# 数据结构实验报告——实验二

## 学号： 20201060330 姓名： 胡诚皓 得分：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 一、实验目的

1. 复习结构体、数组、指针；
2. 掌握数组的静态创建与动态创建；
3. 了解顺序存储的基本访问方法。

### 二、实验内容

1. （必做题）学生成绩信息存储

每个学生的成绩信息包括：学号、语文、数学、英语、总分、加权平均分；采用动态方法创建数组用于存储若干学生的成绩信息；输入学生的学号、语文、数学、英语成绩；计算学生的总分和加权平均分（语文占30%，数学占50%，英语占20%）；输出学生的成绩信息。

1. （必做题）追加和删除学生成绩信息

可以在数组末尾追加新学生的成绩信息；可以根据学号，删除该学生的成绩信息。

1. （选做题）对学生成绩信息进行排序

可以根据学号或总分，升序排序学生的成绩信息。

### 三、数据结构及算法描述

1. 学生成绩信息存储

使用结构体Stu来存储学生的成绩相关信息，其中long long id用于存储学生的学号，double chinese, math, english分别用来存储学生的语文成绩、数学成绩、英语成绩，double total用于存储学生的总分，double average用于存储学生的加权平均分。

为了避免命令行输入时回车键对输入判断的干扰，先读取将要输入的学生的个数，再读取每个学生的信息，在每次读入一个学生的成绩信息后，立刻计算该学生的总分与加权平均分存入。

1. 追加和删除学生成绩信息

在追加学生的成绩信息时，直接将新追加的学生信息放在动态数组的最后。由于需要先查找到要删除id的学生的位置，先用快速排序对Stu数组进行升序排序，再通过二分查找找到需要删除的学生的位置，然后再进行删除。

1. 对学生成绩信息进行排序

此处根据总分进行升序排序，使用的是优化的快速排序算法。

### 四、详细设计

1. 学生成绩信息存储

题1-1

min\_score流程图

1. 追加和删除学生成绩信息

1. 对学生成绩信息进行排序

### 五、程序代码

1. 学生成绩信息存储

（双击图标可以打开文件）

#include <stdio.h>

#include <stdlib.h>

s

typedef struct {

long long id;

double chinese, math, english;

double total, average;

} Stu;

int main() {

Stu \*students;

int count=0;

long long num;

double chi, math, eng;

int n;

students = (Stu \*) malloc(sizeof(Stu));

printf("How many students?\n");

scanf("%d", &n);

while (n--) {

scanf("%lld %lf %lf %lf", &num, &chi, &math, &eng);

(students+count)->id = num;

(students+count)->chinese = chi;

(students+count)->math = math;

(students+count)->english = eng;

(students+count)->total = chi + math + eng;

(students+count)->average = 0.3\*chi + 0.5\*math + 0.2\*eng;

count++;

students = (Stu \*) realloc(students, (count+1)\*sizeof(Stu));

}

printf("-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|\n");

for (int i = 0; i < count; i++) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", students[i].id, students[i].chinese, students[i].math, students[i].english, students[i].total, students[i].average);

}

free(students);

return 0;

}

1. 追加和删除学生成绩信息

（双击图标可以打开文件）

#include <stdio.h>

#include <stdlib.h>

//使用结构体来存储每个学生的学号及成绩信息

typedef struct {

long long id;

double chinese, math, english;

double total, average;

} Stu;

void displayStudents(Stu \*, int);

int readinStudents(Stu \*\*, int, int);

int cmp(const void \*, const void \*);

int deleteStudents(Stu \*\*, int);

int main() {

Stu \*students;

int count=0;

int n;

//申请动态数组的空间

students = (Stu \*) malloc(sizeof(Stu));

//先读取将要输入的学生人数

printf("How many students?\n");

scanf("%d", &n);

//使用封装好的函数读取数据

count = readinStudents(&students, count, n);

displayStudents(students, count);

//再读入一个学生的数据

printf("Add one student at the end of the array\n");

count = readinStudents(&students, count, 1);

displayStudents(students, count);

//使用封装好的函数删除指定id学生

printf("Delete one student by id\n");

count = deleteStudents(&students, count);

displayStudents(students, count);

//释放动态数组占用的空间

free(students);

return 0;

}

//规范格式更美观地输出

//给入需要输出的Stu数组首地址指针，和动态数组的元素个数count

void displayStudents(Stu \*students, int count) {

printf("-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|\n");

for (int i = 0; i < count; i++) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", students[i].id, students[i].chinese, students[i].math, students[i].english, students[i].total, students[i].average);

}

}

//封装读入学生数据

//此处为了使用realloc不断调整动态数组的空间大小，也就是需要修改指向Stu数组首地址的指针

//所以需要传入指向Stu数组首地址指针的指针

//还要传入动态数组的元素个数origin\_num及将要读入的学生的人数

int readinStudents(Stu \*\*students, int origin\_num, int n) {

printf("Input students\' information(id, Chinese score, Math score, English score)\n");

int count = origin\_num;

double chi, math, eng;

long long num;

while (n--) {

scanf("%lld %lf %lf %lf", &num, &chi, &math, &eng);

(\*students)[count].id = num;

