[**【计算机二级C语言】卷008**](https://www.cnblogs.com/mjios/p/12452735.html)

**选择题**

**公共知识**

【1】下列叙述中正确的是()。

〖A〗线性表的链式存储结构与顺序存储结构所需要的存储空间是相同的

〖B〗线性表的链式存储结构所需要的存储空间一般要多于顺序存储结构

〖C〗线性表的链式存储结构所需要的存储空间一般要少于顺序存储结构

〖D〗线性表的链式存储结构与顺序存储结构在存储空间的需求上没有可比性

线性链式存储结构中每个结点都由数据域与指针域两部分组成, 增加了存储空间, 所以一般要多于顺序存储结构。

【2】下列叙述中正确的是()。

〖A〗栈是一种先进先出的线性表

〖B〗队列是一种后进先出的线性表

〖C〗栈与队列都是非线性结构

〖D〗栈与队列都是线性结构

栈是一种先进后出的线性表, 队列是一种先进先出的线性表, 栈与队列都是线性结构。

【3】软件测试的目的是()。

〖A〗评估软件可靠性

〖B〗发现并改正程序中的错误

〖C〗改正程序中的错误

〖D〗发现程序中的错误

软件测试是为了发现错误而执行程序的过程, 测试要以查找错误为中心, 而不是为了演示软件的正确功能, 不是为了评估软件或改正错误。

【4】在软件开发中, 需求分析阶段产生的主要文档是()。

〖A〗软件集成测试计划

〖B〗软件详细设计说明书

〖C〗用户手册

〖D〗软件需求规格说明书

需求分析阶段的工作可以概括为：需求获取、需求分析、编写需求规格说明书、需求评审四个方面。  
所以选择D)。

【5】软件生命周期是指()。

〖A〗软件产品从提出、实现、使用维护到停止使用退役的过程

〖B〗软件从需求分析、设计、实现到测试完成的过程

〖C〗软件的开发过程

〖D〗软件的运行维护过程

通常, 将软件产品从提出、实现、使用维护到停止使用退役的过程称为软件生命周期。  
也就是说, 软件产品从考虑其概念开始, 到该软件产品不能使用为止的整个时期都属于软件生命周期。

【6】面向对象方法中, 继承是指()。

〖A〗一组对象所具有的相似性质

〖B〗一个对象具有另一个对象的性质

〖C〗各对象之间的共同性质

〖D〗类之间共享属性和操作的机制

继承是面向对象的方法的一个主要特征, 是使用已有的类的定义作为基础建立新类的定义技术。  
广义的说, 继承是指能够直接获得已有的性质和特征, 而不必重复定义它们, 所以说继承是指类之间共享属性和操作的机制。

【7】层次型、网状型和关系型数据库划分原则是()。

〖A〗记录长度

〖B〗文件的大小

〖C〗联系的复杂程度

〖D〗数据之间的联系方式

层次模型的基本结构是树形结构, 网状模型是一个不加任何条件限制的无向图, 关系模型采用二维表来表示, 所以三种数据库的划分原则是数据之间的联系方式。

【8】一个工作人员可以使用多台计算机, 而一台计算机可被多个人使用, 则实体工作人员与实体计算机之间的联系是()。

〖A〗一对一

〖B〗一对多

〖C〗多对多

〖D〗多对一

因为一个人可以操作多个计算机, 而一台计算机又可以被多个人使用, 所以两个实体之间是多对多的关系。

【9】数据库设计中反映用户对数据要求的模式是()。

〖A〗内模式

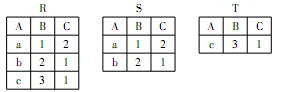
〖B〗概念模式

〖C〗外模式

〖D〗设计模式

数据库系统的三级模式是概念模式、外模式和内模式。  
概念模式是数据库系统中全局数据逻辑结构的描述, 是全体用户公共数据视图。  
外模式也称子模式或用户模式, 它是用户的数据视图, 给出了每个用户的局部数据描述, 所以选择C)。  
内模式又称物理模式, 它给出了数据库物理存储结构与物理存取方法。

