**（参考答案）**

**浙江大学2022–2023学年春夏学期**

**《程序设计专题》课程平时考试试卷**

课程号： 211G0260 ，开课学院： 计算机学院\_\_

考试试卷：√A卷、B卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带 ∕入场

考试日期： 2023 年 04 月 04 日,考试时间： 120 分钟

**诚信考试，沉着应考，杜绝违纪.**

**考生姓名： 学号： 所属院系： \_**

**(注意：答题内容必须写在答题卷上，写在本试题卷上无效)**

**【本次测试要点 】** 结构、递归、算法复杂度、链表、回调函数（基于事件的编程方法），程序设计方法学等

**【本学期复习要点】**1 模块化程序设计与递归函数、2 结构/链表、3 查找/排序与算法分析，涉及教材中的知识点如预处理命令（文件包含、宏定义等）、外部变量、静态全局变量、多文件程序、结构、递归函数、链表和算法（排序、查找）。题型分为五种：单选、改错、问答、程序填空和算法设计。

试题一、单选题（每小题3分，共21分）

1 若有定义struct { int k; char s[30]; } a[5] = {1, "ABC", 2, "abc"}, \*p = a; 那么表达式\*(p++)->s的值是\_\_\_\_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| A．'A' | B．'B' | C．'a' | 1. 'b' |

2 有函数原型为void f(int, int \*)，主函数中有变量定义：int a=2, \*p=&a; 则下列函数调用正确的是\_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| A．f(a, &p) | B．f(\*p, &a) | C．f(a, \*p) | D．f(\*p, a) |

3 void fun(int i)

{

if (i>0) {

fun(i/2);

}

printf("%d ",i);

}

int main()

{

fun(10);

return 0;

}

A、0 1 2 5 10 B、0 C、0 10 D、0 5 10

4 在下面的程序段中，对 x 的赋值语句的频度为( )

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

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

x++;

A. O(2n) B.O(n) C.O(n2) D.O(log2n)

5 程序段 for(i=n-1;i>=1;i--)

for(j=1;j<=i;j++) if(A[j]>A[j+1]) A[j]与 A[j+1]对换;

该算法为冒泡排序算法的核心语句，其中 n 为正整数，则最后一行的语句频度在最坏情况下是 ： A、 O（n） B. O(nlogn) C.O(n) D .O(n2)

6 若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算， 则利用( )存储方式最节省时间。

A.顺序表 B.双链表 C.双循环链表 D.单循环链

【答案】A： 本题考查顺序表的特点。顺序表进行插入或者删除操作时需要移动的元素是待插或者待删位置之后的元素，因此当插入或者删除操作发生在顺序表的表尾时，不需要移动元素。

顺序表支持随机存储，方便于存取任意指定序号的元素，因此本题情况下顺序表最节省时间。

7 单链表 L(带头结点)为空的判断条件是( )

A.L==NULL B.L->next==NULL C.L->next!=NULL D. L!=NULL

【答案】 C 本题考查带头结点的单链表和不带头结点的单链表基本操作的区别。带头结点的单链表判空条件看 head->next 是否为 NULL，不带头结点的单链表是看头指针 head 是否为 NULL。

试题二、指出下面问题的结果正确与否，并改错之（每小题3分，共18分）

1:求出以下算法的时间复杂度。

void fun(int n) {

int i=1,j=100;

while(i<n)

{

j++;   
i+=2;

}

}

时间复杂度 T(n)=O(n2)。

【答案】T(n)=O(n)

**原因分析:**

**第一步**:找出基本操作，确定规模 n。

1**找基本操作**(所谓基本操作，即其重复执行次数和算法的执行时间成正比的操作，通俗点说，这种操作组成了算法，当它们都执行完的时候算法也结束了，多数情况下我们取最深层循环内的语句所描述的操作作为基本操作)，显然题目中 j++; 与 i+=2; 这两行都可以作为基本操作。

