PA1 顺序表实验报告 ： 顺序表的应用实验报告

姓名：钱子贤 学号：2054170 日期：2021 年 10 月 17 日

实验报告格式按照模板来即可，对字体大小、缩进、颜色等不做要求

实验报告要求在文字简洁的同时将内容表示清楚

1. 涉及数据结构和相关背景

1 、掌握线性表的定义及顺序表示;

2 、掌握顺序表实现线性表的基本操作，如建立、查找、插入和删除以及去重等;

3、 掌握顺序表的特点;

2. 实验内容

定义一个包含学生信息（学号，姓名，成绩）的 顺序表，使其具有如下功能:

```
//顺序表数据结构
typedef struct
{
    int no[max];//顺序表元素
    char name[max][20];
    int score[max];
    int length;
    //顺序表当前长度
}student;
```

由于我写代码之前没有看到参考的提示，所以我的顺序表的定义和参考信息不太一样。由于时间原因，没能及时重写一份，我觉得可以做的思考如下:
在 typedef struct student（参考信息）中，可以定义数组 stu[max]，是的 typedef struct sqlist（参考信息）里的*elem 为 stu 数组元素的首地址，这样就可以使得与我原本写的顺序表定义所得到的效果一样。不过我觉得顺序表的定义可以有很多种，我觉得老师应该会更加注重我后面算法的部分。

参考信息：

```
typedef struct {
   char no[8];  //8 位学号
   char name[20]; // 姓名
   double score;   // 成绩
}Student;
typedef Student ElemType;
typedef  struct {
 ElemType  *elem;   // 指向数据元素的基地址
 int  length;     // 线性表的当前长度
 int  listsize;    //线性表的最大容量
 }SqList ;
```

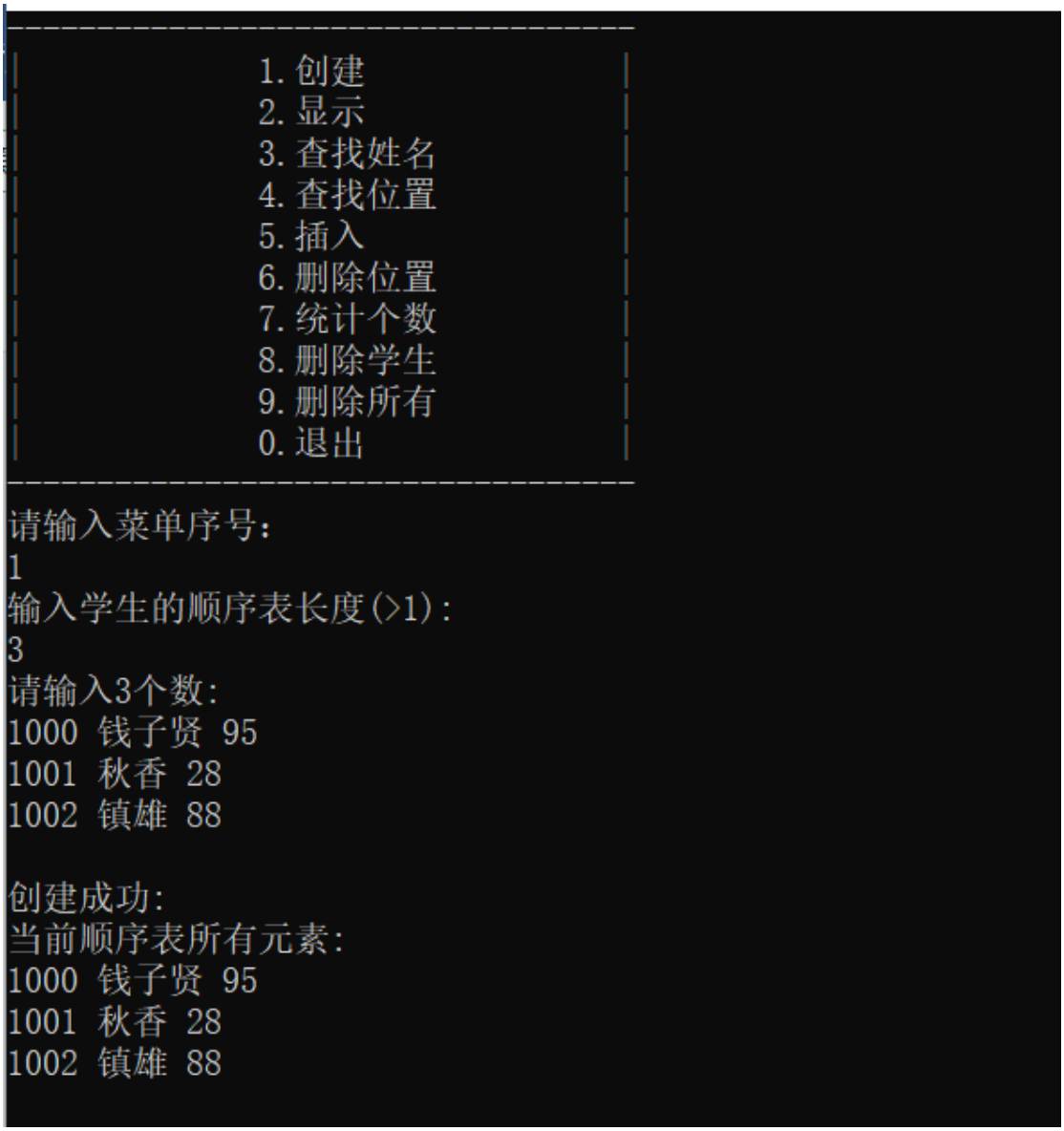
(1) 根据指定学生个数，逐个输入学生信息；

```
//打印顺序表元素
void printit(student p)
{
    cout << "当前顺序表所有元素:" << endl;
    for (int i = 0; i < p.length; i++)
    {
        cout << p.no[i] << " " << p.name[i] << " " << p.score[i] << endl;
    }
    cout << endl;
}

//创建顺序表函数 初始化前n个数据
bool create1(student& p, int n)
{
    if (n<1 || n>max)
        false;//n非法
    for (int i = 0; i < n; i++)
    {
        cin >> p.no[i];
        cin >> p.name[i];
        cin >> p.score[i];
        p.length++;
    }
```

```
    return true;
}
//创建顺序表函数
void create(student& p)
{
    int n;
    bool flag;
    p.length = 0;
    cout << "输入学生的顺序表长度(>1):" << endl;
    cin >> n;
    cout << "请输入" << n << "个数:" << endl;
    flag = create1(p, n);
    if (flag) {
        cout << endl;
        cout << "创建成功:" << endl;
        printit(p);
    }
    else cout << "输入长度非法" << endl;

}
```