【10】有三个关系R、S和T如下：  
则由关系R和S得到关系T的操作是()。



〖A〗自然连接

〖B〗差

〖C〗交

〖D〗并

关系T中的元组是R关系中有而S关系中没有的元组的集合, 所以进行的是差的运算。

**专业知识**

【11】计算机能直接执行的程序是

〖A〗源程序

〖B〗目标程序

〖C〗汇编程序

〖D〗可执行程序

计算机能直接执行的程序是二进制的可执行程序, 扩展名为.exe。  
所以选择D选项。

【12】以下叙述中正确的是

〖A〗C语言规定必须用main作为主函数名, 程序将从此开始执行

〖B〗可以在程序中由用户指定任意一个函数作为主函数, 程序将从此开始执行

〖C〗C语言程序将从源程序中第一个函数开始执行

〖D〗main的各种大小写拼写形式都可以作为主函数名, 如：MAIN, Main等

C语言中的主函数唯一为main()函数不能任意指定, 所以B选项错误。  
C语言从主函数main()开始, 到主函数main()结束, 所以C选项错误。  
主函数必须写成小写的main, 不能混淆大小写, 所以D选项错误。

【13】以下选项中可用作C程序合法实数的是

〖A〗3.0e0.2

〖B〗.1e0

〖C〗E9

〖D〗9.12E

A选项中E后面的指数必须为整数?C语言规定, E之前必须要有数字, 所以C选项错误?E后面必须要有数字, 且必须为整数, 所以D选项错误。

【14】下列定义变量的语句中错误的是

〖A〗int \_int;

〖B〗double int\_;

〖C〗char For;

〖D〗float US$;

C语言规定, 变量命名必须符合标识符的命名规则。  
D选项中包含了非法字符"$", 所以错误。  
标识符由字母、数字或下划线组成, 且第一个字符必须是大小写英文字母或者下划线, 而不能是数字。  
大写字符与小写字符被认为是两个不同的字符, 所以For不是关键字for。

【15】表达式：(int) ((double) 9 / 2) - 9 % 2 的值是

〖A〗0

〖B〗3

〖C〗4

〖D〗5

先将整型数据9强制转换成double型, 然后除以2得到的结果与double型保持一致, 即为4.5, 然后将4.5强制转换成整型数据4, 然后计算9 % 2的值为1, 最后计算4 - 1的值为3, 所以选择B选项。

【16】设变量均已正确定义, 若要通过  
scanf("%d%c%d%c", &a1, &c1, &a2, &c2);  
语句为变量a1和a2赋数值10和20, 为变量c1和c2赋字符X和Y。以下所示的输入形式中正确的是(注：□代表空格字符)

〖A〗10□X<回车> 20□Y<回车>

〖B〗10□X20□Y<回车>

〖C〗10X<回车> 20Y<回车>

〖D〗10□X□20□Y<回车>

输入数据的格式必须与scanf()的格式控制串完全匹配, A、B、D选项中在数字10后面均输入了空格, 则会将空格赋给变量c1, 而不是把X赋给c1, 所以选择C选项。

【17】以下选项中不能作为C语言合法常量的是

〖A〗0.1e + 6

〖B〗'cd'

〖C〗"\a"

〖D〗'\011'

要想表示字符串常量应该用双引号表示, 即"cd", 单引号 用于表示字符常量, 所以B选项中的表示方法错误。

【18】if语句的基本形式是：if (表达式) 语句, 以下关于"表达式"值的叙述中正确的是

〖A〗必须是逻辑值

〖B〗必须是整数值

〖C〗必须是正数

〖D〗可以是任意合法的数值

if语句中的表达式可以是任意合法的数值, 如常量, 变量表达式。  
所以D选项正确。

【19】有如下嵌套的if语句

复制代码

C

|  |  |
| --- | --- |
|  | if (a < b) |
|  | if (a < c) |
|  | k = a; |
|  | else |
|  | k = c; |
|  | else if (b < c) |
|  | k = b; |
|  | else |
|  | k = c; |

