**第2章 线性表作业**

**一、填空**

**1. 在顺序表中插入或删除一个元素，需要平均移动 \_\_\_表中一半\_\_\_ 元素，具体移动的元素个数**

**与 表长及该元素在表中位置 有关。**

**2. 线性表中结点的集合是 有限 的，结点间的关系是 \_\_\_\_\_一对一\_ 的。**

1. **向一个长度为n的向量的第i个元素(1≤i≤n+1)之前插入一个元素时，需向后移动**

**n-i+1 个元素。**

**4. 向一个长度为n的向量中删除第i个元素(1≤i≤n)时，需向前移动(\_\_\_n-i\_\_\_) 个元素。**

**5. 在顺序表中访问任意一结点的时间复杂度均为 O(1) ，因此，顺序表也称为(\_\_\_随机存取\_\_\_)的数据结构。**

**6. 顺序表中逻辑上相邻的元素的物理位置 必定 相邻。单链表中逻辑上相邻的元素的物理位置 不一定 相邻。**

**7. 在单链表中，除了首元结点外，任一结点的存储位置由\_\_\_\_ 其前驱节点链域的值 \_\_\_\_\_指示。**

**8． 在n个结点的单链表中要删除已知结点\*p，需找到它的\_\_\_\_\_前驱节点地址\_ ，其时间复杂度为 O(n) 。**

1. **已知P为单链表中的非首尾结点，在P结点后插入S结点的语句为：\_\_\_\_\_\_\_\_\_\_\_\_S->next=P->next;P->next=S;\_\_ 。  
   10.在非空双向循环链表中，在结点q的前面插入结点p的过程如下：   
   p->prior=q->prior;  
   q->prior->next=p;  
   p->next=q;  
   \_\_\_\_\_\_\_\_\_\_q->prior=p\_\_\_\_\_\_\_\_\_\_\_\_;**

**二、判断正误**

**（ 错 ）1. 链表的每个结点中都恰好包含一个指针。**

可含多个指针域，分别存放多个指针。如双向链表结点含有两个指针域，分别存放直接前驱与直接后继结点的指针

**（ 错 ）2. 链表的物理存储结构具有同链表一样的顺序。**

链表存储结构特点无序，示意图有序

**（ 错 ）3. 链表的删除算法很简单，因为当删除链中某个结点后，计算机会自动将后续各个单元向前移动。**

删除后仅是指针内容改变，结点不动

**（ 错 ）4. 线性表的每个结点只能是一个简单类型，而链表的每个结点可以是一个复杂类型。**

链表是线性表一种，且顺序表也可以存放复杂类型

**（ 错 ）5. 顺序表结构适宜于进行顺序存取，而链表适宜于进行随机存取。**

顺序表结构适宜于进行随机存取，而链表适宜于进行顺序存取

**（ 错 ）6. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高。**

顺序存储方式的优点是存储密度大，但插入、删除运算效率低，需要平均移动表中一半元素。

**（ 错 ）7. 线性表在物理存储空间中也一定是连续的。**

链式存储不要求连续存放

**（ 错 ）8. 线性表在顺序存储时，逻辑上相邻的元素未必在存储的物理位置次序上相邻。**

在顺序存储时，逻辑上相邻的元素一定在存储的物理位置次序上相邻

**（ 错 ）9. 顺序存储方式只能用于存储线性结构。**

完全二叉树是非线性结构但能用顺序存储

**（ 错）10. 线性表的逻辑顺序与存储顺序总是一致的。**

链式存储不要求连续存放，所以不一定一致

**三、单项选择题**

**（ c ）1．数据在计算机存储器内表示时，物理地址与逻辑地址相同并且是连续的，称之为：**

**（A）存储结构 （B）逻辑结构 （C）顺序存储结构 （D）链式存储结构**

**（ b ）2. 一个向量第一个元素的存储地址是100，每个元素的长度为2，则第5个元素的地址是**

**（A）110 （B）108 （C）100 （D）120**

连续存储，所以100+2\*4=108

**（ a ）3. 在n个结点的顺序表中，算法的时间复杂度是O（1）的操作是：**

1. **访问第i个结点（1≤i≤n）和求第i个结点的直接前驱（2≤i≤n）**

通过数组的下标直接定位，时间复杂度是O(1)

1. **在第i个结点后插入一个新结点（1≤i≤n）**O(n)
2. **删除第i个结点（1≤i≤n）** O(n)
3. **将n个结点从小到大排序** O(n^2)或O(nlogn)