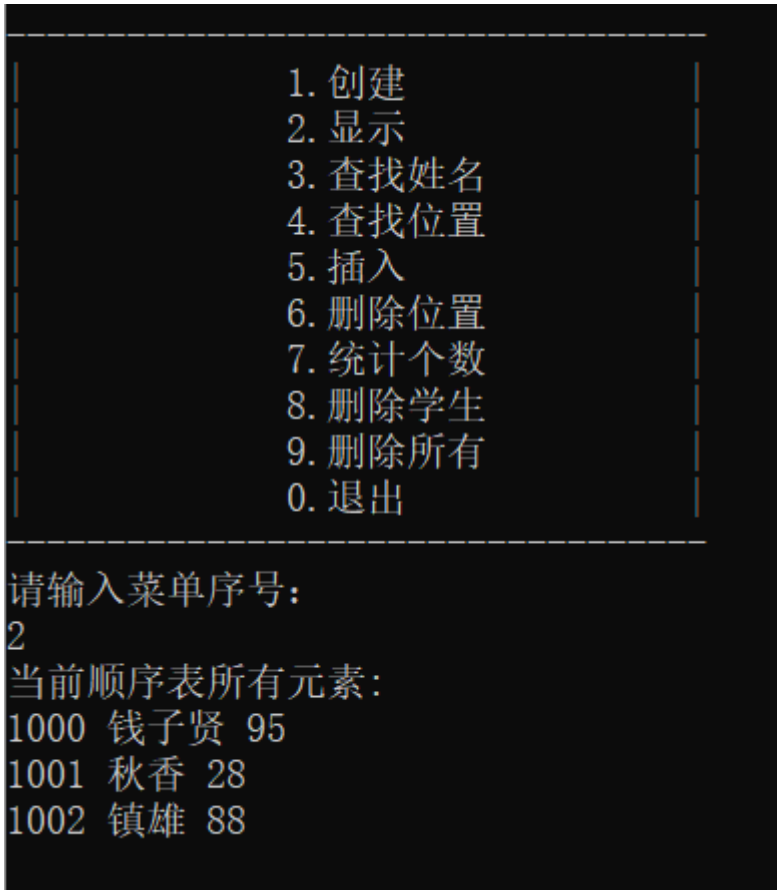


create (student &p,int n) 参数：顺序表 p，顺序表长度 n 功能：创建长度为的顺序表 时间复杂度：O(n)

(2) 逐个显示学生表中所有学生的相关信息;

```
//打印顺序表元素
void printit(student p)
{
    cout << "当前顺序表所有元素:" << endl;
    for (int i = 0; i < p.length; i++)
    {
        cout << p.no[i] << " " << p.name[i] << " " << p.score[i] << endl;
    }
    cout << endl;
```

