[**【计算机二级C语言】卷007**](https://www.cnblogs.com/mjios/p/12452733.html)

**选择题**

**公共知识**

【1】下列数据结构中, 属于非线性结构的是()。

〖A〗循环队列

〖B〗带链队列

〖C〗二叉树

〖D〗带链栈

树是简单的非线性结构, 所以二叉树作为树的一种也是一种非线性结构。

【2】下列数据结构中, 能够按照"先进后出"原则存取数据的是()。

〖A〗循环队列

〖B〗栈

〖C〗队列

〖D〗二叉树

栈是按先进后出的原则组织数据的。  
队列是先进先出的原则组织数据

【3】对于循环队列, 下列叙述中正确的是()。

〖A〗队头指针是固定不变的

〖B〗队头指针一定大于队尾指针

〖C〗队头指针一定小于队尾指针

〖D〗队头指针可以大于队尾指针, 也可以小于队尾指针

循环队列的队头指针与队尾指针都不是固定的, 随着入队与出队操作要进行变化。  
因为是循环利用的队列结构所以对头指针有时可能大于队尾指针有时也可能小于队尾指针。

【4】算法的空间复杂度是指()。

〖A〗算法在执行过程中所需要的计算机存储空间

〖B〗算法所处理的数据量

〖C〗算法程序中的语句或指令条数

〖D〗算法在执行过程中所需要的临时工作单元数

算法的空间复杂度是指算法在执行过程中所需要的内存空间。  
所以选择A)。

【5】软件设计中划分模块的一个准则是()。

〖A〗低内聚低耦合

〖B〗高内聚低耦合

〖C〗低内聚高耦合

〖D〗高内聚高耦合

一般较优秀的软件设计, 应尽量做到高内聚, 低耦合, 即减弱模块之间的耦合性和提高模块内的内聚性, 有利于提高模块的独立性。

【6】下列选项中不属于结构化程序设计原则的是()。

〖A〗可封装

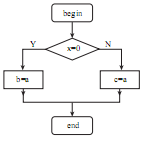
〖B〗自顶向下

〖C〗模块化

〖D〗逐步求精

结构化程序设计的思想包括：自顶向下、逐步求精、模块化、限制使用goto语句, 所以选择A)。

【7】软件详细设计生产的图如下：  
该图是()



〖A〗N - S图

〖B〗PAD图

〖C〗程序流程图

〖D〗E - R图

N - S图提出了用方框图来代替传统的程序流程图, 所以A)不对。  
PAD图是问题分析图, 它是继承程序流程图和方框图之后提出的又一种主要用于描述软件详细设计的图形表示工具, 所以B)不对。  
E - R图是数据库中的用于表示E - R模型的图示工具, 所以D)不对。  
根据图中所示表示方法是进行软件详细设计时使用的程序流程图。

【8】数据库管理系统是()。

〖A〗操作系统的一部分

〖B〗在操作系统支持下的系统软件

〖C〗一种编译系统

〖D〗一种操作系统

数据库管理系统是数据库的机构, 它是一种系统软件, 负责数据库中数据组织、数据操纵、数据维护、控制及保护和数据服务等。  
是一种在操作系统之上的系统软件。

【9】在E - R图中, 用来表示实体联系的图形是()。

〖A〗椭圆形

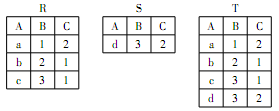
〖B〗矩形

〖C〗菱形

〖D〗三角形

在E - R图中实体集用矩形, 属性用椭圆, 联系用菱形。

【10】有三个关系R、S和T如下：  
则关系T是由关系R和S通过某种操作得到, 该操作为()。



〖A〗选择

〖B〗投影

〖C〗交

〖D〗并

在关系T 中包含了关系R与S中的所有元组, 所以进行的是并的运算。

**专业知识**

【11】以下关于结构化程序设计的叙述中正确的是

〖A〗结构化程序使用goto语句会很便捷

〖B〗在C语言中, 程序的模块化是利用函数实现的

〖C〗一个结构化程序必须同时由顺序、分支、循环三种结构组成

〖D〗由三种基本结构构成的程序只能解决小规模的问题

滥用goto语句将使程序的流程毫无规律, 可读性差, 对于初学者来说尽量不要使用, 所以A错误。  
一个结构化程序可以包含顺序、分支、循环结构中的一种或多种, 所以C错误。  
由三种基本结构构成的程序可以解决任何复杂的问题, 所以D错误。