**（ b ）4. 向一个有127个元素的顺序表中插入一个新元素并保持原来顺序不变，平均要移动 个元素**

**（A）8 （B）63.5 （C）63 （D）7**

n/2=127/2

**（ a ）5. 链接存储的存储结构所占存储空间：**

1. **分两部分，一部分存放结点值，另一部分存放表示结点间关系的指针**
2. **只有一部分，存放结点值**

**（C） 只有一部分，存储表示结点间关系的指针**

**（D） 分两部分，一部分存放结点值，另一部分存放结点所占单元数**

**（ b ）6. 链表是一种采用 存储结构存储的线性表；**

**（A）顺序 （B）链式 （C）星式 （D）网状**

**（ d ）7. 线性表若采用链式存储结构时，要求内存中可用存储单元的地址:**

**（A）必须是连续的 （B）部分地址必须是连续的**

**（C）一定是不连续的 （D）连续或不连续都可以**

**（ b ）8． 线性表Ｌ在 情况下适用于使用链式结构实现。**

**（Ａ）需经常修改Ｌ中的结点值 （Ｂ）需不断对Ｌ进行删除插入**

**（Ｃ）Ｌ中含有大量的结点 （Ｄ）Ｌ中结点结构复杂**

链表插入删除不用移动数据

**（ c ）9． 单链表的存储密度**

**（Ａ）大于1； （Ｂ）等于1； （Ｃ）小于1； （Ｄ）不能确定**

指针空间m

数据项空间(n)/总空间（n+m）

**（ b ）10． 设a1、a2、a3为3个结点，整数P0，3，4代表地址，则如下的链式存储结构称为**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P0 |  |  | **3** |  |  | **4** |  |  |
| **P0** | **🡪** | **a1** | **3** | **🡪** | **a2** | **4** | **🡪** | **A3** | **0** |

**（Ａ）循环链表 （Ｂ）单链表 （Ｃ）双向循环链表 （Ｄ）双向链表**

**四、算法题**

1. **假设两个按元素值递增有序排列的线性表A和B，均以单链表作为存储结构，请编写算法，将A表和B表归并成一个按元素值递减有序排列的线性表C，并要求利用原表（即A表和B表的）结点空间存放表C。**

*方法一，头插法直接插入*

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

using namespace std;

struct lnode

{

int data;

struct lnode\* next;

};

typedef struct lnode\* linklist;

int k;//测试组数

linklist la,lb,lc;

int a[10086] = {0};//输入数据

int b[10086] = {0};

int i;

//创建新链表

linklist lcreate(int a[],int len)

{

lnode\* l=(lnode\*)malloc(sizeof(lnode));

//申请新节点

l->next=NULL;

lnode \*p,\*prear;

prear=l;

for(int i=1;i<=len;i++)

{

cin>>a[i];

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

p->data=a[i];

prear->next=p;

prear=p;

}

prear->next=NULL;

return l;//返回头结点

}

//使用头插法插入链表

bool lmerge(linklist la,linklist lb,linklist &lc)

{

lnode \*pa,\*pb;

if(la==NULL||lb==NULL)

{

//la,lb空链表

return false;

}

pa=la->next;

pb=lb->next;

lc->next=NULL;

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

{

//链表循环未结束

if(pa->data<=pb->data)

{

//pa数据比pb小

la->next=pa->next;

//la指向pa下一项

pa->next=lc->next;

//头插法，pa下一项指向原本头结点所指

lc->next=pa;

//将pa放入lc

//cout<<" a"<<pa->data<<endl;

}

else

{

//pa数据比pb大的情况，同理

lb->next=pb->next;

pb->next=lc->next;

lc->next=pb;

//cout<<" b"<<pb->data<<endl;

}

pa=la->next;//将更新过的指针比较

pb=lb->next;

}

while(la->next!=NULL)

{

//当la有剩余项，将其放入lc

pa=la->next;

la->next=pa->next;

pa->next=lc->next;

lc->next=pa;

}

while(lb->next!=NULL)

{

//当lc有剩余项，将其放入lc

pb=lb->next;

lb->next=pb->next;

pb->next=lc->next;

lc->next=pb;

}

return true;

}

//结果输出函数

void lprint(linklist l)

{

lnode\* p;

if(l==NULL||l->next==NULL)

{

//表明为空结点

return;

}

p=l->next;

while(p!=NULL)

{

cout<<p->data<<" ";

p=p->next;

}

cout<<endl;

}

int main()

{

cin>>k;

while(k--)

{

memset(a,0,sizeof(a));

memset(b,0,sizeof(b));

cin>>a[0]>>b[0];

la=lcreate(a,a[0]);

lb=lcreate(b,b[0]);

//lprint(la);

//lprint(lb);

//lreverse(la);

//lreverse(lb);

//lprint(la);

//lprint(lb);

//1.reverse a,b

//2.头插

lnode\* lc=(lnode\*)malloc(sizeof(lnode));

lmerge(la,lb,lc);

lprint(lc);

free(lc);

}

return 0;

}

*方法二，先将链表逆序再用尾插法*

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

using namespace std;

struct lnode

{

int data;

struct lnode\* next;

};

typedef struct lnode\* linklist;

int k;//测试组数

linklist la,lb,lc;

int a[10086] = {0};//输入数据

int b[10086] = {0};

int i;

//创建新链表

linklist lcreate(int a[],int len)

{

lnode\* l=(lnode\*)malloc(sizeof(lnode));

//申请新节点

l->next=NULL;

lnode \*p,\*prear;

prear=l;

for(int i=1;i<=len;i++)

{

cin>>a[i];

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

p->data=a[i];

prear->next=p;

prear=p;

