**苏州大学 数据结构 课程期中试卷** 共 6 页

考试形式 闭 卷 2018年11月

院系 计算机 年级 专业

学号 姓名 成绩

1. 单项选择题：（每小题2分，共30分）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 答案 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. 在数据结构中，数据的最小单位是（ C ）。
   1. 数据元素
   2. 字节
   3. 数据项
   4. 结点
2. 设n是描述问题规模的非负整数，下面程序片段的时间复杂度是（ A ）。

x=2；

while (x<n/2)

x=2\*x;

1. 下列说法错误的是（ B ）。原答案A有误。
   1. 算法原地工作的含义是指不需要任何额外的辅助空间
   2. 在相同的规模n下，复杂度的算法在时间上总是优于复杂度的算法
   3. 所谓时间复杂度是指最坏情况下，估算算法执行时间的一个上限
   4. 同一个算法，实现语言的级别越高，执行效率就越低
   5. （1）
   6. （1），（2）
   7. （1），（4）
   8. （3）

（（1）算法原地工作是指算法所需的辅助空间是常量。感觉（2）也不太对）

1. 线性表是一个（ A ）。
   1. 有限序列，可以为空
   2. 有限序列，不能为空
   3. 无限序列，可以为空
   4. 无限序列，不能为空
2. 单链表中增加一个头结点的目的是（ C ）
   1. 使单链表至少有一个结点
   2. 标识表结点中首结点的位置
   3. 方便运算的实现
   4. 说明单链表是线性表的链式存储
3. 对顺序存储的线性表，设其长度为n，在任何位置插入或删除操作都是等概率的。删除一个元素时平均要移动表中（ C ）个元素。
   1. n/2
   2. (n+1)/2
   3. (n-1)/2
   4. n

（0+1+2+…+（n-1）次）

1. 将长度为n的单向链表链接在长度为m的单向链表之后的算法的时间复杂度为（ C ）
   1. O(1)
   2. O(n)
   3. O(m)
   4. O(n+m)
2. 对于双向循环链表，在p指针所指的结点之后插入s指针所指结点的操作应为（ D ）
   1. p->next=s; s->back=p; p->next->back=s; s->next=p->next;
   2. p->next=s; p->next->back=s; s->back=p; s->next=p->next
   3. s->back=p; s->next=p->next; p->next=s; p->next->back=s;
   4. s->back=p; s->next=p->next; p->next->left=s; p->next=s;
3. 设栈S和队列Q的初始状态均为空，元素a,b,c,d,e,f,g依次进入栈S。若每个元素出栈后立即进入队列Q，且7个元素出队的顺序是b,d,c,f,e,a,g，则栈S的容量至少是（ C ）。
   1. 1
   2. 2
   3. 3
   4. 4
4. 中缀表达式(A+B)\*(C-D)/(E-F\*G)的后缀表达式是（ B ）。
   1. A+B\*C-D/E-F\*G
   2. AB+CD-\*EFG\*-/
   3. AB+C\*D-E/F-G\*
   4. ABCDEFG+\*-/-\*
5. 已知程序如下：

int s(int n) {

return (n<=0) ? 0 : s(n-1)+n;

}

void main(){

cout<<s(1);

}

程序运行时使用栈来保存调用过程的信息，自栈底到栈顶保存的信息依次对应的是（ A ）

* 1. main()->s(1)->s(0)
  2. s(0)->s(1)->main()
  3. main()->s(0)->s(1)
  4. s(1)->s(0)->main

1. 如进栈序列1,2,3,4,5.可能得到的出栈序列为（ C ）。
   1. 1,2,5,3,4
   2. 3,1,2,5,4
   3. 3,2,5,4,1
   4. 1,4,2,3,5
2. 一个递归算法必须包括（ B ）。
   1. 递归部分
   2. 终止条件和递归部分
   3. 迭代部分
   4. 终止条件和迭代部分
3. 栈和队列的共同点是（ C ）。
   1. 都是先进先出
   2. 都是后进先出
   3. 只允许在端点处插入和删除元素
   4. 没有共同点
4. 已知循环队列存储在一维数组A[0..n-1]中，且队列非空时front和rear分别指向队头元素和队尾元素。若初始时队列为空，且要求第1个进入队列的元素存储在A[0]处，则初始时front和rear的值分别是（ B ）。
   1. 0,0
   2. 0，n-1
   3. n-1,0
   4. n-1，n-1
5. 判断题：（每小题1.5分，共15分）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 |  |  |  |  |  |  |  |  |  |  |

