**2011 年硕士研究生入学考试**

**“数据结构与C语言程序设计”**(**科目代码**:**991**)**试题与答案**

**一、单项选择题(本题共20分，每小题各2分)**

**1**．下列关于线性表的存储结构的叙述中，错误的是。

A．线性表的顺序存储结构中隐式地存储了数据元素之间的逻辑关系

B．线性表的顺序存储结构一定需要占用一片地址连续的存储空间

C．线性表的链式存储结构通过指针来反映数据元素之间的逻辑关系

D．线性表的链式存储结构占用的存储空间一定不连续

**2**．若front 和rear 分别表示链接队列的队头指针与队尾指针，则向队列中插入一个由p

指的新元素的过程是依次执行。

A．rear=p; front=p; B．front=p; rear=p;

C．rear>link=p; rear=p; D．front>link=p; rear=p;

**3**．下列关于二叉树的叙述中，正确的是。

A．二叉树的度可以小于2 B．二叉树的度等于2

C．二叉树中至少有一个结点的度为2 D．二叉树中每一个结点的度都为2

**4**．若某二叉树有40个叶结点，则该二叉树的结点总数最少是。

A．78 B．79 C．80 D．81

**5**．若采用邻接矩阵存储一个有向图，且邻接矩阵主对角线以下元素均为0，则该有向

图的拓扑序列。

A．存在且惟一 B．存在但可能不惟一

C．不存在 D．无法确定

**6**．下面关于AOE 网的叙述中，正确的是。

A．AOE 网是一个带权的连通图

B．AOE 网是一个带权的强连通图

C．AOE 网是一个带权的无回路的连通图

D．AOE 网是一个带权且无回路的有向图

**7**．下列关于线性表查找方法的叙述中，错误的是。

A．顺序查找法适合于采用顺序存储结构和链式存储结构的线性表的查找

B．对于相同元素，顺序查找法一定能够查找到表中首次出现的元素

C．对于相同元素，折半查找法一定能够查找到表中首次出现的元素

D．对于相同元素，折半查找法不一定能够查找到表中首次出现的元素

**8**．在二叉排序树中进行查找的平均时间效率主要与下列因素之一有关，该因素是。

A．二叉排序树的深度 B．二叉排序树中结点的个数的多少

C．被查找结点的度 D．二叉排序树的存储结构

**9**．下列4 种排序方法中，每一趟排序结束时不一定能够确定一个元素排序最终位置的

是。

A．插入排序 B．快速排序

C．堆积(Heap)排序 D．二路归并排序

2

**10**．下列4 种排序方法中，当待排序的序列中元素初始时已经按值有序，排序所花费的

时间反而有可能最多的是。

A．泡排序 B．谢尔(Shell)排序

C．快速排序 D．堆积(Heap)排序

**二、简答题(本题共20分，每小题各5 分)**

**1**．等概率情况下，在长度为n 的顺序表中插入和删除一个数据元素分别需要平均移动

多少个元素？移动的元素个数主要取决于哪几个因素？

**2**．在采用循环单链表作为某队列的存储结构时，可以只设置一个队头指针，也可以只

设置一个队尾指针。请问：从操作的时间效率考虑，采用哪种方案更合适？为什么？

**3**．对于具有n 个顶点、e 条边的稀疏图和稠密图，就空间性能而言，采用邻接矩阵存

储方法和邻接表存储方法哪一种更合适？为什么？

**4**．什么是小顶堆积(Heap)？在小顶堆积中，值最大的元素可能处在什么位置？（可以

借助一棵二叉树描述）

**三、综合题(本题共20分，每小题各5 分)**

**1**．下列算法的功能是删除长度为n的顺序表A中重复出现的多余元素，即对于重复出

现的元素，表中只保留一个。请在算法的空白处填上必要的内容，使算法完整。

void PURGE(ElemType A[ ], int &n)

{ int i=0,j,k;

while(i<n){

j=i+1; /\* 从第i+1 个元素开始逐个与第i个元素比较\*/

while(j<n)

if(A[j]==A[i]){ /\* 若A[j]与A[i]相同，删除A[j] \*/

for( ① )