【12】对于一个正常运行的C程序, 以下叙述中正确的是

〖A〗程序的执行总是从main函数开始, 在程序的最后一个函数中结束

〖B〗程序的执行总是从程序的第一个函数开始, 在main函数结束

〖C〗程序的执行总是从main函数开始

〖D〗程序的执行总是从程序的第一个函数开始, 在程序的最后一个函数中结束

任何一个C程序都是从主函数main开始, 至主函数main结束, 所以选择C选项。

【13】以下选项中能表示合法常量的是

〖A〗1, 200

〖B〗1.5E2.0

〖C〗'\'

〖D〗"\007"

A选项中整型常量应表示为1200, 不能包含","。  
B选项中E后面的指数必须为整数。  
C选项中转义字符以 "\"开始,若要表示字符 "\", 应写为 "\\"。

【14】以下定义语句中正确的是

〖A〗int a = b = 0;

〖B〗char A = 65 + 1, b = 'b';

〖C〗float a = 1, \*b = &a, \*c = &b;

〖D〗double a = 0.0; b = 1.1;

A选项语句中b变量还没有定义不能直接用于给a变量赋值。  
C选项语句中\*c = &b, 因为变量b已经为指针变量, 取指针变量b的地址, 这时变量c应该是指针的指针。  
D选项语句中a = 0.0后面应该为逗号, 不能是分号。

【15】若变量x、y已正确定义并赋值, 以下符合C语言语法的表达式是

〖A〗x + 1 = y

〖B〗++x, y = x--

〖C〗x = x + 10 = x + y

〖D〗double(x) / 10

A选项中不能将变量y赋给表达式, C选项中错误与A选项一样, D选项中强制类型转换表达式应写成(double) x / 10。

【16】若变量已正确定义为int型, 要通过语句  
scanf("%d,%d,%d", &a, &b, &c);  
给a赋值1、给b赋值2、给 c赋值3, 以下输入形式中错误的是(注：□代表一个空格符)

〖A〗1, 2, 3<回车>

〖B〗□□□1, 2, 3 <回车>

〖C〗1, □□□2, □□□3<回车>

〖D〗1□2□3<回车>

scanf()的格式控制串可以使用其他非空白字符, 如本题中的逗号, 但在输入时必须输入这些字符, 以保证匹配。  
所以逗号必须输入。

【17】以下不能输出字符A的语句是(注：字符A的ASCII码值为65, 字符a的ASCII码值为97)

〖A〗printf("%c \n", 'a' - 32);

〖B〗printf("%d \n", 'A');

〖C〗printf("%c \n", 65);

〖D〗printf("%c \n", 'B' - 1);

B选项输出的格式为%d, 即为整型格式, 所以输出字符A的ASCII码值65, 而不是字符A。  
由于大写字母和其对应的小写字母的ASCII码值小32, 所以A正确。  
字符A的ASCII码值就是65, 所以以%c格式输出可以输出字符A, 所以C正确。  
由于字符A的ASCII码值比字符B的小1, 所以D正确。

【18】若a是数值类型, 则逻辑表达式(a == 1) || (a != 1)的值是

〖A〗1

〖B〗0

〖C〗2

〖D〗不知道a的值, 不能确定

逻辑或||要求只要两边的运算对象有一个非零, 结果就为真, 虽然不知道a的值, 但是若a为1, 则左边运算对象为1, 若a的值不是1, 则右边运算对象的值为1, 所以总能保证一边非零, 所以结果为真, 即1。

【19】设有定义：

复制代码

C

int a = 1, b = 2, c = 3;