1. 数据的逻辑结构是指数据的各数据项之间的逻辑关系。（ x ）

(逻辑结构指数据元素逻辑上的关系，一般分为集合、线性、树形、图形四种

存储结构指数据元素连同其逻辑关系在存储器上的存放形式，主要的有四类：顺序、链接、索引、散列)

1. 顺序存储结构要求连续的存储区域，在存储管理上不够灵活，因此不常用。（ x ）
2. 算法独立于具体的程序设计语言，与具体的计算机无关。（ √ ）
3. 线性表的插入、删除总是伴随着大量数据的移动。（ x ）

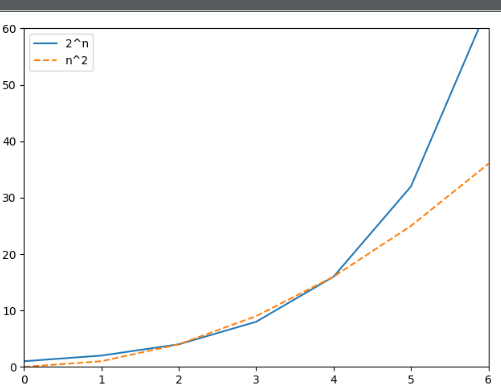
(要看线性表类型。只有顺序表是如此。)

1. 对任何数据结构，链式存储结构一定优于顺序存储结构。（ x ）当然不是
2. 在一个设有头指针和尾指针的单链表中，执行删除该单链表中最后一个元素的操作与链表的长度无关。（ x ）？？
3. 栈和队列都是操作受限的线性表。栈是插入和删除只能在一端进行的线性表；队列是插入在一端进行，删除在另一端进行的线性表。（ √ ）
4. 通常使用队列来处理函数或过程的调用。（ x ）堆栈！
5. 任何一个递归过程都可以转换成非递归过程。（ √ ）理论上而言，所有递归程序都可以用非递归程序来实现
6. 串长度是指串中不同字符的个数。（ x ）所有字符的个数
7. 填空题：（每个空1.5分，共15分）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  | 2 |  | 3 |  | 4 |  | 5 |  |
| 6 |  | 7 |  | 8 |  | 9 |  | 10 |  |

1. 若某栈初始为空，push和pop分别表示对栈进行一次进栈与出栈操作，那么对于输入序列a,b,c,d,e，经过push,push,pop,push,pop,push,push以后，输出序列是 （1）bc, 栈中尚有ade 。
2. 在进行入栈运算时应先判别栈是否 （2）满 ；在进行出栈运算时应先判别栈是否 （3）空 ；当栈中元素为n个，进行入栈运算时发生上溢，则说明该栈的最大容量为 （4）n 。
3. 顺序存储结构是通过 （5）结点物理上相邻 表示元素之间的关系的；链式存储结构是通过 （6）结点指针 表示元素之间的关系的。
4. 数据结构中评价算法的两个重要指标是（7）时间复杂度 和（8）空间复杂度 。
5. 对于一个具有n个结点的单链表，在已知的结点\*p后插入一个新结点的时间复杂度为（9）O(1) ， 在给定值为x的结点后插入一个新结点的时间复杂度为（10）O(n) 。
6. 应用题：（每题6分，共18分）
7. 有实现同一功能的两个算法A1和A2，其中A1的时间复杂度为T1=，A2的时间复杂度为T2=，仅就时间复杂度而言，请具体分析这两个算法哪一个好。

对于算法A1和A2的时间复杂度取对数，得nlog2和2logn。显然，当n<4时，A1和A2相差不大，n=4时，两者一样 ，当n>4时算法A2明显好于A1

（图像更明显）

1. 线性表有两种存储结构，一是顺序表，二是链表。如果有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变化，线性表的总数也会自动改变。在此情况下，应选用哪种存储结构？为什么？

