|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构》实验报告** |
|  |
| 实验一  线性结构及其应用  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 朱海峰 | | 学 号: | 190110716 | | 专 业: | 计算机科学与技术 | | 日 期: | 2020-04-04 | |

# 问题分析

问题：

学期结束，辅导员需要收集两个班级的同学的 C 语言课程成绩信息并整理。

请你为辅导员做一个成绩录入统计，帮助辅导员更好地工作

（1）成绩信息录入——线性表的建立

操作 1：初始化空表；

操作 2：按成绩降序插入所有同学们的学号班级信息和成绩到各自班级

链表；

操作 3：查找/删除（按学号），如果对应班级链表中无该同学信息，则

输出无法找到

操作 4：信息输出——线性表的遍历

（2）信息统计——将同学们的成绩合并到总表；

（3）将总表中信息翻转到新表中

分析：

显然，这是一个线性表结构的应用练习，涉及到以下几个方面：

一、初始化空表、降序插入节点： 分配内存，写入数据，插入节点；

二、查找信息、删除节点、信息输出： 顺序查找，删除节点，线性遍历；

三、合并到总表、翻转到新表： 归并排序，反转链表。

# 二、详细设计

## 2.1 设计思想

用线性表存储数据，通过节点的新增、删除、遍历、重排序等操作，完成插入、删除、打印、归并、反转等要求。

## 2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）

实验的存储结构是带头结点的单链表。

两个班级用一个含两个指针元素的数组作为头节点，单个节点是含有用来存储成绩的整型、存储学号的整型和指向下一个节点的指针的结构体。其中节点按成绩降序排列。此外，另有两个指针，一个指向合并后总表头节点，一个总表反转后的头节点。

(2) 涉及的操作（一般为自定义函数，可不写过程，但要注明该函数的含义。）

StudentLinkedListNode\* studentLinkedListAdd(StudentLinkedListNode\* head, StudentLinkedListNode\* node)

/\*\* 按照降序插入学生的成绩情况,并返回链表头指针 \*/

//若head==NULL，则为head分配内存，使其指向头节点，即进行初始化；

//若head!=NULL，则按降序插入已经分配内存，已经写入数据的节点node。

//最后返回头指针（因为表头存在，头指针不改变）。

void searchByID(StudentLinkedListNode\*head ,int id)

/\*\* 根据学号搜索某个学生的成绩情况 \*/

StudentLinkedListNode\* deleteByID(StudentLinkedListNode\* head,int id)

/\*\* 根据学号删除某个学生的成绩情况 \*/

//删除中包含有查找，从head开始，顺序查找id——

//若找到，查找则打印学号，删除则删除节点释放空间；

//若没找到，则打印该节点不存在。

void outputStudentLinkedList(StudentLinkedListNode\* head)

//遍历打印

StudentLinkedListNode\* mergeLinkedList(StudentLinkedListNode\* heads[])

/\*\* 合并两个班级的成绩情况 \*/

//从head[0]和head[1]两个头指针开始，进行归并，成绩数值更大的就先排，再比较下一个。当一个表排完时，直接将剩下数据排入总表。

StudentLinkedListNode\* studentLinkedListCopy(StudentLinkedListNode\* src)

/\*\* copy一个已有的node \*/

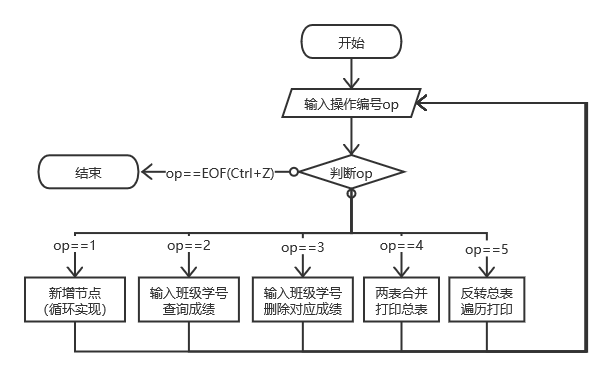
//将src指针对应节点数据复制到新分配内存的节点，返回新的节点。

StudentLinkedListNode\* reverseLinkedList(StudentLinkedListNode\*head)

