**电 子 科 技 大 学**

**实 验 报 告**

**课程名称： 数据结构与算法**

**学生姓名：**  **郭洋**

**学 号：**  **2015220103021**

**指导教师： 刘勇国**

**实验地点： 清水河科技实验大楼**

**信息与软件工程学院**

**电 子 科 技 大 学**

**实 验 报 告（1）**

**学生姓名：** 郭洋 **学 号：** 2015220103021  **指导教师： 刘勇国**

**实验地点：**清水河科技实验大楼 **实验时间：2016.03**

**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" \t "_blank)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 {

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，并能实现年龄递增的两个单向链表合并一个链表，合并后的链表按年龄递减，可认为同名同年龄是同一个学生，每个学生在合并后的链表中仅出现一次。最后打印输出合并后的链表元素，验证结果的正确性。

**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。

1. **数据结构与程序：**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**typedef struct stuinfo**

**{**

**char stuName[10];**

**int Age;**

**}Stuinfo;**

**typedef struct node**

**{**

**Stuinfo data;**

**struct node \*next;**

**}ListNode,\*ListPtr;**

**ListPtr ListCreate(void);**

**void ListInsert(ListPtr L,ListPtr h,char \*x,int age);**

**void ListDelete(ListPtr L,char \*s);**

**ListPtr ListMerge(ListPtr La,ListPtr Lb);**

**void OutPut(ListPtr L);**

**ListPtr ListLocate(ListPtr L,char \*name);**

**ListPtr ListReverse(ListPtr L);**

**void ListFree(ListPtr L);**

**int main(void)**

**{**

**ListPtr La,Lb = NULL;**

**printf("Create a List == 1\n");**

**printf("Inserst an element == 2\n");**

**printf("Delete an element == 3\n");**

**printf("Create another List == 4\n");**

**printf("Merge two lists == 5\n");**

**printf("Exit == 0\n\n");**

**int choice;**

**bool mark = false,mark0 = false;**

**do{**

**printf("Enter your choice:");**

**scanf("%d",&choice);**

**switch(choice){**

**case 1: La = ListCreate();**

**OutPut(La);**

**mark0 = true;**

**break;**

**case 2: if (mark0 == true){**

**printf("Which place do you want to insert? and enter the name of that place:");**

**char name1[10],name2[10];**

**int age;**

**scanf(" %s",name1);**

**printf("Enter the name you want to insert: ");**

**scanf(" %s",name2);**

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

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