2 确定规模，由循环条件 i<n 可以知道，循环执行的次数，即基本操作执行的次数和参数 n 有关，因此参数 n 就是我们所说的规模 n。

**第二步:**计算出 n 的函数 f(n)。

显然，n 确定以后，循环的结束与否与 i 有关，i 的初值为 1，每次自增 2，假设 i 自 增 m 次后循环结束，则 i 最后的值为 1+2×m，因此有 1+2×m+K=n(其中 K 为一个常数， 因为在循环结束时 i 的值稍大于 n，为了方便表述和进一步计算，用 K 将 1+2×m 修正成 n。 因为 K 为常数，所以这样做不会影响最终时间复杂度的计算)，解得 m=(n-1-K)/2，即 f(n)=(n-1-K)/2，可以发现其中增长最快的项为 n/2,

因此时间复杂度： T(n)=O(n)。

2:分析以下算法的时间复杂度。

void fun(int n)

{

int i，j，x=0;

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

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

x++;

}

时间复杂度： T(n)=O(n2)。

【答案】正确

x++;处于最内层循环，因此取 x++;做为基本操作。显然 n 为规模。可以算出 x++;的 执行次数为 f(n)=n(n-1)/2，变化最快的项为 n2，因此时间复杂度为 T(n)=O(n2)。

3:分析以下算法的时间复杂度。

void fun(int n)

 {

int i=0;s=0;

while(s<n)

{

i++;   
s=s+i;

}

}

时间复杂度： T(n)=O(n log 2 n )。

【答案】 T(n)=O(√n)

分析:

显然 n 为规模，基本操作为 i++; s=s+i; i 与 s 都从 0 开始，假设循环执行 m 次结束， 则有 s1=1,s2=1+2=3,s3=1+2+3=6, ......,sm=m(m+1)/2(其中 sm 为执行到第 m 次的时候 s 的值)，则有 m(m+1)/2+K=n，(K 为起修正作用的常数) 由求根公式求解关于m的一元二次方程，

由此可知时间复杂度为:

T(n)=O(√n)

说明:在计算时间复杂度的时候有可能会出现这种情况，即对于相同的规模，因输入序列不同会出现不同的时间复杂度，这时我们一般取最坏的情况下的输入序列(即使得基本操作执行次数最多的序列)来计算时间复杂度。

试题三、阅读与问答题（共4题，共20分）

1、浙大有三种身份的学生：本科生、研究生、博士，设计数据结构存储浙大学生的身份信息，有以下三种方法：

1) int classID;

classID = 0; /\*表示学生身份是本科生\*/

classID = 1; /\*表示学生身份是研究生\*/

classID = 2; /\*表示学生身份是博士\*/

2) 定义具名常量

#define 本科生 0

#define 研究生 1

#define 博士 2

int classID;

classID = 本科生;

classID = 研究生;

classID = 博士;

3) 定义枚举类型

typedef enum {本科生，研究生，博士} IDTYPE；

IDTYPE classID；

classID = 本科生;

classID = 研究生;

classID = 博士;

请你评价三种设计方法（从程序的可读性、可维护性、时空效益、调试和赋值）。

【答案】

方法1)：不合适，不直观

方法2)：绝大多数调试程序无法打印具名常量的值;

具名常量的值由程序员手工赋值，可能会引起冲突。

方法3)：用enum关键字说明的常量由编译程序自动生成，程序员不需要用手工对常量一一赋值。使用enum方法使程序更容易维护。

用enum关键字说明常量使程序更清晰易读，因为在定义enum常量的同时也定义了一个枚举类型标识符。

在调试程序时通常可以检查枚举常量，这一点是非常有用的，尤其在不得不手工检查头文件中的常量值时。使程序调试起来更方便。

不过，用enum关键字说明常量比用#define指令说明常量要占用更多的内存，因为前者需要分配内存来存储常量。

2、下列程序运行时输入3 5 1 2 4 -1<回车>，程序的输出是\_\_\_\_\_\_\_.

