电子科技大学信息与软件工程学院

**实 验 报 告**

学 号

姓 名

（实验） 课程名称 数据结构与算法

理论教师 郝宗波

实验教师 李美蓉

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告（1）**

**学生姓名：**  **学 号：**  **指导教师： 郝宗波**

**实验地点：**清水河科技实验大楼 **实验时间：2017.04.15**

**一、实验室名称：**学校实验中心软件实验室

**二、实验项目名称：**编程实现线性表的合并

**三、实验学时：**4

**四、实验原理：**

在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系是由指针域来确定的。链式存储方式即可以用于表示线性结构，也可用于表示非线性结构。一般来说，在[线性表](http://baike.soso.com/v105830.htm?ch=ch.bk.innerlink)的链式存储结构中，各数据结点的存储符号是不连续的，并且各结点在存储空间中的位置关系与逻辑关系也不一致。对于线性链表，可以从头指针开始，沿各结点的指针扫描到链表中的所有结点。

线性表的链接存储中，为了方便在表头插入和删除结点的操作，经常在表头结点（存储第一个元素的结点）的前面增加一个结点，称之为头结点或表头附加结点。这样原来的表头指针由指向第一个元素的结点改为指向头结点，头结点的数据域为空，头结点的指针域指向第一个元素的结点。

**五、实验目的：**

本实验通过定义单向链表的数据结构，设计创建链表、插入结点、遍历结点等基本算法，使学生掌握线性链表的基本特征和算法，并能熟练编写C程序，培养理论联系实际和自主学习的能力，提高程序设计水平。

**六、实验内容：**

使用数据结构typedef struct node {

Elemtype data；

struct node \*next；

} ListNode, \*ListPtr；

typedef struct stuInfo {

int stuID；

char stuName[10]; /\*学生姓名\*/

int Age /\*年龄\*/

} ElemType;

实现带头结点的单向链表的创建、删除链表、插入结点等操作，可每个学生的学号互不相同，学号不同而姓名相同则为不同的学生，每个学生的学号在合并后的链表中不重复，如果出现重复，则删除年龄较小结点。最后打印输出合并后的链表元素，验证结果的正确性。

（1）设计学生信息结点的数据结构；

（2）用C语言实现创建升序链表的函数，每个结点的学号不同，按照学号升序排列；

（3）用C语言实现结点的插入的函数，插入后仍然为升序；

（4）编程实现两个单向链表合并，合并后仍然升序；

（5）编程实现合并后链表逆序排列的算法；

（6）打印输出合并后的链表元素。

**七、实验器材（设备、元器件）：**

PC机一台，装有C语言集成开发环境。

**八、数据结构与程序：**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 20

typedef struct stuInfo {

    int   stuID;

    char  stuName[10];      /\*学生姓名\*/

    int    Age; /\*年龄\*/

} ElemType;

typedef struct node {

    ElemType  data;

    struct  node  \*next;

} ListNode, \*ListPtr;

void LinkListInit(ListPtr\* L){

    \*L = (ListPtr) malloc(sizeof(ElemType));

    if (\*L == NULL) {

        printf("申请空间分配失败");

    }

    (\*L) -> next = NULL;

}

void LinkListInit(ListPtr\* L){

    \*L = (ListPtr) malloc(sizeof(ElemType));

    if (\*L == NULL) {

        printf("申请空间分配失败");

    }

    (\*L) -> next = NULL;

}

void Bubble(ListPtr\* L){

    int i,j;

    ListPtr p = \*L;

    ElemType temp;

    int count = 0;

    while(p -> next != NULL)

    {

        count++;

        p = p->next;

    }

    for(i=0;i<count-1;i++) /\* 冒泡法排序 \*/

    {

        p = (\*L) -> next;

        for(j=0;j<count-i-1;j++)

        {

            if(p->data.stuID > p->next->data.stuID)

            {

                temp = p->data;

                p->data = p->next->data;

                p->next->data=temp;

            }

            p = p->next;

        }

    }

void LinkedListCreate(ListPtr\* L, ElemType a[], int n){

    ListPtr p;

    LinkListInit(L);

    for (int i = 0; i < n; i++){

        p = (ListPtr) malloc(sizeof(ListNode));

        if (p == NULL){

            printf("分配空间失败");

        }

        p -> data = a[i];

        p -> next = (\*L) -> next;

        (\*L) -> next = p;

