# 数据结构实验报告——实验三

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

### 一、实验目的

1. 复习结构体、指针；
2. 掌握链表的创建、遍历等操作；
3. 了解函数指针。

### 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；采用链表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

可以在链表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，struct Stu \*next用于指向下一个结点。

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。

为了方便输出，使用apply2All(StuList head, void (\*func)(Stu item))函数，使用函数指针传入想要对每个结点进行的void (\*)(Stu)类型的操作函数。

1. 追加和删除学生成绩信息

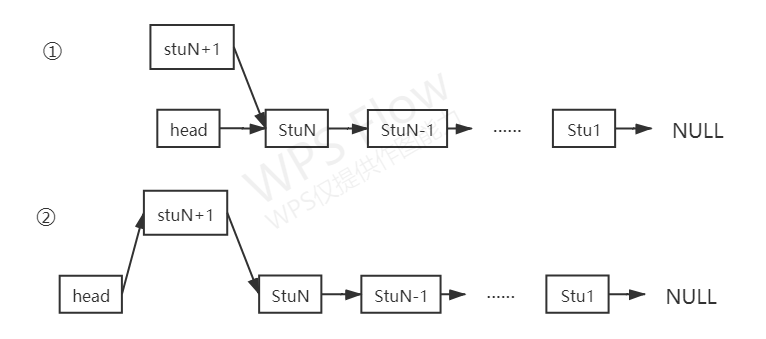
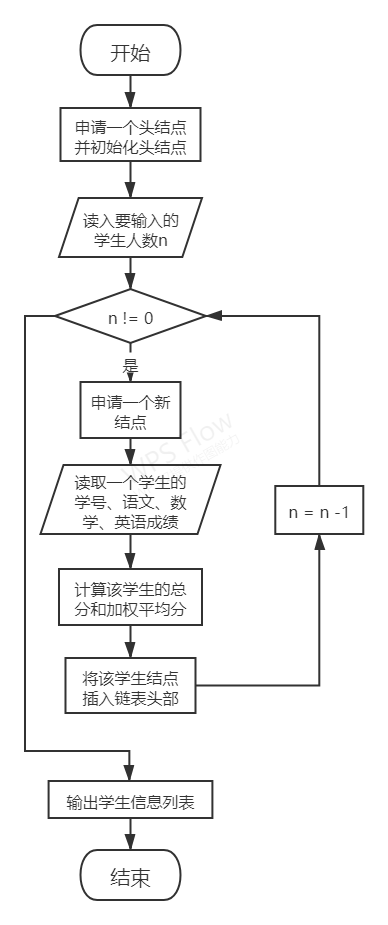
在追加学生的成绩信息时，使用头插法将新的Stu结点放在studentsList链表的头部，使头结点指向这个最新的Stu结点。对于单链表来说，由于只能顺序访问单链表的各个结点，所以使用顺序查找来查找到要删除id的学生的位置。若输入的id存在，从链表中删除该学生结点，然后会重新输出学生信息列表；若输入的id不存在，会提示“id Not Found!”

