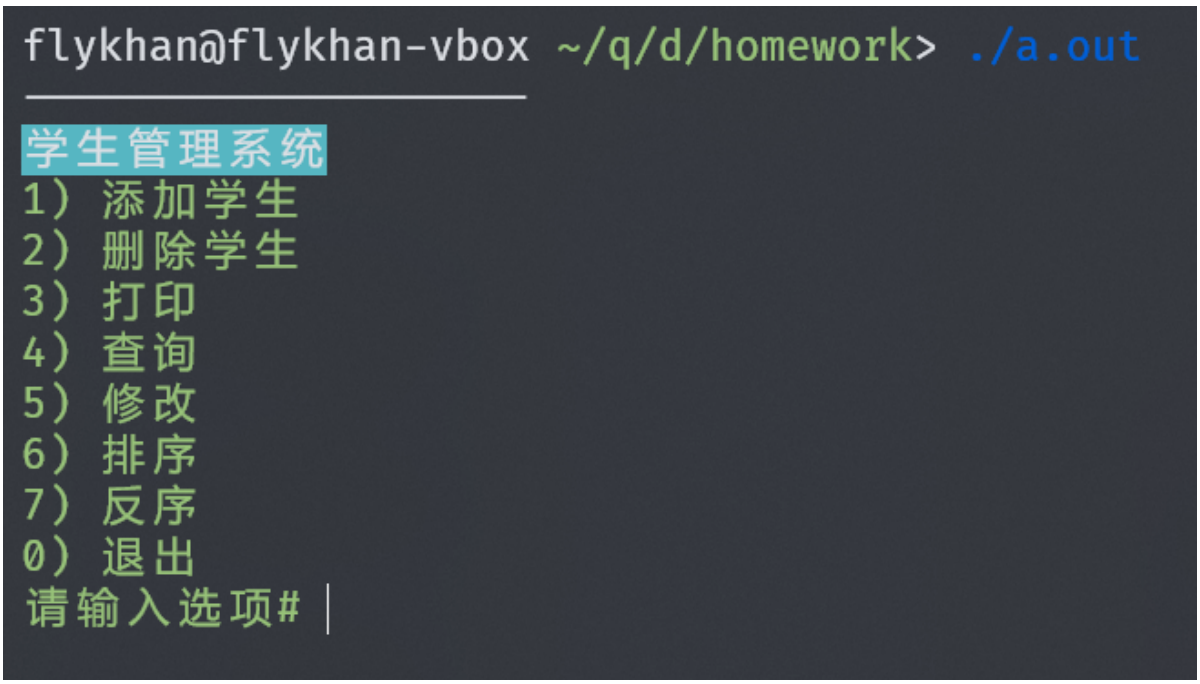
```
            {
                STU *q = p->next; // 创建一个临时指针,用来保存头指针,防止头指针丢失
                while (NULL != q)
                {
                    if (p->age > q->age)
                    {
                        STU *tp = (STU *)malloc(sizeof(STU));
                        tp->sid = p->sid;
                        strcpy(tp->name, p->name);
                        tp->age = p->age;

                        p->sid = q->sid;
                        strcpy(p->name, q->name);
                        p->age = q->age;

                        q->sid = tp->sid;
                        strcpy(q->name, tp->name);
                        q->age = tp->age;
                    }
                    q = q->next;
                }
                p = p->next;
            }
        }
        else if (2 == choose2) // 降序
        {
            STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
            while (NULL != p)
            {
                STU *q = p->next; // 创建一个临时指针,用来保存头指针,防止头指针丢失
                while (NULL != q)
                {
                    if (p->age < q->age)
                    {
                        STU *tp = (STU *)malloc(sizeof(STU));
                        tp->sid = p->sid;
                        strcpy(tp->name, p->name);
                        tp->age = p->age;

                        p->sid = q->sid;
                        strcpy(p->name, q->name);
                        p->age = q->age;

                        q->sid = tp->sid;
                        strcpy(q->name, tp->name);
                        q->age = tp->age;
                    }
                    q = q->next;
                }
                p = p->next;
            }
        }
        else
        {
            // 选择错误
            printf("\033[%dm选择错误\033[0m\n", RED);
```

```
        }
    }
    else
    {
        // 选择错误
        printf("\033[%dm选择错误\033[0m\n", RED);
    }
    return head; // 返回头指针
}

// 反序学生
STU *reverse_stu(STU *head)
{
    if (NULL == head)
    {
        printf("\033[%dm无数据,无法反序\033[0m\n", RED);
        return NULL;
    }
    STU *p = head; // 创建一个临时指针,用来保存头指针,防止头指针丢失
    STU *q = NULL; // 创建一个临时指针,用来保存尾部指针
    STU *tp = NULL;
    while (NULL != p) // 当 NULL 不等于 p 时,说明 p 不是尾部,则反序
    {
        // 交换 p 和 q 的数据
        q = p->next;  // 保存 p 的下一个节点，用作临时变量
        p->next = tp; // 将 p 的下一个节点指向 tp
        tp = p;       // 将 tp 指向 p
        p = q;        // 将 p 指向 q
    }
    head = tp; // 将头指针指向 tp