以下语句中执行效果与其他三个不同的是

〖A〗if (a > b) { c = b, a = b, b = c; }

〖B〗if (a > b) { c = a, a = b, b = c; }

〖C〗if (a > b) c = a; a = b; b = c;

〖D〗if (a > b) { c = a; a = b; b = a; }

因为变量a = 1, b = 2所以条件a > b不成立, 选项A、B和D相当于一条语句, 选项A是逗号表达式, 选项B和选项D是复合语句, 所以都没有执行。  
选项C中, 语句"c=a"不执行, 语句"a=b;b=c;"执行。  
故选项C符合题意。

【20】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int y = 10; |
|  | while (y--); |
|  | printf("y=%d\n", y); |
|  | } |

程序执行后的输出结果是

〖A〗y=0

〖B〗y=-1

〖C〗y=1

〖D〗while构成无限循环

执行y--直到值为0, 由于y--是先用再减, 所以退出循环时, y的值为-1。

【21】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i, j; |
|  | for (i = 1; i < 4; i++) { |
|  | for (j = i; j < 4; j++) |
|  | printf("%d\*%d=%d", i, j, i \*j); |
|  | printf("\n"); |
|  | } |
|  | } |

程序运行后的输出结果是

〖A〗 1\*1=1 1\*2=2 1\*3=3

22=4 23=6

3\*3=9

〖B〗 1\*1=1 1\*2=2 1\*3=3

21=2 22=4

3\*1=3

〖C〗 1\*1=1

12=2 22=4

13=3 23=6 3\*3=9

〖D〗 1\*1=1

21=2 22=4

31=3 32=6 3\*3=9

外循环第一次i的值为1, 内循环第一次j的初值为1, 打印1 \*1 = 1, 内循环第二次j的值为2, 打印1 \*2 = 2, 内循环第三次j的值为3, 打印1 \*3 = 3, 打印回车换行, 退出内循环。  
外循环第二次i的值为2, 内循环第一次j的初值为2, 打印2 \*2 = 4, 内循环第二次j的值为3, 打印2 \*3 = 6, 打印回车换行, 退出内循环。  
外循环第3次, i的值为3, 内循环第一次j的初值为3, 打印3 \*3 = 9, 打印回车换行, 退出内循环。

【22】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i = 5; |
|  | do { |
|  | if (i % 3 == 1) |
|  | if (i % 5 == 2) { |
|  | printf("\*%d", i); |
|  | break; |
|  | } |
|  | i++; |
|  | } while (i != 0); |
|  | printf("\n"); |
|  | } |

程序的运行结果是

〖A〗\*2 \* 6

〖B〗\*3 \* 5

〖C〗\*5

〖D〗\*7

do...while语句是先执行后判断, 所以第一次先执行循环体, 判断i % 3 == 1的条件, 由于i为5所以余数为2, 条件不成立, 执行i++, i的值为6, 判断while条件为真, 第二次执行循环体, 同第一次循环体的执行过程一样, i的值变为7, 判断while条件为真, 第三次执行循环体, 此时i % 3 == 1条件成立, 判断i % 5 == 2也成立, 打印\*与i的值, 即\*7, 然后执行break语句, 跳出循环。  
所以选择D选项。

【23】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | double f(double x); |
|  | void main() { |
|  | double a = 0; |
|  | int i; |
|  | for (i = 0; i < 30; i += 10) |
|  | a += f((double) i); |
|  | printf("%3.0f\n", a); |
|  | } |
|  | double f(double x) { |
|  | return x \* x + 1; |
|  | } |

程序运行后的输出结果是

〖A〗500

〖B〗401

〖C〗503

〖D〗1404

本题重点考查函数的定义和调用, 第一次循环, a = 0, i = 0; 返回值a = 0 \* 0 + 1 = 1; 第二次循环, a = 1, i = 10; 返回值a = 10 \* 10 + 1 + 1 = 102; 第三次循环, a = 102, i = 20; 返回值a = 20 \* 20 + 1 + 102 = 503; 第四次循环, a = 503, i = 30; 不符合i < 30, 跳出循环, 最后结果输出a = 503。

