**西南石油大学实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目编号** | 001063000501 | **项目名称** | 一般线性表的应用 | **成绩** |  |
| **专业年级** | **软件工程2022级** | | | **指导教师** | **胡卫东** |
| **姓名** | **李浩楠** | **学号** | **202231060920** | **实验日期** | **2023/5/31** |

1. **实验目的**
2. 掌握一般线性表的结构的定义。
3. 掌握一般线性表的基本操作。
4. 使用C/C++实现相应的基本操作。
5. **实验工具**

PC微机、Windows、PTA或Dev c++。

1. **实验步骤**
2. 密接者最终系统（单链表的插入与查询）。

新冠疫情泛滥，为了控制疫情传播速度，对密接者的追踪和管理非常重要。现在开发一款密接者追踪系统，存放所有密接者信息，该系统能够按照姓名查询密接者信息，也能在系统中插入新的密接者。已知该系统采用单链表实现密接者信息的存储（已存有4位密接者的信息，包括姓名和症状），编写函数实现密接者的查询和新增。

1. 有序链表合并。
2. 统计计算机专业学生人数。

本题要求实现一个函数，统计计算机专业的学生人数。学生学号存储在单链表中，通过识别学号来判断该学生是否为计算机专业的学生，学生的学号共7位数字，其中第2、3位是专业编号，计算机专业的编号为02。例如，‘8026823’则为计算机专业学生，‘8062391’为非计算机专业学生。  
链表结点定义如下：  
struct ListNode {  
char code[8];  
struct ListNode \*next;  
};

1. 顺序表求交集。

两个线性表La和Lb均采用顺序存储结构，数据元素均为正整数，两个线性表中各有若干个整数，编写程序求出两个线性表的交集。  
已知顺序表的结构定义如下：

typedef struct

{

ElemType \*elem; //存储空间基址，体现动态性

int length; //当前表的长度

int listsize; //当前分配的存储容量

}SqList;

主程序框架为：

int main()

{

SqList La,Lb,Lc;

InitList\_Sq(La); //初始化3个线性表

InitList\_Sq(Lb);

InitList\_Sq(Lc);

Listinput(La); //输入La的值

Listinput(Lb);//输入Lb的值

steMUl(Lc,La,Lb);//求出La和Lb的交集，放在Lc中

print\_List(Lc);//打印交集

return 0;

}

### 输入格式:

输入两行正整数，每行均以0结束（0之前的整数为线性表中数据元素，不包括0）。

### 输出格式:

输出两个线性表中交集，每个整数之间以一个空格隔开；如果两个线性表没有交集则输出“empty class”。

1. **实验结果**
2. 测试用例：张三 咳嗽 2 李林

代码：LinkList locate\_LinkList(LinkList H, char key[]) //H为单链表的头指针，key为要查询的姓名

{

LinkList p=H->next;

while(p&&strcmp(p->data.name,key)!=0)

{p=p->next;}

if(strcmp(p->data.name,key)==0)

return p;

return 0;

}

Status ListInsert(LinkList &H,int i,ElemType e)//H为单链表的头指针，i为插入位置，e为新增的密接者

{

LinkList p=H,s;

int j=0;

while(p&&(j<i-1))

{p=p->next;j++;}

if(!p||j>i-1)

return 0;

s=(LNode\*)malloc(sizeof(LNode));

s->data=e;

s->next=p->next;

p->next=s;

return 1;

}

运行结果：接触者：陈晨,症状为：无症状

接触者：张三,症状为：咳嗽

接触者：李俐,症状为：轻微咳嗽

接触者：李林,症状为：无症状

接触者：张伟,症状为：乏力

接触者李林的症状为：无症状

1. 测试用例：1 3 5 8 -1

2 6 8 -1

代码：LinkList Read( )//按顺序依次读入元素，生成一个带头结点的单链表，表中元素值排列顺序与输入顺序一致，若输入-1，则创建链表结束（链表中不包含-1）。此处要求元素值按非递减顺序录入