    printf("\033[%dm反序成功\033[0m\n", BLUE);
    return head; // 返回头指针
}

int main(int argc, char const *argv[])
{
    STU *head = NULL; // 定义全局的头指针

    // 声明临时变量,用来保存学生信息
    char tsid;
    char tname[32];
    int tage;

    // 创建几个学生信息，用于测试
    STU *stu1 = (STU *)malloc(sizeof(STU));
    stu1->sid = 1;
    strcpy(stu1->name, "张三");
    stu1->age = 18;
    STU stu2 = {2, "李四", 21, NULL};
    STU stu3 = {3, "王五", 15, NULL};
    STU stu4 = {4, "赵六", 13, NULL};
    head = insert_stu(head, stu1);
    head = insert_stu(head, &stu2);
    head = insert_stu(head, &stu3);
    head = insert_stu(head, &stu4);
```

```
    while (1)
    {
        // printf("\033[2J");
        printf("--------------------\n");
        printf("\033[%d;%dm学生管理系统\033[0m\n", BOLD, CYAN_B);
        printf("\033[%d;%dm1) 添加学生 \n2) 删除学生 \n3) 打印  \n4) 查询  \n5) 修改
\n6) 排序  \n7) 反序 \n0) 退出\n请输入选项# \033[0m", BOLD, GREEN);
        int cmd = 0;
        scanf("%d", &cmd);
        switch (cmd)
        {
        case 0:
            return 1;
        case 1:
            printf("输入学生的 学号 姓名 年龄: ");
            // 键盘收集数据并动态插入链表
            scanf("%hhd %s %d", &tsid, tname, &tage); // 键盘收集数据
            STU *item = (STU *)malloc(sizeof(STU));    // 申请一个节点的内存，用来保存
学生信息
            item->sid = tsid;                          // 保存学生学号到新建的节点
            strcpy(item->name, tname);                 // 保存学生姓名到新建的节点
            item->age = tage;                          // 保存学生年龄到新建的节点
            item->next = NULL;                         // 新建的节点的下一个节点指向空，
因为是尾部插入
            head = insert_stu(head, item);             // 将新建的节点插入到链表中
            break;
        case 2:
            printf("输入要删除的学生的学号: ");
            scanf("%hhd", &tsid);
            head = delete_stu(head, tsid);
            break;
        case 3:
            shows(head);
            break;
        case 4:
            printf("输入要查询的学生的学号: ");
            scanf("%hhd", &tsid);
            head = find_stu(head, tsid);
            if (head == NULL)
            {
                printf("\033[%dm未找到 sid = %d 的学生\033[0m\n", RED, tsid);
            }
            else
            {
                printf("查询到的学生信息如下: \n");
                printf("\033[%dm学号\t姓名\t年龄\033[0m\n", BLUE_B);
                printf("\033[%dm%d\t%s\t%d\033[0m\n", BLUE, head->sid, head-
>name, head->age);
            }
            break;
        case 5:
            printf("输入要修改的学生的学号:");
            scanf("%hhd", &tsid);
            head = update_stu(head, tsid);
            break;
        case 6:
```