}

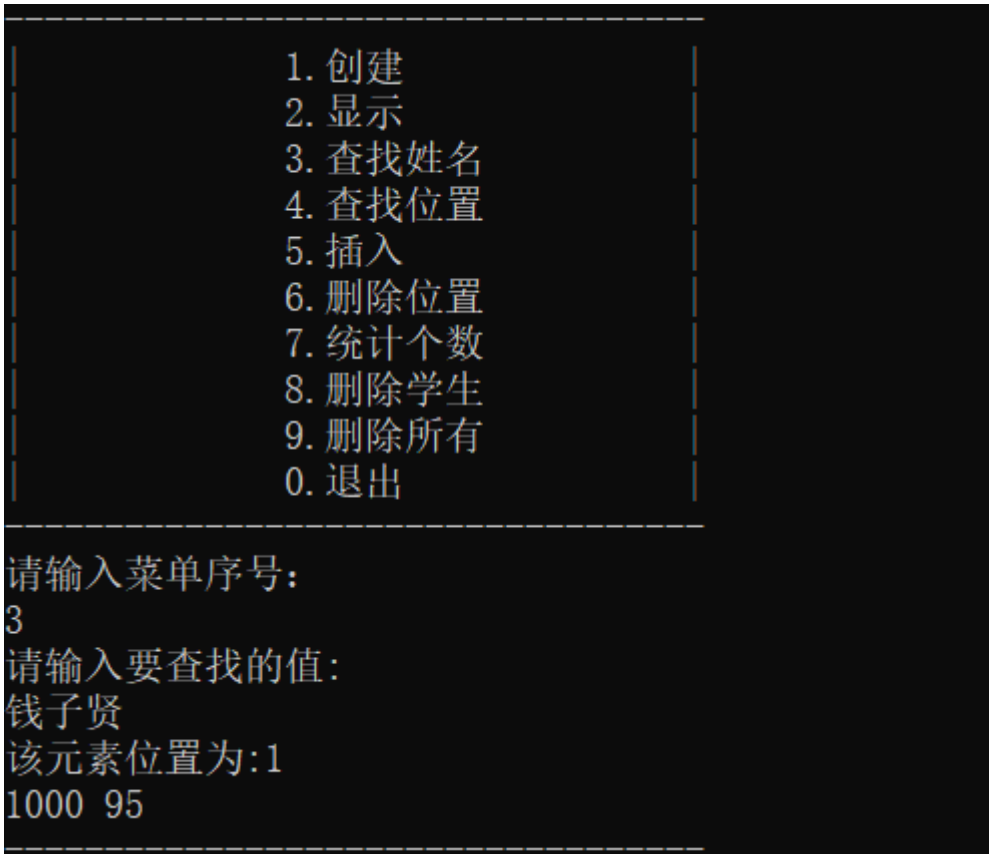


printit(student p) 参数:顺序表p 功能：遍历p，并输出

(3) 根据姓名进行查找，返回此学生的学号和成绩;

```
//查找函数 按位置从小到大查找第一个值等于e的元素 并返回位置
int search1(student p, char name[20])
{
    for (int i = 0; i < p.length; i++)//从低位置查找
    {
        if (*p.name[i] == *name)
            return i + 1;
    }
    return 0;
}
```

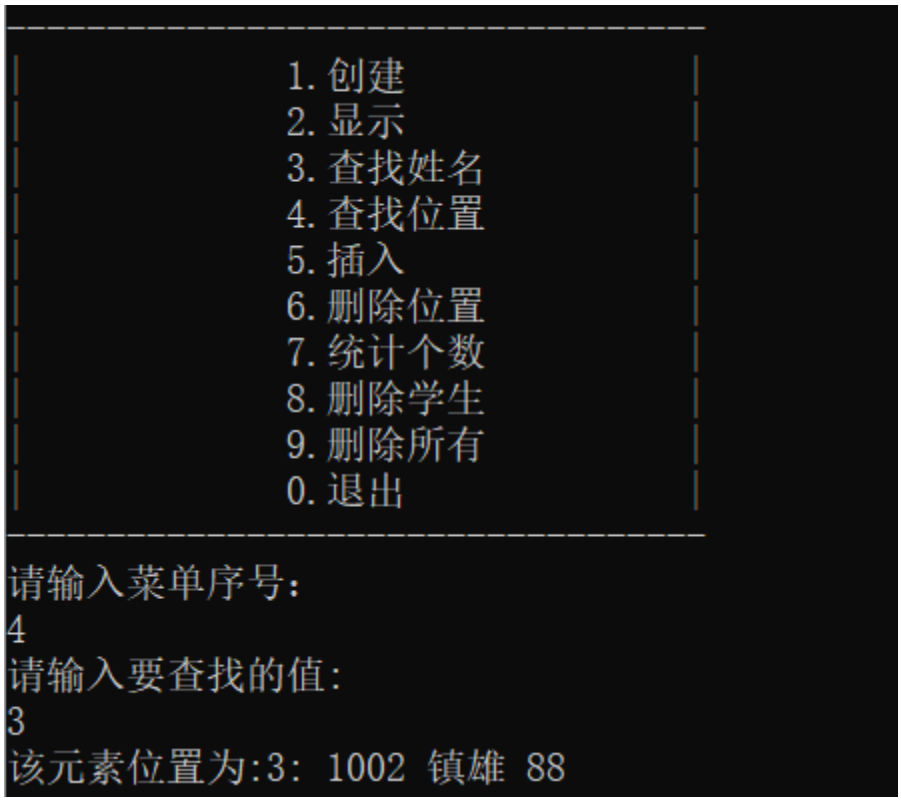
```
//查找功能函数 调用search1查找元素
void search(student p)
{
    char name[10]; int flag;
    cout << "请输入要查找的值:" << endl;
    cin >> name;
    flag = search1(p, name);
    if (flag)
    {
        cout << "该元素位置为:" << flag << "" << endl;
        cout << p.no[flag - 1] << " " << p.score[flag - 1] << endl;
    }
    else
        cout << "未找到该元素！" << endl;
}
```

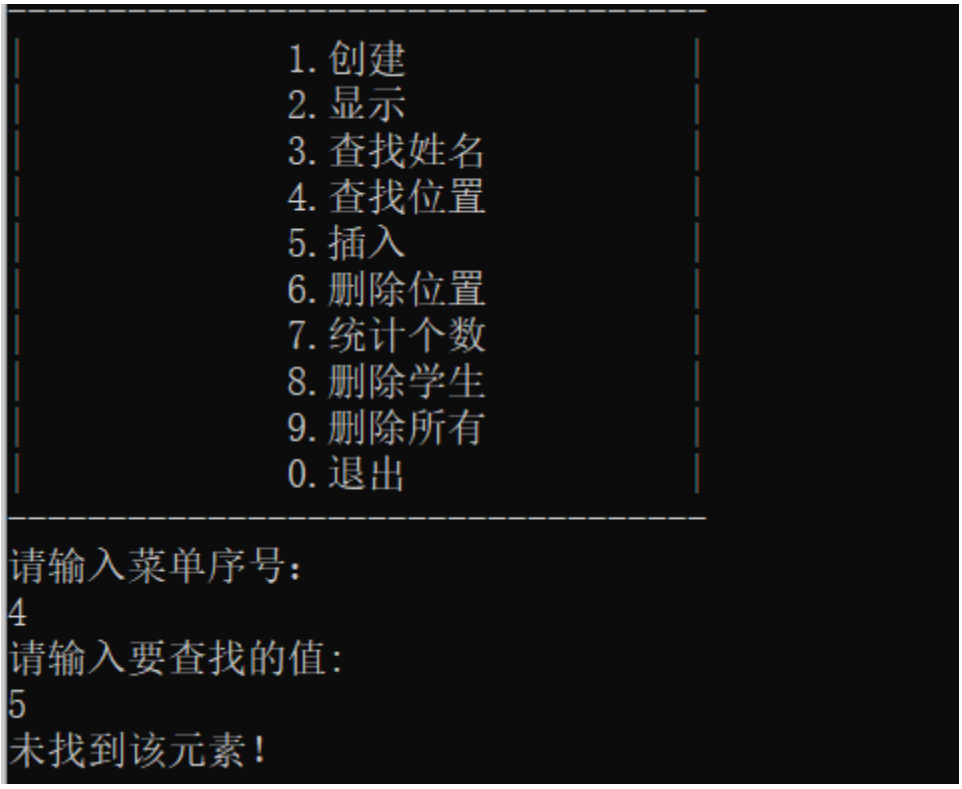


search(student p) 参数:顺序表p 功能：返回找到的名字的元素的值 时间复杂度:O(n)

