第一章 绪论

一、选择题

1.组成数据的基本单位是（ ）

（A）数据项（B）数据类型（C）数据元素（D）数据变量

2.数据结构是研究数据的（ ）以及它们之间的相互关系。

（A）理想结构，物理结构 （B）理想结构，抽象结构

（C）物理结构，逻辑结构 （D）抽象结构，逻辑结构

3.在数据结构中，从逻辑上可以把数据结构分成（ ）

（A）动态结构和静态结构 （B）紧凑结构和非紧凑结构

（C）线性结构和非线性结构（D）内部结构和外部结构

4.数据结构是一门研究非数值计算的程序设计问题中计算机的 （①）以及它们之间的（②）和运算等的学科。

① （A）数据元素（B）计算方法（C）逻辑存储（D）数据映像

② （A）结构 （B）关系 （C）运算 （D）算法

5.算法分析的目的是（）。

（A） 找出数据结构的合理性 （B）研究算法中的输入和输出的关系

（C）分析算法的效率以求改进（D）分析算法的易懂性和文档性

6.计算机算法指的是（①）,它必须具备输入、输出和（②）等 5 个特性。

① （A）计算方法（B）排序方法（C）解决问题的有限运算序列（D）调度方法

② （A）可执行性、可移植性和可扩充性（B）可行性、确定性和有穷性

（C）确定性、有穷性和稳定性 （D）易读性、稳定性和安全性

二、判断题

1.数据的机内表示称为数据的存储结构。（ ）

2.算法就是程序。（ ）

3.数据元素是数据的最小单位。（ ）

4.算法的五个特性为：有穷性、输入、输出、完成性和确定性。（ ）

5.算法的时间复杂度取决于问题的规模和待处理数据的初态。（ ）

三、填空题

1.数据逻辑结构包括_____、_____、_____ 和_____四种类型,其中树形结构和图形结构合称为_____。

2.在线性结构中,第一个结点____前驱结点,其余每个结点有且只有_____个前驱结点；最后一个结点_____后续结点,其余每个结点有且只有_____个后续结点。

3.在树形结构中,树根结点没有_____结点,其余每个结点有且只有_____个前驱结点；叶子结点没有_____结点,其余每个结点的后续结点可以_____。

4.在图形结构中,每个结点的前驱结点数和后续结点数可以_____。

5.线性结构中元素之间存在_____关系,树形结构中元素之间存在_____关系,图形结构中元素之间存在_____关系。

6.算法的五个重要特性是_____、_____、_____、_____、_____。

7.数据结构的三要素是指_____、_____和_____。

8.链式存储结构与顺序存储结构相比较,主要优点是_____。

9.设有一批数据元素,为了最快的存储*元素,数据结构宜用_____结构,为了方便插入一个元素,数据结构宜用_____结构。

四、算法分析题

1.求下列算法段的语句频度及时间复杂度

参考答案:

一、选择题

1. C 2.C 3. C 4. A、B 5. C 6.C、B

二、判断题:

1、√ 2、 × 3、× 4、× 5、√

三、填空题

1、线性、树形、图形、集合 ；非线性（网状） 2、没有；1；没有；1 3、前驱；1；后继；任意多个 4、任意多个 5、一对一；一对多；多对多 6、有穷性；确定性；可行性；输入；输出 7、数据元素；逻辑结构；存储结构 8、插入、删除、合并等操作较方便 9、顺序存储；链式存储

四、算法分析题

for(i=1; i<=n; i++)
for(j =1; j <=i ; j++)
=+1;
分析：该算法为一个二重循环，执行次数为内、外循环次数相乘，但内循环次数不固定，与外循环有关，因些，时间频度 T(n)=1+2+3+… +n=n*(n+1)/2
有 1/4≤T(n)/n2≤ 1，故它的时间复杂度为 O(n2), 即 T(n)与 n2 数量级相同。 2、分析下列算法段的时间频度及时间复杂度
for （i=1;i<=n;i++ ）
for (j=1;j<=i;j++)
for (k=1;k<=j;k++)
*=i+j-k;
分析算法规律可知时间频度 T(n)=1+(1+2)+(1+2+3)+…+(1+2+3+… +n)
由于有 1/6 ≤T□ n）/ n3 ≤ 1，故时间复杂度为 O(n3)