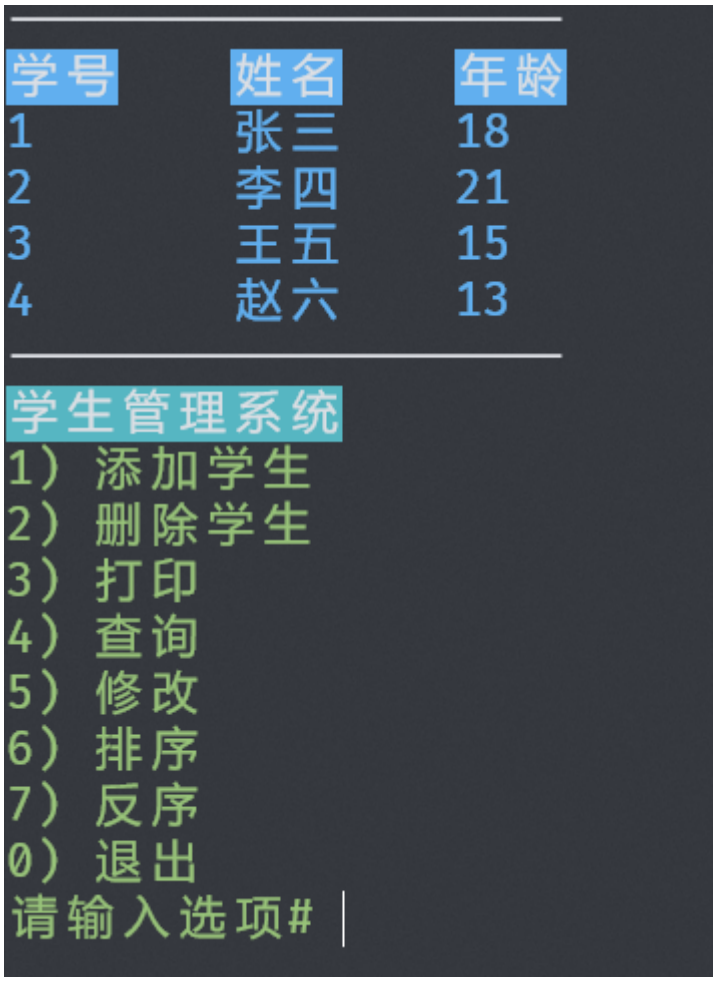
```
            printf("排序学生信息:\n");
            head = sort_stu(head);
            break;
        case 7:
            printf("反序学生信息:\n");
            head = reverse_stu(head);
            break;
        default:
            printf("\033[%dm选择错误\033[0m\n", RED);
            break;
        }
    }
    return 0;
}
```