**ListPtr temp;**

**temp = ListLocate(La,name1);**

**if (temp == NULL){**

**printf("The name isn't in this list!\n");**

**}else{**

**ListInsert(La,temp,name2,age);**

**}**

**OutPut(La);**

**}else{**

**printf("You haven't create a list . Please select option 1 to create it.\n");**

**}**

**break;**

**case 3: if (mark0 == true){**

**printf("Enter the name you want to delete:");**

**char name3[10];**

**scanf(" %s",name3);**

**ListDelete(La,name3);**

**OutPut(La);**

**}else{**

**printf("You haven't create a list . Please select option 1 to create it.\n");**

**}**

**break;**

**case 4: if (mark0 == true){**

**Lb = ListCreate();**

**OutPut(Lb);**

**mark = true;**

**}else{**

**printf("Select option 1 at first!\n");**

**}**

**break;**

**case 5: ListPtr Lc;**

**if (mark == true && mark0 == true){**

**Lc = ListMerge(La,Lb);**

**OutPut(Lc);**

**}else{**

**printf("You haven't create another list yet!\n");**

**}**

**break;**

**default:break;**

**}**

**}while(choice != 0);**

**ListFree(La);**

**}**

**ListPtr ListCreate(void)**

**{**

**ListPtr L;**

**L = (ListNode \*)malloc(sizeof(ListNode));**

**if (L == NULL){**

**printf("Fail to apply the space!");**

**exit(0);**

**}**

**ListPtr tail; //create a tail pointer**

**L -> next = NULL;**

**tail = L;**

**printf("Enter the length of this list: ");**

**int length,i; //i is used to count number**

**scanf(" %d",&length);**

**for(i = 0;i < length;i++){**

**ListPtr p;**

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

**if (p == NULL){**

**printf("Fail to apply the space!");**

**exit(0);**

**}**

**printf("Enter the %d th student's name: ",i+1);**

**scanf(" %s",(p -> data).stuName);**

**printf("Enter the %d th student's age: ",i+1);**

**scanf(" %d",&(p -> data).Age);**

**p -> next = NULL;**

**tail -> next = p;**

**tail = p;**

**}**

**return L;**

**}**

**void ListInsert(ListPtr L,ListPtr h,char \*name,int age)**

**{**

**ListPtr q,pre;**

**q = (ListNode \*)malloc(sizeof(ListNode));**

**if (q == NULL){**

**printf("Fail to apply the space!");**

**exit(0);**

**}**

**strcpy((q -> data).stuName,name);**

**(q -> data).Age = age;**

**pre = L; // find prior**

**while(pre != NULL && pre -> next != h){**

**pre = pre -> next;**

**}**

**q -> next = pre -> next;**

**pre -> next = q;**

**}**

**void ListDelete(ListPtr L,char \*name)**

**{**

**ListPtr pre,p;**

**pre = L;**

**while(pre -> next != NULL && (strcmp((pre -> next -> data).stuName,name)) != 0){**

**pre = pre -> next;**

**}**

**p = pre -> next;**

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

