|  |
| --- |
| 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: | 段佳慧 | | 学 号: | 200110807 | | 专 业: | 计算机科学与技术 | | 日 期: | 2021-04-02 | |

# 问题分析

1. 建立两个非空的线性表，从文件中读取数据元素（班级、学习成绩、学号）之后计算机将其存储在两个线性链表中，并根据头插法按照成绩降序排列。
2. 计算机根据过程1产生的两个单链表进行链表反转操作，即将单列表的数据顺序进行反转。例如输入一个列表的头结点，反转该列表并输出反转后的头结点。
3. 在两个由各班成绩组成的线性链表中，找到链表交点即相同结点。此时应按照地址访问，遍历两个链表最终得到两个链表的第一个共同交点。
4. 遍历链表，按顺序打印所欲数据元素

# 二、详细设计

## 2.1 设计思想

1. 链表的每个指针指向一个结构体，结构体包括学生的学号和成绩。依次取文件中的数据元素，按照班级分别存储到两个链表中，通过分组逐个比较实现按成绩降序排列。

2. 通过遍历来输出每个链表的数据元素

3. 使用头插法将链表中的数据元素逆置

4. 通过改变链表中数据元素的指针指向来反转链表

5. 通过遍历每个链表来逐个比较两个链表中每个数据元素的地址来判断相同结点的位置

## 2.2 存储结构及操作

(1) 存储结构 存储结构如下：

typedef struct node{

char StuID[11];

int Grade;

struct node \*next;

} StudentLinkedListNode;

(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。）

**1、/\* 创建相交链表 \*/**

Void createCrossLink(StudentLinkedListNode\*a, StudentLinkedListNode\* b, int beforeCross1, int beforeCross2)

//该函数为创建相交链表函数a和b后面若干结点值相同，beforeCross1为跳过的a中的个数，从第beforeCross1 + 1个结点开始相交，beforeCross2为跳过的b中的个数，从第beforeCross2 + 1个结点开始相交，相交方法是将b中的前一结点指向a中的首个相交结点。

**2、/\*销毁相交链表\*/**

void destroyCrossLink(StudentLinkedListNode\* a,StudentLinkedListNode\* b, StudentLinkedListNode\* crossNode)

**3、/\*打印单个节点 \*/**

void printLinkedListNode(StudentLinkedListNode \* node)

//通过遍历 逐个打印单个结点

**4、/\*输出该表的成绩情况 \*/**

void outputStudentLinkedList(StudentLinkedListNode\* head)

**5、/\*\*新建一个链表node并返回 \*/**

StudentLinkedListNode\* studentLinkedListCreate(char student\_id[], int grade)

**6、/\*按照降序插入学生的成绩情况,并返回链表头指针 \*/**

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

**7、/\*反转链表 \*/**

StudentLinkedListNode\*reverseLinkedList(StudentLinkedListNode\*head)

/\*指向关系为：pPre->pCur->pNext \*/

//通过遍历两个链表，改变链表的指针指向 使得指向顺序发生变化

**8、/\* 找到相交的第一个结点 \*/**

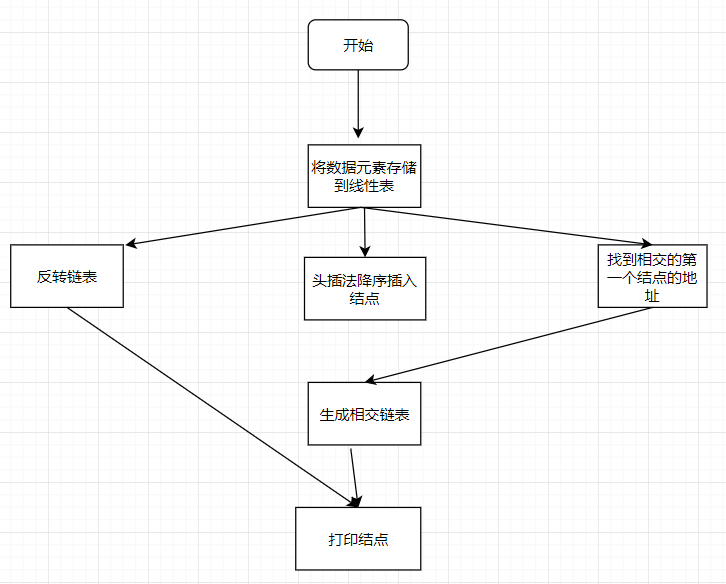
StudentLinkedListNode\* findCrossBeginNode(StudentLinkedListNode\* class1, StudentLinkedListNode\* class2)