主菜单



3. 查询学生

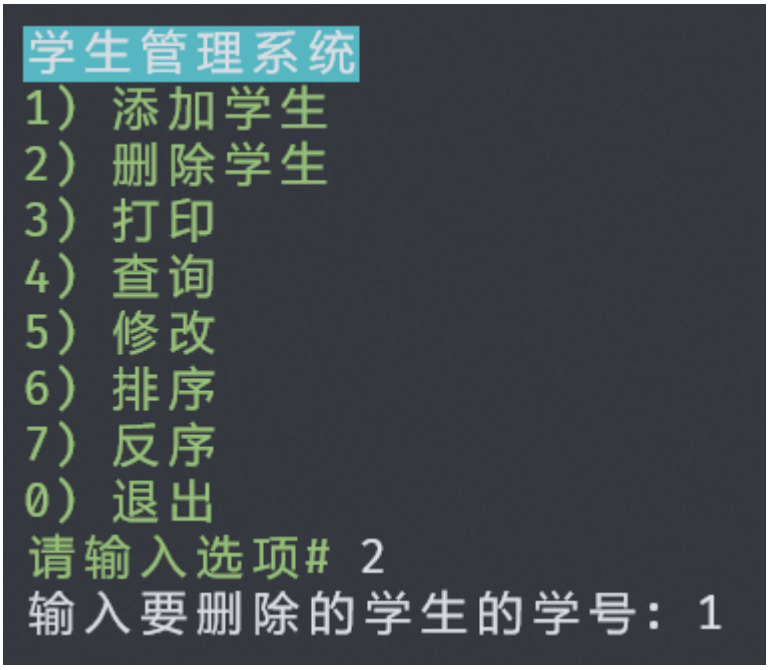
默认4位学生打印为



1. 添加学生

再次打印结果为

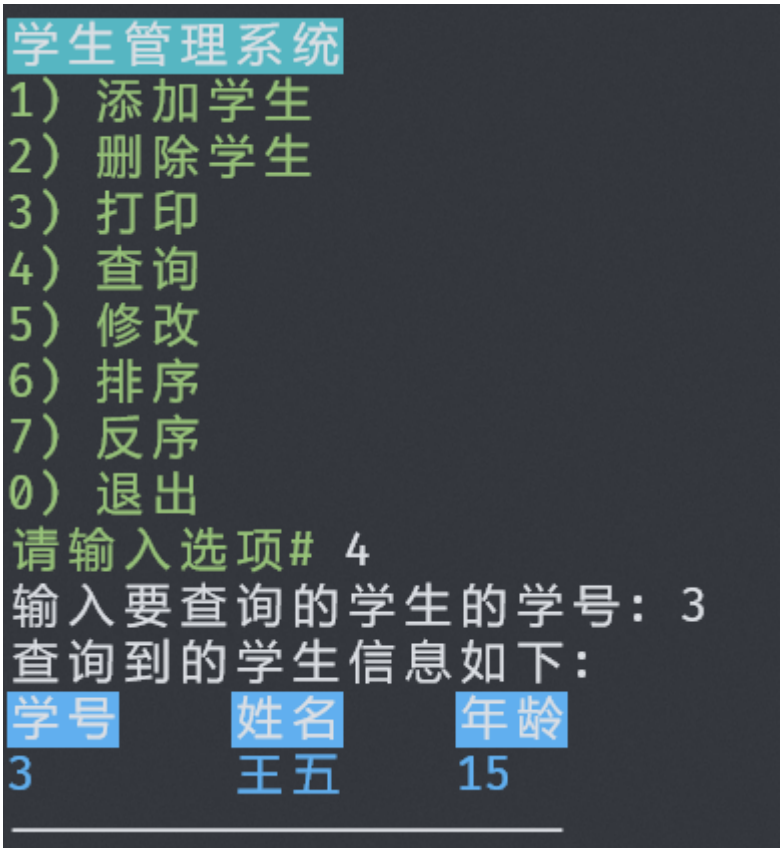
2. 删除学生


学生管理系统
1） 添加学生
2） 删除学生
3） 打印
4） 查询
5） 修改
6） 排序
7） 反序
0） 退出
请输入选项# 2
输入要删除的学生的学号: 1