(4) 根据指定的位置可返回相应的学生信息（学号，姓名，成绩）;

```
//查找函数 查找位置 并返回位置
void searchloc(student p)
{
    int i;
    cout << "请输入要查找的值:" << endl;
    cin >> i;
    if (i>=1&&i<=max&&p.no[i-1]!=0)
    {
        cout << "该元素位置为:" << i << ": " ;
        cout << p.no[i - 1] << " " << p.name[i - 1] << " " << p.score[i - 1] << endl;
    }
    else
        cout << "未找到该元素！" << endl;
}
```



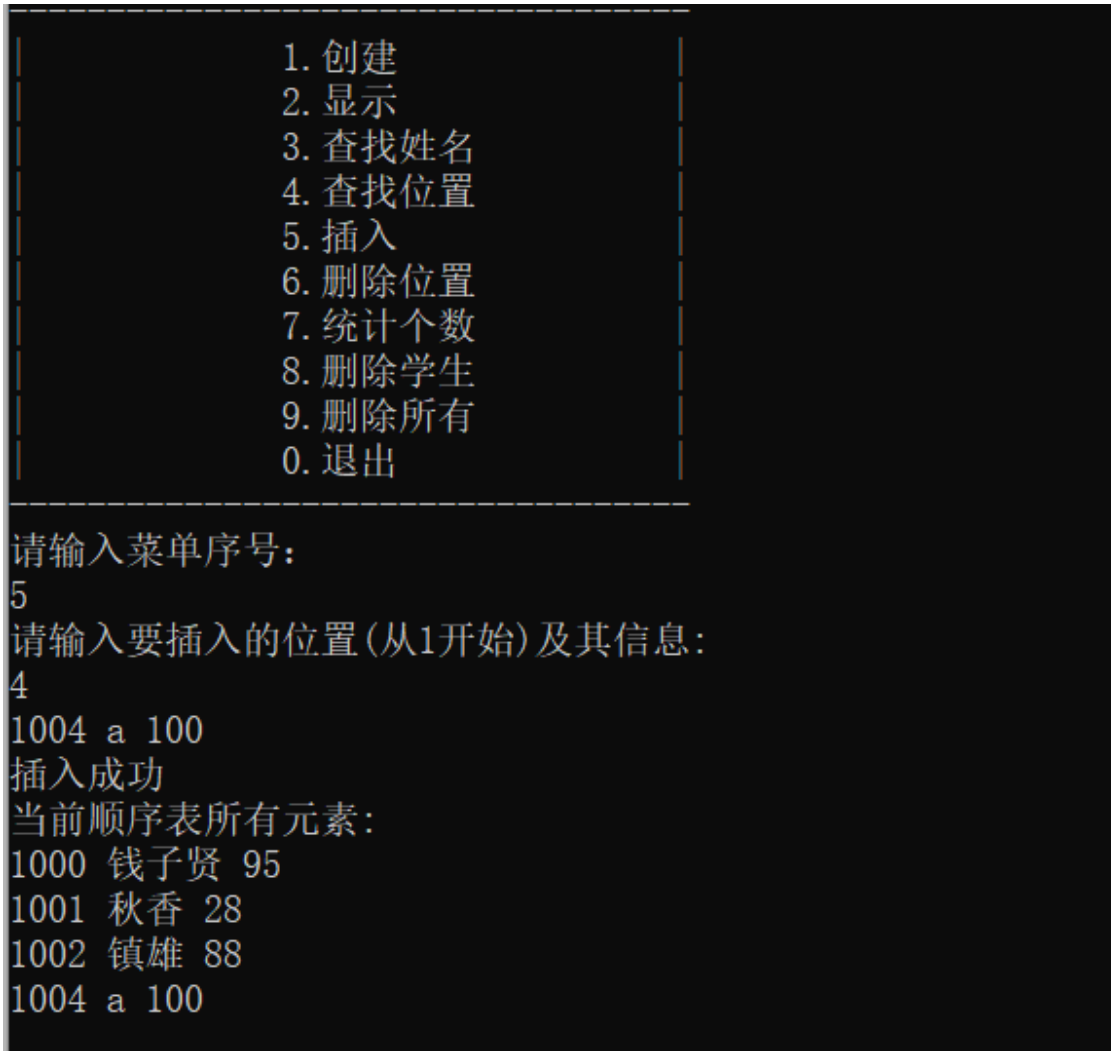


searchloc(student p) 参数:顺序表p 功能：返回找到的位置的元素值 时间复杂度:O(n)