1. 对学生成绩信息进行排序

此处根据总分进行升序排序，由于单链表只能进行顺序访问，选择排序在排序中也是按顺序进行比较排序，因此使用选择排序对链表进行排序。

### 详细设计

1. 学生成绩信息存储



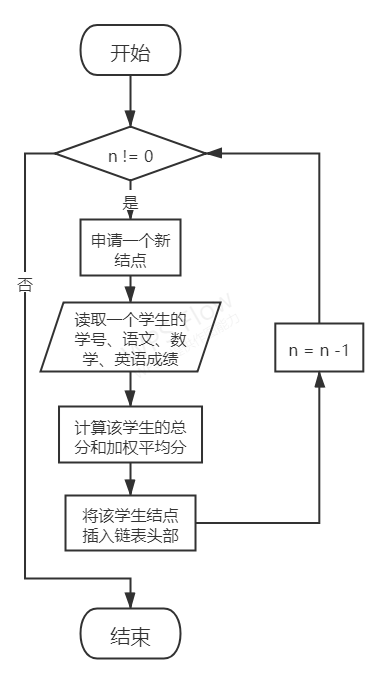
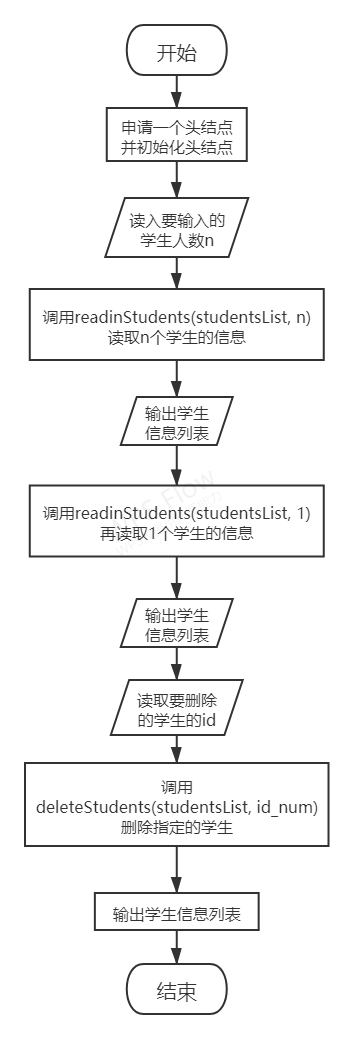
题3-1-1

