**一、填空**

1. 数据结构主要包括\_\_\_\_数据的存储结构\_\_\_\_\_\_\_\_\_、\_\_\_\_\_数据的逻辑结构\_\_\_\_\_\_\_\_\_\_\_\_及数据的基本运算三方面内容。
2. 下列程序的时间复杂度为 \_\_O(n^3)\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。f(n) = n^2\*(n+1)/2-n\*(n^2-1)/6

int f(int n)

{ int i,j，k,sum= 0;

for(i=l; i<n+1;i++)

{

for(j=n;j>i-1; j--)

for(k=1;k<j+1;k++ )

sum++;

printf("sum=%d\n",sum)；

}

return (sum);

}

1. 顺序存储方式是通过\_\_\_元素的相对物理位置\_\_\_\_\_\_\_来表示数据元素之间的逻辑关系的。
2. 链式存储方式是利用\_\_\_\_指针域\_\_\_\_\_\_来表示数据元素之间的逻辑关系的。
3. 若某表最常用的操作是在最后一个结点之后插入一个结点或删除最后一个结点。则采用\_\_\_\_链式\_\_\_\_\_\_存储方式最节省运算时间。
4. 在循环队列中用数组*A*[0..*m*-1] 存放队列元素，其队头和队尾指针分别为*front*和*rear*，则当前队列中的元素个数是\_(rear – front + m) % m \_\_\_\_\_\_\_\_\_\_\_。
5. 递归过程或函数调用时，处理参数及返回地址，要用一种称为\_\_\_\_栈\_\_\_\_\_\_\_的数据结构。
6. 两个字符串相等的充分必要条件是\_\_长度相等且对应位置上的字符相等\_\_。
7. 矩阵压缩存储的原则是\_\_\_多个相同的非零元素只分配一个存储空间\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_对零元素不分配存储空间\_\_\_\_\_\_\_\_\_\_\_\_。
8. 数组A[5][7]若第一个元素A[1][1]的存储地址为100，每个元素占4个字节，则元素A[3][5]以行优先存储地址是\_\_176\_\_\_，列优先的存储地址是\_\_188\_\_\_\_。
9. 若将n阶三对角矩阵A压缩存储在一维数组B中，则数组B需要\_3n-2\_\_\_\_\_个存储空间。
10. 对称矩阵A压缩存储在SA[ ]中，若第一个元素A[1][1]存储在SA[0]，则A[3][5]存储在SA[ \_\_12\_\_ ]。

**二、 简答题**

1. 用链表表示线性表的优点是？

（1）插入和删除比较灵活，不需要大量移动结点

（2）动态分配空间比较灵活，不需要预先申请最大的连续空间。

1. 什么是顺序队列的“假溢出”现象？如何解决？

（1）顺序队列的“假溢出”现象：当元素被插入到数组中下标最大的位置上之后，队列的空间就用尽了，尽管此时数组的低端还有空闲空间，这种现象叫做假溢出。

（2）解决办法：将存储队列的数组头尾相接，形成循环队列。队头、队尾指针加1时用语言的取模(余数)运算实现。

队头指针进1: q.front = (q.front+1) % max\_size;

队尾指针进1: q.rear = (q.rear+1) % max\_size;

1. 运算是数据结构的一个重要方面。试举一例，说明两个数据结构的逻辑结构和存储方式完全相同，只是对于运算的定义不同。因而两个结构具有显著不同的特性，是两个不同的结构。

栈和队列的逻辑结构相同，其存储表示也可相同（顺序存储和链式存储），但由于其运算集合不同而成为不同的数据结构。

1. 什么是模式匹配？KMP算法对比BF算法，有哪些改进？已知模式串t为：”abababc”，求出next[]的值。

模式匹配是数据结构中字符串的一种基本运算，给定一个子串，要求在某个字符串中找出与该子串相同的所有子串，这就是模式匹配。

KMP算法的核心是利用匹配失败后的信息，尽量减少模式串与主串的匹配次数以达到快速匹配的目的。具体实现就是通过一个next数组实现，数组本身包含了模式串的局部匹配信息。通过next数组可以有效避免主串循环变量的回溯，从而提高了匹配效率。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 模式串P | a | b | a | b | a | b | c |
| 下标i | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| next[i] | -1 | 0 | 0 | 1 | 2 | 3 | 4 |

**三、按要求解答：**

1. 已知对栈S入栈操作为push(e)，出栈操作为pop(&e)，经过如下：

S.push(11); S.push(22);

S.pop(e);

S.push(33);S.push(44);

S.pop(e); S.push(55);S.pop(e);

则要求写出依次出栈的数据，并画出上述操作后栈中的状态。

22,44,55

|  |
| --- |
|  |
| 33 |
| 11 |

1. 已知上三角矩阵A（5×5）压缩在数组sk[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16},要求画出矩阵A。

1 2 3 4 5

0 6 7 8 9

0 0 10 11 12

0 0 0 13 14

0 0 0 0 15

1. 已知线性表L，试编写算法判断元素是否是递增。要求自选存储方式，并写出数据类型。

//返回0表示不是递增， 返回1表示是递增，采用顺序表存储，数据类型为整型

int is\_incremental(int l[], int len)

{

int max = l[0];

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

{

if (max <= l[i])

{

max = l[i];

}

else

{

return 0;

}

}

return 1;

}

1. 已知双向链表L，要求写出程序段实现在结点r之前插入结点p的操作。

p->prev = r->prev;

if ( r->prev!= NULL){

r->prev->next = p;

}

p->next = r;

r->prev = p;

1. 已知稀疏矩阵A如下，要求分别画出其压缩后的三元组表和十字链表。

7 0 0 0 0

0 2 0 9 0

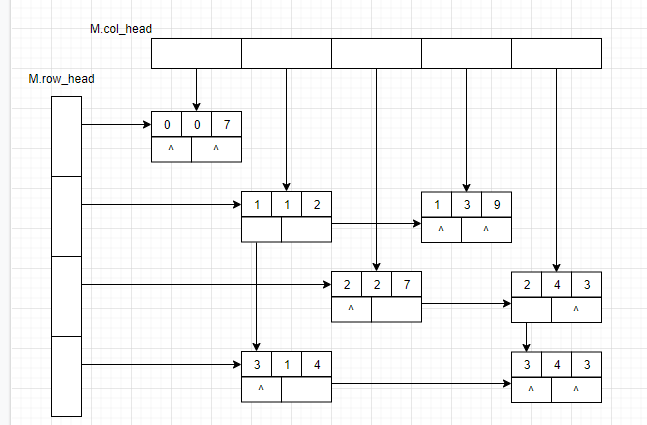
0 0 7 0 3

0 4 0 0 3

三元组表:

(0,0,7),(1,1,2),(1,3,9),(2,2,7),(2,4,3),(3,1,4),(3,4,3)

十字链表：



1. 已知广义表: A=(a,b), B=(A), C=(a,(b，A),B), 要求
   1. 写出gettail(gethead(gettail(C)))的结果。
   2. 画出其单链表示法。

a) b

b) 