#include<stdio.h>

#include<stdlib.h>

struct node{

int num;

struct node \*next;

};

void List(struct node \*p) {

while(p!=NULL){

printf("%d ",p->num);

p=p->next;

}

}

void main()

{

int num;

struct node \*head=NULL,\*p,\*p1,\*p2;

scanf("%d",&num);

while(num!=-1){

p=(struct node \*)malloc(sizeof(struct node));

p->num=num;

if(head==NULL){

head=p;

head->next=NULL;

}else{

p1=p2=head;

while((p1->num>p->num)&&(p1->next!=NULL)){

p2=p1;

p1=p1->next;

}

if(p1->num<=p->num){

if(head==p1) head=p;

else p2->next=p;

p->next=p1;

}else{

p1->next=p;

p->next = NULL;

}

}

scanf("%d",&num);

}

List(head);

p=head;

while(p!=NULL){

if(p==head){

p=p->next;

head->next=NULL;

}else{

p1=p->next;

p->next=head;

head=p;

p=p1;

}

}

List(head);

return;

}

【答案】5 4 3 2 1 1 2 3 4 5

3、下列程序段执行后，z的值是\_\_\_\_\_\_\_。

static struct {

int x, y[3];

} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, \*p=&a[-1];

int z;

z=\*((int \*)(++p+1)-1);

【答案】4

4、有一单向链表h如下图所示，表示整数序列（1，1，2，4，4，3，5，5，5，4）。

h

4



5

1

2

1

有关变量定义：

struct node {

int info;

struct node \*next;

} \*h, \*p, \*q;

现执行以下程序段，则输出的结果是：\_\_\_\_\_\_\_。

q=h;

if (h && h->next!=NULL) p=h->next;

else return 0;

while (p!=NULL) {

if (q->info!=p->info) {

q=p; p=p->next;

} else {

p=p->next; free(q->next); q->next=p;

}

}

p=h;

while (p!=NULL) {

printf("%d", p->info);

p=p->next;

}

【答案】 124354

试题四、程序填空题（共20空，每空2分，共40分）

1. 下面的程序将输入的数存在一个单向链表中（这些数均大于或等于0，输入小于0的数表示输入结束）。在进行过处理后，将链表中的数分别存在文件out1.dat和out2.dat中，同时释放链表。其中，out1.dat存放大于THRESHOLD的数，out2.dat存放小于等于THRESHOLD的数。

#include <stdio.h>

#include <stdlib.h>

#define THRESHOLD 50

struct NODE {

int score;

struct NODE \*next;

};

int main(void)

{

struct NODE \*p, \*q, \*head = NULL;

FILE \*fp1, \*fp2;

int score;

for(;;) {

scanf("%d", &score);

if(score < 0) break;

if((p = (struct NODE \*) malloc(sizeof(struct NODE))) == NULL)

return 1;

p->score = score;

p->next = NULL;

if (head)

p->next = head;

**(1)**　 　;

}

if( (fp1 = fopen("out1.dat", "w")) == NULL ||

(　　 **(2)**　 　) == NULL)

return 2;

for(p = head; p != NULL; 　　 **(3)**　 　) {

fprintf(p->score>THRESHOLD? fp1 : fp2, "%d\n", p->score);

q = p->next;

free(　　 **(4)**　 　);

}

fclose(fp1);

**(5)**　 　;

return 0;

}

【答案】

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (1) | | head=p | | (2) | fp2=fopen(“out2.dat”,”w”) |
| (3) | | p=q | | (4) | p |
| (5) | | fclose(fp2) | |

2、有两个循环单链表，链表头指针分别为h1和h2，编写一个函数将链表h2链接到链表h1之后，要求链接后的链表仍保持循环链表形式。

算法思想：先找到两个链表的尾指针，将第一个链表的尾指针与第二个链表的头结点链接起来，再使之成为循环的。

struct Node{

...

struct NODE \*next;

};

typedef struct Node \*PNODE;

PNODE link(PNODE h1, PNODE h2)

{

PNODE tail1, tail2; //分别指向两个链表的尾结点

tail1 = h1;

while (tail1->next != NULL) //寻找 h1 的尾结点

（1）

（2）

while (tail2->next != NULL) //寻找 h2 的尾结点

（3）

tail1->next = h2; //将h2链接到h1之后

（4） //令h2的尾结点指向h1

return（5）

}

//将循环链表h2链接到循环链表h1之后，使之仍保持循环链表的形式

【答案】（1） tail1 = tail1->next; （2）tail2 = h2; （3）tail2 = tail2->next; （4）tail2->next = h1; （5） h1;

2、用 typedef 写出类型POINT的定义\_\_\_\_\_（6） ，使得POINT表示含有10个元素的整型指针数组类型。

【答案】typedef int \*POINT[10]