主函数流程图

题3-1-2

insert函数示意图

1. 追加和删除学生成绩信息

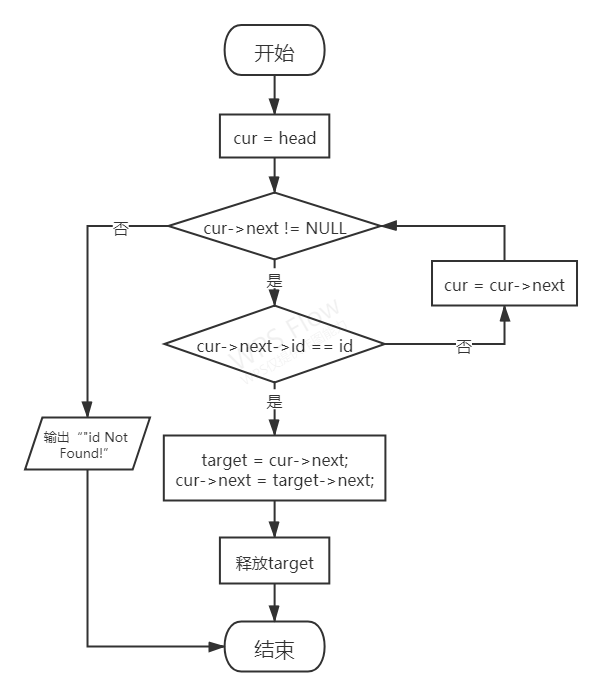


题3-2

readinStudents函数流程图

题3-2

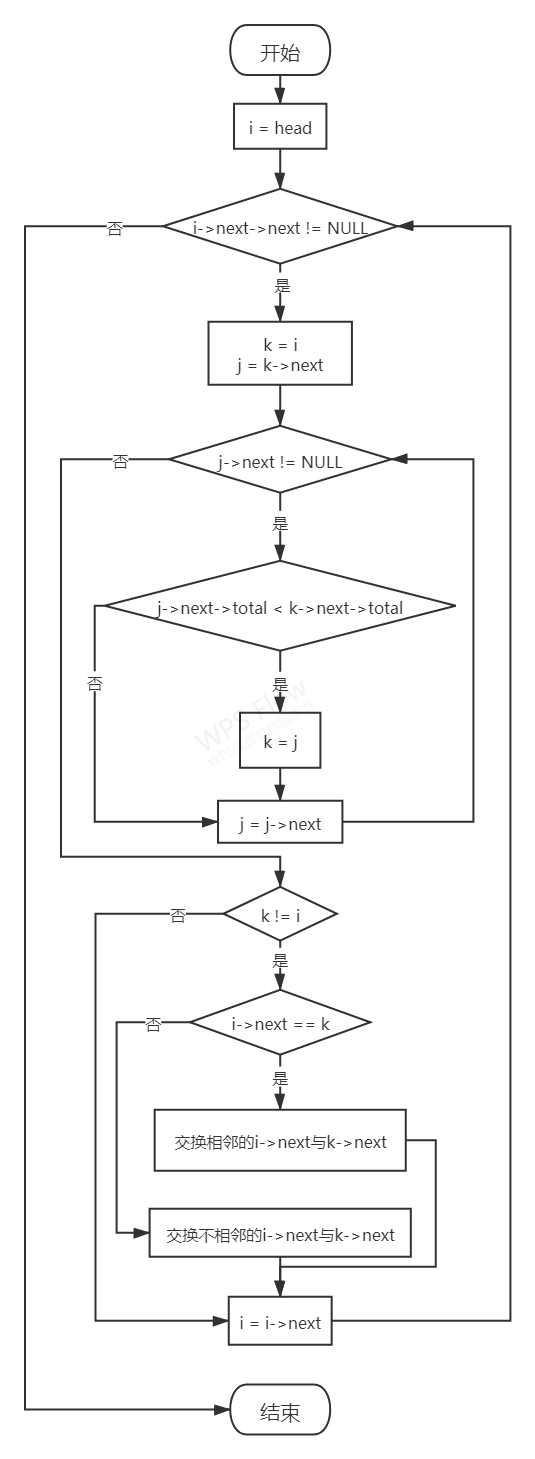
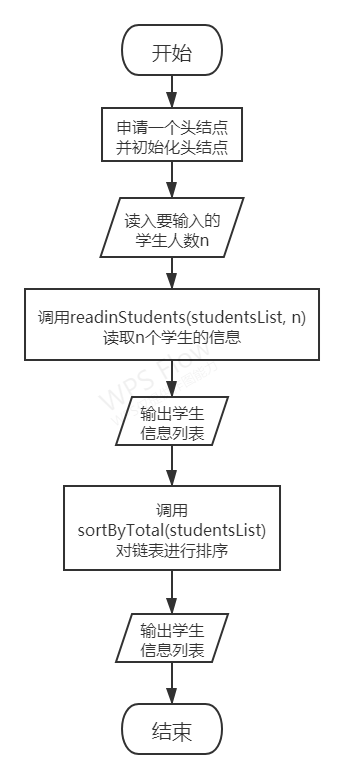
主函数流程图



题3-2

deleteStudents函数流程图

1. 对学生成绩信息进行排序



题2-3

sortByTotal函数流程图

题3-3

主函数流程图

### 五、程序代码

1. 学生成绩信息存储



#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

struct Stu {

long long id;

double chinese, math, english;

double total, average;

struct Stu \*next;

};

typedef struct Stu \*StuList;

typedef struct Stu Stu;

void insert(StuList, Stu \*);

void createList(StuList);

void apply2All(StuList, void (\*)(Stu));

void display(Stu stu);

int main() {

StuList studentsList;

long long num;

double chi, math, eng;

int n;

Stu \*tmp;

//创建一个用于存储Stu的链表并初始化

studentsList = (StuList) malloc(sizeof(Stu));

createList(studentsList);

//先读取将要输入的学生人数

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

scanf("%d", &n);

//读入每个学生的相关信息

while (n--) {

//创建新结点

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

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

tmp->id = num;

tmp->chinese = chi;

tmp->math = math;

tmp->english = eng;

tmp->total = chi + math + eng;

tmp->average = 0.3 \* chi + 0.5 \* math + 0.2 \* eng;

//每次读入之后就往链表中插入

insert(studentsList, tmp);

}

//输出表头

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

apply2All(studentsList, display);

system("pause");

return 0;

}

//头插法把结点插入链表

void insert(StuList head, Stu \*new\_student) {

new\_student->next = head->next;

head->next = new\_student;

}

//初始化链表

void createList(StuList head) {

head->next = NULL;

head->id = -1;

head->chinese = head->math = head->english = -1;

head->total = head->average = -1;

}

//对Stu链表中的每个元素应用func函数

void apply2All(StuList head, void (\*func)(Stu item)) {

Stu \*cur = head;

while (cur->next != NULL) {

cur = cur->next;

(\*func)(\*cur);

}

}

//规范格式化输出

void display(Stu stu) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", stu.id, stu.chinese, stu.math, stu.english, stu.total,

stu.average);

}

1. 追加和删除学生成绩信息



#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

struct Stu {

long long id;

double chinese, math, english;

double total, average;

struct Stu \*next;

