**顺序表实验报告**

姓名：陆明奇 学号：2050283 日期：2021年10月15日

**1. 实验内容**

**1.1 问题描述**

运用顺序表实现学生管理系统

**1.2 基本要求**

定义一个包含学生信息（学号，姓名，成绩）的的 顺序表，使其具有如下功能：

(1) 根据指定学生个数，逐个输入学生信息；

(2) 逐个显示学生表中所有学生的相关信息；

(3) 根据姓名进行查找，返回此学生的学号和成绩；

(4) 根据指定的位置可返回相应的学生信息（学号，姓名，成绩）；

(5) 给定一个学生信息，插入到表中指定的位置；

(6) 删除指定位置的学生记录；

(7) 统计表中学生个数。

(8) 删除某学生的所有记录

(9) 删除所有重复记录

**1.3功能说明（函数、类）**

student\* read(int n)

{

student\* stu = (student\*)malloc(n \* sizeof(student));//动态申请内存

if (stu == NULL) {

cout << "动态内存申请失败" << endl;

exit(-1);

}

cout << "请依次输入" << n << "个学生信息" << endl;

student\* p = stu;

for (int i = 1; i <= n; i++) {//读入学生信息

scanf("%s%s%lf", p->no, p->name, &(p->score));

p++;

}

return stu;

}

功能：根据指定学生个数，逐个输入学生信息；

算法复杂度：O(n);

void dispaly(student\* stu,int n)

{

student\* p = stu;

for (int i = 1; i <= n; i++) {//输出学生信息

cout << "第" << i << "名同学的学号是" << p->no << ",姓名是" << p->name << ",成绩是" << p->score << endl;

p++;

}

return;

}

功能：逐个显示学生表中所有学生的相关信息；

算法复杂度：O(n);

int ListSearchName(student\*& stu, char name1[20], int& n)

{

student\* p = stu;

for (int i = 1; i <= n; i++) {

if (!strcmp(p->name, name1)) {//遍历按名字查找

cout << "该学生的学号是" << p->no << ",成绩是" << p->score << endl;

return i;

}

else {

p++;

}

}

return 0;

}

功能：根据姓名进行查找，返回此学生的学号和成绩；

算法复杂度：O(n);

void location(student\*& stu,int pos)

{

student\* p = stu;

cout<<"该学生的学号是"<<(p + pos - 1)->no << ",姓名是" << (p + pos - 1)->name << ",成绩是" << (p + pos - 1)->score << endl;

}

功能：根据指定的位置可返回相应的学生信息（学号，姓名，成绩）；

算法复杂度：O(1);

bool ListInsert(student\*& stu, int pos, char no1[8], char name1[20], double score1,int& n)

{

if (pos<1 || pos>n + 1) {

cout << "位置非法" << endl;

return false;

}

stu = (student\*)realloc(stu, (n + 1) \* sizeof(student));

if (stu == NULL) {

cout << "动态内存申请失败" << endl;

exit(-1);

}

student\* q = &stu[pos - 1];//插入的位置

for (student\* p = &stu[n - 1]; p >= q; p--) {//依次向后移位

strcpy((p + 1)->no, p->no);

strcpy((p + 1)->name, p->name);

(p + 1)->score = p->score;

}

strcpy(q->no, no1);//插入的元素

strcpy(q->name, name1);

q->score = score1;

n++;

return true;

}

功能：给定一个学生信息，插入到表中指定的位置；

算法复杂度：O(n);

bool ListDelete(student\*& stu, int id, int& n)

{

if (id<1 || id>n) {

cout << "位置非法" << endl;

return false;

}

student\* p = &stu[id - 1];//删除的位置

student\* q = &stu[n - 1];//表尾

for (p++; p <= q; p++) {//依次向前一位

strcpy((p - 1)->no, p->no);

strcpy((p - 1)->name, p->name);

(p - 1)->score = p->score;

}

stu = (student\*)realloc(stu, (n - 1) \* sizeof(student));//重新申请空间

n--;

return true;

}

功能：删除指定位置的学生记录；

算法复杂度：O(n);

void show\_num(int n)

{

cout << "学生总人数为" << n << endl;

}

功能：统计表中学生个数；

算法复杂度：O(1);

bool Delete\_by\_info(student\*& stu, char name1[20], char no1[8], double score1, int& n)

{

student\* p = stu;//表头位置

student\* q = &stu[n - 1];//表尾位置

bool flag = false;//是否找到该学生

for (; p <= q; p++) {

if (!strcmp(p->name, name1) && !strcmp(p->no, no1) && p->score == score1) {//遍历查找

for (p++; p <= q; p++) { //被删除元素之后的元素都向前移一位

strcpy((p - 1)->no, p->no);

strcpy((p - 1)->name, p->name);