第二章 线性表

一、选择题

1.一个线性表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

2. 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）64（B）63 （C）63.5 （D）7

3.线性表采用链式存储结构时，其地址（ ）。

(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) 连续与否均可以

4. 在一个单链表中，若 p 所指结点不是最后结点，在 p 之后插入 s 所指结点，则执行（ ）

（A）s->ne*t=p;p->ne*t=s; （B） s->ne*t=p->ne*t;p->ne*t=s;

（C）s->ne*t=p->ne*t;p=s; （D）p->ne*t=s;s->ne*t=p;

5.在一个单链表中，若删除 p 所指结点的后续结点，则执行（ ）

（A）p->ne*t=p->ne*t->ne*t; （B）p=p->ne*t; p->ne*t=p->ne*t->ne*t;

（C）p->ne*t=p->ne*t; （D）p =p->ne*t->ne*t;

6.下列有关线性表的叙述中，正确的是（ ）

（A）线性表中的元素之间隔是线性关系

（B）线性表中至少有一个元素

（C）线性表中任何一个元素有且仅有一个直接前趋

（D）线性表中任何一个元素有且仅有一个直接后继

7.线性表是具有 n 个（ ）的有限序列（n≠ 0）

（A）表元素 （B）字符 （C）数据元素 （D）数据项

二、判断题

1.线性表的链接存储，表中元素的逻辑顺序与物理顺序一定相同。（ ）

2.如果没有提供指针类型的语言，就无法构造链式结构。（ ）

3.线性结构的特点是只有一个结点没有前驱，只有一个结点没有后继，其余的结点只有一个前驱和后继。（ ）

4.语句 p=p->ne*t 完成了指针赋值并使 p 指针得到了 p 指针所指后继结点的数据域值。（ ）

5.要想删除 p 指针的后继结点，我们应该执行 q=p->ne*t ； p->ne*t=q->ne*t； free(q)。（ ）

三、填空题

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：

_____ 。

2.顺序表中逻辑上相邻的元素物理位置（ 相邻， 单链表中逻辑上相邻的元素物理位置

_____相邻。

3.线性表 L＝（a1，a2，..，an）采用顺序存储，假定在不同的 n＋1 个位置上插入的概率相同，则插入一个新元素平均需要移动的元素个数是_____

4.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->ne*t=p;

p->ne*t=q;

_____;

5.已知 L 是无表头结点的单链表，是从下列提供的答案中选择合适的语句序列，分别实现：

（1）表尾插入 s 结点的语句序列是_____

(2) 表尾插入 s 结点的语句序列是_____

1. p->ne*t=s;
2. p=L;
3. L=s;
4. p->ne*t=s->ne*t;
5. s->ne*t=p->ne*t;
6. s->ne*t=L;

7. s->ne*t=null;

8.

9. while(p->ne*t!=null) p=p->ne*t;

四、算法设计题

1.试编写一个求已知单链表的数据域的平均值的函数（数据域数据类型为整型）。

2.已知带有头结点的循环链表中头指针为 head,试写出删除并释放数据域值为*的所有结点的 c 函数。

3.*百货公司仓库中有一批电视机，按其价格从低到高的次序构成一个循环链表，每个结点有价格、数量和链指针三个域。现出库（销售）m 台价格为 h 的电视机，试编写算法修改原链表。

4.*百货公司仓库中有一批电视机，按其价格从低到高的次序构成一个循环链表，每个结点有价格、数量和链指针三个域。现新到 m 台价格为 h 的电视机，试编写算法修改原链表。