};

typedef struct Stu \*StuList;

typedef struct Stu Stu;

void insert(StuList, Stu \*);

void createList(StuList);

void apply2All(StuList, void (\*)(Stu));

void display(Stu stu);

void readinStudents(StuList, int);

void deleteStudents(StuList, long long);

int main() {

StuList studentsList;

int n;

long long id\_num;

//创建一个用于存储Stu的链表并初始化

studentsList = (StuList) malloc(sizeof(Stu));

createList(studentsList);

//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

readinStudents(studentsList, n);

//输出表头

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

apply2All(studentsList, display);

//再读入一个学生的数据

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

readinStudents(studentsList, 1);

//输出表头

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

apply2All(studentsList, display);

//使用封装好的函数删除指定id学生

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

printf("input id:\n");

scanf("%lld", &id\_num);

deleteStudents(studentsList, id\_num);

//输出表头

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

apply2All(studentsList, display);

system("pause");

return 0;

}

//封装了往以head为头结点的列表中读入n个Stu结点的功能

void readinStudents(StuList head, int n) {

long long num;

double chi, math, eng;

Stu \*tmp;

StuList res;

while (n--) {

//创建新结点

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

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

tmp->id = num;

tmp->chinese = chi;

tmp->math = math;

tmp->english = eng;

tmp->total = chi + math + eng;

tmp->average = 0.3 \* chi + 0.5 \* math + 0.2 \* eng;

//每次读入之后就往链表中插入

insert(head, tmp);

}

}

//封装从链表中删除学生的功能

void deleteStudents(StuList head, long long id) {

Stu \*cur = head;

Stu \*target;

while (cur->next != NULL) {

if (cur->next->id == id) {

target = cur->next;

cur->next = target->next;

printf("Deleted\n");

free(target);

return;

}

cur = cur->next;

}

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

}

//头插法把结点插入链表

void insert(StuList head, Stu \*new\_student) {

new\_student->next = head->next;

head->next = new\_student;

}

//初始化链表

void createList(StuList head) {

head->next = NULL;

head->id = -1;

head->chinese = head->math = head->english = -1;

head->total = head->average = -1;

}

//对Stu链表中的每个元素应用func函数

void apply2All(StuList head, void (\*func)(Stu item)) {

Stu \*cur = head;

while (cur->next != NULL) {

cur = cur->next;

(\*func)(\*cur);

}

}

//规范格式化输出

void display(Stu stu) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", stu.id, stu.chinese, stu.math, stu.english, stu.total,

stu.average);

}

1. 对学生成绩信息进行排序



#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

struct Stu {

long long id;

double chinese, math, english;

double total, average;

struct Stu \*next;

};

typedef struct Stu \*StuList;

typedef struct Stu Stu;

void insert(StuList, Stu \*);

void createList(StuList);

void apply2All(StuList, void (\*)(Stu));

void display(Stu stu);

void readinStudents(StuList, int);

void sortByTotal(StuList);

Stu \*getCopy(Stu \*);

int main() {

StuList studentsList;

int n;

long long id\_num;

//创建一个用于存储Stu的链表并初始化

studentsList = (StuList) malloc(sizeof(Stu));

createList(studentsList);

//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

readinStudents(studentsList, n);

//输出表头

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

apply2All(studentsList, display);

//进行排序并输出

printf("\nAfter sorting:\n");

sortByTotal(studentsList);

//输出表头

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

apply2All(studentsList, display);

system("pause");

return 0;

}

//封装了往以head为头结点的列表中读入n个Stu结点的功能

void readinStudents(StuList head, int n) {

long long num;

double chi, math, eng;

Stu \*tmp;

StuList res;

while (n--) {

//创建新结点

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

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

tmp->id = num;

tmp->chinese = chi;

tmp->math = math;

tmp->english = eng;

tmp->total = chi + math + eng;

tmp->average = 0.3 \* chi + 0.5 \* math + 0.2 \* eng;

//每次读入之后就往链表中插入

insert(head, tmp);

}

}

//头插法把结点插入链表

void insert(StuList head, Stu \*new\_student) {

new\_student->next = head->next;

head->next = new\_student;

}

//初始化链表

void createList(StuList head) {

head->next = NULL;

head->id = -1;

head->chinese = head->math = head->english = -1;

head->total = head->average = -1;

