电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221302006

姓 名 周玉川

（实验） 课程名称 数据结构

理论教师 刘勇国

实验教师 肖老师

**电 子 科 技 大 学**

**实 验 报 告（1）**

**学生姓名：** 周玉川  **学 号：2017221302006**  **指导教师： 刘勇国，肖老师**

**实验地点：**清水河基础实验大楼 **实验时间：2018.4.13**

**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：**

**// done by zhouyuchuan --2018.4.13**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <stdbool.h>**

**#define MAXNUM 50**

**typedef struct node{**

**char name[20];**

**int age;**

**char stuID[14];**

**}ElemType;**

**typedef struct Node{**

**ElemType data;**

**struct Node\*next;**

**}ListNode,\*ListPtr;**

**void List\_Init(ListPtr L);**

**void Input(ListPtr L);**

**void List\_SortInsert(ListPtr L,ElemType x);**

**void List\_Destory(ListPtr h);**

**ListPtr List\_Merge(ListPtr La,ListPtr Lb);**

**void Output(ListPtr La);**

**void List\_Check(ListNode \*a, ListNode \*b);**

**void List\_Delete(ListPtr L);**

**int main(void)**

**{**

**ListPtr L=(ListPtr)malloc(sizeof(ListNode)),L1=(ListPtr)malloc(sizeof(ListNode));**

**List\_Init(L);**

**List\_Init(L1);**

**int time;**

**printf("Operation prompt: \nInsert: i\nDelete: D\nPrint: p\nQuit: q\n");**

**char operate;**

**bool con=true;;**

**printf("As for the link L:\n");**

**for (;con;){**

**printf("Please input operation: ");**

**operate=getchar();**

**printf("\n");**

**while (getchar()!='\n');**

**switch(operate){**

**case 'q':case 'Q': con=false;break;**

**case 'I':case 'i':Input(L); break;**

**case 'D':case 'd':List\_Delete(L); break;**

**case 'P':case 'p':Output(L);break;**

**default : break;**

**}**

**}**

**con=true;**

**printf("As for the link L1:\n");**

**for (;con;){**

**printf("Please input operation: ");**

**operate=getchar();**

**printf("\n");**

**while (getchar()!='\n');**

**switch(operate){**

**case 'q':case 'Q': con=false;break;**

**case 'I':case 'i':Input(L1); break;**

**case 'D':case 'd':List\_Delete(L1); break;**

**case 'P':case 'p':Output(L1);break;**

**default : break;**

**}**

**}**

**List\_Check(L, L1);**

**ListPtr LL=List\_Merge(L,L1);**

**printf("After merge L and L1: \n");**

**Output(LL);**

**return 0;**

**}**

**void List\_Init(ListPtr L){**

**L->next=NULL;**

**}**

**void List\_Check(ListNode \*a, ListNode \*b){**

**ListPtr la=a,lb=b,L,p;**

**while (la->next!=NULL){**

**L=la;**

**while(L->next!=NULL)**

**{**

**if (strcmp(la->next->data.name,L->next->data.name)==0)**

**{**

**L->next->data.age=L->next->data.age < la->next->data.age ? L->next->data.age : la->next->data.age;**

**p=la->next;**

**la->next=la->next->next;**

**free(p);**

**return ;**

**}**

**}**

**}**

**}**

**void Input(ListPtr L){**

**ElemType x;**

**printf("Enter the & to end.\n");**

**do{**

**printf("Enter the name: ");**

**gets(x.name);**

**if (strcmp(x.name,"&")==0)**

**{**

**return ;**

**}**

**printf("Enter the age: ");**

**scanf("%d",&x.age);**

**printf("Enter the student id:");**

**scanf("%s",x.stuID);**

**while(getchar()!='\n');**

**List\_SortInsert(L,x);**

**}while(1);**

**}**

**void List\_SortInsert(ListPtr L,ElemType x){**

**ListPtr Lp=L;**

**for (;Lp->next!=NULL&&Lp->next->data.age < x.age;){**

**if (strcmp(Lp->next->data.name,x.name)==0){**

**Lp->next->data.age=(L->next->data.age < x.age)? L->next->data.age:x.age;**

**return ;**

**}**

**Lp=Lp->next;**

**}**

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

**p->data=x;**

**p->next=Lp->next;**

**Lp->next=p;**

**}**

**void List\_Destory(ListPtr h){**

**ListPtr L;**

**for (;h->next!=NULL;){**

**L=h->next;**

**free(h);**

**h=L;**

**}**

**free(h);**

**}**

**ListPtr List\_Merge(ListPtr La,ListPtr Lb){ //二路并归算法**

**ListPtr L=(ListPtr)malloc(sizeof(ListNode)),p,q,Lp;**

**Lp=L;**

**p=La->next;**

**q=Lb->next;**

**while(q!=NULL&&p!=NULL){**

**if (p->data.age <q->data.age){**

**Lp->next=p;**

**p=p->next;**

**Lp=Lp->next;**

**}**

**else{**

**Lp->next=q;**

**q=q->next;**

**Lp=Lp->next;**

**}**

**}**

**while (p!=NULL){**

**Lp->next=p;**

**p=p->next;**

**Lp=Lp->next;**

**}**

**while (q!=NULL){**

**Lp->next=q;**

**q=q->next;**

**Lp=Lp->next;**

**}**

**return L;**

**}**

**void List\_Delete(ListPtr L){**

**char deleteId[14];**

**printf("Please enter the id you delete: ");**

**scanf ("%s",deleteId);**

**while(getchar()!='\n');**

**ListPtr p=L->next,pre=L;**

**for (;p!=NULL;){**

**if (strcmp(p->data.stuID,deleteId)==0)**

**{**

**pre->next=p->next;**

**}**

**pre=p;**

**p=p->next;**

**}**

**}**

**void Output(ListPtr La){**

**ListPtr L=La->next;**

**printf("姓名 年龄 学号\n");**

**for (;L!=NULL;)**

**{**

**printf("%s %4d %s\n",L->data.name,L->data.age,L->data.stuID);**

**L=L->next;**

**}**

**}**

**九、程序运行结果：**

|  |
| --- |
| **(1)运行结果1：** |
| **评价：界面正常，清晰易懂** |
| **（2）运行结果2：** |
| **评价：满足实验重复人物取较小年龄** |
| **（3）运行结果3：** |
| **评价：链表2也操作正确，运行正常。** |
| **（4）运行结果4：** |
| **评价：按实验要求，链表成功合并成有序表，成功退出。**  **使用二路并归算法，极大减少算法时间和空间复杂度，认真研究书本知识，并加以运用，值得一赞。** |

**十、实验结论：**

**实验成功，结论正确。**

**十一、总结及心得体会：**

1. 认真研究书本，虽然对新手来说有些困难，但是认真吸收后会受益无穷。
2. 平时多写代码，多练练，不能只在上机的时候才敲敲键盘，切勿眼高手低，没有简单的知识，只有勤快的学生。
3. 学习线性表知识从最简单初始化，判空，判满，插入，删除，销毁学起，然后再去研究线性表的应用，循循渐进，脚踏实地。
4. 对我来讲还是有点简单，都是平常都写过的操作，并不难，但自己还是有致命的短板，打字母太慢，还容易打错，容易犯低级错误，还是多练。
5. 以上4点就是我的切身感悟，版权所有，违者必究。

---author:

Zhouyuchuan