4/24/2023

循环队列的相关条件和公式：

队尾指针是rear,队头是front，其中QueueSize为循环队列的最大长度

1.队空条件：rear==front

2.队满条件：(rear+1) %QueueSIze==front

3.计算队列长度：（rear-front+QueueSize）%QueueSize

4.入队：（rear+1）%QueueSize

5.出队：（front+1）%QueueSize

下面哪项是数组优于链表的特点

A

方便删除

B

方便插入

C

长度可变

D

占用空间小

链表和数组的区别：

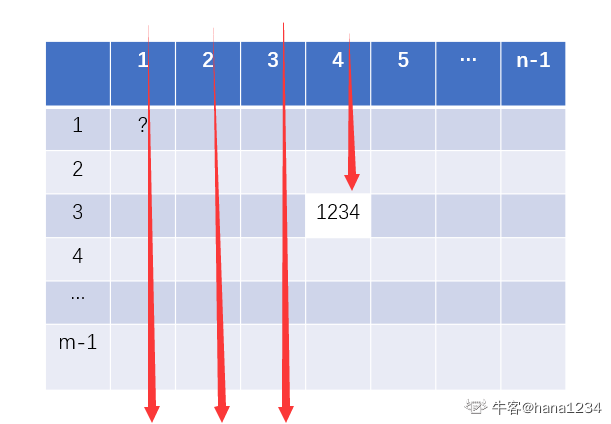
链表是链式存储结构，数组是顺序存储结构；

链表通过指针连接元素，而数组则是把所有元素按顺序进行存储；

链表插入和删除元素不需要移动元素，数组删除和增加元素需要移动元素。

ABC都是链表的特点，数组所需存储空间小于链表，所以正确答案选D

已知二维数组A[1: 4, 1: 6]采用列序为主序方式存储，每个元素占用4个存储单元，并且A[3，4]的存储地址为1234，元素A[1, 1]的存储地址是（）1178



每个元素占用4个存储单元，所以A[1,1]+(3\*4+2)\*4 =A[3,4]

1234-(3\*4+2)\*4=1178

以下代码执行后，array的结果是？

var array=[-1,1,3,4,6,10];

array.sort((a,b)=>Math.abs(a-3)-Math.abs(b-3));

array. sort(sortFunction)是按照sortFunction结果大小排序。 一言不合先把比较的对象定义在数组上方，再手工分析 class Untitled { public static void main(String[] args) { //4 2 0 1 3 7 var array=[-1,1,3,4,6,10]; array.sort((a,b)=>Math.abs(a-3)-Math.abs(b-3)); System.out.println(array); } } 发现最小的是算好后的第三个数字，对应原数组原位置上面是，3。看看选项中只有一个是以3排在前面的数组。好了，后面的不用比了。就是C

原数组var array=[-1,1,3,4,6,10];

参数(a,b)=>Math.abs(a-3)-Math.abs(b-3)：即数值减3的绝对值[4，2，0，1，3，7]作为参数决定排序。

根据绝对值减3大小排列[0，1，2，3，4，7]，对应的原数组排序后为[3，4，1，6，-1，10]

4/25/2023

java 数组初始化有三种方式：

动态初始化：数组定义与为数组分配空间和赋值的操作分开进行；

静态初始化：在定义数字的同时就为数组元素分配空间并赋值；

默认初始化：数组是引用类型，它的元素相当于类的成员变量，因此数组分配空间后，每个元素也被按照成员变量的规则被隐式初始化。

//动态初始化

int[] a;//int a[];两者都可以

a = new int[10];

a[0] = 0;

a[1] = 1;

a[2] = 2;