}

//对Stu链表中的每个元素应用func函数

void apply2All(StuList head, void (\*func)(Stu item)) {

Stu \*cur = head;

while (cur->next != NULL) {

cur = cur->next;

(\*func)(\*cur);

}

}

//规范格式化输出

void display(Stu stu) {

printf("%12lld|%11.2f|%8.2f|%11.2f|%9.2f|%7.2f|\n", stu.id, stu.chinese, stu.math, stu.english, stu.total,

stu.average);

}

//使用选择排序思想进行排序

void sortByTotal(StuList head) {

Stu \*k, \*i = head, \*j;

Stu \*small, \*origin;

//i及之前的认为已经排好

while (i->next->next != NULL) {

k = i;

j = k->next;

while (j->next != NULL) {

if (j->next->total < k->next->total)

k = j;

j = j->next;

}

//找到比k->next总分更小的，就交换k->next和i->next

//相邻交换比较特殊，单独处理

if (k != i) {

if (i->next == k) {

i->next = k->next;

k->next = k->next->next;

i->next->next = k;

} else {

small = getCopy(k->next);

origin = getCopy(i->next);

small->next = origin->next;

origin->next = k->next->next;

free(i->next);

i->next = small;

free(k->next);

k->next = origin;

}

}

i = i->next;

}

}

//返回一个指向与\*source一模一样的Stu深拷贝的指针

Stu \*getCopy(Stu \*source) {

Stu \*res;

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

res->id = source->id;

res->chinese = source->chinese;

res->math = source->math;

res->english = source->english;

res->total = source->total;

res->average = source->average;

res->next = source->next;

return res;

}

### 六、测试和结果

1. 学生成绩信息存储

**Input:**

4

20201060001 86.5 98 90

20201060002 87.5 88 70.2

20201060003 90.5 87 88

20201060004 91 65 78

**Output:**

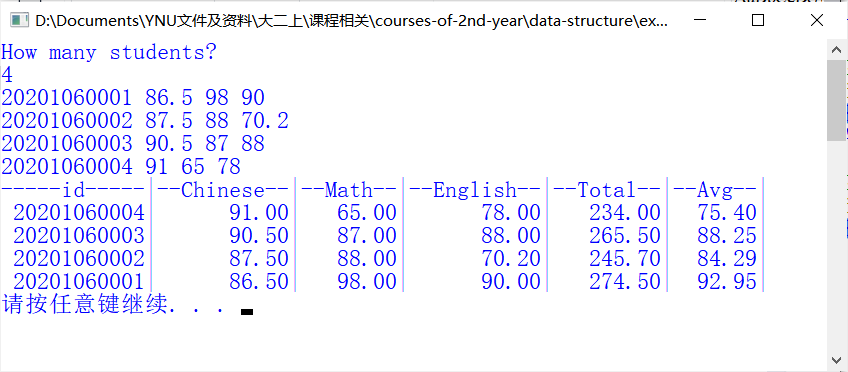
-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060002| 87.50| 88.00| 70.20| 245.70| 84.29|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|



1. 追加和删除学生成绩信息

**Input:**

4

20201060001 86.5 98 90

20201060002 87.5 88 70.2

20201060003 90.5 87 88

20201060004 91 65 78

20201060005 90 87 60

20201060002

**Output:**

-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060002| 87.50| 88.00| 70.20| 245.70| 84.29|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|

-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060005| 90.00| 87.00| 60.00| 237.00| 82.50|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060002| 87.50| 88.00| 70.20| 245.70| 84.29|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|