以下选项中与上述if语句等价的语句是

〖A〗k = (a < b) ? ((b < c) ? a : b) : ((b > c) ? b : c);

〖B〗k = (a < b) ? ((a < c) ? a : c) : ((b < c) ? b : c);

〖C〗k = (a < b) ? a : b; k = (b < c) ? b : c;

〖D〗k = (a < b) ? a : b; k = (a < c) ? a : c;

题目中嵌套语句的含义是当a < b且a < c成立, 则将a的值赋给k, 如果a < b成立而a < c不成立将c的值赋给k。  
如果a < b不成立而b < c成立, 将b的值赋给k, 如果如果a < b不成立且b < c也不成立, 将c的值赋给k。  
判断条件表达式只有B选项可以表示这个含义。

【20】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int k = 5; |
|  | while (--k) |
|  | printf("%d", k -= 3); |
|  | printf("\n"); |
|  | } |

执行后的输出结果是

〖A〗1

〖B〗2

〖C〗4

〖D〗死循环

--k先自减在使用, 所以第一次判断条件即为while(4), 条件为真执行打印k的值k = 4 - 3, 结果为1, 第二次判断条件即为while(0), 条件为假, 结束循环, 打印回车换行符。  
所以选择A选项。

【21】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i, j; |
|  | for (i = 3; i >= 1; i--) { |
|  | for (j = 1; j <= 2; j++) |
|  | printf("%d ", i + j); |
|  | printf("\n"); |
|  | } |
|  | } |

程序的运行结果是

〖A〗4 3

2 5

4 3

〖B〗4 5

3 4

2 3

〖C〗2 3

3 4

4 5

〖D〗2 3

3 4

2 3

第一次执行外循环, i的值为3, 第一次执行内循环j的值为1, j <= 2条件成立, 打印i + j的值, 即4; 第二次执行内循环j的值为2, j <= 2条件成立, 打印i + j的值, 即5; 再判断条件不成立, 打印回车换行符, 跳出内循环。  
第二次执行外循环, i的值为2, 内循环的执行同第一次一样, 所以打印3和4, 打印回车换行, 结束内循环。  
第三次执行外循环i的值为1, 同理内循环中打印2和3, 打印回车换行符, 结束内循环。

【22】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int k = 5, n = 0; |
|  | do { |
|  | switch (k) { |
|  | case 1: |
|  | case 3: |
|  | n += 1; |
|  | k--; |
|  | break; |
|  | default: |
|  | n = 0; |
|  | k--; |
|  | case 2: |
|  | case 4: |
|  | n += 2; |
|  | k--; |
|  | break; |
|  | } |
|  | printf("%d", n); |
|  | } while (k > 0 && n < 5); |
|  | } |

程序运行后的输出结果是

〖A〗02356

〖B〗0235

〖C〗235

〖D〗2356

k的值为5, 逐个对case语句进行匹配, 均不匹配, 执行default下的语句n = 0; k--。  
再执行case2 : case 4: 后面的语句n += 2; k--; 。  
执行break语句跳出switch判断, 打印n的值, 即2。  
经过第一次循环后, k的值为3, 然后进行k > 0 && n < 5条件判断为真。  
第二次执行循环体, 执行case3:后面的n += 1; k--; 。  
再执行break语句跳出循环, 此时打印n的值3, 此时k的值为2。  
第三次执行循环体k的值为2, 满足条件, 执行case2 : case 4: 后面的语句n += 2, k--, 。  
执行break语句跳出循环体, 打印n的值5。  
此时n不满足条件k > 0 && n < 5, 所以结束循环。  
整个程序的打印结果为235。

【23】以下关于return 语句的叙述中正确的是

〖A〗一个自定义函数中必须有一条return 语句

〖B〗一个自定义函数中可以根据不同情况设置多条return 语句

〖C〗定义成void类型的函数中可以有带返回值的return 语句

〖D〗没有return 语句的自定义函数在执行结束时不能返回到调用处

在函数中允许有多个return 语句, 但每次调用只能有一个return 语句被执行, 因此只能返回一个函数值, A选项描述错误。  
定义成void类型的函数, 不允许从该函数取得返回值, 也不允许使用return 语句, C选项描述错误。  
没有return 语句的函数在执行到函数的最后一条语句后会自动返回到调用处, D选项描述错误。  
因此B选项正确。