5.线性表中的元素值按递增有序排列，针对顺序表和循环链表两种不同的存储方式，分别编写 C 函数删除线性表中值介于 a 与 b（a≤ b□ 之间的元素。

6.设 A=(a0,a1,a2,...,an-1),B=(b0,b1,b2,...,bm-1)是两个给定的线性表,它们的结点个数分别是 n 和 m,且结点值均是整数。

若 n=m，且 ai= bi （0≤i<n ），则 A=B;

若 n<m ，且 ai=bi （0≤i<n ），则 A<B;

若存在一个 j， j<m ，j<n ，且 ai=bi （0≤i<j ）， 若 aj<bj，则 A<B，否则 A>B。

试编写一个比较 A 和 B 的 C 函数,该函数返回 -1或 0 或 1,分别表示 A<B 或 A=B 或 A>B。

7.试编写算法,删除双向循环链表中第 k 个结点。

8.线性表由前后两部分性质不同的元素组成(a0,a1,...,an-1,b0,b1,...,bm-1),m 和 n 为两部分元素的个数,若线性表分别采用数组和链表两种方式存储,编写算法将两部分元素换位成(b0,b1,...,bm-1,a0,a1,...,an-1),分析两种存储方式下算法的时间和空间复杂度。

9.用循环链表作线性表(a0,a1,...,an-1)和（b0,b1,...,bm-1）的存储结构,头指针分别为 ah 和 bh，设计 C 函数，把两个线性表合并成形如（a0,b0,a1,b1,...）的线性表，要求不开辟新的动态空间,利用原来循环链表的结点完成合并操作,结构仍为循环链表,头指针为 head,并分析算法的时间复杂度。

10.试写出将一个线性表分解为两个带有头结点的循环链表，并将两个循环链表的长度放在各自的头结点的数据域中的 C 函数。其中,线性表中序号为偶数的元素分解到第一个循环链表中，

序号为奇数的元素分解到第二个循环链表中。

11.试写出把线性链表改为循环链表的 C 函数。

12.已知非空线性链表中*结点的直接前驱结点为 y,试写出删除*结点的 C 函数。

参考答案：

一、选择题

1. B 2.C 3. D 4. B 5. A 6.A 7. C

二、判断题：

参考答案：1、×2、√3、×4、×5、√

三、填空题

1、s->ne*t=p->ne*t; p->ne*t=s; 2、一定；不一定 3、n/2 4、q->prior=p; 5、(1)6) 3)

(2) 2))91) 7)

四、算法设计题

1、

```
typedef struct node
{int data;
struct node *link;
}NODE;
int aver(NODE *head)
{int i=0,sum=0,ave; NODE *p;
p=head;
while(p!=NULL)
{p=p->link;++i;
sum=sum+p->data;}
```

ave=sum/i;

return (ave);}

2、

```
typedef struct node
{
int data; /* 假设数据域为整型 */
struct node *link;
}NODE;
void del_link(NODE *head,int *) /* 删除数据域为*的结点*/
{
NODE *p,*q,*s;
p=head;
q=head->link;
while(q!=head)
{if(q->data==*)
{p->link=q->link;
s=q;
q=q->link;
free(s);}
else
{
p=q;
q=q->link;
}
}
}
```

3、

```
void del(NODE *head,float price,int num)
```

```
{
NODE *p,*q,*s;
p=head;q=head->ne*t;
while(q->price<price&&q!=head)
{
p=q;
q=q->ne*t;
}
if(q->price==price)
q->num=q->num-num;
else
        无此产品
if(q->num==0)
{
p->ne*t=q->ne*t;
free(q);
}
}
```