(5) 给定一个学生信息，插入到表中指定的位置;

```
//插入函数 位置i插入数据 i及之后元素后移  1=<i<=length+1
bool insert1(student& p, int i, int no1,char name[20],int score1)
{
    if (i<1 || i>p.length + 1) //判断位置是否有效
    {
        cout<<"位置不对"<<endl;
        return false;
    }
    if (p.length >= max)//判断存储空间是否已满
    {
        cout<<"当前存储人数已满"<<endl;
        return false;
    }
    for (int j = p.length; j >= i; j--)//位置i及之后元素后移
    {
        p.no[j] = p.no[j - 1];
        *p.name[j] = *p.name[j - 1];
        p.score[j] = p.score[j - 1];
    }
    p.no[i - 1] = no1;
    *p.name[i - 1] = *name;
```

```
    p.score[i - 1] = score1;
    p.length++;
    return true;
}
//插入功能函数 调用insert1完成顺序表元素插入 调用printit函数显示插入成功后的结果
void insert(student& p)
{
    int place;
    int no1;
    char name[20];
    int score1;
    bool flag;
    cout<<"请输入要插入的位置(从1开始)及其信息:"<<endl;
    cin >> place ;
    cin >> no1;
    cin >> name;
    cin >> score1;
    flag = insert1(p, place, no1, name, score1);
    if (flag)
    {
        cout<<"插入成功"<<endl;
        printit(p);
    }
}
```

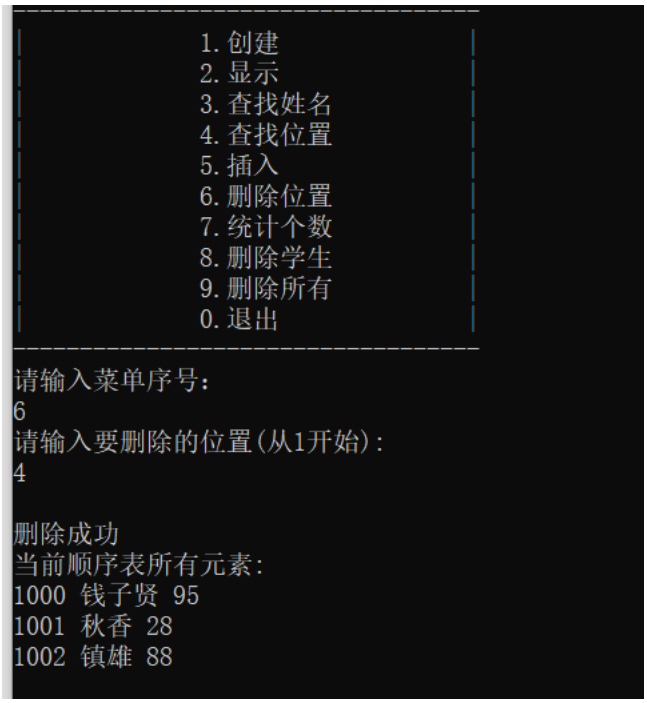


insert1(student &p,int i, int no1,char name[20],int score1) 参数:顺序表p,位置i,功能：位置i处插入新元素， 时间复杂度:O(n)

(6) 删除指定位置的学生记录;

```
//删除函数 删除位置i的元素 i之后的元素依次前移
bool deletes1(student& p, int i)
{
    if (i<1 || i>p.length)
    {
        cout<<"位置无效"<<endl;
        return false;
    }
    for (int j = i; j <= p.length - 1; j++)//位置i之后元素依次前移覆盖
    {
        p.no[j - 1] = p.no[j];
```

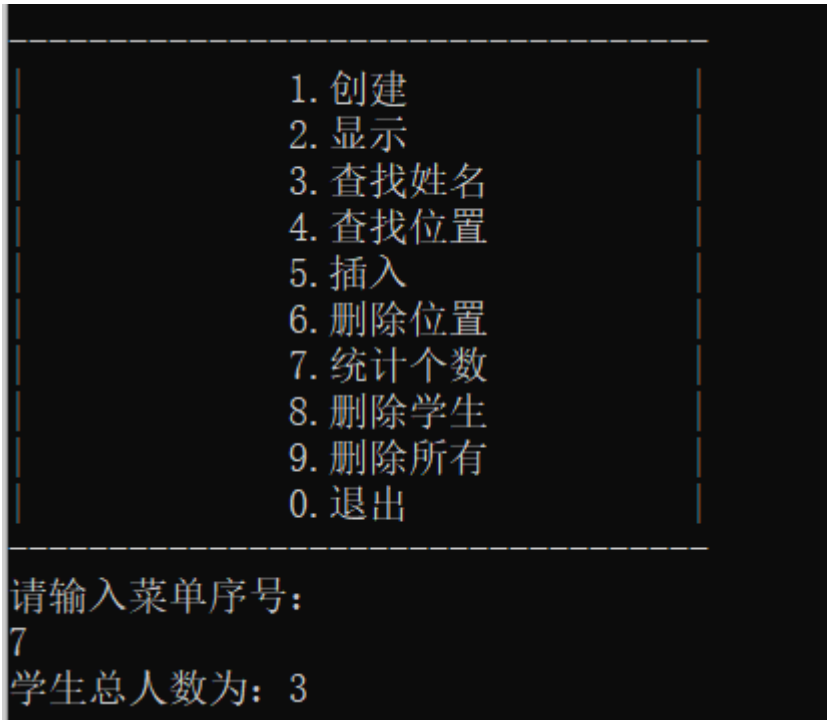
```
            *p.name[j - 1] = *p.name[j];
            p.score[j - 1] = p.score[j];
        }
        p.length--;
        return true;
}
void deletes(student& p)
{
        int place;
        bool flag;
        cout<<"请输入要删除的位置(从1开始):"<<endl;
        cin>> place;
        flag = deletes1(p, place);
        if (flag)
        {
            cout << endl;
            cout<<"删除成功"<<endl;
            printit(p);
        }
}
```



deletes1(student &p,int i) 参数:顺序表p,位置i 功能：删除位置i处元素 时间复杂度:O(n)