【24】已定义以下函数

复制代码

C

|  |  |
| --- | --- |
|  | int fun(int \*p) { |
|  | return \*p; |
|  | } |
|  | fun函数返回值是 |

〖A〗一个整数

〖B〗不确定的值

〖C〗形参p中存放的值

〖D〗形参p的地址值

返回值为指针变量指向的数据, 该数据已被定义为整型。  
因此A选项正确。

【25】以下程序段完全正确的是

〖A〗int \*p; scanf("%d", &p);

〖B〗int \*p; scanf("%d", p);

〖C〗int k, \*p = &k; scanf("%d", p);

〖D〗int k, \*p; \*p = &k; scanf("%d", p);

A选项中错在没有对指针进行初始化, 无效指针, 并且在scanf("%d", &p)中无需再进行取地址操作; B选项中没有对指针进行初始化, 无效指针; D选项中语句\*p = &k; 的左端\*p 是指针所指内存空间的值, &k 是地址, 应为p = &k。  
因此C选项正确。

【26】设有定义：

复制代码

C

double a[10], \*s = a;

以下能够代表数组元素a[3]的是

〖A〗(\*s)[3]

〖B〗\*(s + 3)

〖C〗\*s[3]

〖D〗\*s + 3

数组a是10个元素的数组, a是数组名, 字符指针指向数组a。  
选项A中(\*s)[3]无法表示元素a[3]; 选项C中s[3]相当于a[3], \*s[3]是不正确的表达方式; 选项D中\*s + 3, \*s表示元素a[0], \*s + 3也就是a[0] + 3不能表示元素a[3], 本题答案为B选项。

【27】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void f(int \*q) { |
|  | int i = 0; |
|  | for (; i < 5; i++) |
|  | (\*q)++; |
|  | } |
|  | void main() { |
|  | int a[5] = { 1, 2, 3, 4, 5 }, i; |
|  | f(a); |
|  | for (i = 0; i < 5; i++) |
|  | printf("%d,", a[i]); |
|  | } |

程序运行后的输出结果是

〖A〗6, 2, 3, 4, 5,

〖B〗2, 2, 3, 4, 5,

〖C〗1, 2, 3, 4, 5,

〖D〗2, 3, 4, 5, 6,

本题考查的是函数的地址调用, 将数组名作为数组首地址进行传递, 循环中, 将数组首元素5次加1等于6, 因此A选项正确。

【28】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun(int(\*s)[4], int n, int k) { |
|  | int m, i; |
|  | m = s[0][k]; |
|  | for (i = 1; i < n; i++) |
|  | if (s[i][k] > m) |
|  | m = s[i][k]; |
|  | return m; |
|  | } |
|  | void main() { |
|  | int a[4][4] = { { 1, 2, 3, 4 }, { 11, 12, 13, 14 }, { 21, 22, 23, 24 }, { 31, 32, 33, 34 } }; |
|  | printf("%d\n", fun(a, 4, 0)); |
|  | } |

程序的运行结果是

〖A〗4

〖B〗34

〖C〗31

〖D〗32

本题重点考查二维数组名作为实参进行参数传递, 在主函数中调用了fun函数, 实参为二维数组名a和两个整数4、0, 这样对应定义fun函数首部也有三种形式, 这里采用了第一种形式, 也就是行指针, 这样在fun函数对s[i][j]进行操作实际上就是对主函数中的a[i][j]进行操作, 再分析fun函数的作用便可知, 是求二维数组第0列中最大的那个元素。  
因此C选项正确。

【29】以下选项中正确的语句组是

〖A〗char \*s; s = { "BOOK!" };

〖B〗char \*s; s = "BOOK!";