(\*students+count)->chinese = chi;

(\*students+count)->math = math;

(\*students+count)->english = eng;

(\*students+count)->total = chi + math + eng;

(\*students+count)->average = 0.3\*chi + 0.5\*math + 0.2\*eng;

count++;

\*students = (Stu \*) realloc(\*students, (count+1)\*sizeof(Stu));

}

return count;

}

//封装从动态数组中删除学生的功能

//与readinStudents函数相同，由于需要修改指向Stu数组首地址的指针

//需要传入指向Stu数组首地址指针的指针

//还需要传入动态数组中的元素个数origin\_num

int deleteStudents(Stu \*\*students, int origin\_num) {

long long id;

int left, right, mid;

int new\_num=origin\_num;

qsort(\*students, origin\_num, sizeof(Stu), cmp);

scanf("%lld", &id);

left = 0;

right = origin\_num - 1;

while (left <= right) {

mid = (left + right)/2;

if ((\*students)[mid].id == id)

break;

else if ((\*students)[mid].id > id)

right = mid - 1;

else if ((\*students)[mid].id < id)

left = mid + 1;

}

if (left > right)

printf("id Not Found!\n");

else {

for (int i = mid; i < new\_num-1; i++)

students[i] = students[i+1];

new\_num--;

\*students = (Stu \*) realloc(\*students, (new\_num+1)\*sizeof(Stu));

}

printf("input id:\n");

return new\_num;

}

//用于qsort的比较函数

int cmp(const void \*a, const void \*b) {

const Stu \*first=(const Stu \*) a;

const Stu \*second=(const Stu \*) b;

return first->id - second->id;

}

1. 对学生成绩信息进行排序

（双击图标可以打开文件）

#include <stdio.h>

#include <stdlib.h>

//使用结构体来存储每个学生的学号及成绩信息

typedef struct {

long long id;

double chinese, math, english;

double total, average;

} Stu;

void displayStudents(Stu \*, int);

int readinStudents(Stu \*\*, int, int);

void sortByTotal(Stu \*, int, int);

int main() {

Stu \*students;

int count=0;

int n;

//申请动态数组的空间

students = (Stu \*) malloc(sizeof(Stu));

//先读取将要输入的学生人数

printf("How many students?\n");

scanf("%d", &n);

//使用封装好的函数读取数据

count = readinStudents(&students, count, n);

displayStudents(students, count);

//进行排序并输出

printf("After sorting:\n");

sortByTotal(students, 0, count-1);

displayStudents(students, count);

//释放动态数组占用的空间

free(students);

return 0;

}

//规范格式更美观地输出

//给入需要输出的Stu数组首地址指针，和动态数组的元素个数count

void displayStudents(Stu \*students, int count) {

printf("-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|\n");

for (int i = 0; i < count; i++) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", students[i].id, students[i].chinese, students[i].math, students[i].english, students[i].total, students[i].average);

}

}

//封装读入学生数据

//由于需要修改指向Stu数组首地址的指针，所以需要传入指向Stu数组首地址指针的指针

//还要传入动态数组的元素个数origin\_num及将要读入的学生的人数

int readinStudents(Stu \*\*students, int origin\_num, int n) {

printf("Input students\' information(id, Chinese score, Math score, English score)\n");

int count = origin\_num;

double chi, math, eng;

long long num;

while (n--) {

scanf("%lld %lf %lf %lf", &num, &chi, &math, &eng);

(\*students)[count].id = num;

(\*students+count)->chinese = chi;

(\*students+count)->math = math;

(\*students+count)->english = eng;

(\*students+count)->total = chi + math + eng;

(\*students+count)->average = 0.3\*chi + 0.5\*math + 0.2\*eng;

count++;

\*students = (Stu \*) realloc(\*students, (count+1)\*sizeof(Stu));

}

return count;

}

//使用优化的快速排序对Stu数组进行排序

//由于需要修改指向Stu数组首地址的指针，所以需要传入指向Stu数组首地址指针的指针

//left, right为当前进行处理的数组区间，且为闭区间

void sortByTotal(Stu \*students, int left, int right) {

int i=left, j=right;

double mid;

Stu tmp;

mid = students[(left+right)/2].total;

while (i <= j) {

while (students[i].total < mid) i++;

while (students[j].total > mid) j--;

if (i <= j) {

tmp = students[i];

students[i] = students[j];

students[j] = tmp;

i++;

j--;

}

}

if (j > left) sortByTotal(students, left, j);

if (i < right) sortByTotal(students, i, right);

}

### 六、测试和结果

1. 学生成绩信息存储
2. 追加和删除学生成绩信息
3. 对学生成绩信息进行排序

### 用户手册

1. 学生成绩信息存储

先输入想要录入的学生人数，再依次输入每个学生的信息，按顺序分别为学号、语文成绩、数学成绩、英语成绩。学号为不超过长整型大小的整数，成绩既可以以整数形式输入，也可以以小数形式输入。

1. 追加和删除学生成绩信息

学生的相关信息的录入的输入规则与第1题中相同。在显示第一次输入的学生的相关信息后，会提示

1. 对学生成绩信息进行排序

输入的n必须为正整数。