4、

```
typedef struct node
{
float price;
int num;
struct node *ne*t;
}NODE;
void ins(NODE *head,float price,int num)
{
NODE *p,*q,*s;
```

```
p=head;q=head->ne*t;
while(q->price<price&&q!=head)
{
p=q;
q=q->ne*t;
}
if(q->price==price)
q->num=q->num+num;
else
{
s=(NODE *)malloc(sizeof(NODE));
s->price=price;
s->num=num;
s->ne*t=p->ne*t;
p->ne*t=s;
}
}
```

5、顺序表：

算法思想：从 0 开始扫描线性表，用 k 记录下元素值在 a 与 b 之间的元素个数，对于不满足该条件的元素，前移 k 个位置，最后修改线性表的长度。

```
void del（elemtype list[]，int *n，elemtype a，elemtype b）
{
int i=0，k=0；
while（i<n)
{
if(list[i]>=a&&list[i]<=b) k++;
else
list[i-k]=list[i];
i++;
}
```

修改线性表的长度*/

}

循环链表:

```
void del(NODE *head,elemtype a,elemtype b)
{
NODE *p,*q;
p= head;q=p->link; /*  假设循环链表带有头结点  */
while(q!=head && q->data<a)
{
p=q;
q=q->link;
}
while(q!=head && q->data<b)
{
r=q;
q=q->link;
free(r);
}
if(p!=q)
p->link=q;
}
```

6、

```
*define MA*SIZE 100
int listA[MA*SIZE],listB[MA*SIZE];
int n,m;
int compare(int a[],int b[])
{
int i=0;
while(a[i]==b[i]&&i<n&&i<m)
i++;
```

```
if(n<m&&i==n) return(-1);
if(n>m&&i==m) return(1);
if(i<n&&i<m)
if(a[i]<b[i]) return(-1);
else if(a[i]>b[i]) return(1);
}
```

7、

```
void del(DUNODE    **head,int i)
{
DUNODE *p;
if(i==0)
{
*head=*head->ne*t;
*head->prior=NULL;
return(0);
}
Else
{for(j=0;j<i&&p!=NULL;j++)
p=p->ne*t;
if(p==NULL||j>i) return(1);
p->prior->ne*t=p->ne*t;
p->ne*t->prior=p->proir;
free(p);
return(0);
}
```

8.

顺序存储:

```
void convert(elemtype list[],int l,int h) /*将数组中第 l 个到第 h 个元素逆置*/
{
```

```
elemtype temp;
for(i=h;i<=(l+h)/2;i++)
{
temp=list[i];
list[i]=list[l+h-i];
list[l+h-i]=temp;
}
}
void e*change(elemtype list[],int n,int m);
{
convert(list,0,n+m-1);
convert(list,0,m-1);
convert(list,m,n+m-1);
}
```

该算法的时间复杂度为 O(n+m),空间复杂度为 O(1)

链接存储:(不带头结点的单链表)

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void convert(NODE **head,int n,int m)
{
NODE *p,*q,*r;
int i;
p=*head;
q=*head;
for(i=0;i<n-1;i++)
q=q->link; /*q 指向 an-1 结点 */
r=q->link;
```

```
while(r->link!=NULL)
r=r->link; /*r 指向最后一个 bm-1 结点 */
*head=q;
r->link=p;
}
```

该算法的时间复杂度为 O(n+m),但比顺序存储节省时间(不需要移动元素,只需改变指针),空间复杂度为 O(1)

9.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
NODE    *union(NODE    *ah,NODE *bh)
{
NODE    *a,*b,*head,*r,*q;
head=ah;
a=ah;
b=bh;
while(a->link!=ah&&b->link!=bh)
{
r=a->link;
q=b->link;
a->link=b;
b->link=r;
a=r;
b=q;
}
if(a->link==ah) /*a 的结点个数小于等于 b 的结点个数 */
{
```

```
while(b->link!=bh)
b=b->link;
b->link=head;
}
if(b->link==bh) /*b 的结点个数小于 a 的结点个数 */
{
r=a->link;
a->link=b;
b->link=r;
}
return(head);
}
```

该算法的时间复杂度为 O(n+m),其中 n 和 m 为两个循环链表的结点个数.