〖C〗char s[10]; s = "BOOK!";

〖D〗char s[]; s = "BOOK!";

A选项"char \*s; s={"BOOK !"};"字符指针不能这样赋值; C选项和D选项数组s应在定义时赋初值。  
因此B选项正确。

【30】若有定义语句：

复制代码

C

char \*s1 = "OK", \*s2 = "ok";

以下选项中, 能够输出"OK"的语句是

〖A〗if (strcmp(s1, s2) != 0) puts(s2);

〖B〗if (strcmp(s1, s2) != 0) puts(s1);

〖C〗if (strcmp(s1, s2) == 1) puts(s1);

〖D〗if (strcmp(s1, s2) == 0) puts(s1);

strcmp(s1, s2)是字符串比较函数, 比较规则是两个字符串自左向右逐个字符相比(按照ASCII码值大小), 以第一个不相同的字符的大小作为比较结果。  
因为'OK' < 'ok', 所以strcmp(s1, s2)为负值, 只有B选项正确。

【31】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(char \*\*p) { |
|  | ++p; |
|  | printf("%s\n", \*p); |
|  | } |
|  | void main() { |
|  | char \*a[] = { "Morning", "Afternoon", "Evening", "Night" }; |
|  | fun(a); |
|  | } |

程序的运行结果是

〖A〗Afternoon

〖B〗fternoon

〖C〗Morning

〖D〗orning

本题主函数中定义了一个指针数组a, 可以将它看成一个以字符指针为元素的一维数组。  
和-般的一维数组名能赋给同类型指针变量一样, 该字符指针数组也可以赋给指向字符指针的指针, 所以数组名a可以用作函数fun()的实参。  
在fun()函数中, ++p操作使形参p往后移动一个字符指针的位置, 即指针a[1]的位置。  
故最后通过printf()函数输出的字符串为"Afternoon"。  
因此A选项正确。

【32】有以下程序, 程序中库函数islower(ch)用以判断ch中的字母是否为小写字母：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <ctype.h> |
|  | void fun(char \*p) { |
|  | int i = 0; |
|  | while (p[i]) { |
|  | if (p[i] == ' ' && islower(p[i - 1])) |
|  | p[i - 1] = p[i - 1] - 'a' + 'A'; |
|  | i++; |
|  | } |
|  | } |
|  | void main() { |
|  | char s1[100] = "ab cd EFG !"; |
|  | fun(s1); |
|  | printf("%s\n", s1); |
|  | } |

程序运行后的输出结果是

〖A〗ab cd EFg !

〖B〗Ab Cd EFg !

〖C〗ab cd EFG !

〖D〗aB cD EFG !

调用fun函数后, 实参数组名s1传给形参指针p, p也指向数组s1。  
fun函数的功能是遍历字符数组中的所有字符, 如果某一个字符是空格并且前一个字符是小写字母的话, 就把前一个字符转换为大写。  
程序中语句p[i - 1] = p[i - 1] - 'a' + 'A'; 即p[i - 1] = p[i - 1] - 32, 意思是将p[i - 1]中的字符转换为大写。  
因此D选项正确。

【33】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int x) { |
|  | int y; |
|  | if (x == 0 || x == 1) |
|  | return (3); |
|  | y = x \* x - f(x - 2); |
|  | return y; |
|  | } |
|  | void main() { |
|  | int z; |
|  | z = f(3); |
|  | printf("%d\n", z); |
|  | } |

程序的运行结果是

〖A〗0

〖B〗9

〖C〗6

〖D〗8

主函数中调用f函数, 同时将数值3传递给变量x, 执行条件与if(x == 0 || x == 1); 此时条件不成立, 跳过return (3);, 直接执行语句y = x \* x - f(x - 2); 这里f(x - 2)为函数的递归调用, 再次调用f函数, 此时x = 1, 条件语句条件成立, 返回3给f(1), y = 3 \* 3 - 3 = 6, 最后将6返回主函数, 并进行输出。  
因此C选项正确。