//将head对应表反转，返回反转后的头指针。通过一次将原表每个简单插入到头节点来实现。

## 2.3 程序整体流程

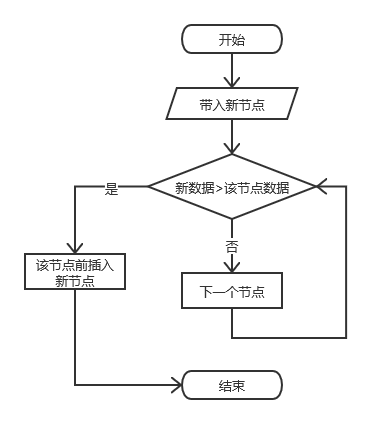
整体流程：



总流程

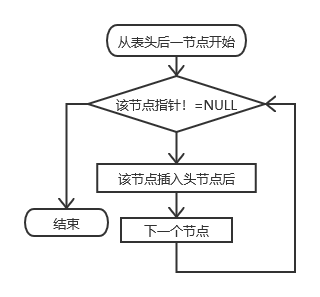
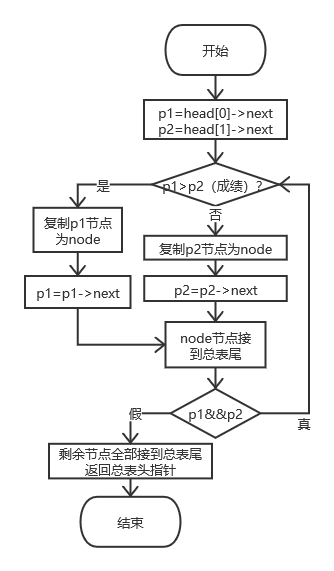
核心算法流程：

1. 降序插入：



降序插入

1. 合并总表：
2. 翻转总表：



合并总表 翻转总表

# 三、用户手册

（英文均为程序提示）

1.insert 2.search by StuID 3.delete by StuID 4.merge 5.reverse 6.output

输入数字1插入数据（空表则先初始化）；

输入数字2按班级和学号查询学生的成绩；

输入数字3按班级和学号查找节点并删除；

输入数字4合并两班数据为总表并打印；

输入数字5反转总表并打印；

输入数字6分别打印两班数据。

注意：一般来说，首先要初始化空表插入数据，然后可进行查询、删除、合并和打印操作，翻转总表也需要再合并后进行。

1. 插入数据：

How many rows of data do you need to input?