【24】若有以下函数首部

复制代码

C

|  |  |
| --- | --- |
|  | int fun(double x[10], int \*n) |
|  |  |

则下面针对此函数的函数声明语句中正确的是

〖A〗int fun(double, int);

〖B〗int fun(double \*, int \*);

〖C〗int fun(double \*x, int n);

〖D〗int fun(double x, int \*n);

函数声明的一般形式为：类型说明符 被调函数名(类型 形参, 类型 形参…); 或为：类型说明符 被调函数名(类型, 类型…); 由于函数首部为int fun(double x[10], int \*n), 因此B选项正确。

【25】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int m = 1, n = 2, \*p = &m, \*q = &n, \*r; |
|  | r = p; |
|  | p = q; |
|  | q = r; |
|  | printf("%d,%d,%d,%d\n", m, n, \*p, \*q); |
|  | } |

程序运行后的输出结果是

〖A〗2,1,1,2

〖B〗1,2,1,2

〖C〗2,1,2,1

〖D〗1,2,2,1

int m = 1, n = 2, \*p = &m, \*q = &n, \*r; 即指针变量p指向m, 指针变量q指向n, r = p; p = q; q = r; 即通过指针变量r, 将指针p和指针q指向交换。  
因此最后输出1, 2, 2, 1。

【26】若有以下定义

复制代码

C

:int x[10], \*pt = x;

则对x数组元素的正确引用是

〖A〗pt + 3

〖B〗\*&x[10]

〖C〗\*(pt + 10)

〖D〗\*(x + 3)

A选项中, pt + 3是数组x的第4个元素地址; B选项中x[10]下标越界; \*(pt + i)表示引用指针pt所指元素后的第i个元素, 所以C选项错误, 最大只能为\*(pt + 9)。  
D选项中, 引用的是x数组的第4个元素, 因此D选项正确。

【27】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i, s = 0, t[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; |
|  | for (i = 0; i < 9; i += 2) |
|  | s += \*(t + i); |
|  | printf("%d\n", s); |
|  | } |

程序执行后的输出结果是

〖A〗20

〖B〗25

〖C〗45

〖D〗36

统计1 ~ 9九个数中的奇数和, 此题考察指向数组的指针。  
C语言规定数组名代表数组的首地址, 也就是第一个元素的地址。  
因此\*(t + i)代表数组的第i + 1个元素。  
程序运行的结果是1 + 3 + 5 + 7 + 9 = 25。

【28】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #define N 4 |
|  | void fun(int a[][N], int b[]) { |
|  | int i; |
|  | for (i = 0; i < N; i++) |
|  | b[i] = a[i][i]; |
|  | } |
|  | void main() { |
|  | int x[][N] = { { 1, 2, 3 }, { 4 }, { 5, 6, 7, 8 }, { 9, 10 } }, y[N], i; |
|  | fun(x, y); |
|  | for (i = 0; i < N; i++) |
|  | printf("%d,", y[i]); |
|  | printf("\n"); |
|  | } |

程序的运行结果是

〖A〗1, 0, 7, 0,

〖B〗1, 2, 3, 4,

〖C〗1, 4, 5, 9,

〖D〗3, 4, 8, 10,

该程序首先在定义变量时, 对二维数组x[][N]进行赋值操作; 调用函数fun, 函数fun的功能是将二维数组中的a[0][0]、a[1][1]、a[2][2]和a[3][3]赋值给一维数组。  
最后将一维数组1, 0, 7, 0, 输出。

【29】设有定义：

复制代码

C

char \*c;

以下选项中能够使c正确指向一个字符串的是

〖A〗char str[] = "string"; c = str;

〖B〗scanf("%s", c);

〖C〗c = getchar();

〖D〗\*c = "string";

A选项为正确用法, 先将字符串存于字符数组中, 然后将数组名赋给字符指针。  
(数组名代表数组首地址, 定义数组时为其分配确定地址)。  
C选项错误, getchar()函数输入个字符给字符型变量, 而不是字符指针。  
B选项和D选项有类似的错误, 两个选项并无语法错误, 但运行时可能会出现问题。  
因为在B选项和D选项中, 字符指针没有被赋值, 是个不确定的值, 指向一个不确定的内存区域, 这个区域可能存放有用的指令或数据。  
在这个不确定的区域重新存放字符串, 可能会发生无法预知的错误。  
因此A选项正确。