【34】有以下程序：

复制代码

C

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

程序执行后的输出结果是

〖A〗55

〖B〗50

〖C〗45

〖D〗60

内部静态变量是始终存在的, 当函数被调用退出后, 内部静态变量会保存数据, 再次调用该函数时, 以前调用时的数值仍然保留着。  
fun(a, 5)的值是15, 再次调用后sum = 15, 所以fun(b, 4) = 45, s = 45 + 15 = 60。  
因此D选项正确。

【35】有以下结构体说明、变量定义和赋值语句：

复制代码

C

|  |  |
| --- | --- |
|  | struct STD { |
|  | char name[10]; |
|  | int age; |
|  | char sex; |
|  | } s[5], \*ps; |
|  | ps = &s[0]; |

则以下scanf函数调用语句有错误的是

〖A〗scanf("%s", s[0].name);

〖B〗scanf("%d", &s[0].age);

〖C〗scanf("%c", &(ps->sex));

〖D〗scanf("%d", ps->age);

本题考查结构体的相关知识, 题目中需要输入一个变量, scanf要求参数为指针, 而D选项中ps->age为一个变量, 不是指针, 所以错误。

【36】若有以下语句：

复制代码

C

|  |  |
| --- | --- |
|  | typedef struct S { |
|  | int g; |
|  | char h; |
|  | } T; |

以下叙述中正确的是

〖A〗可用S定义结构体变量

〖B〗可用T定义结构体变量

〖C〗S是struct 类型的变量

〖D〗T是struct S类型的变量

本题考查typedef重新声明一种结构体类型, 那么T为结构体类型, 而不是结构体变量, 所以B选项正确。

【37】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | struct A { |
|  | int a; |
|  | char b[10]; |
|  | double c; |
|  | }; |
|  | struct A f(struct A t); |
|  | void main() { |
|  | struct A a = { 1001, "ZhangDa", 1098.0 }; |
|  | a = f(a); |
|  | printf("%d,%s,%6.1f\n", a.a, a.b, a.c); |
|  | } |
|  | struct A f(struct A t) { |
|  | t.a = 1002; |
|  | strcpy(t.b, "ChangRong"); |
|  | t.c = 1202.0; |
|  | return t; |
|  | } |

程序运行后的输出结果是

〖A〗1002,ZhangDa,1202.0

〖B〗1002,ChangRong,1202.0

〖C〗1001,ChangRong,1098.0

〖D〗1001,ZhangDa,1098.0

本题考查结构体变量的引用以及作为函数参数, 题目虽然看似复杂, 其实比较容易, f函数的参数是结构体变量, 然后对参数重新赋值并返回, 所以该题目的答案为B选项。

【38】设有宏定义：

复制代码

C

#define IsDIV(k, n) ((k % n == 1) ? 1 : 0)

且变量m已正确定义并赋值, 则宏调用：IsDIV(m, 5) && IsDIV(m, 7)  
为真时所要表达的是

〖A〗判断m是否能被5和7整除

〖B〗判断m被5和7整除是否都余1

〖C〗判断m被5或者7整除是否余1

〖D〗判断m是否能被5或者7整除

宏定义isDIV(k, n)表达式为((k % n == 1) ? 1 : 0), 替换字符串为问号表达式, 如果k % n是否为1, 为1返回1否则返回0。  
IsDIV(m, 5) && IsDIV(m, 7)这个表达式为真时, 表示m % 5和m % 7都为1, 所以本题答案为B选项。

【39】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 1, b = 2, c = 3, x; |
|  | x = (a ^ b) & c; |
|  | printf("%d\n", x); |
|  | } |

程序的运行结果是

〖A〗3

〖B〗1

〖C〗2

〖D〗0

本题考查位运算符以及相关运算, ^为按位异或, &为按位与, 那么a ^ b为3, 再与c按位与仍然为3, 所以答案为A选项。

【40】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*fp; |
|  | int k, n, a[6] = { 1, 2, 3, 4, 5, 6 }; |
|  | fp = fopen("d2.dat", "w"); |
|  | fprintf(fp, "%d%d%d\n", a[0], a[1], a[2]); |
|  | fprintf(fp, "%d%d%d\n", a[3], a[4], a[5]); |
|  | fclose(fp); |
|  | fp = fopen("d2.dat", "r"); |
|  | fscanf(fp, "%d%d", &k, &n); |
|  | printf("%d %d\n", k, n); |
|  | fclose(fp); |
|  | } |