{

LinkList p,H,s;

H=(LNode\*)malloc(sizeof(LNode));

H->next=NULL;

p=H;

int n;

scanf("%d",&n);

while(n!=-1)

{

s=(LNode\*)malloc(sizeof(LNode));

s->data=n;

s->next=p->next;

p->next=s;

p=s;

scanf("%d",&n);

}

return H;

}

LinkList Merge( LinkList L1, LinkList L2 )//合并L1与L2。已知L1与L2中的元素非递减排列，要求合并后的单链表中元素也按值非递减排列。

{

/\*if(L1->next==NULL&&L2->next==NULL)

return NULL;\*/

LinkList p1,p2,p3,L3;

p1=L1->next;

p2=L2->next;

L3=L1;

p3=L3;

while(p1&&p2)

{

if(p1->data<=p2->data)

{

p3->next=p1;

p3=p1;

p1=p1->next;

}

else

{

p3->next=p2;

p3=p2;

p2=p2->next;

}

}

p3->next=(p1?p1:p2);

return L3;

}

运行结果：1 2 3 5 6 8 8

1. 测试用例：1021202

2022310

8102134

1030912

3110203

4021205

#

代码：int countcs( struct ListNode \*head )

{

int num=0;

Linklist p=head->next;

while(p)

{

if(p->code[1]=='0'&&p->code[2]=='2')

num++;

p=p->next;

}

return num;

}

运行结果：3

1. 测试用例：1 5 9 11 15 0

2 4 5 6 9 10 0

代码：#include<string.h>

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef int ElemType;

typedef struct

{

ElemType \*elem; //存储空间基址，体现动态性

int length; //当前表的长度

int listsize; //当前分配的存储容量

}SqList;

int InitList\_Sq(SqList \*l)

{

l->elem=(ElemType \*)malloc(30\*sizeof(ElemType));

if(!l->elem)

exit(0);

l->length=0;

l->listsize=30;

return 1;

}

void Listinput(SqList \*La)

{

int i=0;

int a;

scanf("%d",&a);

while(a!=0)

{

La->elem[i]=a;

La->length++;

i++;

scanf("%d",&a);

}

}

int steMUl(SqList \*Lc, SqList \*La, SqList \*Lb)

{

int i,j,num=0;

for(i=0;i<La->length;i++)

{

for(j=0;j<Lb->length;j++)

{

if(La->elem[i]==Lb->elem[j])

{

Lc->elem[num]=La->elem[i];

num++;

}

}

}

Lc->length = num;

return 1;

}

void print\_List(SqList \*Lc)//打印交集

{

int i;

for(i=0;i<Lc->length;i++)

{

printf("%d ",Lc->elem[i]);

}

if(Lc->length==0)

printf("empty class");

}

int main()

{

SqList La,Lb,Lc;

InitList\_Sq(&La); //初始化3个线性表

InitList\_Sq(&Lb);

InitList\_Sq(&Lc);

Listinput(&La); //输入La的值

Listinput(&Lb);//输入Lb的值

steMUl(&Lc,&La,&Lb);//求出La和Lb的交集，放在Lc中

print\_List(&Lc);//打印交集

return 0;

}

运行结果：5 9

1. **实验总结**
2. 对于顺序表而言，可以随机存取表中任意一元素，简单易操作。但是在做插入删除操作时需要移动大量元素，同时但数据表中元素较多且变化较大时，操作过程复杂，导致空间的浪费。
3. 对于单链表而言，频繁的插入与删除操作不需要移动大量元素，操作简单、高效，同时能够开拓空间，解决顺序表大小固定的问题，并不会造成空间的浪费。
4. 当线性表长度变化大，难以预估存储规模，宜采用链表存储；变化不大，宜采用顺序表存储；很少做插入或删除操作时，采用顺序表；平凡插入或删除时选择链式存储。
5. 链表的插入与删除时，应当注意避免断链情况的发生而导致数据的丢失。