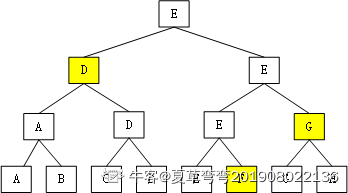
//静态初始化

int[] b = {0, 1, 2};//int b[] = {0, 1, 2};两者都可以

//默认初始化

int[] c = new int[10];//int c[] = new int[10];两者都可以

在一个有8个int数据的数组中，随机给出数组的数据，找出最大和第二大元素一定需要进行几次比较 9 次



第一次分组8个分成4组 两两一组 比4次 第二次分组为上面比较下来的4个还是两两分组 分2组 第三次分组就剩下两个直接比较出了最大 这时一共比较了4+2+1=7次 接着拿第三次分组中与最大比较的那个数分别和 之前与最大那个数比较过的数比较 分别是第一次分组有一个 第二次分组有一个 7+2=9

设有序表的关键字序列为{1，4，6，10，18，35，42，53，67，71，78，84，92，99}，当用二分查找法查找键值为35的结点时，经（）次比较后查找成功

C 第1次 1，4，6，10，18，35，42，53，67，71，78，84，92，99 left=0,right=13,mind=6; mid值为42，太大 第2次 1，4，6，10，18，35，42，53，67，71，78，84，92，99 left=0,right=5,mind=2; mid值为6，太小 第3次 1，4，6，10，18，35，42，53，67，71，78，84，92，99 left=3,right=5,mind=4; mid值为18，太小 第4次 1，4，6，10，18，35，42，53，67，71，78，84，92，99 left=5,right=5,mind=5; mid值为35，找到了

C++ 在下列语句中，正确的是

char a[] [3]={"a","i"}; 二维数组声明时一定要指定列数 单引号表示字符，不能用来给二维数赋值

线性表的顺序存储结构是一种（随机存取）的存储结构

所谓“随机存取”，指的是当存储器中的消息被读取或写入时，所需要的时间与这段信息所在的位置无关，所以叫随机。

将一个n×n的对称矩阵A的下三角部分按行存放在一个一维数组B中，

A[0][0]存放在B[0]中，那么第i行的对角元素A[i][i]在B中的存放位置是 (i+3)×i/2

就是求等差数列，因为是下三角矩阵原因 所以有a=1,d=1,n=n；An=(n\*n+n)/2

由题知道A起始地址是0，所以n=i+1带入得An=（i\*i+3i+2）/2,

根据B起始地址为0，所以An=(i\*i+3i+2)/2 -1 = (i\*i+3i)/2 =(i+3)x(i/2)

稀疏矩阵采用三元组表形式进行压缩存储，三元组转置：行列互换，然后再按行排序

声明一个指向含有10个元素的数组的指针，其中每个元素是一个函数指针，该函数的返回值是int，参数是int\*： int (\*(\*p)[10])(int \*)

首先题目说要声明一个数组指针, 一般我们想到的数组指针是 随便来一个 int(\*p)[10],

然后又说每个元素是一个函数指针,那么我们随便来一个 函数指针 int (\*pf)(int \*) . 然后把(\*p)[10]作为一个整体替代

pf 即 int(\*(\*p)[10]))(int \*); 分析:

判断一个复杂式子看最高优先级的,\*p是一个指针,然后(\*p)外面是[],所以是数组指针,(\*p)[10])描述完毕,然后再看外面int(\*)(int \*)很明显,这是一个函数指针,所以这个数组中每个元素是函数指针

设有一个10阶对称矩阵A[10][10]，采用压缩存储方式按行将矩阵中的下三角部分的元素存入一维数组B[ ]中，A[0][0]存入B[0]中，则A[8][6]在B[ ]的（ 42）位置

(1+8) \* 8 / 2 + 7 = 43,下标从0开始，减1=42

线性结构的是 串 链式存储栈 顺序存储栈

线性是线性，顺序是顺序，线性是逻辑结构，顺序是存储结构，两者不是一个概念，线性是指一个元素后继只有唯一的一个元素或节点，非线性是一个元素后面可以有多个后继或前继节点，顺序是指存储结构连续，例如数组是顺序的，链表不是顺序的，但他们都是线性的。当然顺序也可以是非线性的，例如顺序结构存储非线性结构的二叉树！！！