10.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void analyze(NODE *a)
{
NODE   *rh，*qh，*r,*q,*p;
int i=0，j=0；/*i 为序号是奇数的结点个数 j 为序号是偶数的结点个数 */
p=a；
rh=（NODE *）malloc（sizeof（NODE））；/*rh 为序号是奇数的链表头指针 */
qh=(NODE *)malloc(sizeof(NODE)); /*qh 为序号是偶数的链表头指针 */
r=rh;
q=qh;
while(p!=NULL)
{
```

```
r=p;
i++;
p=p->link;
if(p!=NULL)
{
q->link=p;
q=p;
j++;
p=p->link;
}
}
rh->data=i;
r->link=rh;
qh->data=j;
q->link=qh;
}
```

11.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void change(NODE    *head)
{
NODE     *p;
p=head;
if(head!=NULL)
{
while(p->link!=NULL)
p=p->link;
```

```
p->link=head;
}
}
```

12.

```
typedef struct node
{
elemtype data;
struct node *link;
}NODE;
void del(NODE **,NODE *y)
{
NODE *p,*q;
elemtype d1;
p=y;
q=*;
while(q->ne*t!=NULL) /* 把后一个结点数据域前移到前一个结点*/
{
p->data=q->data;
q=q->link;
p=q;
p->link=NULL; /*  删除最后一个结点*/
free(q);
}
```

第三章 栈和队列一、选择题

1. 一个栈的入栈序列是 a,b,c,d,e,则栈的不可能的输出序列是（ ）。

（A） edcba（B）decba（C）dceab （D）abcde

2.栈结构通常采用的两种存储结构是（ ）。

（A） 线性存储结构和链表存储结构（B）散列方式和索引方式

（C）链表存储结构和数组 （D）线性存储结构和非线性存储结构

3.判定一个栈 ST(最多元素为 m0)为空的条件是（ ）。

（A） ST-〉top!=0 （B）ST-〉top==0

（C）ST-〉top!=m0 （D）ST-〉top=m0

4.判定一个栈 ST(最多元素为 m0)为栈满的条件是（ ）。

（A）ST->top!=0 （B）ST->top==0

（C）ST->top!=m0-1（D）ST->top==m0-1

5.一个队列的入列序列是 1,2,3,4,则队列的输出序列是（ ）。

（A）4,3,2,1（B）1,2,3,4（C）1,4,3,2（D）3,2,4,1

6.循环队列用数组 A[0,m-1]存放其元素值,已知其头尾指针分别是 front 和 rear 则当前队列中的元素个数是（ ）

（A）(rear-front+m)%m （B） rear-front+1 （C）rear-front-1（D）rear-front

7.栈和队列的共同点是（ ）

（A） 都是先进后出 （B）都是先进先出

（C）只允许在端点处插入和删除元素（D）没有共同点

8.表达式 a*(b+c)-d 的后缀表达式是（ ）。

（A）abcd*+-（B）abc+*d- （C）abc*+d-（D）-+*abcd

9.4个元素 a1，a2，a3 和 a4 依次通过一个栈，在 a4 进栈前，栈的状态，则不可能的出栈序是（ ）

(A)a4，a3，a2，a1 (B)a3，a2，a4，a1

(C)a3，a1，a4，a2 (D)a3，a4，a2，a1

10.以数组 Q[0..m－1]存放循环队列中的元素，变量 rear 和 qulen 分别指示循环队列中队尾元素的实际位置和当前队列中元素的个数，队列第一个元素的实际位置是（ ）

(A)rear－qulen (B)rear－qulen＋m

(C)m－qulen (D)1＋（rear＋m－qulen）% m

二、填空题

1.栈的特点是_____,队列的特点是_____。

2.线性表、栈和队列都是_____结构,可以在线性表的_____位置插入和删除元素，对于栈只能在_____插入和删除元素，对于队列只能在_____插入元素和_____删除元素。