程序运行后的输出结果是

〖A〗1 2

〖B〗1 4

〖C〗123 4

〖D〗123 456

fprintf()函数向文件输出, 将输出的内容输出到硬盘上的文件或是相当于文件的设备上执行两次fprintf后文件中有123 456, 所以D选项正确。

**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), 将所指数组中小于平均值的数据移至数组的前部, 大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：47、30、32、40、6、17、45、15、48、26, 其平均值为30.500000。  
移动后的输出为：30、6、17、15、26、47、32、40、45、48。  
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) x[i] / N |
|  | (2) j++或++j |
|  | (3) i++或++i |

填空1 : av代表平均值, 本题考查了怎样求平均值, 因此本空应该填写x[i] / N。  
填空2:通过for循环和if判断找到x[i]中比平均值小的数, 并把这些值赋值给y[j], 因此本空应该填写j++或者++j。  
填空3:通过while循环语句, 把x[i]中比平均值大的数放在数组y的后半部分, 因此本空应该填写i++或者++i。

【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。在此程序中, 函数fun的功能是：统计字符串中各元音字母(即A、E、I、O、U)的个数。注意：字母不分大小写。  
例如, 输入"THIs is a boot", 则应输出是1 0 2 2 0。  
请改正程序中的错误, 使它能得出正确的结果。  
注意：部分源程序在文件modi1.c中。  
不要改动main函数, 不得增行或删行, 也不得更改程序的结构 !

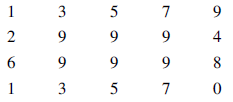
复制代码

C

|  |  |
| --- | --- |
|  | (1) void fun(char \* s, int num[5]) |
|  | (2) num[k] = 0; |
|  | (3) switch (\*s) |

switch语句说明如下：  
(1) switch后的表达式, 可以是整型或字符型, 也可以是枚举类型。  
在新的ANSIC标准中允许表达式的类型为任何类型。  
(2) 每个case后的常量表达式只能是由常量组成的表达式, 当switch后的表达式的值与某一个常量表达式的值一致时, 程序就转到此case后的语句开始执行。  
如果没有一个常量表达式的值与switch后的值一致, 就执行default后的语句。  
(3) 各个case后的常量表达式的值必须互不相同。  
(4) 各个case的次序不影响执行结果, 一般情况下, 尽量将出现概率大的case放在前面。  
(5) 在执行完一个case后面的语句后, 程序会转到下一个case后的语句开始执行, 因而必须使用break语句才能跳出。

【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, 编写函数fun, 该函数的功能是：求出二维数组周边元素之和, 作为函数值返回。二维数组中的值在主函数中赋予。  
例如, 若二维数组中的值为：



则函数值为61。  
注意：部分源程序在文件prog1.c中。  
请勿改主动函数main和其他函数中的任何内容, 仅在函数fun的花括号中填入你编写的若干语句。

复制代码

C

|  |  |
| --- | --- |
|  | int fun(int a[M][N]) { |
|  | int i, j, sum = 0; |
|  | for (i = 0; i < M; i++) |
|  | for (j = 0; j < N; j++) |
|  | if (i == 0 || i == M - 1 || j == 0 || j == N - 1) |
|  | /\*只要下标中有一个为0或M - 1或N - 1, 则它一定是周边元素\*/ |
|  | sum = sum + a[i][j]; |
|  | /\*将周边元素相加\*/ |
|  | return sum; |
|  | } |

本题采用逐一判断的方式, 周边元素的规律是, 其下标值中一定有一个是0或M - 1或N - 1。  
程序中循环语句用来控制数组的行和列, 条件语句用来判断数组元素是否为周边元素。