在Java中的话，

char类型的字符做运算的时候，提升为int型，如果返回值为char的话，if中的return需要强制类型转换

设某数据结构的二元组形式表示为 A=(D ， R) ， D={01 ， 02 ， 03 ， 04 ， 05 ， 06 ， 07 ， 08 ， 09} ， R={r} ， r={<01 ， 02> ， <01 ， 03> ， <01 ， 04> ， <02 ， 05> ， <02 ， 06> ， <03 ， 07> ， <03 ， 08> ， <03 ， 09>} ，则数据结构A是（树型结构）

数据结构的二元组形式为：DS = (D, S)。

其中 D 是数据元素的集合； S 是 D 中数据元素之间的关系集合，并且数据元素之间的关系是使用序偶来表示的。序偶是由两个元素 x 和 y 按一定顺序排列而成的二元组，记作<x , y>， x 是它的第一元素， y 是它的第二元素。

1.如果 D != null，而S == null，则该数据结构为集合结构。

2.如果 D = {01, 02, 03, 04, 05}，S = {<02,04>, <03,05>, <05,02>, <01,03>}，则该数据结构是线性结构。

在这些数据元素中有一个可以被称为“第一个”的数据元素；还有一个可以被称为“最后一个”的数据元素；除第一个元素以外每个数据元素有且仅有一个直接前驱元素，除最后一个元素以外每个数据元素有且仅有一个直接后续元素。这种数据结构的特点是数据元素之间是 1对 1 的联系，即线性关系。

3.D = {01, 02, 03, 04, 05, 06}，S = {<01,02>, <01,03>, <02,04>, <02,05>, <03,06>}

除了一个数据元素（元素 01）以外每个数据元素有且仅有一个直接前驱元素，但是可以有多个直接后续元素。这种数据结构的特点是数据元素之间是 1 对 N 的联系，即树结构。

4.D = {01, 02, 03, 04, 05}

S = {<01,02>, <01,05>, <02,01>, <02,03>, <02,04>, <03,02>,<04,02>, <04,05>, <05,01>, <05,04>}:

每个数据元素可以有多个直接前驱元素，也可以有多个直接后续元素。这种数据结构的特点是数据元素之间是 M 对 N 的联系，即图结构

4/26/2023

若声明一个浮点数数组如下： float average[]=new float[30];假设该数组的内存起始位置为200， average[15]的内存地址是 260

float一般为4个字节，以0做下标，计算第16个，所以只有15个间隔（15-0）

15\*4+200=260

JDK8之前版本，HashMap的数据结构是怎样的 数组+链表/红黑树

数组与链表的区别是 前者长度固定,后者长度可变

关联数组是一种键值对集合，类似于hash表，每一个key对应一个value，不是线性表

数组不是原生类 原生类有8种， int double boolean float byte short long char

数组元素在栈区，链表元素在堆区

char s1[10], s2[10] = {"books"};则能将字符串 books 存放到数组 s1 的正确语句是 strcpy(s1, s2);

char s[ ]=”china”; char \*p; p=s;

A s 和 p 完全相同

B 数组 s 中的内容和指针变量 p 中的内容相等

C s 数组长度和 p 所指向的字符串长度相等

p=s;将指针P指向数组S的内存首地址。 p申请的内存空间存放的是s首地址的内存空间，而s的连续内存空间存放的china，所以A错误。 数组s的内容是china,而指针p内容为s的地址，所以B错误。 s数组的长度为6（包含结束符'\0'），p所指向的字符串长度为5。所以C错误。

请问下面哪种方式可以在不改变原来数组的情况下，拷贝出数组 b ，且满足 b!=a 。例如数组 a 为 [1,2,3](js)

let b=a.slice(); let b=a.concat();