(7) 统计表中学生个数。

```
                cout<<"学生总人数为："<<p.length<<endl; break;
```



过于简单，直接输出就行

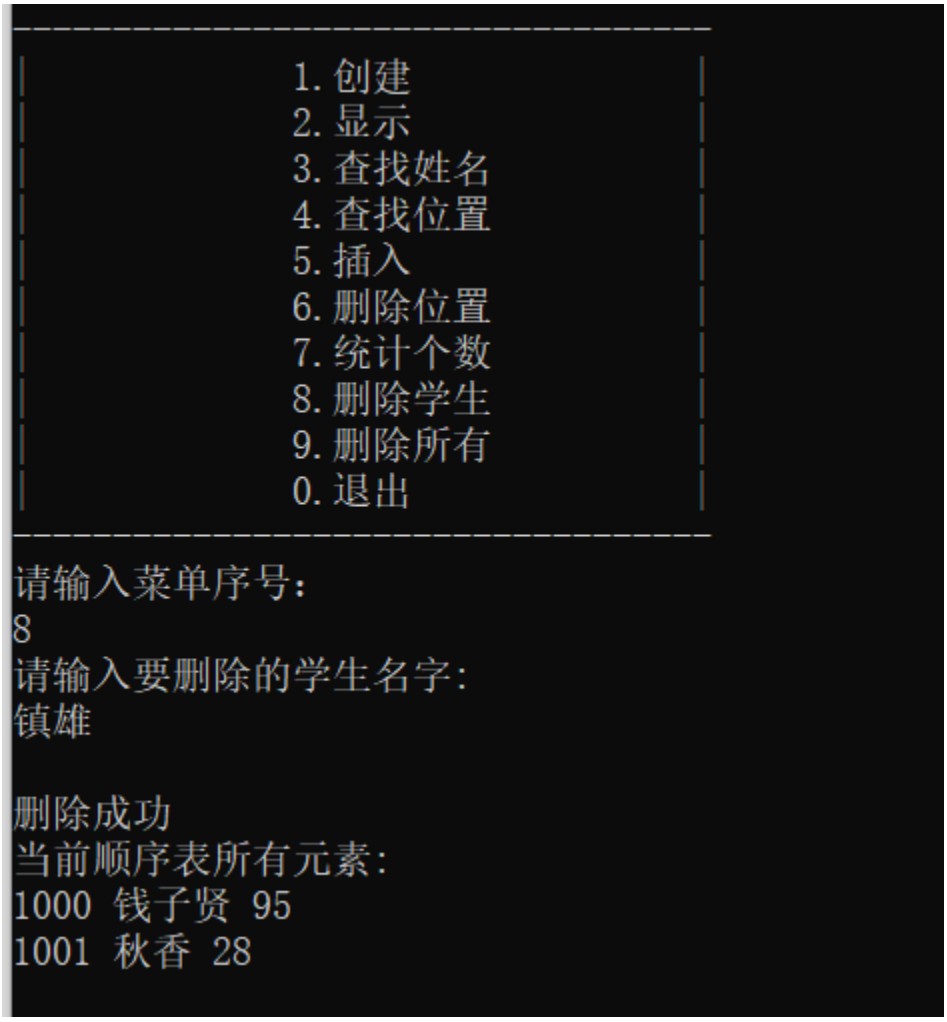
(8) 删除某学生的所有记录

```
//删除功能函数 调用deletes_one1函数完成顺序表的删除 调用printit函数显示插入成功后的结果

bool deletes_one1(student& p, char name[20])
{
    int judge = 0; int i = 0;
    for (i=0; i < p.length; i++)
    {
        if (*p.name[i] == *name)
        {
            judge = 1;
            break;

        }
    }
    i++;
    if (judge==0)
    {
        cout << "没找到名字" << endl;
        return false;
    }
    else
```

```
    for (int j = i; j <= p.length - 1; j++)//位置i之后元素依次前移覆盖
    {
        p.no[j - 1] = p.no[j];
        *p.name[j - 1] = *p.name[j];
        p.score[j - 1] = p.score[j];
    }
    p.length--;
    return true;
}
void deletes_one(student& p)
{
    char name[20];
    bool flag;
    cout << "请输入要删除的学生名字:" << endl;
    cin >> name;
    flag = deletes_one1(p, name);
    if (flag)
    {
        cout << endl;
        cout << "删除成功" << endl;
        printit(p);
    }
}
```