// class1和class2在后一部分完全重合（结点的内存地址相同），找出并返回开始相交的第一个结点。

## 2.3 程序整体流程

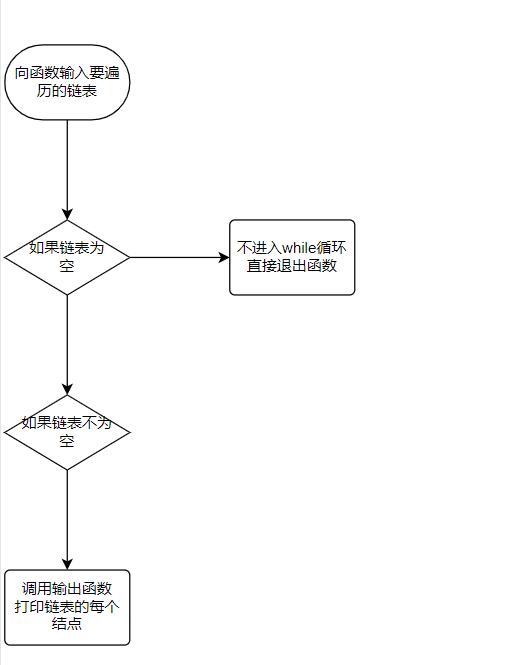
画出整体流程，及核心算法流程。

整体流程

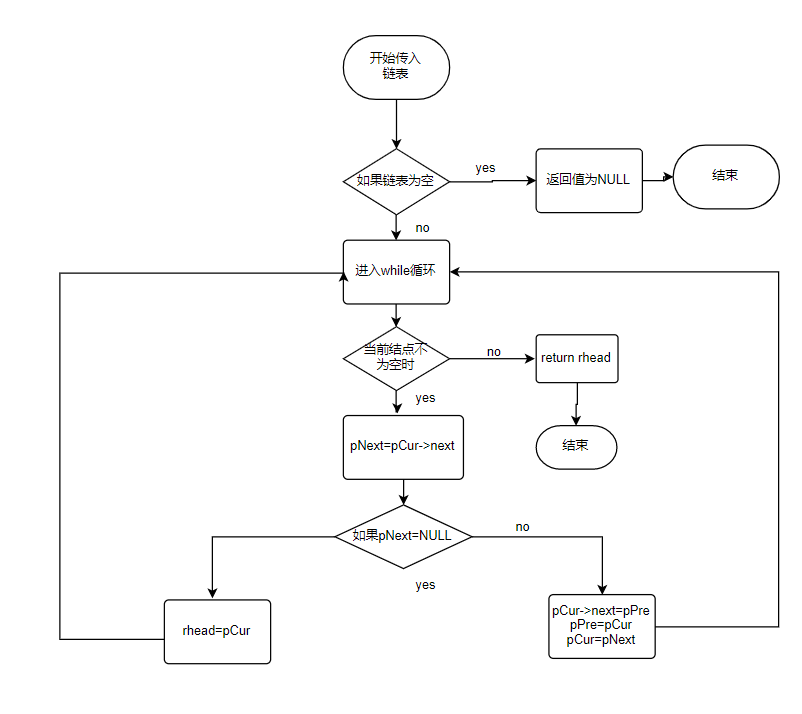


核心算法流程

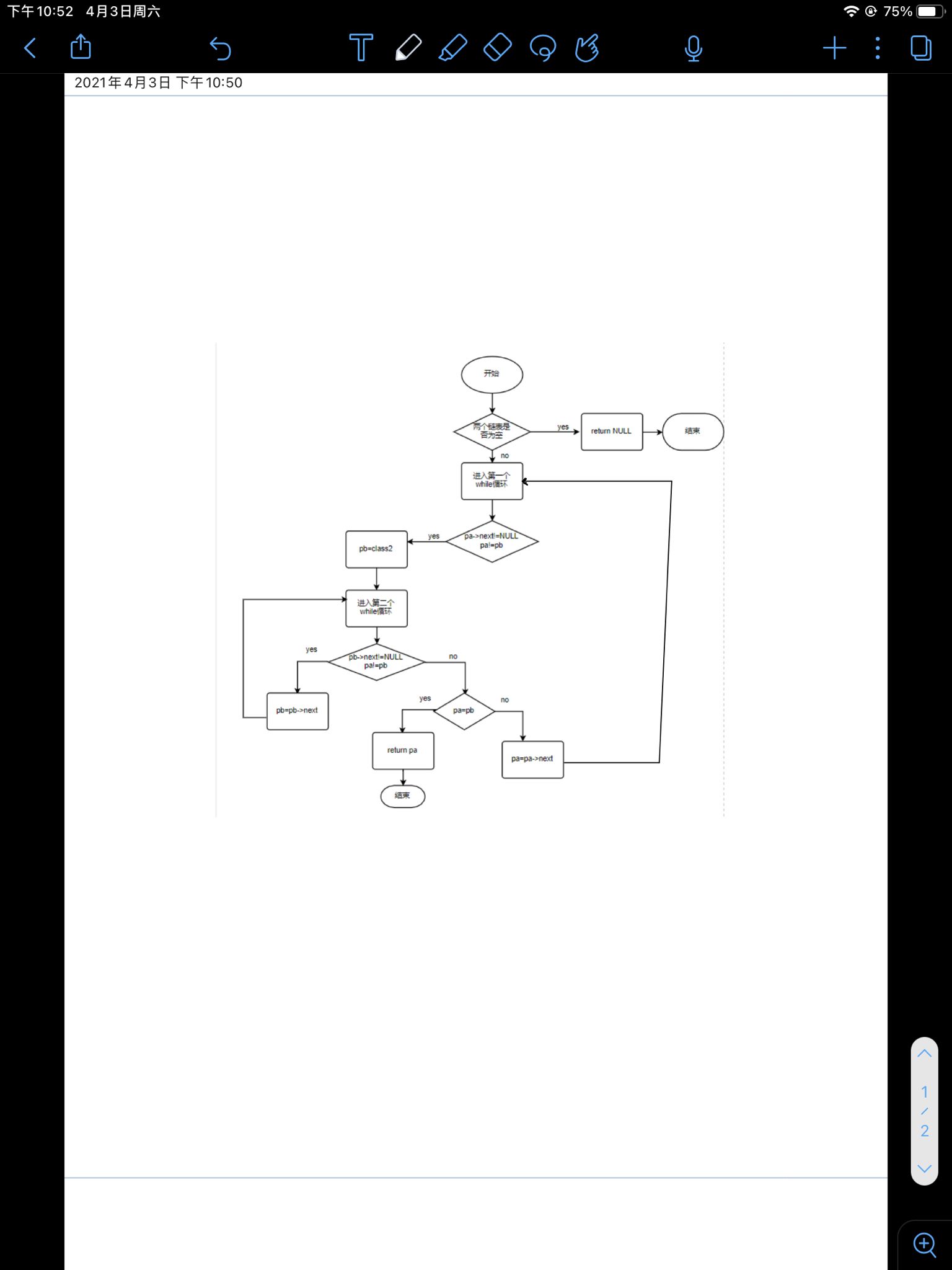
（1）遍历



（2）反转链表



（3）找到两个链表第一个相同结点

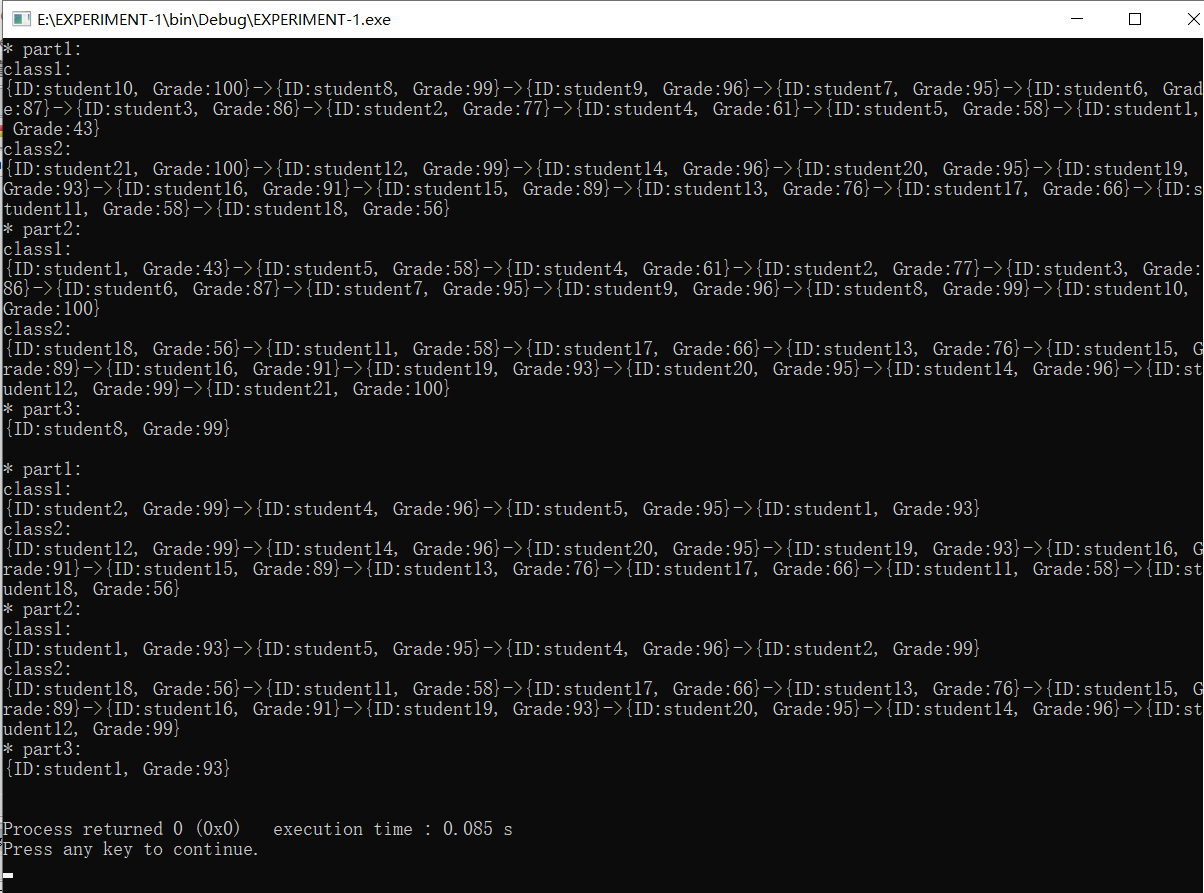


# 三、用户手册

如：(1)输入数据的方式；(2)实现各种功能的操作方式等。

1. 计算机直接从目标文件中读取相关数据信息
2. 用户直接运行整个代码即可得到反转链表与合并后的链表

# 四、结果



# 五、总结

该实验涉及到的数据结构和算法，以及遇到的问题和收获。

本此实验涉及的数据结构为线性链表。

主要的算法为（1）新建链表、（2）遍历输出链表、（3）按照降序使用头插法插入新结点、（4）反转链表以及找到（5）两个链表相交的第一个结点。

（1）新建链表

StudentLinkedListNode\* studentLinkedListCreate(char student\_id[], int grade){

StudentLinkedListNode\* p=NULL;

p=(StudentLinkedListNode\*)malloc(sizeof(struct node));

strcpy(p->StuID,student\_id);