3、下列程序段执行后，z的值是 \_（7）\_\_.

static struct {

int x, y[3];

} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, \*p=a+3;

int z;

z=((int \*)(p-2))[-1];

【答案】4

1. 下面程序段的功能是实现冒泡排序算法，请在下划线处填上正确的语句。

void bubble(int r[n])

{

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

{

for(exchange=0,j=0; j< j<n-i;j++)

if (r[j]>r[j+1]){temp=r[j+1];\_\_\_ （8）\_\_\_;r[j]=temp;exchange=1;}

if (exchange==0) return；

}

}

【答案】r[j+1]=r[j]

1. 下面程序段的功能是实现二分查找算法，请在下划线处填上正确的语句。

struct record{int key; int others;};

int bisearch(struct record r[ ], int k)

{

int low=0,mid,high=n-1;

while(low<=high)

{

\_\_\_\_\_\_\_\_\_\_\_\_（9）\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if(r[mid].key==k) return(mid+1); else if(\_\_\_（10）\_\_) high=mid-1;else low=mid+1;

}

return(0);

}

【答案】 mid=(low+high)/2，r[mid].key>k

1. 有下列运行时间函数: (1) f1(n)=1000; (2) f2(n)=n2+1000n; (3) f3(n)=3n3+100n2+n+1; 分别写出相应的大 O 表示的运算时间。

（1） （11） （2） （12） （3） （13）

【答案】 (1) T1(n)=O(1000/1000)=O(1)

(2) T2(n)=O(n2/1)=O(n2)

(3) T3(n)=O(3n3/3)=O(n3)

试题五、算法分析与设计（共4空，共24分）

1书写在顺序表中插入数据元素的算法 （需画程序程序操作的示意图）

在顺序表 L 的第 p (1≤p≤length+1)个位置上插入新的元素 e。如果 p 的输入不正确，则返回 0，代表插入失败;如果 p 的输入正确则将顺序表第 p 个元素及以后元素右移一个位置，腾出一个空位置插入新元素，顺序表长度增加 1，插入操作成功，返回 1。

#define MAX 100 //这里定义一个整型常量 MAX，值为 100。

typedef struct {

int data[MAX]; //存放顺序表元素的数组(默认是 int 型)

int length; //存放顺序表的长度。

}Sqlist; //顺序表类型的定义。

int insert(Sqlist &L, int p, int e)

{

int i;

if(p<1||p>L.length+1||L.length==MAX) // 位置错误或表长已经达到

return 0; //顺序表的最大允许值，此时插入不成功，返回 0。

for(i=L.length;i>=p;i--)

（1） // 从后往前逐个将元素往后移一个位置。

L.data[p] = e; //将 x 放在插入位置 p 上。

（2）

return 1; //插入成功返回 1。

}

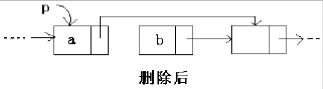
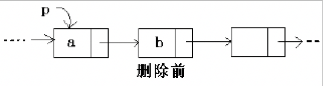
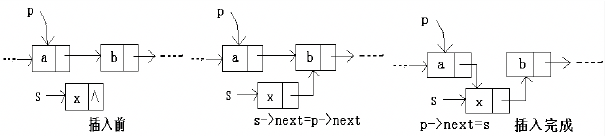
【答案】L.data[i+1]=L.data[i] ；// 从后往前逐个将元素往后移一个位置。

L.length++；//表内元素多了一个，因此表长自增 1。

2 A 和 B 是两个单链表(带表头结点)，其中元素递增有序。设计一个算法将 A 和 B 归并成一个按元素递增有序的链表 C，C 由 A 和 B 中的结点组成。

已知 A，B 中的元素递增有序，怎样使归并后的 C 中元素依然有序呢。我们可以从 A， B 中挑出最小的元素插入 C 的尾部，这样当 A，B 中所有元素都插入 C 中的时候，C 一定是 递增有序的。哪一个元素是 A，B 中最小的元素呢?很明显，由于 A，B 是递增的，所以 A 中的最小元素是其开始结点中的元素，B 也一样。我们只需从 A，B 的开始结点中选出一个 较小的来插入 C 的尾部即可。这里还需注意， A 与 B 中的元素有可能一个已经全部被插入到 C 中，另一个还没有插完，比如 A 中所有元素已经全部被插入到 C 中而 B 还没有插完，这 说明 B 中所有元素都大于 C 中元素，因此只要将 B 链接到 C 的尾部即可，如果 A 没有插完 则用类似的方法来解决。