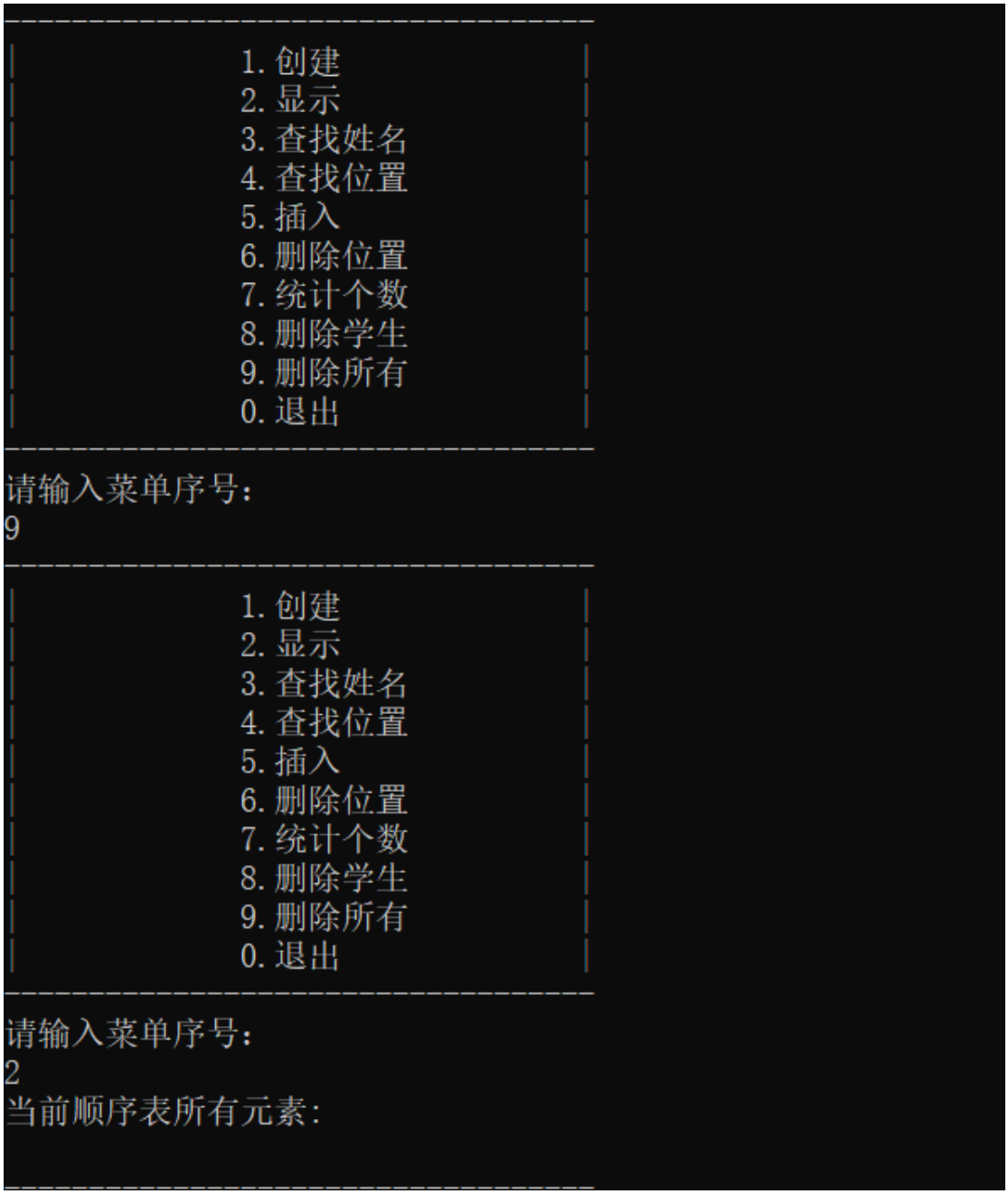



```
bool deletes_one1(student& p, char name[20])
```

参数:顺序表p,char name[20] 功能：删除所属名字处元素 时间复杂度:O(n)

(9) 删除所有重复记录

```
//清空顺序表
void clearall(student& p) {
    p.length = 0;
}
```



clearall(student &p) 参数：顺序表p 功能：清空顺序表

2.1.5 调试分析（遇到的问题和解决方法）

调试分析已在上图展示

会出现的问题在于 name 的输入输出会比较麻烦，有时用了 name[i-1]=name[i] 发现不对，最后的解决方案是因为定义的是*name[]，所以需要加上*，*name[i-

1]=*name[i]。其实这道题目遇到的困难还是比较少的，还有一些问题比如删除后出现汉字乱码等问题，通过对于+-i 的细心分析，最后也得到了解决。

2.1.6 总结和体会

需要释放空间，是堆和栈的区别。堆和栈的区别就是申请方式不同：栈是系统自动分配空间，而堆则是程序员根据需要申请空间。由于栈上的空间是自动分配自动回收的，所以，栈内数据的生存周期只是在函数的运行过程中，运行后就释放掉。而堆上的数据只要不释放空间，就一直可以访问到，缺点是一旦忘记释放会造成内存泄露。顺序表不需要进行销毁，顺序表最大长度也固定了，各有好坏，如果是动态分配的话需要释放空间。

3. 实验总结

线性表、顺序表基本操作的编程实现，掌握线性表、顺序表的建立、遍历、插入、删除、读取等基本操作的编程实现，也可以进一步编程实现逆序等操作，存储结构可以采用顺序存储结构和链表存储结构之一，可以依次完成主要功能来体现功能的正确性，用菜单进行管理完成大部分功能，要求可以重复运行。线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个元素，使得线性表在逻辑结构上相邻的元素存储在连续的物理存储单元中，即：通过数据元素物理存储的连续性来反应元素之间逻辑上的相邻关系。采用顺序存储结构存储的线性表通常简称为顺序表。顺序存储的线性表的特点在于线性表的逻辑顺序与物理顺序一致;数据元素之间的关系是以元素在计算机内“物理位置相邻”来体现。