输入一个大于零的整数，如10；

input the i row data format as:class\_id,student\_id,grade

按照a,bbbbbbbbb,cc的格式输入班级、学号、成绩，a=0或1，如：

1,190010010,83

0,190010012,85

1,190010020,86

1,190010015,87

1,190010023,88

0,190010013,89

0,190010022,90

1,190010017,91

1,190010033,92

0,191151115,93

1. 查询成绩：

input the data format as: ClassID, StuID

格式相似，按照提示输入班级、学号，如：

0,190010010

即可查询，若无此人也有提示

1. 删除节点

提示与（2）相同，输入格式也相同，即可删除，若无此人也有提示。

1. 合并总表

系统自动合并并打印。

1. 翻转总表

系统自动翻转并打印。

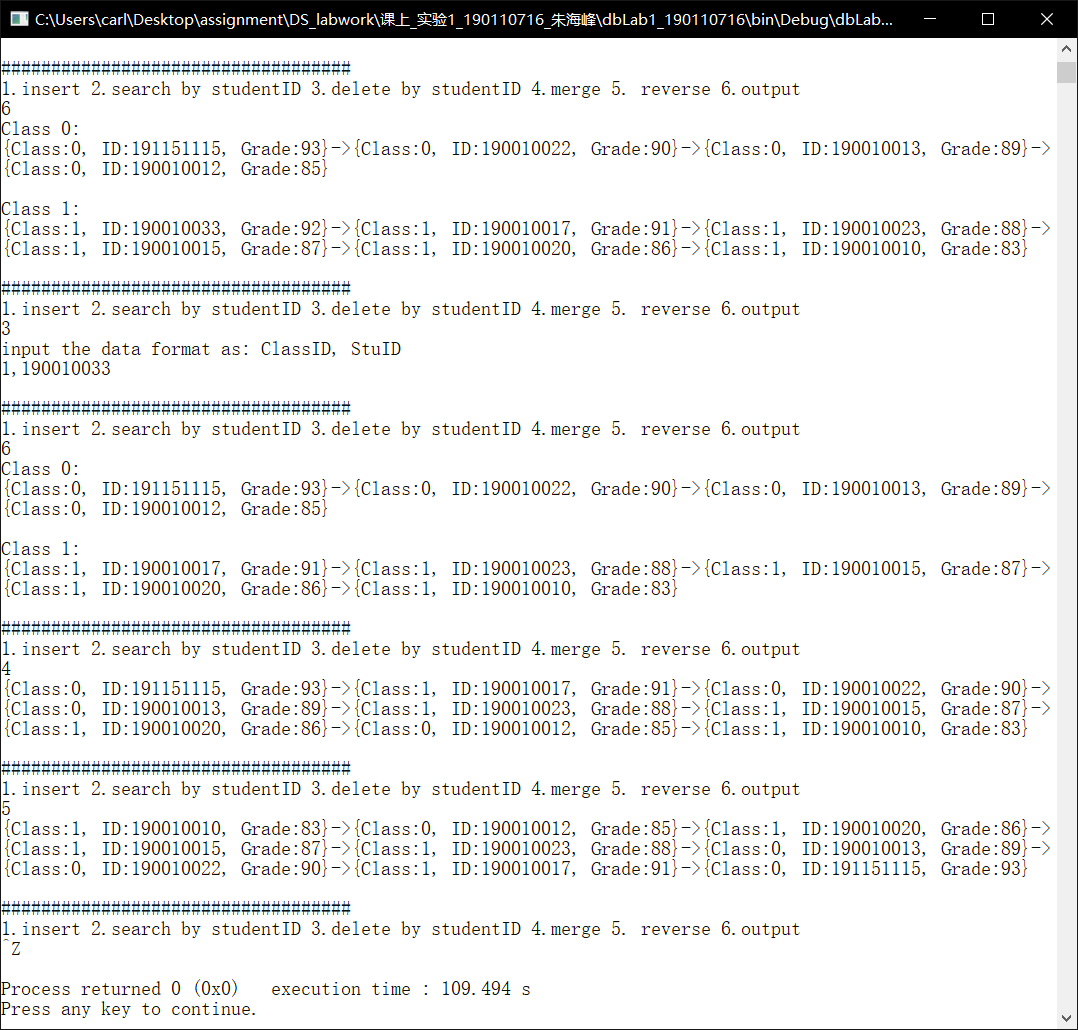
1. 输出数据

系统分别打印输出两班数据。

# 四、结果

程序正确运行的结果截图。





# 五、总结

实验中用到了带表头的单链表，单个节点存储一个学生的数据。算法有按顺序插入节点、顺序查找、删除节点、归并排序、翻转链表和链表遍历等。

第一次实验，也有很多问题，不一而足：

心里素质问题。上课就要求交一部分，时间限制导致紧张，这方面还是需要提前复习学习过的内容，多打代码，多调试，有所准备自然可以克服。

具体技术问题。插入节点的部分调试了很多次，有的地方思路还不够严密，特别是边界情况要多考虑：指针一定初始化为NULL，头节点是否存在、是否要新建也要考虑等。

当然收获也不少，第一次用有模板的来写，不像以前都是按自己的写，这是一个“标准化”的过程，自己初学，算法实现的格式不一定有老师的规范。如何在“规范”的模板基础上，完善每个函数的内容，这是以前所没有接触到的。相信这对以后实际应用的作用也很大。