A[k-1]=A[k];

n--; /\* 修改表的长度\*/

}else

②

③

}

}

**2**．请将由题三2 图给定的树转换为一棵二叉树。

（只须画出转换后的二叉树）

题三 2 图

**3**．已知某3阶B-树如题三3图所示，请画出在

该 B-树中插入关键字20 以后得到的B-树。

题三 3 图

60

30 50 70

15 25 35 45 55 65 75

A

B C D

E F G H I J

3

**4**．请分别写出对数据元素序列(49,38,65,97,76,13,27,49)按从小到大进行谢尔(Shell）排

序时每一趟的结果。设排序的间隔数(也称为增量)依次为4,2,1。

**四、算法设计题(本题15 分)**

已知某哈夫曼树采用二叉链表存储，结点构造为lchild data rchild ，其中，叶

结点的 data 域中已经存放了该叶结点对应的权值。请写一**非递归算法**，该算法的功能是计算

根结点指针为T的哈夫曼树的带权路径长度(WPL)。

要求：

**1**．用文字简要给出算法的基本思想；（5 分）

**2**．根据算法的基本思想写出相应算法。（10 分）

**五、程序阅读题(本题共20分，每小题各2分)**

**1**．下列程序的输出结果是。

main( )

{

char ch=‘A’;

printf(“ch(1)=%d,ch(2)=%c\n”,ch,ch+1);

}

**2**．下列程序段的输出结果是。

k=1; t=3;

do{

t+=k++;

if(t%7==0)

continue;

else

++k;

}while(t<15);

printf(“%d”,k);

**3**．下列程序的输出结果是。

#include <stdio.h>

main( )

{ int s[12]={1,2,3,4,4,3,2,1,1,1,2,3},a[5]={0},i;

for(i=0;i<12;i++)

a[s[i]]++;

for(i=1;i<5;i++)

printf(“%d”,a[i]);

printf(“\n”);

}

**4**．下列程序的输出结果是。

#include <string.h>

main( )

{ char str1[15]= “good”,str2[10]= “morning”;

printf(“%d\n”,strlen(strcat(str1,str2)));

}

**5**．下列程序的输出结果是。

main( )

4

{ int a[5]={1,2,3,4,5},\*p;

p=a;

printf(“%d\n”,\*(++p));

}

**6**．下列程序的输出结果是。

main( )

{ char \*s=“13579”;

s++;

printf(“%c%c%c”,\*s,\*(s+1),\*s+1);

}

**7**．下列程序的输出结果是。

#define MAX(A,B) (A)>(B)?(A):(B)

#define PRINT(Y) printf(“Y=%d\t”,Y)

main( )

{ int a=1,b=2,c=3,d=4,temp;

temp=MAX(a+b,c+d);

PRINT(temp);

}

**8**．下列程序的输出结果是。

int fun(int x,int y)

{ return(x+y); }

main( )

{

int a=2,b=5,c=8;

printf(“%d\n”,fun(fun(a+c,b),a-c));

}

**9**．下列程序的输出结果是。

#include <stdio.h>

main( )

{

struct date{

int year,month,day;

}today;

printf(“%d\n”,sizeof(struct date));

}

**10**．执行下列程序后，文件file2.txt中的内容是。

#include <stdio.h>

main( )

{ FILE \*in,\*out;

char \*str1=“YOU PLAN TO FAIL.”;

char \*str2=“IF YOU FAIL TO PLAN.”;

if((in=fopen(“file1.txt”,“w”))!=NULL)

while(\*str1!=‘.’)

fputc(\*str1++,in);

fclose(in);

if(((in=fopen(“file1.txt”,“r”))!=NULL)&&

((out=fopen(“file2.txt”,“w”)!=NULL)){

while(!eof(in)){

fgetc(in);

5

fputc(\*str2++,out);

}

}

fclose(in);

fclose(out);

}

**六、填空题(本题共20分，每小题各4 分)**

**1**．对于下列程序，为了使输出结果为t=4，输入量x和y应该满足的条件是。

main( )

{ int x,y,s=1,t=1;

scanf(“%d,%d”,&x,&y);

if(x>0)

s=s+1;

if(x>y)

t=s+t;

else if(x==y)

t=5;

else

t=2\*s;

printf(“t=%d”,t);

}

**2**．若已有下列定义，则表达式p->b/n.a 的值是① ，表达式p->b/n.a\*++p->b 的值是

② ，表达式(\*p).a+p->c的值是③ 。

struct num{

int a;

int b;

float c;

}n={1,3,5.0};

struct num \*p=&n;