}

prear->next=NULL;

return l;//返回头结点

}

//使用尾插法插入节点

bool lmerge(linklist la,linklist lb,linklist &lc)

{

lnode \*pa,\*pb,\*prear;//尾项指针 prear

if(la==NULL||lb==NULL)

{

return false;

}

pa=la->next;

pb=lb->next;

lc=la;

prear=lc;

lc->next=NULL;

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

{

if(pa->data>=pb->data)

{

//pa数据大于pb

prear->next=pa;

prear=pa;//尾指针向后

pa=pa->next;

//cout<<"a"<<prear->data<<endl;

}

else

{

prear->next=pb;

prear=pb;

pb=pb->next;

//cout<<prear->data<<endl;

}

}

while(pa)

{//pa有剩余项放入lc

prear->next=pa;

prear=pa;

pa=pa->next;

}

while(pb)

{//pb有剩余项放入lc

prear->next=pb;

prear=pb;

pb=pb->next;

// cout<<"?"<<prear->data<<endl;

}

prear->next=NULL;

return true;

}

//1.reverse2

void lreverse(linklist& l)

{

lnode \*p,\*q,\*tmp;

p=l->next;

q=p->next;

p->next=NULL;

//将p和q之间的指向方向改变

while(q!=NULL)

{

tmp=q->next;

q->next=p;//逆转方向

p=q;

q=tmp;

//pq均向后移

}

l->next=p;//头指针指向第一个元素

}

//结果输出函数

void lprint(linklist l)

{

lnode\* p;

if(l==NULL||l->next==NULL)

{

//表明为空结点

return;

}

p=l->next;

while(p!=NULL)

{

cout<<p->data<<" ";

p=p->next;

}

cout<<endl;

}

int main()

{

cin>>k;

while(k--)

{

memset(a,0,sizeof(a));

memset(b,0,sizeof(b));

cin>>a[0]>>b[0];

la=lcreate(a,a[0]);

lb=lcreate(b,b[0]);

//lprint(la);

//lprint(lb);

lreverse(la);

lreverse(lb);

//lprint(la);

//lprint(lb);

//1.先将a,b由升序变降序再头插

//2.头插

lmerge(la,lb,lc);

lprint(lc);

free(lc);

}

return 0;

}

**2、 设线性表A=(a1, a2,…,am)，B=(b1, b2,…,bn)，试写一个按下列规则合并A、B为线性表C的算法，使得：**

**C= (a1, b1,…,am, bm, bm+1, …,bn) 当m≤n时；**

**或者 C= (a1, b1,…,an, bn, an+1, …,am) 当m>n时。**

**线性表A、B、C均以双链表作为存储结构，且C表利用A表和B表中的结点空间构成。**

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

using namespace std;

struct lnode

{

int data;

struct lnode\* next;

};

typedef struct lnode\* linklist;

int k;//测试组数

linklist la,lb,lc;

int a[10086] = {0};//输入数据

int b[10086] = {0};

int i;

//创建新链表

linklist lcreate(int a[],int len)

{

lnode\* l=(lnode\*)malloc(sizeof(lnode));

//申请新节点

l->next=NULL;

lnode \*p,\*prear;

prear=l;

for(int i=1;i<=len;i++)

{

cin>>a[i];

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

p->data=a[i];

prear->next=p;

prear=p;

}

prear->next=NULL;

return l;//返回头结点

}

//使用尾插法插入链表

bool lmerge(linklist la,linklist lb,linklist &lc)

{

lnode \*pa,\*pb,\*prear;//尾项指针

if(la==NULL||lb==NULL)

{

return false;

}//la,lb空链表返回

pa=la->next;

pb=lb->next;

lc=la;

prear=lc;

lc->next=NULL;

int len=0;

len=min(a[0],b[0]) ;

//先处理a,b都有的项数

for(int i=1;i<=len;i++)

{

//一次循环中先插入a，再插入b

prear->next=pa;

prear=pa;

pa=pa->next;

prear->next=pb;

prear=pb;

pb=pb->next;

//cout<<prear->data<<endl;

}

while(pa)

{

//a项数多则按顺序放入

prear->next=pa;

prear=pa;

pa=pa->next;

}

while(pb)

{

//b项数多则按顺序放入

prear->next=pb;

prear=pb;

pb=pb->next;

// cout<<"?"<<prear->data<<endl;

}

prear->next=NULL;

return true;

}

//结果输出函数

void lprint(linklist l)

{

lnode\* p;

if(l==NULL||l->next==NULL)

{

//表明为空结点

return;

}

p=l->next;

while(p!=NULL)

{

cout<<p->data<<" ";

p=p->next;

}

cout<<endl;

}

int main()

{

cin>>k;

while(k--)

{

memset(a,0,sizeof(a));

memset(b,0,sizeof(b));

cin>>a[0]>>b[0];

la=lcreate(a,a[0]);

lb=lcreate(b,b[0]);

//lprint(la);

//lprint(lb);

lnode\* lc=(lnode\*)malloc(sizeof(lnode));

lmerge(la,lb,lc);

lprint(lc);

free(lc);

}

return 0;

}