(p - 1)->score = p->score;

}

flag = true;

break;

}

}

if (flag) {

stu = (student\*)realloc(stu, (n - 1) \* sizeof(student));

n--;

return true;

}

else {

cout << "该学生信息错误" << endl;

return false;

}

}

功能：删除某学生的所有记录；

算法复杂度：O(n);

void ListUnique(student\* &stu,int& n)

{

student\* p = stu;

for (int i = 2; i <= n; i++) {//插入排序

for (int j = i; j > 1; j--) {

if (strcmp((p+j-1)->no,(p+j-2)->no)>0) {

break;

}

else {

swap((p + j - 1)->no, (p + j - 2)->no);

swap((p + j - 1)->name, (p + j - 2)->name);

double tmp = (p + j - 1)->score;

(p + j - 1)->score = (p + j - 2)->score;

(p + j - 2)->score = tmp;

}

}

}

int n1 = n;//去重过程中n将变化，先保存n

int samenum = 0;

for (int i = 1; i <= n1-1; i++) {

if (!strcmp((p + i - 1-samenum)->no, (p + i-samenum)->no)) {

student\* q = p + i - 1 - samenum;//删除元素后现在正在检查的元素的位置

student\* r = p + n1 - 1 - samenum;//删除元素后最后一个元素位置

for (q++; q <= r; q++) {//向前移动一位

strcpy((q - 1)->no, q->no);

strcpy((q - 1)->name, q->name);

(q - 1)->score = q->score;

}

n--;

samenum++;

}

}

stu = (student\*)realloc(stu, n \* sizeof(student));

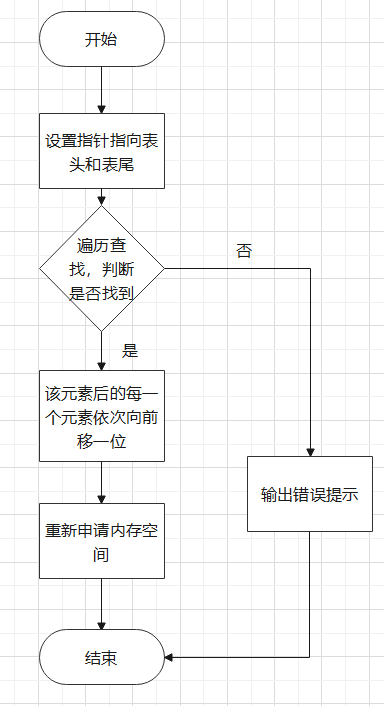
}

功能：删除所有重复记录；

算法复杂度：O(n^2);

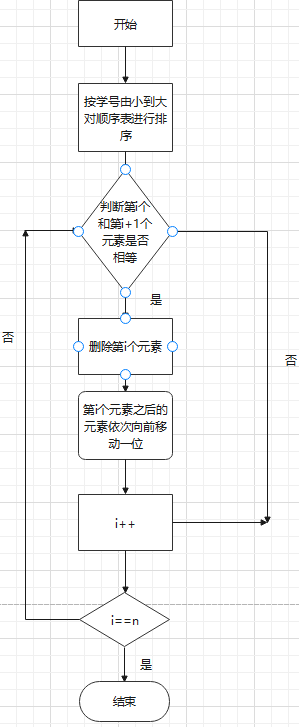
**1.4 流程图及复杂复分析**

删除某学生的所有记录：



这个算法首先从表头开始查找对应该信息的学生，若找到，停止查找并对其之后的所有元素向前移一位。若没找到，向后继续查找。不管找到没有，都对该表进行了一次遍历，所以算法复杂度为O(n)。

功能：删除所有重复记录；



首先对表中的元素按学号大小进行排序。这里采用插入排序。将一个元素插入到已经排好序的有序表中，从而一个新的、记录数增1的有序表。在其实现过程使用双层循环，外层循环对除了第一个元素之外的所有元素，内层循环对当前元素前面有序表进行待插入位置查找，并进行移动。对排好序的表进行遍历，若一个元素和它后一位的元素相等，则删去该元素，并将该元素之后的元素依次向前移动一位。因为插入排序使用二层循环，所以算法复杂度为O(n^2)。

**2. 实验总结**

这次实验中我完成了一个功能较为全面的学生信息管理系统，涉及元素的插入，删除，查找，去重等。通过这些代码的训练，我对于顺序表的性质，使用，实现有了更深入的了解。