实验项目名称： 线性表

**（所属课程：** 数据结构 **）**

**学 院： 计算机科学与信息工程学院 专业班级：网络工程18-1**

**姓 名： 邓二浦 学 号：18031110129**

**实验日期： 2019-10-11 实验地点：a06-203**

**合作者： 指导教师：孙高飞**

**本实验项目成绩： 教师签字： 日期：**

1. **实验目的**

本次实验的目的掌握顺序表的存储结构形式及其描述和基本运算的实现；掌握动态链表结构及相关算法设计

1. **实验内容**

2.1 实现顺序表的创建、插入、删除和输出功能。

2.2线性表的单链表设计以及各种基本操作的实现。

2.3 实现两个有序单链表的合并。

**三 、实验步骤与结果**

2.1实验步骤

* 建立main函数。

int main()

{

SeqList L;

int i,e;

initlist(&L);

creatlist(&L);

printf("创建的顺序表为:\n");

printlist(&L);

printf("请输入要插在第几个数以及要插入的数\n");

scanf("%d%d",&i,&e);

InsList(&L,i,e);

printf("插入后的链表为:\n");

printlist(&L);

printf("请输入要删除第几个数\n");

scanf("%d",&i);

dellist(&L,i,&e);

printf("删除后的链表为:\n");

printlist(&L);

return 0;

}

* 输入42页 线性表SeqList的定义和相关的常量定义。

typedef struct

{

int a[max];

int last;

}SeqList;

* 输入调试算法2.2 InsList 方法，为顺序表插入元素。

int InsList(SeqList \*L,int i,int e) //插入

{

int k;

if((i<1)||(i>L->last+2))

{

printf("插入位置i值不合法");

return 0;

}

if(L->last>=max-1)

{

printf("表已满，无法插入\n");

return 0;

}

for(k=L->last;k>=i-1;k--)

L->a[k+1]=L->a[k];

L->a[i-1]=e;

L->last++;

return 1;

}

* 在main函数中调用InsList为顺序表插入多个数据。
* 编写线性表输出函数，通过循环依次输出SqList中的各个元素的内容。

void printlist(SeqList \*L) //输出顺序表

{

int i;

for(i=0;i<=L->last;i++)

printf("%d ",L->a[i]);

printf("\n");

}

* 输入调试算法2.3，删除元素（在删除元素方法前后分别输出顺序表，观察删除结果是否正确）。

int dellist(SeqList \*L,int i,int \*e) //删除

{

int k;

if(i<1||i>L->last+1)

{

printf("删除位置不合法\n");

return 0;

}

\*e=L->a[i-1];

for(k=i-1;k<L->last;k++)

L->a[k]=L->a[k+1];

L->last--;

return 1;

}

2.2实验步骤

* 建立main函数。

int main()

{

int i,e;

Linklist L,\*p;

initlist(&L);

creatlist(&L);

printf("创建的链表为:");

printlist(&L);

printf("请输入要查找链表中的第几个数\n");

scanf("%d",&i);

p=get(&L,i);

printf("第%d个元素为%d\n",i,p->date);

printf("请输入插在第几个数以及要插入的值\n");

scanf("%d%d",&i,&e);

insertlist(&L,i,e);

printf("插入后的链表为:");

printlist(&L);

printf("请输入要删除第几个数\n");

scanf("%d",&i);

dellist(&L,i,&e);

printf("删除后的链表为\n");

printlist(&L);

return 0;

}

* 输入48页链表LinkList和结点Node的定义。

typedef struct node

{

int date;

struct node \*next;

}Node,Linklist;

* 实现算法2.5，初始化单链表。

void initlist(Linklist \*L) //初始化链表

{

L=(Linklist \*)malloc(sizeof(struct node));

L->next=NULL;

}

* 实现算法2.6 或2.7 ，初始化链表数据。
* 编写链表表输出函数，通过While循环依次输出LinkList中的各个元素的内容。输出刚创建的链表。

void printlist(Linklist \*L) //输出链表

{

Linklist \*p;

p=L->next;

while(p!=NULL)

{

printf("%d ",p->date);

p=p->next;

}

printf("\n");

}

* 实现算法2.8，获得第i个数据元素

Linklist \* get(Linklist \*L,int i) //在链表中查找第i个元素

{

Linklist \*p;int j;

if(i<0)

{printf("查找不合法");return NULL;}

p=L;j=0;

while(p->next!=NULL&&j<i)

{

p=p->next;

j++;

}

if(i==j)

return p;

else

return NULL;

}

* 实现算法2.11，在链表中插入数据元素，可通过输入链表检查结果。

int insertlist(Linklist \*L,int i,int e) //插入操作

{

Linklist \*p,\*s;

int k;

if(i<0) return 0;

p=L;k=0;

while(p!=NULL&&k<i-1)

{

p=p->next;

k++;

}

if(p==NULL)

{

printf("插入位置不合法\n");return 0;

}

s=(Linklist \*)malloc(sizeof(struct node));

s->date=e;

s->next=p->next;

p->next=s;

return 1;

}

* 实现算法2.12，在链表中删除元素，可通过输入链表检查结果。

int dellist(Linklist \*L,int i,int \*e) //删除操作

{

Linklist \*p,\*s;

int k;

p=L;k=0;

while(p->next!=NULL&&k<i-1)

{

p=p->next;k++;

}

if(p->next==NULL)

{

printf("删除结点的位置i不合理");

return 0;

}

s=p->next;

p->next=s->next;

\*e=s->date;

free(s);

return 1;

}

1. **实验总结**

线性表的特点:

1. 同一性。线性表由同类数据元素组成，每一个ai必须属于同一个数据类型。
2. 有穷性。线性表由有限个数据元素组成，表长度就是表中数据元素。
3. 有序性。线性表中相邻数据元素之间存在着序偶关系<ai,ai+1>.

线性表的顺序存储

 优点： 具有简单、运算方便等优点，特别是对于小线性表或长度固定的线性表，采用顺序存储结构的优越性更为突出；

 缺点：1.顺序存储插入与删除一个元素，必须移动大了的数据元素，以此对大的线性表，特别是在元素的插入和删除很频繁的情况下，采取顺序存储很是不方便，效率低；

2.顺序存储空间容易满，出现上溢，程序访问容易出问题，顺序存储结构下，存储空间不便扩充；

3.顺序存储空间的分配问题，分多了浪费，分少了空间不足上溢。

单链表的优点：①元素的存储单元是任意的，可连续也可不连续。②不需要限定长度。

单链表的缺点：①其查找时间复杂度为O(n)。②存放元素时需要另外开辟一个指针域的空间。

分析下列情况下，采用何种存储结构更好些。

（1）若线性表的总长度基本稳定，且很少进行插入和删除，但要求以最快的速度存取线性表中的元素。采用顺序存储结构，其插入和删除操作需要移动大量元素，但可按数组下标直接查找元素。

（2）如果n个线性表同时并存，并且在处理过程中各表的长度会动态发生变化。

采用链接存储结构，链表的长度不需要限制，可动态发生变化。

（3）描述一个城市的设计与规划。

采用链接存储结构，因为城市的设计与规划过程中的变化大，需要不断的插入和删除操作。