3.一个栈的输入序列是 12345,则栈有输出序列 12345 是_____。(正确/错误)

4.设栈 S 和队列 Q 的初始状态皆为空，元素 a1，a2，a3，a4，a5 和 a6 依次通过一个栈，一个元素出栈后即进入队列 Q，若 6 个元素出队列的顺序是 a3，a5，a4，a6，a2，a1 则栈 S 至少

应该容纳_____个元素。

三、算法设计题

1.假设有两个栈 s1 和 s2 共享一个数组 stack[M],其中一个栈底设在 stack[0]处，另一个栈底设在 stack[M-1]处。试编写对任一栈作进栈和出栈运算的 C 函数 push（*,i)和 pop(i),i=l,2。其中 i=1 表示左边的栈，,i=2 表示右边的栈。要求在整个数组元素都被占用时才产生溢出。

2.利用两个栈 s1,s2 模拟一个队列时,如何用栈的运算来实现该队列的运算 写出模拟队列的插入和删除的 C 函数。

一个栈 s1 用于插入元素,另一个栈 s2 用于删除元素.

参考答案：一、选择题

1. C 2.A 3. B 4. B 5. B 6.B 7 C 8、C 9、C 10、D

二、填空题

1、先进先出；先进后出 2、线性 ； 任何 ；栈顶；队尾；对头 3、正确的 4、3

三、算法设计题

1.

```
*define M 100
elemtype stack[M];
int top1=0,top2=m-1;
int push(elemtype *,int i)
{
if(top1-top2==1) return(1); /*上溢处理*/
else
if(i==1) stack[top1++]=*;
if(i==2)stack[top2--]=*;
return(0);
}
int pop(elemtype *p*,int i)
```

{

if(i==1)

if(top1==0) return(1);

else

{

top1--;

p=stack[top1];

return(0);

}

else

if(i==2)

if(top2==M-1) return(1);

else

{

top2++;

p=stack[top2];

return(0);

}

}

2.

elemtype s1[MA*SIZE],s2[MAZSIZE];

int top1,top2;

void enqueue(elemtype *)

{

if(top1==MA*SIZE) return(1);

else

{

push(s1,*);

return(0);

}}

void dequeue(elemtype *p*)

{

elemtype *;

top2=0;

while(!empty(s1))

{

pop(s1,&*);

push(s2,*);

}

pop(s2,&*);

while(!empty(s2))

{

pop(s2,&*);

push(s1,*);

}

}

第四章串一、选择题

1.下列关于串的叙述中，正确的是（ ）

(A)一个串的字符个数即该串的长度 (B)一个串的长度至少是 1

(C)空串是由一个空格字符组成的串 (D)两个串 S1 和 S2 若长度相同，则这两个串相等

2.字符串 的 ne*tval值为

(A)(0,1,01,1,0,4,1,0,1) (B)(0,1,0,0,0,0,2,1,0,1)

(C)(0,1,0,1,0,0,0, 1,1) (D)(0,1,0,1,0,1,0,1,1)

3.字符串满足下式,其中 head 和 tail的定义同广义表类似,如 head(‘ *yz’)=

‘*’ ,tail(‘ *yz’)= ‘yz’ ,则s=( )。 concat(head(tail(s)),head(tail(tail(s))))= ‘dc’ 。

(A)abcd (B)acbd (C)acdb (D)adcb

4.串是一种特殊的线性表,其特殊性表现在（ ）

(A)可以顺序存储 (B)数据元素是一个字符

(C)可以链式存储 (D)数据元素可以是多个字符

5．设串 S1=‘ ABCDEFG’ ,s2=‘ PQRST’ ,函数 CONCAT（*，Y）返回*和 Y 串的连接串，SUBSTR（S，I，J）返回串 S 从序号 I 开始的 J 个字符组成的字串，LENGTH（S）返回串 S