【30】若有定义语句：

复制代码

C

|  |  |
| --- | --- |
|  | char s[10] = "1234567\0\0"; |
|  | 则strlen(s)的值是 |

〖A〗7

〖B〗8

〖C〗9

〖D〗10

strlen()函数：是计算字符串长度时, 遇到结束标识为止, 且长度不包括结束标识。  
本题中字符串, 从第一个字符开始, 遇到第一个结束标识'\0'为止, 注意'\0'不占字符串长度, 所以字符串长度为7。  
因此A选项正确。

【31】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | void main() { |
|  | char p[20] = { 'a', 'b', 'c', 'd' }, q[] = "abc", r[] = "abcde"; |
|  | strcat(p, r); |
|  | strcpy(p + strlen(q), q); |
|  | printf("%d\n", strlen(p)); |
|  | } |

程序运行后的输出结果是

〖A〗11

〖B〗9

〖C〗6

〖D〗7

strcpy:字符串拷贝函数; strlen:求字符串长度函数(注意：不包含字符串结束标记字符'\0'); strcat:字符串连接函数。  
执行完语句strcat(p, r); 后, p数组中存储的元素为a, b, c, d, a, b, c, d, e; 执行语句strcpy(p + strlen(q), q); 得到的结果是将q所指向的字符串拷贝至p + strlen(q)开始的存储位置, 因为strlen的值为3, 即p + 3开始存储q中的元素。  
所以执行完strcpy(p + strlen(q), q)语句后, 字符数组p[20]的存储元素为a, b, c, a, b, c; 所以strlen(p)的结果为6。  
因此C选项正确。

【32】有以下程序(说明：字母A的ASCII码值是65)

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(char \*s) { |
|  | while (\*s) { |
|  | if (\*s % 2) |
|  | printf("%c", \*s); |
|  | s++; |
|  | } |
|  | } |
|  | void main() { |
|  | char a[] = "BYTE"; |
|  | fun(a); |
|  | printf("\n"); |
|  | } |

程序运行后的输出结果是

〖A〗BT

〖B〗YE

〖C〗YT

〖D〗BY

本题中执行fun(a)时, 会取a的第一个字母B, 因为B的ASCII码为66, 所以不会输出B, 函数只会输出ASCII码为奇数的字母, Y的ASCII码为89, T的ASCII码为84, E的ASCII码为69。  
因此B选项正确。

【33】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun() { |
|  | static int x = 1; |
|  | x += 1; |
|  | return x; |
|  | } |
|  | void main() { |
|  | int i, s = 1; |
|  | for (i = 1; i <= 5; i++) |
|  | s += fun(); |
|  | printf("%d\n", s); |
|  | } |

程序运行后的输出结果是

〖A〗120

〖B〗11

〖C〗6

〖D〗21

fun函数中的变量x为静态局部变量, 占用固定的内存单元, 下一次调用时仍可以保留上次调用时的值。  
也就是说, 如果多次调用fun函数, x的定义只在第一个调用时有效, 从第二次调用开始, x的定义相当于不存在, 直接使用上次x的值。  
fun函数被调用了5次, 每次调用后的返回值累加到s上。  
5次调用后, x的值为6, s的值为21。  
因此D选项正确。

【34】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun2(char a, char b) { |
|  | printf("%c %c", a, b); |
|  | } |
|  | char a = 'A', b = 'B'; |
|  | void fun1() { |
|  | a = 'C'; |
|  | b = 'D'; |
|  | } |
|  | void main() { |
|  | fun1(); |
|  | printf("%c %c ", a, b); |
|  | fun2('E', 'F'); |
|  | } |