**3**．下列程序段的功能是计算1000！的末尾含有多少个零。请在程序段的空白处填上必

要的内容，使程序段完整。（提示：只要计算出1000！中含有因数5 的个数即可）

for(k=0,i=5;i<=1000;i+=5){

m=i;

while( ){

k++;

m=m/5;

}

}

**4**．下列程序的功能是通过指针操作，找出并输出三个整数中的最小者。请在程序的空白

处填上必要的内容，使程序完整。

#include <stdlib.h>

main( )

{

int \*a,\*b,\*c,num,x,y,z;

a=&x;

b=&y;

c=&z;

printf(“Input a,b,c:”);

6

scanf(“%d%d%d”,a,b,c);

printf(“%d,%d,%d\n”,\*a,\*b,\*c);

num=\*a;

if(\*a>\*b)

① ;

if(num>\*c)

② ;

printf(“The minimun=%d\n”,num);

}

**5**．下列程序的功能是先由用户通过键盘输入一个文件名，然后向此文件输入一串字符

(假设输入以字符“#”结束)，最后再将当前日期写到文件的尾部。请在程序的空白处填上

必要的内容，使程序完整。

#include <stdio.h>

main( )

{

char ch,date[20],fname[30];

FILE \*fp;

printf(“Input the file name:”);

scanf(“%s”,fname);

if((fp=fopen( ① ))==NULL){

printf(“Can not open file %s !\n”,fname);

exit(0);

}

printf(“Input a string:\n”);

while((ch=getchar( ))!=‘#’)

fputc( ② );

printf(“Enter date:\n”);

scanf(“%s”,date);

fprintf( ③ );

fclose(fp);