A：堆内存是用来存放由new创建的对象和数组，

栈内存是用来存放在函数中定义的一些基本类型的变量和对象的引用变量，

例子：new函数和malloc函数申请的内存在堆；

局部变量存放在栈；函数调用参数，函数返回值，函数返回地址存放在栈

B：链表与数组插入和删除时间复杂度都是O (n)

C：数组利用下标 index 定位，时间复杂度为O(1) “无论有序数组，还是无序数组”

链表定位元素时间复杂度O(n)； “最好情况下第一个就是即o(1)，最坏情况o(n)，取最坏情况o(n)。”

D：数组：查询速度快， 增删元素慢；

链表： 查询速度慢， 增删元素快；

本题需要将数组a=[1,2,3]变成[1,2,3,4],需要改变原数组a。

Array对象常用方法中：

不改变原数组：

1、 concat()

连接两个或多个数组

不改变原数组

返回被连接数组的一个副本

2、join()

把数组中所有元素放入一个字符串

不改变原数组

返回字符串

3、 slice()

从已有的数组中返回选定的元素

不改变原数组

返回一个新数组

4、 toString()

把数组转为字符串

不改变原数组

返回数组的字符串形式

改变原数组：

5、 pop()

删除数组最后一个元素，如果数组为空，则不改变数组，返回undefined

改变原数组

返回被删除的元素

6、 push()

向数组末尾添加一个或多个元素

改变原数组

返回新数组的长度

7、 reverse()

颠倒数组中元素的顺序

改变原数组

返回该数组

8、 shift()

把数组的第一个元素删除，若空数组，不进行任何操作，返回undefined

改变原数组

返回第一个元素的值

9、 sort()

对数组元素进行排序(ascii)

改变原数组

返回该数组

10、 splice()

从数组中添加/删除项目

改变原数组

返回被删除的元素

11、 unshift()

向数组的开头添加一个或多个元素

改变原数组

返回新数组的长度

A选项，a.reverse()后返回值即数组a变成[3,2,1],再使用unshift()方法在数组开头添加一个4，a数组就变成了[4,3,2,1]。注，unshift()返回新数组的长度，此处为4。正确。

B选项，push()方法返回新数组的长度，是一个number类型，不是数组，所以不能再用“.”操作符继续执行reverse()方法。错误。

C选项，正确。

D选项，splice()语法：arrayObject.splice(index,howmany,item1,.....,itemX)。

参数：

index：必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置。

howmany：必需。要删除的项目数量。如果设置为 0，则不会删除项目。

item1, ..., itemX：可选。向数组添加的新项目。

返回值：

Array：包含被删除项目的新数组，如果有的话。

由于a.splice(3,1,4)中的index为3，超过了数组a的下标，所以3后面的1也不会起作用。a.splice(3,1,4)会将4插入到数组a末尾。但splice()返回值是被删除项目的新数组，由于a.splice(3,1,4)并没有删除元素，所以返回的新数组为空，对空数组reverse()还是空。(如果D选项改成a.splice(3,1,4);a.reverse()就正确了。)错误。

设一维数组中有n个数组元素，则读取第i个数组元素的平均时间复杂度为 O(1)

用下标读取，不是查找

假设以行优先顺序存储三维数组A[5][6][7],其中元素A[0][0][0]的地址为1100，且每个元素占2个存储单元，则A[4][3][2]的地址是()1482

位置为4\*(6\*7)+3\*7+2=191

每个元素两个存储单元，最终结果为191\*2+1100=1482

若二维数组 a 有 m 列，则计算任一元素 a[i][j] 在数组中的位置公式为（ ）。

（假设 a[0][0] 位于数组的第一个位置上）

i\*m+j+1

有一个100\*90的稀疏矩阵,非0元素有10个,设每个整型数占2字节,则用三元组表示该矩阵时,所需的字节数是()66

将非零元素所在行、列、非零元素的值构成一个三元组（i，j，v）

；

对于该题：

每个非零元素占3\*2=6个字节，共10个非零元素，需6\*10 = 60 个字节；

此外，还一般要用三个整数来存储矩阵的行数、列数和总元素个数，又需要3\*2 = 6个字节；

总共：60 + 6 = 66 个字节。

如有定义：char str[20];，能将从键盘输入的字符串“How are you”保存到 str 数组的语句是（ ）gets(str);