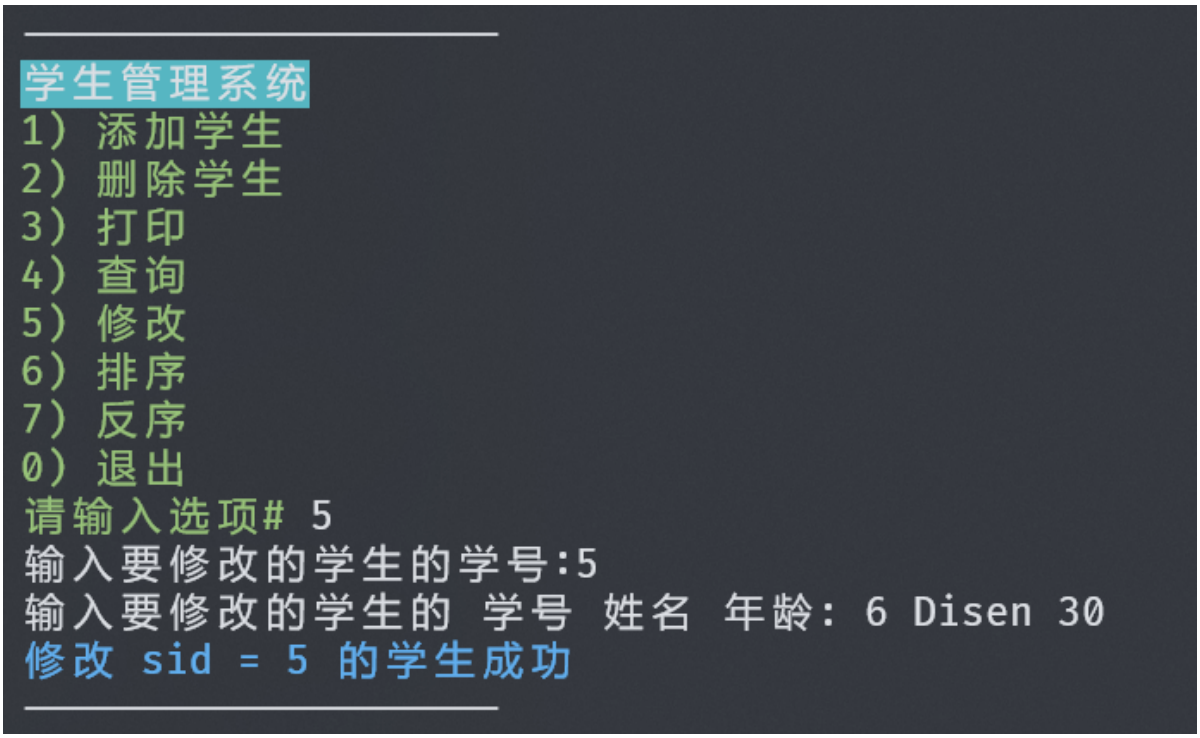
再次打印


学号 姓名 年龄
2 李四 21
3 王五 15
4 赵六 13
5 disen 19

4. 查询学生


学生管理系统
1) 添加学生
2) 删除学生
3) 打印
4) 查询
5) 修改
6) 排序
7) 反序
0) 退出
请输入选项# 4
输入要查询的学生的学号: 3
查询到的学生信息如下:
学号 姓名 年龄
3 王五 15

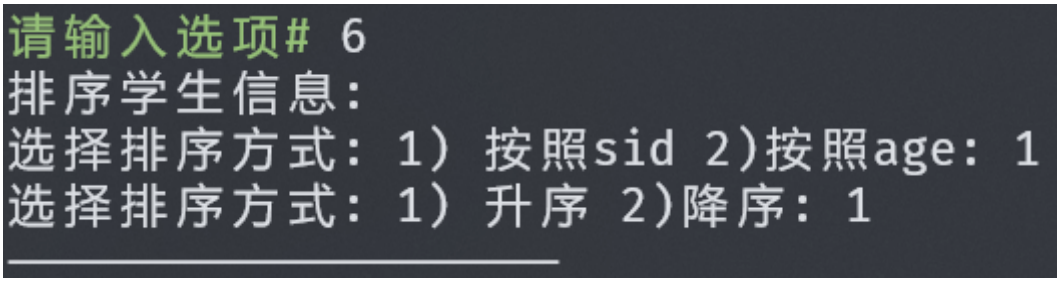
5. 修改学生


学生管理系统
1) 添加学生
2) 删除学生
3) 打印
4) 查询
5) 修改
6) 排序
7) 反序
0) 退出
请输入选项# 5
输入要修改的学生的学号:5
输入要修改的学生的 学号 姓名 年龄: 6 Disen 30
修改 sid = 5 的学生成功