}

**七、程序设计题(本题15 分)**

请编写一C语言程序，该程序的功能是先通过键盘输入一个整数n，然后调用一个**递归**

**函数**fun(int n)计算1+2+3+…+n，最后输出计算结果。

**八、程序设计题(本题20 分)**

请设计一C语言函数(注：只须写出函数，不必写出完整程序)，该函数的功能是用尽可

能高的时间效率与空间效率将一个int类型的数组A[0..n-1]的所有元素依次循环右移k个位

置。

例如，对于某数组，当k=3时(即把数组所有元素循环右移3 个位置)，是将

10 20 30 40 50 60 70 转换为50 60 70 10 20 30 40

7

**参考答案：**

**一、单项选择题**

**1．D 2．C 3．A 4．B 5．B**

**6．D 7．C 8．A 9．D 10．C**

**二、简答题**

**1**．答：在等概率情况下，在长度为n的顺序表中插入一个数据元素需要平均移动n/2

个元素，删除一个数据元素需要平均移动(n-1)/2 个元素。具体移动的元素个数主要取决于

表的长度n以及插入或删除的位置，位置越接近n，做需要移动的元素就越少。

**2**． 答：只设置一个队尾指针更合适。因为对于采用循环单链表作为存储结构的队列而

言，可以通过队尾指针在O(1)的时间内找到队头指针，而只设置队头指针要在O(n)的时间

内才能找到队尾指针。因此，只设置队尾指针，进队和出队操作的时间复杂度均为O(1)；

而只设置队头指针，出队操作的时间复杂度为O(1)，但进队操作的时间复杂度为O(n)。

**3**． 答：一般情况下，采用邻接矩阵存储图需要一个一维数组存储顶点的数据信息和一

个二维数组(称之为邻接矩阵)存储边或弧的信息，因此，空间复杂度为(n2)，与图中边或弧

的数量无关，可见邻接矩阵适合存储稠密图；而采用邻接表需要分别将以某顶点为出发点的

所有边对应的边结点链接为一个线性链表，同时用一个一维数组存储图中顶点的数据信以及

指向以该顶点为出发点的第一条边对应的边结点的指针，因此，空间复杂度为(n+e)，可见

图中边(或弧)数越少需要的存储空间就越少，因此，邻接表适合存储稀疏图。

**4**． 答：如果借助二叉树来描述，小顶堆积是一棵完全二叉树，二叉树中任意分支结点

的值均小于或等于其左孩子和右孩子(若右孩子存在)的值。堆积中值最大的元素对应的结点

一定是叶结点，否则，该结点必定有大于它的孩子结点，这与小顶堆积的定义相矛盾；因此，

值最大的元素对应的结点只能作为叶结点出现在二叉树的最下面两层中的一层中。

**三、综合题**

**1．**①k=j;k<n;k++ ② j++; ③ i++;

**2． 3．**

**4．**

趟序 间隔数各趟排序结果

初始 49 38 65 97 76 13 27 49

第1趟4 49 13 27 4976 38 65 97

第2趟2 27 13 49 38 65 4976 97

第3趟1 13 27 38 49 4965 76 97

A

C

B

F D

E

G

H

I

J

30 60

15 25 35 45 55 65 75

20 50 70

8

**四、算法设计题**

（1）算法的基本思想：

本题宜采用二叉树的后序遍历的非递归算法完成。在遍历过程中，访问一个叶结点时，

将该叶结点的数据域值(该叶结点的权值)与该叶结点的路径长度(即当前栈顶指针值加1)相

乘，并进行WPL值的累加。遍历结束时便求的该哈夫曼树的WPL。

（2）算法：

#define MaxNum 50 /\* 定义二叉树中结点最大数目\*/

int POSTORDER\_WPL(BTREE T)

{

/\* T为二叉树根结点所在链结点的地址\*/

BTREE STACK1[MaxNum],p=T;

int STACK2[MaxNum],flag,top=1;

WPL=0;

if(T!=NULL)

do{

while(p!=NULL){

STACK1[++top]=p; /\* 当前p指结点的地址进栈\*/

STACK2[top]=0; /\* 标志0 进栈\*/

p=p>lchild; /\* 将p移到其左孩子结点\*/

}

p=STACK1[top];

flag=STACK2[top]; /\* 退栈\*/

if(flag==0){

STACK1[++top]=p; /\* 当前p指结点的地址再次进栈\*/

STACK2[top]=1; /\* 标志1 进栈\*/

p=p>rchild; /\* 将p移到其右孩子结点\*/

}

else{

if(p->lchild==NULL&& p->rchild==NULL) /\* p指结点为叶结点\*/

WPL=WPL+p->data\*(top+1);

p=NULL;

}

}while(!(p==NULL&& top==1))；

returnWPL;

}

**五、程序阅读题**

**1．**ch(1)=65,ch(2)=B **2．**8 **3．**4332 **4．**11

**5．**2 **6．**354 **7．**Y=7 **8．**9

**9．**6 **10．**IF YOU FAIL TO PL

**六、填空题**

**1．**0<x<y **2．**①3 ②12 ③6.0

**3．**m%5==0 **4．**①num=\*b ②num=\*c

**5．**① fname, “w” ② ch,fp ③ fp,“%s”,date

**七、程序设计题**

double fun(int n)

{

double s;

if(n==1)

return 1;

s=n+fun(n-1);

return s;

}

9

main( )

{ double sum;

int n;

printf(“Input n:”);

scanf(“&d”,&n);

sum=fun(n);

printf(“The result is %lf\n”,sum);

}

**八、程序设计题**

（1）算法的基本思想：

根据k 值将数组A[0..n-1]分成前后两个部分，其中，前一部分为数组的前n-k 个元素，

后一部分为数组的后k个元素；然后先将后一部分中的k个元素进行逆置(即前后对应位置的

元素依次颠倒位置)，接着将前一部分中的n-k个元素进行逆置，最后将整个数组的所有元素

再进行一次逆置，即得到所需要的结果。

（2）程序：

void MOVE2(int A[ ],int n,int k)

{

REVERSE(A,n-k,n-1); /\* 逆置数组的后k个元素\*/

REVERSE(A,0,n-k-1); /\* 逆置数组的前n-k个元素\*/

REVERSE(A,0,n-1); /\* 逆置数组的所有元素\*/

}

void REVERSE(int A[ ],int from,int to)

{

int i, temp;

for(i=0;i<(to-from+1)/2;i++){ /\* 逆置下标为from到to之间的所有元素\*/

temp=A[from+i];

A[from+i]=A[to-i];

A[to-i]=temp; /\* 交换元素A[from+i]与A[to-i]的位置\*/

}

}

效率分析：第1 次调用REVERSE 函数的时间复杂度为(k)，第2次调用REVERSE 函数

的时间复杂度为(n-k)，第3 次调用REVERSE 函数的时间复杂度为(n)，因此，整个程序的

时间复杂度为(n)。只用了1 个数组元素的辅助空间temp。

**指定参考书：**

**1．《数据结构教程第二版》唐发根 编著 北京航空航天大学出版社 2005**

**2．《C程序设计第三版》 谭浩强 编著 清华大学出版社 2005**

**建议选用2010年7月**

**第 6次印刷的书**\_\_