因为scanf不能输入空格，而gets函数却可以。

下面哪个Java语句声明了一个适合于创建50个字符串对象数组的变量？

A：char[][]

定义了二位字符数组。在Java中，使用字符串对char数组赋值，必须使用toCharArray()方法进行转换。所以A错误。

B、C：在Java中定义String数组，有两种定义方式：String a[]和String[]

a。所以B、C正确。

D、E：数组是一个引用类型变量

，因此使用它定义一个变量时，仅仅定义了一个变量

，这个引用变量还未指向任何有效的内存

，因此定义数组不能指定数组的长度。所以D、E错误。

F：Object类是所有类的父类。子类其实是一种特殊的父类，因此子类对象可以直接赋值给父类引用变量，无须强制转换，这也被称为向上转型。这体现了多态的思想。所以F正确。

最后选B、C、F

char a[] = "a0\0a0\0";

printf("%d, %d", sizeof(a), strlen(a));

在定义字符串的时候，会有一个'\0'作为字符串的结束标志，所以完整的字符串应为"a0\0a0\0\0"，使用sizeof()函数计算数组a的占用的空间大小应为7，strlen()函数计算字符串长度是碰到第一个字符串结束符'\0'为止，且不包含'\0'，所以strlen(a)为2。

若数组S[1..n]作为两个栈S1和S2的存储空间，对任何一个栈，只有当[1..n]全满时才不能进行进栈操作。为这两个栈分配空间的最佳方案是（

S1的栈底位置为1，S2的栈底位置为n

两个栈的栈顶

一个在数组第一个元素，朝着数组索引增加方向增加

一个栈顶在数据最后一个元素，朝着数组索引减少方向增加

当两个栈的栈顶位置相差1时，数组被写满

在程序设计中，要对两个16K×16K的多精度浮点数二维数组进行矩阵求和时，行优先读取和列优先读取的区别是

行优先快

char a[3];

char b[] = "good";

a = b;

printf("%s", a);

编译出错 a和b均为数组名，代表数组的首地址，无法相互赋值。

1：vector 底层数据结构为数组，支持快速随机访问 2：list 底层数据结构为双向链表，支持快速增删 3：map、set都是STL关联容器，支持快速增删

4/27/2023

Which statement is true for the class java.util.ArrayList

A Serializable, Cloneable , Iterable <E>, Collection <E>, List <E>, RandomAccess List接口是有序的，通常允许重复，因此可以确定A对，C错；ArrayList是实现List 接口的大小可变数组，所以B错；D是Map的特性，所以D错；查看手册： Note that this implementation is not synchronized. ArrayList的实现是不是线程同步的，所以E错。

char a[2][3] = {{'a', 'b', 'c'}, {'1', '\0', '2'}};

printf("%s", a[0]);

a[0]为二维数组中第0行的起始地址，等同于二维数组a的起始地址。打印字符串时，以'\0'为字符串的结束标志，所以打印输出的字符串应为“abc1”

数组采用顺序存储方式表示

已知一稀疏矩阵的三组元表为：（1，2，3），（1，6，1），（3，1，5）（3，2，-1），（5，4，5）（5，1，-3），则其转置矩阵的三元组表中第3个三元组为（）

原始的

1，2，3

1，6，1

3，1，5

3，2，-1

5，4，5

5，1，-3

转置之后相当于交换xy之后再存储

2, 1, 3

6, 1, 1,

1, 3, 5

2, 3，-1

4, 5, 5

1, 5，-3

然后按照xy增序排列：

1, 3, 5

1, 5，-3

2, 1, 3

2, 3，-1

4, 5, 5

6, 1, 1,

所以选A-- 2,1,3