程序的运行结果是

〖A〗A B E F

〖B〗C D E F

〖C〗A B C D

〖D〗C D A B

本题考查的是变量的作用范围。  
如果在同一个源文件中, 外部变量与局部变量同名, 则在局部变量的作用范围内, 外部变量被"屏蔽", 即不起作用。  
本题中, 在fun1()和main()函数内访问的是全局变量a和b, 在fun2()函数中访问的是形参变量a和b。  
所以, main()函数中的printf()函数输出的是在fun1()函数中被改变的全局变量a和b的值C D, 而fun2()函数中输出的是传递给它的实参值E F。  
因此B选项正确。

【35】以下关于typedef的叙述错误的是

〖A〗用typedef可以增加新类型

〖B〗typedef只是将已存在的类型用一个新的名字来代表

〖C〗用typedef可以为各种类型说明一个新名, 但不能用来为变量说明一个新名

〖D〗用typedef为类型说明一个新名, 通常可以增加程序的可读性

typedef并不是增加了新类型, 而是用一个新名字替代已存在的类型, 不能为变量说明一个新名, 使用typedef可以增强移植性。  
所以A选项错误。

【36】程序中已构成如下图所示的不带头结点的单向链表结构, 指针变量s、p、q均已正确定义, 并用于指向链表结点, 指针变量s总是作为指针指向链表的第一个结点。

复制代码

C

<img border = "0" alt = "" src = "http://res.weilaijiaoyu.cn/CourseFile/2CVip/PC/Images/ms4-36-1.png">

若有以下程序段：  
q = s; s = s->next; p = s;  
while (p->next) p = p->next;  
p->next = q; q->next = NULL;  
该程序段实现的功能是

〖A〗删除尾结点

〖B〗尾结点成为首结点

〖C〗删除首结点

〖D〗首结点成为尾结点

本题考查链表的操作, 本题中首先是指针q, 通过语句q = s使q指向a结点, 指针s指向了它的下一个节点b, 指针p也指向b结点。  
然后通过while循环, 指针p指向链表的最后一个元素。  
通过语句p->next = q改变最后一个元素的指向, 并将第一个元素q的next赋值为NULL值, 所以本题实现的效果是使首节点成为尾节点。  
故本题答案为D选项。

【37】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #define S(x) 4 \* (x) \* x + 1 |
|  | void main() { |
|  | int k = 5, j = 2; |
|  | printf("%d\n", S(k + j)); |
|  | } |

程序运行后的输出结果是

〖A〗33

〖B〗197

〖C〗143

〖D〗28

本题考查带参数的宏定义, S为带参数的宏定义, 运行S(k + j)为4 \* (k + j) \* k + j + 1 = 143, 选项C正确。

【38】若有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | int r = 8; |
|  | printf("%d\n", r >> 1); |

输出结果是

〖A〗16

〖B〗8

〖C〗4

〖D〗2

本题考查位运算, 属于基础知识, 题目中将整型变量8, 二进制表达为00001000, 右移一位为00000100, 即4, 选项C正确。

【39】以下叙述中错误的是

〖A〗gets函数用于从终端读入字符串

〖B〗getchar函数用于从磁盘文件读入字符

〖C〗fputs函数用于把字符串输出到文件

〖D〗fwrite函数用于以二进制形式输出数据到文件

本题考查字符处理函数和文件做操函数, 属于基础知识, 其中B选项getchar是用于从终端读入字符。

【40】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*pf; |
|  | char \*s1 = "China", \*s2 = "Beijing"; |
|  | pf = fopen("abc.dat", "wb+"); |
|  | fwrite(s2, 7, 1, pf); |
|  | rewind(pf); |
|  | /\*文件位置指针回到文件开头\*/ |
|  | fwrite(s1, 5, 1, pf); |
|  | fclose(pf); |
|  | } |

以上程序执行后abc.dat文件的内容是

〖A〗China

〖B〗Chinang

〖C〗ChinaBeijing

〖D〗BeijingChina

本题考查文件操作函数, fwrite和rewind函数, 题目中先是将s2字符串写入adc.dat中, 然后将写指针回到文件开头, 然后写入s1字符串, 那么s1字符串就将前五个字符覆盖, 所以最终结果为Chinang, 选项B正确。