p->Grade=grade;

p->next=NULL;

return p;

}

（2）遍历输出链表

void outputStudentLinkedList(StudentLinkedListNode\* head){

StudentLinkedListNode \*p;

p=head;/

while(p!=NULL){

printLinkedListNode(p);

p=p->next;

}

}

1. 按照降序使用头插法插入新结点

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

if (node == NULL){

return head;

}

node->next = head;

return node;

}

1. 反转链表

StudentLinkedListNode\* reverseLinkedList(StudentLinkedListNode\*head){

if(head==NULL)

return NULL;

/\*\*指向关系为：pPre->pCur->pNext \*/

StudentLinkedListNode \*pPre=NULL;

StudentLinkedListNode \*pCur=head;

StudentLinkedListNode \*rhead=NULL;

while(pCur!=NULL){

StudentLinkedListNode \*pNext=pCur->next;

if(pNext==NULL) {

rhead=pCur;

}

pCur->next=pPre;

pPre=pCur;

pCur=pNext;

}

return rhead;

}

1. 找到相交的第一个结点

StudentLinkedListNode\* findCrossBeginNode(StudentLinkedListNode\* class1, StudentLinkedListNode\* class2){

if(class1==NULL||class2==NULL) {

return NULL;

}

StudentLinkedListNode \*pa=class1;

StudentLinkedListNode \*pb=class2;

while(pa->next!=NULL&&pa!=pb){

pb=class2;

while(pb->next!=NULL&&pa!=pb) {

pb=pb->next;

}

if(pa==pb){

return pa;

}

pa=pa->next;

}

if(pa==pb)

return pa;

}

在实验过程中遇到了几个问题，首先就是一开始对于线性表的理解不够导致刚上手写代码很多地方模棱两可，需要不停地翻看PPT和书本。其次在完成函数StudentLinkedListNode\* findCrossBeginNode 部分时 由于一开始没有想到正确的算法导致花费了很多时间，没有理解到位相交的第一个结点的有关概念。最后就是在写代码的过程中，很多时候自己有一些想法但是因为害怕出错总是犹犹豫豫没有上机实验造成时间浪费的很多。

总的来说，这次实验使我对线性链表的操作更加熟练，也使我更深刻地认识了这种数据结构在生活中的作用，在以后的学习过程中，我要多多注意代码规范，追求简洁、细致的编程方式，并且要考虑边界与特殊情况，力求更进一步。