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

**free(p);**

**}else{**

**printf("The name isn't in this list!\n");**

**}**

**}**

**ListPtr ListMerge(ListPtr La,ListPtr Lb)**

**{**

**ListPtr Lc,pa,pb,pc;**

**pa = La -> next;**

**pb = Lb -> next;**

**Lc = La;**

**pc = Lc;**

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

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

**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;**

**}**

**Lc = ListReverse(Lc);**

**free(Lb);**

**return Lc;**

**}**

**ListPtr ListReverse(ListPtr L)**

**{**

**ListPtr p,q;**

**if (L -> next && L -> next -> next){**

**p = L -> next;**

**q = p -> next;**

**p -> next = NULL;**

**while(q){**

**p = q;**

**q = q -> next;**

**p -> next = L -> next;**

**L -> next = p;**

**}**

**}**

**return L;**

**}**

**ListPtr ListLocate(ListPtr L,char \*name)**

**{**

**ListPtr p;**

**p = L -> next;**

**while((p != NULL) && strcmp((p -> data).stuName,name) != 0){**

**p = p -> next;**

**}**

**if (p == NULL){**

**return NULL;**

**}else{**

**return p;**

**}**

**}**

**void OutPut(ListPtr L)**

**{**

**ListPtr p;**

**p = L -> next;**

**int i = 1;**

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

**printf("The %d th student's information :%s(name) %d(age)\n",i,(p -> data).stuName,(p -> data).Age);**

**p = p -> next;**

**i++;**

**}**

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

**}**

**void ListFree(ListPtr L)**

**{**

**ListPtr pre,pre0;**

**pre = L;**

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

**pre0 = pre -> next;**

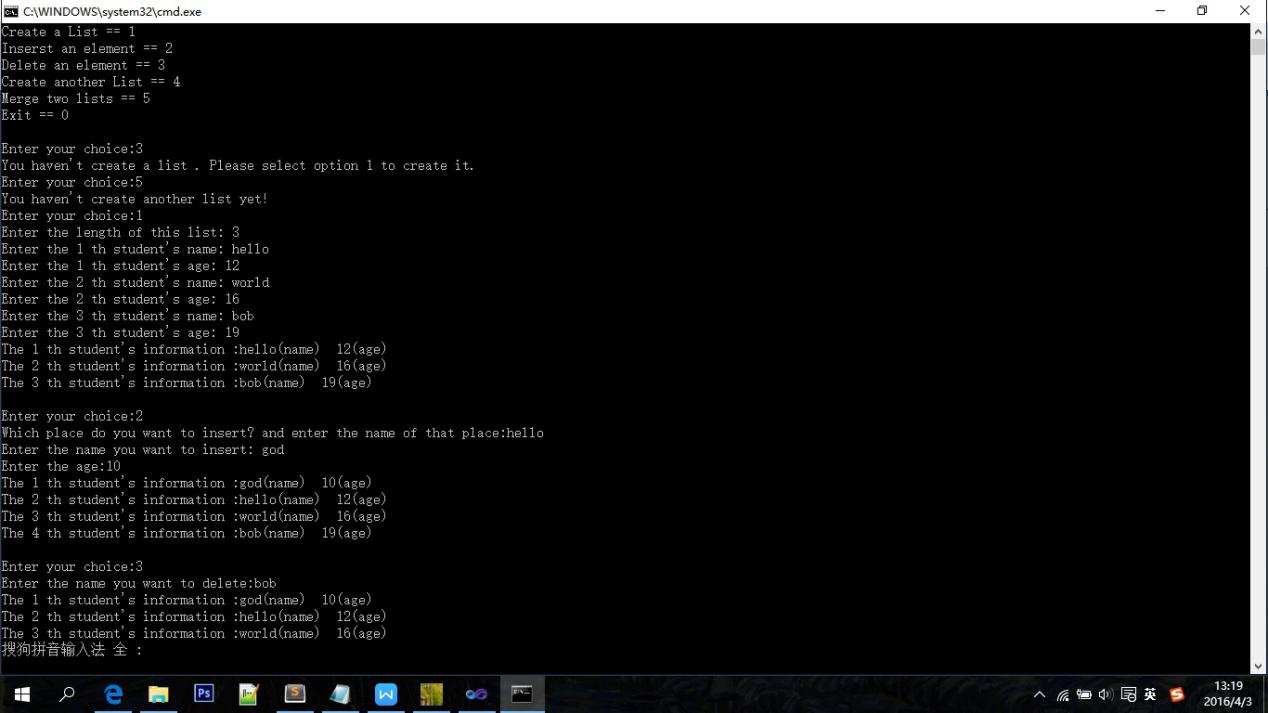
**free(pre);**

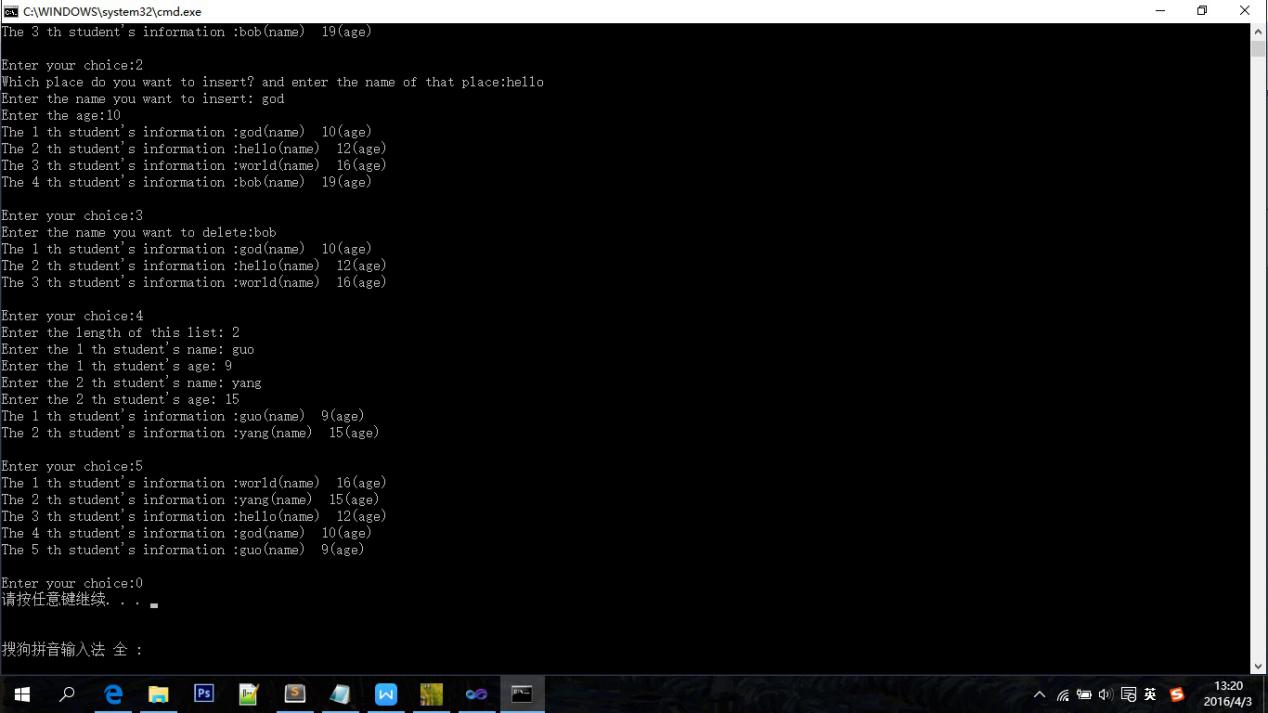
**pre = pre0;**

**}**

**}**

1. **程序运行结果：**

****

****

1. **实验结论：**

该程序通过线性表基本上能够实现对学生信息的创建、插入、删除、合并等操作。同时对于一些误操作能够给出提醒。

1. **总结及心得体会：**

通过这次实验，我对线性表的认识以及应用有了更多的认识，学会了利用线性表进行一般的常规操作，如创建、插入、删除等等，同时也了解到了线性表的一些实际应用的实例如学生信息的存储。不过，在此过程中也发现了一些问题，比如编写过程中的不熟练，起初对于malloc()应用之后未及时的进行free()操作等坏的编写习惯，以及对于部分非预期的输入的处理上还没有完全考虑的周全，故可能还存在些bug。