    }

    Bubble(L);

}

}

void DelRepetition(ListPtr\* L){

        ListPtr p, pre;

        pre = (\*L)->next;

        while ((pre->next != NULL) && ((pre->data.stuID) != (pre->next->data.stuID))) {

            pre = pre -> next;

        }

        p = pre -> next;

        if (p != NULL){

            if (p->data.Age < pre->data.Age){

                pre -> next = p -> next;

            }

            else {

                pre->data = p->data;

                pre -> next = p -> next;

            }

           free(p);

        }

}

void LinkedListMerge(ListPtr\* La, ListPtr\* Lb, ListPtr\* Lc){

    ListPtr pa, pb, pc;

    pa = (\*La) -> next;

    pb = (\*Lb) -> next;

    (\*Lc) = (\*La);

    pc = \*Lc;

    while ((pa != NULL) && (pb != NULL)) {

        if (pa -> data.stuID <= pb -> data.stuID){

            pc -> next = pa;

            pc = pa;

            pa = pa -> next;

        }

        else {

            pc -> next = pb;

            pc = pb;

            pb = pb -> next;

        }

    }

    if (pa != NULL){

        pc -> next = pa;

    }

    else {

        pc -> next = pb;

    }

    DelRepetition(Lc);

    free(\*Lb);

    free(\*La);

}

void LinkedListTraverse(ListPtr\* L){

    ListPtr p;

    p = (\*L)->next;

    while(p != NULL){

        printf("%d  %s  %d\n", p->data.stuID, p->data.stuName, p->data.Age);

        p = p->next;

    }

    printf("\n");

}

void LinkedListReverseSort(ListPtr\* La, ListPtr\* Lb){

    ListPtr pa = (\*La)->next, pb = (\*Lb)->next, temp = NULL;

    (\*La) -> next = NULL;

    while (pa != NULL && pb != NULL) {

        if ((pa ->data.stuID) <= (pb->data.stuID)){

            temp = pa->next;

            pa->next = (\*La)->next;

            (\*La)->next = pa;

            pa = temp;

        }

        else {

            temp = pb->next;

            pb->next = (\*La)->next;

            (\*La)->next = pb;

            pb = temp;

        }

    }

    while (pa != NULL){

        temp = pa -> next;

        pa->next = (\*La)->next;

        (\*La)->next = pa;

        pa = temp;

    }

    while (pb != NULL){

        temp = pb->next;

        pb->next = (\*La)->next;

        (\*La)->next = pb;

        pb = temp;

    }

    DelRepetition(La);

}

void LinkedListTraverse(ListPtr\* L){

    ListPtr p;

    p = (\*L)->next;

    while(p != NULL){

        printf("%d  %s  %d\n", p->data.stuID, p->data.stuName, p->data.Age);

        p = p->next;

    }

    printf("\n");

}

int main() {

    ListPtr La, Lb, Lc;

    int i;

    ElemType stu[MAX\_SIZE],elem;

    FILE \*fp, \*fp2;

fp = fopen("/Users/ryshen/Desktop/la.txt","r");

    if(!fp)

    {

        printf("The file does not exist!\n");

        return 0;

    }

    for(i = 0; !feof(fp); i++)

    {

        fscanf(fp, "%d %s %d", &stu[i].stuID, stu[i].stuName, &stu[i].Age);

    }

    fclose(fp);

    LinkedListCreate(&La, stu, i);

    printf("la:\n");

    LinkedListTraverse(&La);

    printf("输入插入数据\n");

    scanf("%d %s %d", &elem.stuID, elem.stuName, &elem.Age);

    LinkedListInsert(&La, elem, La->next);

    printf("insert\n");

    LinkedListTraverse(&La);

    fp2 = fopen("/Users/ryshen/Desktop/lb.txt","r");

    if(!fp2)

    {

        printf("The file does not exist!\n");

        return 0;

    }

    for(i = 0; !feof(fp2); i++)

    {

        fscanf(fp2, "%d %s %d", &stu[i].stuID, stu[i].stuName, &stu[i].Age);

    }

    fclose(fp2);

    LinkedListCreate(&Lb, stu, i);

    printf("lb:\n");

    LinkedListTraverse(&Lb);

    LinkedListMerge(&La, &Lb, &Lc);

    printf("合并:\n");

    LinkedListTraverse(&Lc);