**【注意与提示】一是尾插法建立单链表，二是单链表的归并操作**



q=p->next;

p->next=p->next->next;

free(q);//调用 free 函数来释放 q 所指结点的内存空间。

**请填入部分代码，并用图示表示算法的操作过程。**

void merge(LNode \*&A, LNode \*&B, LNode \*&C)

{

LNode \*p=A->next; //p 来跟踪 A 的最小值结点。

LNode \*q=B->next; //q 来跟踪 B 的最小值结点。

LNode \*r; //r 始终指向 C 的终端结点。

C=A; //用 A 的头结点来做 C 的头结点。

C->next=NULL;

free(B); //B 的头结点已无用，则释放掉。

r=C; //r 指向 C，因为此时头结点也是终端结点。

while(p!=NULL&&q!=NULL)

//当 p 与 q 都不空时选取 p 与 q 所指结点中的较小者插入 C 的尾部。

{

/\*以下的 if else 语句中，r 始终指向当前链表的终端节点，作为接纳新结 点的一个媒介，通过它新结点被链接入 C 并且重新指向新的终端结点以便于接受下一个新 结点，这里体现了建立链表的尾插法思想。\*/

／\*填入部分代码，并用图示及注解 表示算法的操作过程。\*／

if(p->data<=q->data)

{

／\* （3） 填入部分代码\*／

}

else

{

r->next=q; q=q->next;

r=r->next;

}

}

r->next=NULL; /\*以下两个 if 语句将还有剩余结点的链表链接在 C 的尾部\*/

if(p!=NULL)

r->next=p;

if(q!=NULL)

r->next=q;

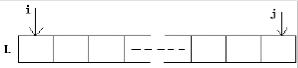
}

【答案】 r->next=p; p=p->next;

r=r->next;

1. 设计一个算法，将顺序表中的所有元素逆置。

**【分析】**



上图即可说明问题，两个变量 i，j 指示顺序表的第一个元素和最后一个 元素，交换 i，j 所指元素，然后 i 向后移动一个位置，j 向前移动一个位置，如此循环， 直到 i 与 j 相遇时结束，此时顺序表 L 中的元素已经逆置。

由此可写出以下代码:

void reverse(Sqlist &L) // L 要改变，用引用型

{

int i,j;

int temp; //辅助变量，用于交换

for(i=1,j=L.length;i<j;i++,j--) //当 i 与 j 相遇时循环结束

{

／\*（4） 请填入代码 \*／

}

}

【答案】

temp = L.data[i];

L.data[i] = L.data[j];

L.data[j] = temp;

【注意】:本题中 for循环的执行条件要写成 i<j而不要写成 i!=j。如果数组中元素有偶数个则 i与 j会出现下图所示状态，此时 i继续往右走，j继续往左走，会互相跨越对方，循环不会结束。



4用字符指针数组，来表示多行文本

#define MAXLINES 1000

#define MAXLENGTH 81

char \*line[MAXLINES];

char CurrentLine[MAXLENGTH];

for (i = 0; i < MAXLINES; i++) {

getline(CurrentLine, MAXLENGTH); // 读入一行文本

line[i] = (char\*)malloc(strlen(CurrentLine) + 1);

／\*（5） 请填入代码 \*／

}

【答案】

strcpy(line[i], CurrentLine);

【附加题】书写一函数 ，实现打印出链表中各元素中 含有“Key”值的小标（元素在链表中的序号）。

struct Node

{

char \* MyStr ;

struct Node next;

} \*head, \*p ;

／\*\*\*\*\*

\*函数原型 int Print(struct Node \* List,char \*key);

\*@brief 功能：输出链表

\*@param head: 链表头指针

\*@reurn void

\*@设计要求：链表为空时，调用此函数，要提示“链表为空”

\*@输出样例：12（从1开始计数） 或 链表为空

\*\*\*\*/

int Print(struct Node \* List,char \*key)

{

／\*（6） 请填入代码 \*／