**编程题**

[编程题代码下载](https://github.com/CoderMJLee/NCRE)

【41】使用VC++2010打开考生文件夹下blank1中的解决方案。此解决方案的项目中包含一个源程序文件blank1.c。在此程序中, 函数fun的功能是：计算形参x所指数组中N个数的平均值(规定所有数均为正数), 将所指数组中大于平均值的数据移至数组的前部, 小于等于平均值的数据移至x所指数组的后部, 平均值作为函数值返回, 在主函数中输出平均值和移动后的数据。  
例如, 有10个正数：46、30、32、40、6、17、45、15、48、26, 平均值为30.500000。  
移动后的输出为：46、32、40、45、48、30、6、17、15、26。  
请在程序的下画线处填入正确的内容并把下画线删除, 使程序得出正确的结果。  
注意：部分源程序在文件blank1.c中。  
不得增行或删行, 也不得更改程序的结构 !

复制代码

C

|  |  |
| --- | --- |
|  | (1) s / N |
|  | (2) j++ |
|  | (3) - 1 |

填空1:由原程序可知, av代表的是平均值, 而平均值的求法是所有数的总和除以数的个数, 因而本空应该填写s / N。  
填空2 : y数组代表暂时存放x数组, if (x[i] > av) 表达的是当x数组中的数大于平均值时, 应该把这些大于平均值的数放在y数组的前半部分, 因而此空应该填y[j++]。  
填空3:此空表明当x[i]不等于什么时, x[i]中的数要赋值给y数组, 由题意可知此空只能填写-1。

【42】使用VC++2010打开考生文件夹下modi1中的解决方案。此解决方案的项目中包含一个源程序文件modi1.c。此程序的功能是：读入一个英文文本行, 将其中每个单词的第一个字母改成大写, 然后输出此文本行(这里"单词"是指由空格隔开的字符串)。  
例如, 若输入"I am a student to take the examination", 则应输出"I Am A Student To Take The Examination"。  
请改正程序中的错误, 使程序能得出正确的结果。  
注意：部分源程序在文件modi1.c中。  
不得增行或删行, 也不得更改程序的结构 !

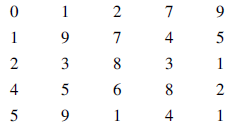
复制代码

C

|  |  |
| --- | --- |
|  | void upfst(char \*p) |
|  |  |

主函数中fun函数的调用方式说明函数fun的参数应为指针类型。

【43】使用VC++2010打开考生文件夹下prog1中的解决方案。此解决方案的项目中包含一个源程序文件prog1.c。在此程序中, 定义了N×N的二维数组, 并在主函数中赋值。请编写函数fun, 函数的功能是：求出数组周边元素的平均值并作为函数值返回给主函数中的s。例如, 若a 数组中的值为：



则返回主程序后s的值应为3.375。  
注意：部分源程序在文件prog1.c中。  
请勿改动主函数main和其他函数中的任何内容, 仅在函数fun的花括号中填入所编写的若干语句。

复制代码

C

|  |  |
| --- | --- |
|  | double fun(int w[][N]) { |
|  | int i, j, k = 0; |
|  | double sum = 0.0; |
|  | for (i = 0; i < N; i++) |
|  | for (j = 0; j < N; j++) |
|  | if (i == 0 || i == N - 1 || j == 0 || j == N - 1) |
|  | /\*只要下标中有一个为0或N - 1, 则它一定是周边元素\*/ { |
|  | sum = sum + w[i][j]; |
|  | /\*将周边元素求和\*/ |
|  | k++; |
|  | } |
|  | return sum / k; |
|  | /\*求周边元素的平均值\*/ |
|  | } |

本题要求计算二维数组周边元素的平均值, for循环语句控制循环过程, if条件语句根据数组元素的下标判断该元素是否为二维数组的周边元素。  
本题采用逐一判断的方式, 周边元素的规律为下标中有一个是0或N - 1, 所以只要下标中有一个为0或N - 1, 那么它一定是周边元素。  
计算周边元素个数的方式是当给sum累加一个值时, k也加1。