的长度，则 CONCAT（SUBSTR（S1，2，LENGTH（S2））， SUBSTR（S1，LENGTH（S2），2））的结果串是（ ）

（A）BCDEF (B) BCDEFG (C)BCPQRST (D)BCDEFEF

二、算法设计

1.分别在顺序存储和一般链接存储两种方式下,用 C 语言写出实现把串 s1 复制到串 s2 的串复制函数 strcpy(s1,s2)。

2.在一般链接存储(一个结点存放一个字符)方式下,写出采用简单算法实现串的模式匹配的 C 语言函数 int L_inde*(t,p)。

参考答案：

一、选择题

1. A 2.B 3. D 4. D 5. D

二、算法设计

1.

顺序存储：

```
*define MA*N 100
char s[MA*N];
int S_strlen(char s[])
{
int i;

return(i);
}
void S_strcpy(char s1[],char s2[]) //4.3题
{
int i;

s2[i]=s1[i];
```

}

一般链接存储：

```
typedef struct node
{
char data;
struct node *link;
}NODE;
NODE *L_strcpy(NODE *s1)
{
NODE *s2,*t1,*t2,*s;
if(s1==NULL) return(NULL);
else
{
t1=s1;
t2=(NODE *)malloc(sizeof(NODE));
s2=t2;
while(t1!=NULL)
{
s=(NODE *)malloc(sizeof(NODE));
s->data=t1->data;
t2->link=s;
t2=s;
t1=t1->link;
}
t2->link=NULL;
s=s2;
s2=s2->link;
free(s);
return(s2);
```

}

}

2.

```
typedef struct node
{
char data;
struct node *link;
}NODE;
int L_inde*(NODE *t,NODE *p)
{
NODE *t1,*p1,*t2;

t1=t;i=1;
while(t1!=NULL)
{
p1=p;
t2=t1->link;
while(p1->data==t1->data&&p1!=NULL)
{
p1=p1->link;
t1=t1->link;
}
if(p1==NULL) return(i);
i++;
t1=t2;
}
return(0);
}
```

第五章 数组和广义表

一、选择题

1. 常对数组进行的两种基本操作是（ ）

（A）建立与删除（B）索引和修改（C）查找和修改（D）查找与索引

2.二维数组 M 的元素是 4 个字符(每个字符占一个存储单元)组成的串,行下标 i 的范围从 0 到 4,列下标 j 的范围从 0 到 5,M 按行存储时元素 M[3][5]的起始地址与 M 按列存储时元素（ ）的起始地址相同。

（A）M[2][4]（B）M[3][4]（C）M[3][5]（D）M[4][4]

3.数组 A[8][10]中,每个元素 A 的长度为 3 个字节,从首地址 SA 开始连续存放在存储器内,存放该数组至少需要的单元数是（ ）。

（A）80（B）100（C）240（D）270

4.数组 A[8][10]中,每个元素 A 的长度为 3 个字节,从首地址 SA 开始连续存放在存储器内,该数组按行存放时,元素 A[7][4]的起始地址为（ ）。

（A）SA+141（B）SA+144（C）SA+222（D）SA+225

5.数组 A[8][10]中,每个元素 A 的长度为 3 个字节,从首地址 SA 开始连续存放在存储器内,该数组按列存放时,元素 A[4][7]的起始地址为（ ）。

（A）SA+141（B）SA+180（C）SA+222（D）SA+225

6.稀疏矩阵一般的压缩存储方法有两种,即（ ）。

（A） 二维数组和三维数组（B）三元组和散列

（C）三元组和十字链表 （D）散列和十字链表

7.若采用三元组压缩技术存储稀疏矩阵,只要把每个元素的行下标和列下标互换,就完成了对该矩阵的转置运算,这种观点（ ）。

（A）正确（B）错误

8.设矩阵 A 是一个对称矩阵,为了节省存储,将其下三角部分按行序存放在一维数组 B[1,n(n-1)/2]中,对下三角部分中任一元素 ai,j(i<=j),在一组数组 B 的下标位置 k 的值是（ ）。