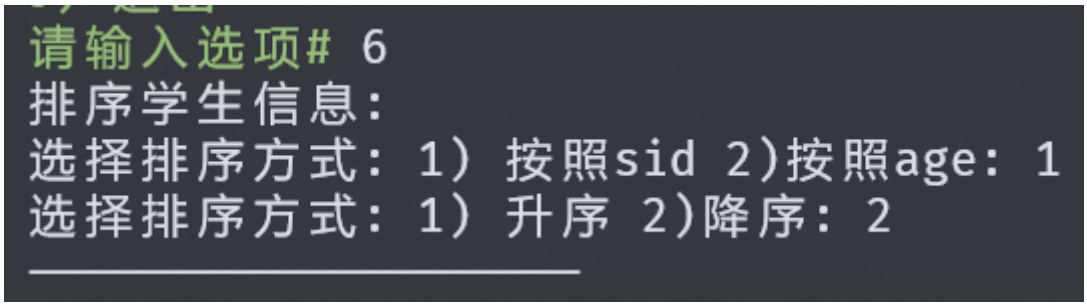
再次打印如下

6. 排序（按照学号 sid-升序、sid-降序、年龄 age-升序、age-降序）

sid-升序



sid-降序



age-升序

```
请输入选项# 6
排序学生信息:
选择排序方式: 1) 按照sid 2)按照age: 2
选择排序方式: 1) 升序 2)降序: 1
```

学号	姓名	年龄
4	赵六	13
3	王五	15
6	Disen	30

age-降序

```
请输入选项# 6
排序学生信息:
选择排序方式: 1) 按照sid 2)按照age: 2
选择排序方式: 1) 升序 2)降序: 2
```

学号	姓名	年龄
6	Disen	30
3	王五	15
4	赵六	13

错误选项

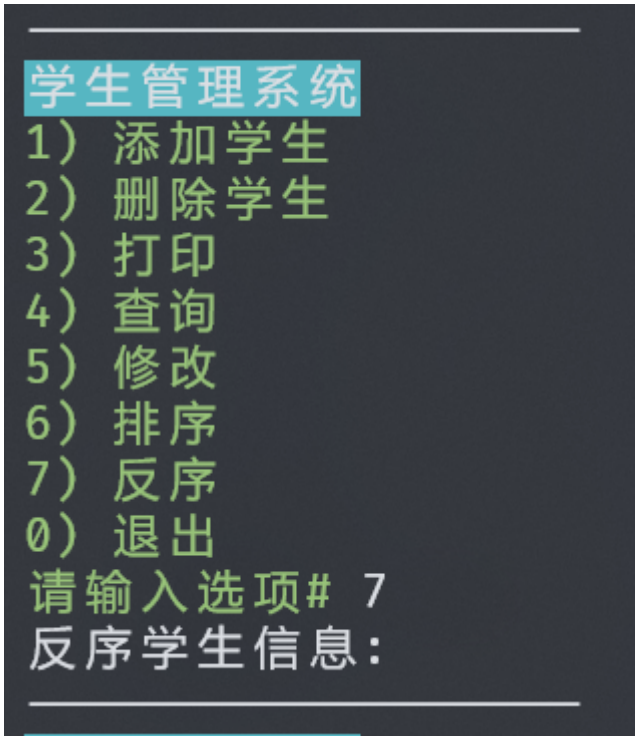
```
请输入选项# 6
排序学生信息:
选择排序方式: 1) 按照sid 2)按照age: 1
选择排序方式: 1) 升序 2)降序: 3
选择错误
--------------------------------

学生管理系统
1) 添加学生
2) 删除学生
3) 打印
4) 查询
5) 修改
6) 排序
7) 反序
0) 退出
请输入选项# 6
排序学生信息:
选择排序方式: 1) 按照sid 2)按照age: 4
选择错误
--------------------------------
```

7. 逆序（反序）：在当前排序的基础上

反序前输出

```
--------------------------------

学号      姓名      年龄
6         Disen     30
3         王五      15
4         赵六      13
--------------------------------
```

执行反序操作



反序后输出