    LinkedListReverseSort(&La, &Lb);

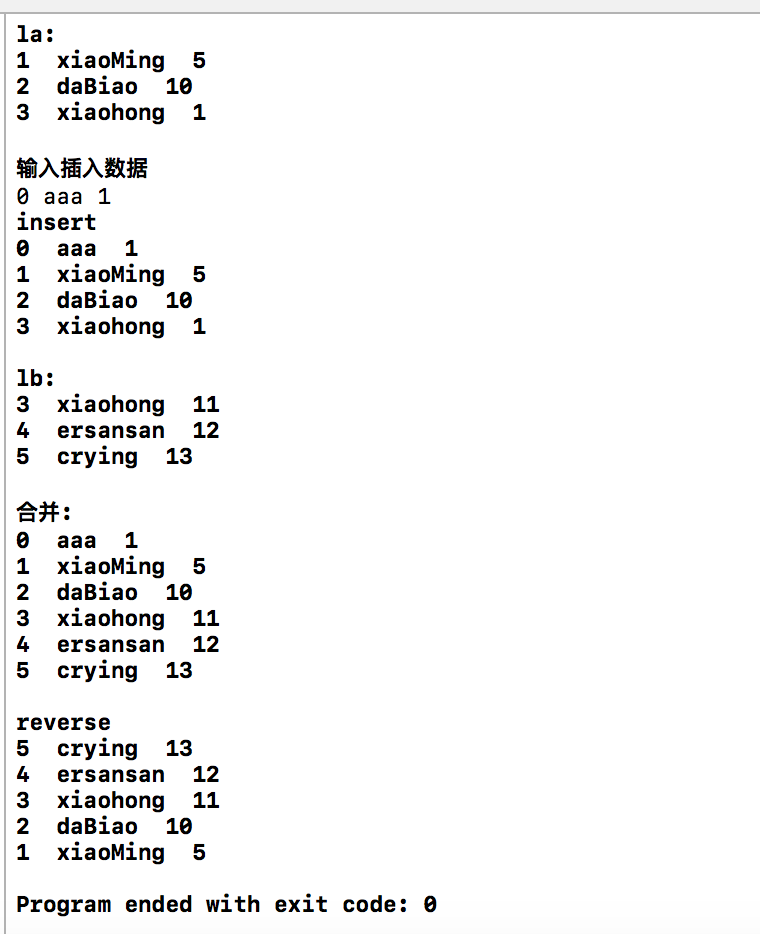
    printf("reverse\n");

    LinkedListTraverse(&La);

    return 0;

}

**九、程序运行结果：**

****

**十、实验结论：**

测试正确，程序满足项目需求。

**十一、总结及心得体会：**

总结：

这次的实验主要考察了对带头结点的链表的基本操作，主要包括设计创建链表、插入结点、遍历结点等算法，更加掌握了线性链表的基本特征和算法，也加强了对**C**语言对文件的操作的使用。

这次实验还是遇到了很多的问题，其中最大的问题就是对上学期**C**语言知识的遗忘，很多对文件的操作都忘了要怎么做，一直查资料花费了较多的时间。所以学过的知识还是得多加复习才能真正的掌握啊。。还有就是写**txt**文件的时候，一不小心敲了回车，就造成数据读入错误，始终不对，也找了很久的错误。。不过经历了这次以后也学会了如何检查**txt**文件有没有手贱多敲空格，以及要注意不要手贱。

其次，就是将**merge**函数和**reverse**函数一起使用时忘记了他们之间的相互影响，即使用了一个函数以后，**la**或者**lb**就被改变不复存在，导致第二个函数不能执行。。也是找了很久的错，发现后也没有找到适合的解决方法，求助了同学后才知道较好的解决方法。。。反映出我的思维还是比较呆板啊，需要加强这方面的锻炼，摆脱惯性思维。

心得体会：

实验是工科学生必须做的，只有自己动手尝试才能真正掌握知识，才能将理论的东西化为实践。

只要能够静下心来，就会发现编程的乐趣，不断调试程序的过程是既快乐又痛苦的，但是这是每个人成长必经的过程，希望以后自己还能以这样的心态，慢慢的认真的编写和调试程序。

理解不是很透彻时，可以尝试去问了同学的意见，同学的提示可以打开我的思维。这次的实验让我感受到了学习还是不能闭门造车，有时候思想的交流会有意想不到的效果。