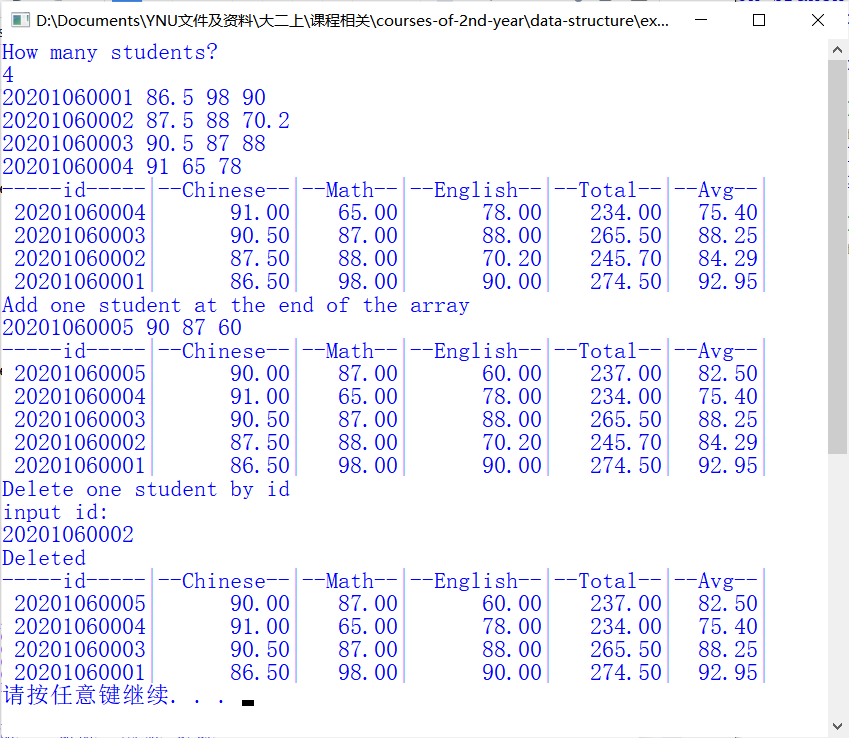
-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060005| 90.00| 87.00| 60.00| 237.00| 82.50|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|



1. 对学生成绩信息进行排序

**Input:**

4

20201060001 86.5 98 90

20201060002 87.5 88 70.2

20201060003 90.5 87 88

20201060004 91 65 78

**Output:**

-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060002| 87.50| 88.00| 70.20| 245.70| 84.29|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|

After sorting:

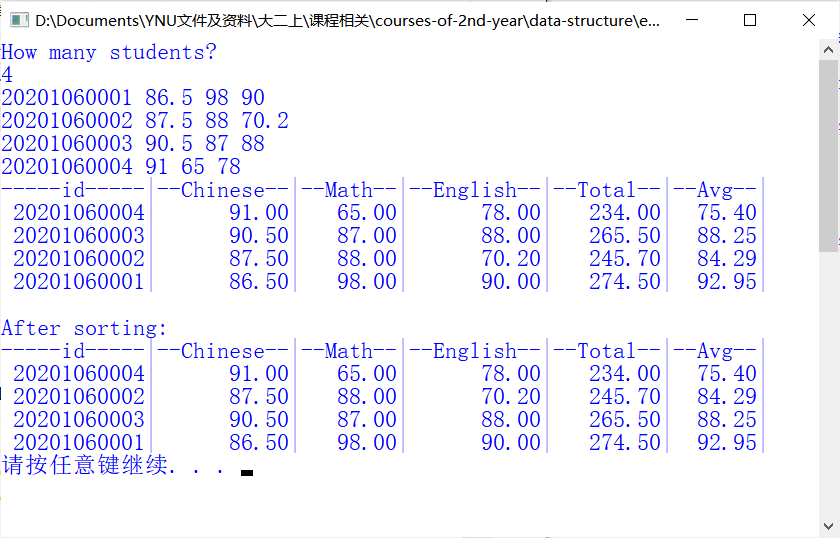
-----id-----|--Chinese--|--Math--|--English--|--Total--|--Avg--|

20201060004| 91.00| 65.00| 78.00| 234.00| 75.40|

20201060002| 87.50| 88.00| 70.20| 245.70| 84.29|

20201060003| 90.50| 87.00| 88.00| 265.50| 88.25|

20201060001| 86.50| 98.00| 90.00| 274.50| 92.95|



### 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，会要求输入要添加的学生的相关信息，输入格式与之前相同。回车确认后，会显示添加了新的学生之后的学生信息列表。接着会要求输入想要删除的学生的id，若id存在，删除该学生后会重新输出学生信息列表；若输入的id不存在，会提示“id Not Found!”

1. 对学生成绩信息进行排序

学生的相关信息的录入的输入规则与第1题中相同。输入完成后，会依次输出排序前和排序后的学生信息列表。