源代码:

```
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
#include<iostream>
#define max 100
using namespace std;
//顺序表数据结构
typedef struct
{
    int no[max];//顺序表元素
    char name[max][20];
    int score[max];
    int length;
    //顺序表当前长度
}student;

//初始化顺序表函数，构造一个空的顺序表
int initial(student& p)
{
    memset(p.no, 0, sizeof(p.no));
    memset(p.name, 0, sizeof(p.name));
    memset(p.score, 0, sizeof(p.score));//初始化数据为0
    p.length = 0;                //初始化长度为0
    return 0;
}
//打印顺序表元素
void printit(student p)
{
    cout << "当前顺序表所有元素:" << endl;
    for (int i = 0; i < p.length; i++)
    {
        cout << p.no[i] << " " << p.name[i] << " " << p.score[i] << endl;
    }
    cout << endl;
}
//创建顺序表函数 初始化前n个数据
bool create1(student& p, int n)
{
    if (n<1 || n>max)
        false;//n非法
    for (int i = 0; i < n; i++)
```

```
    {
        cin >> p.no[i];
        cin >> p.name[i];
        cin >> p.score[i];
        p.length++;
    }
    return true;
}
//创建顺序表函数
void create(student& p)
{
    int n;
    bool flag;
    p.length = 0;
    cout << "输入学生的顺序表长度(>1):" << endl;
    cin >> n;
    cout << "请输入" << n << "个数:" << endl;
    flag = create1(p, n);
    if (flag) {
        cout << endl;
        cout << "创建成功:" << endl;
        printit(p);
    }
    else cout << "输入长度非法" << endl;

}
//查找函数 按位置从小到大查找元素 并返回位置
int search1(student p, char name[20])
{
    for (int i = 0; i < p.length; i++)//从低位置查找
    {
        if (*p.name[i] == *name)
            return i + 1;
    }
    return 0;
}
//查找功能函数 调用search1查找元素
void search(student p)
{
    char name[10]; int flag;
    cout << "请输入要查找的值:" << endl;
    cin >> name;
    flag = search1(p, name);
    if (flag)
```

```
    {
        cout << "该元素位置为:" << flag << "" << endl;
        cout << p.no[flag - 1] << " " << p.score[flag - 1] << endl;
    }
    else
        cout << "未找到该元素！" << endl;
}
//查找函数 查找位置 并返回位置
void searchloc(student p)
{
    int i;
    cout << "请输入要查找的值:" << endl;
    cin >> i;
    if (i>=1&&i<=max&&p.no[i-1]!=0)
    {
        cout << "该元素位置为:" << i << ": " ;
        cout << p.no[i - 1] << " " << p.name[i - 1] << " " << p.score[i - 1] << endl;
    }
    else
        cout << "未找到该元素！" << endl;
}

//插入函数 位置i插入数据 i及之后元素后移  1=<i<=length+1
bool insert1(student& p, int i, int no1,char name[20],int score1)
{
    if (i<1 || i>p.length + 1) //判断位置是否有效
    {
        cout<<"位置不对"<<endl;
        return false;
    }
    if (p.length >= max)//判断存储空间是否已满
    {
        cout<<"当前存储人数已满"<<endl;
        return false;
    }
    for (int j = p.length; j >= i; j--)//位置i及之后元素后移
    {
        p.no[j] = p.no[j - 1];
        *p.name[j] = *p.name[j - 1];
        p.score[j] = p.score[j - 1];
    }
    p.no[i - 1] = no1;
    *p.name[i - 1] = *name;
    p.score[i - 1] = score1;
```

```
    p.length++;
    return true;
}
//插入功能函数 调用insert1完成顺序表元素插入 调用printit函数显示插入成功后的结果
void insert(student& p)
{
    int place;
    int no1;
    char name[20];
    int score1;
    bool flag;
    cout<<"请输入要插入的位置(从1开始)及其信息:"<<endl;
    cin >> place ;
    cin >> no1;
    cin >> name;
    cin >> score1;
    flag = insert1(p, place, no1, name, score1);
    if (flag)
    {
        cout<<"插入成功"<<endl;
        printit(p);
    }
}
//删除函数 删除位置i的元素 i之后的元素依次前移
bool deletes1(student& p, int i)
{
    if (i<1 || i>p.length)
    {
        cout<<"位置无效"<<endl;
        return false;
    }
    for (int j = i; j <= p.length - 1; j++)//位置i之后元素依次前移覆盖
    {
        p.no[j - 1] = p.no[j];
        *p.name[j - 1] = *p.name[j];
        p.score[j - 1] = p.score[j];
    }
    p.length--;
    return true;
}
void deletes(student& p)
{
    int place;
    bool flag;
```

```
    cout<<"请输入要删除的位置(从1开始):"<<endl;
    cin>> place;
    flag = deletes1(p, place);
    if (flag)
    {
        cout << endl;
        cout<<"删除成功"<<endl;
        printit(p);
    }
}
//删除功能函数 调用deletes_one1函数完成顺序表的删除 调用printit函数显示插入成功后的结果

bool deletes_one1(student& p, char name[20])
{
    int judge = 0; int i = 0;
    for (i=0; i < p.length; i++)
    {
        if (*p.name[i] == *name)
        {
            judge = 1;
            break;

        }
    }
    i++;
    if (judge==0)
    {
        cout << "没找到名字" << endl;
        return false;
    }
    else
    for (int j = i; j <= p.length - 1; j++)//位置i之后元素依次前移覆盖
    {
        p.no[j - 1] = p.no[j];
        *p.name[j - 1] = *p.name[j];
        p.score[j - 1] = p.score[j];
    }
    p.length--;
    return true;
}
void deletes_one(student& p)
{
    char name[20];
    bool flag;
```

```
    cout << "请输入要删除的学生名字:" << endl;
    cin >> name;
    flag = deletes_one1(p, name);
    if (flag)
    {
        cout << endl;
        cout << "删除成功" << endl;
        printit(p);
    }
}

//清空顺序表
void clearall(student& p) {
    p.length = 0;
}









//菜单
void menu()
{
    cout << "-----------------------------------" << endl;
    cout << "|              1.创建               |" << endl;
    cout << "|              2.显示               |" << endl;
    cout << "|              3.查找姓名           |" << endl;
    cout << "|              4.查找位置           |" << endl;
    cout << "|              5.插入               |" << endl;
    cout << "|              6.删除位置           |      " << endl;
    cout << "|              7.统计个数           |" << endl;
    cout << "|              8.删除学生           |" << endl;
    cout << "|              9.删除所有           |" << endl;
    cout << "|              0.退出               |" << endl;
    cout << "-----------------------------------" << endl;


}


int main()
{
```

```
    student p;
    int choice;
    initial(p);
    while (1)
    {
        menu();
        cout << "请输入菜单序号：" << endl;
        cin>>choice;
        if (0 == choice) break;
        switch (choice)
        {
            case 1:
                create(p); break;
            case 2:
                printit(p); break;
            case 3:
                search(p); break;
            case 4:
                searchloc(p); break;
            case 5:
                insert(p); break;
            case 6:
                deletes(p); break;
            case 7:
                cout<<"学生总人数为："<<p.length<<endl; break;
            case 8:
                deletes_one(p); break;
            case 9:
                clearall(p); break;
            default:
                cout<<"输入错误！！！"<<endl;
        }
    }
    return 0;
}
```