答：选链式存储结构。可以动态申请内存空间，不受表长度（即表中元素个数）的影响，插入、删除时间复杂度为O（1）

1. 有5个元素，其入栈次序为A,B,C,D,E，在各种可能的出栈序列中，第一个出栈元素为C，且第二个出栈元素为D的出栈序列有哪几个？

答：

（1）出栈序列CDEBA：A进B进C进C出D进D出E进E出B出A出

（2）出栈序列CDBEA：A进B进C进C出D进D出B出E进E出A出

（3）出栈序列CDBAE：A进B进C进C出D进D出B出A出E进E出

1. 算法设计题：（共22分）
2. 设计递归算法以求解从集合{1..n}中选取k（k<=n）个元素的所有组合。例如，从集合{1..4}中选取2个元素的所有组合的输出结果为：1 2,1 3,1 4,2 3,2 4,3 4。 （10分）

**#include<iostream>**

**using namespace std;**

**void combine(int\* arr, int start, int\* result, int count, const int m, const int n)**

**{**

**int i = 0;**

**for (i = start; i < n + 1 - count; i++)**

**{**

**result[count - 1] = i;**

**if (count - 1 == 0)**

**{**

**int j;**

**for (j = m - 1; j >= 0; j--)**

**cout<<arr[result[j]];**

**cout<<endl;**

**}**

**else**

**combine(arr, i + 1, result, count - 1, m, n);**

**}**

**}**

**void combine\_incre(int \*arr,int n)**

**{**

**for(int m=1; m<n+1; m++)**

**{**

**int result[m];**

**combine(arr, 0, result, m, m, n);**

**}**

**}**

1. 设有两个有序单链表，一为升序，一为降序。试编写算法，将这两个链表合并为一个有序链表。 （12分）

**#include<malloc.h>**

**typedef struct LinkNode{**

**int data;**

**struct LinkNode \* next;**

**} LinkNode ,\* LinkList;**

**LinkList merge(LinkList A,LinkList B)**

**{**

**LinkList C=(LinkNode\*)malloc(sizeof LinkNode);**

**C->next=0;**

**LinkList Bt=B->next;**

**while(Bt)**

**{**

**LinkList Ct=(LinkNode\*)malloc(sizeof LinkNode);**

**Ct->data=Bt->data;**

**Ct->next=C->next;**

**C=Ct->next;**

**Bt=Bt->next;**

**}//逆序B,C为逆序结果**

**B=C;**

**LinkList At=A->next;**

**LinkList Bt=B->next;**

**LinkList D=(LinkNode\*)malloc(sizeof LinkNode);**

**D->next=0;//待生成的合并后的链表头结点**

**while(!At&&!Bt)//从当前A和C遍历结点中选择小的加入到D的头结点之后(降序，升序的话选择大的同时上面的逆序部分要换成逆序A)**

**{**

**if(At->data<Bt->data)**

**{**

**LinkList Dt=(LinkNode\*)malloc(sizeof LinkNode);**

**Dt->data=At->data;**

**Dt->next=D->next;**

**D->next=Dt;**

**At=At->next;**

**}**

**else**

**{**

**LinkList Dt=(LinkNode\*)malloc(sizeof LinkNode);**

**Dt->data=Bt->data;**

**Dt->next=D->next;**

**D->next=Dt;**

**Bt=Bt->next;**

**}**

**}**

**//A和B其中一个可能没有遍历结束，所以要遍历结束一波。**

**while(!At)**

**{**

**LinkList Dt=(LinkNode\*)malloc(sizeof LinkNode);**

**Dt->data=At->data;**

**Dt->next=D->next;**

**D->next=Dt;**

**At=At->next;**

**}**

**while(!Bt)**

**{**

**LinkList Dt=(LinkNode\*)malloc(sizeof LinkNode);**

**Dt->data=Bt->data;**

**Dt->next=D->next;**

**D->next=Dt;**

**Bt=Bt->next;**

**}**

**return D;**

}