（A）i(i-1)/2+j-1（B）i(i-1)/2+j（C）i(i+1)/2+j-1 (D)i(i+1)/2+j

二、填空题

1.已知二维数组 A[m][n]采用行序为主方式存储,每个元素占 k 个存储单元,并且第一个元素的存储地址是 LOC(A[0][0]),则 A[0][0]的地址是_____。

2.二维数组 A[10][20]采用列序为主方式存储,每个元素占一个存储单元,并且 A[0][0]的存储地址是 200,则 A[6][12]的地址是_____。

3.有一个 10 阶对称矩阵 A,采用压缩存储方式(以行序为主,且 A[0][0]=1),则 A[8][5]的地址是_____。

4.设 n 行 n 列的下三角矩阵 A 已压缩到一维数组 S[1..n*(n+1)/2]中,若按行序为主存储,则 A[i][j]对应的 S 中的存储位置是_____。

5.若 A 是按列序为主序进行存储的 4×6 的二维数组,其每个元素占用 3 个存储单元,并且 A[0][0]的存储地址为 1000,元素 A[1][3]的存储地址为_____,该数组共占用_____个存储单元。

三、算法设计

1.如果矩阵 A 中存在这样的一个元素 A[i][j]满足条件:A[i][j]是第 i 行中值最小的元素,且又是第 j 列中值最大的元素,则称之为该矩阵的一个马鞍点。编写一个函数计算出 1×n 的矩阵 A 的所有马鞍点。

2.n 只猴子要选大王,选举办法如下:所有猴子按 1,2,...,n 编号围坐一圈,从 1 号开始按 1、2、...、m 报数,凡报 m 号的退出到圈外,如此循环报数,直到圈内剩下只猴子时,这只猴子就是大王。n 和 m 由键盘输入,打印出最后剩下的猴子号。编写一程序实现上述函数。

3.数组和广义表的算法验证程序

编写下列程序:

(1)求广义表表头和表尾的函数 head()和 tail()。

(2)计算广义表原子结点个数的函数 count_GL()。

(3)计算广义表所有原子结点数据域(设数据域为整型〉之和的函数 sum_GL()。

参考答案:一、选择题

1. C 2.B 3. C 4. C 5. B 6.C 7、B 8、B

二、填空题

1、loc(A[0][0])+(n*i+j)*k 2、33 3、42 4、i*(i+1)/2+j+1 5、1039;72

三、算法设计题

1.

算法思想:依题意,先求出每行的最小值元素,放入 min[m]之中,再求出每列的最大值元素,放入 ma*[n]之中,若*元素既在 min[i]中,又在 ma*[j]中,则该元素 A[i][j]便是马鞍点,找出所有这样的元

,即找到了所有马鞍点。因此,实现本题功能的程序如下:

```
*include <stdio.h>
*define m 3
*define n 4
void minma*(int a[m][n])
{
int i1,j,have=0;
int min[m],ma*[n];
for(i1=0;i1<m;i1++)/*计算出每行的最小值元素,放入 min[m]之中*/
{
min[i1]=a[i1][0];
for(j=1;j<n;j++)
if(a[i1][j]<min[i1]) min[i1]=a[i1][j];
}
for(j=0;j<n;j++)/*计算出每列的最大值元素,放入 ma*[n]之中*/
{
ma*[j]=a[0][j];
for(i1=1;i1<m;i1++)
if(a[i1][j]>ma* [j]) ma*[j]=a[i1][j];
}
for(i1=0;i1<m;i1++)
for(j=0;j<n;j++)
if(min[i1]==ma*[j])
{


have=1;
}
                没有鞍点\
}
```

2.

算法思想:本题用一个含有 n 个元素的数组 a,初始时 a[i]中存放猴子的编号 i,计数器似的值为 0。