[**【计算机二级C语言】卷006**](https://www.cnblogs.com/mjios/p/12452728.html)

**选择题**

**公共知识**

【1】下列叙述中正确的是()。

〖A〗栈是"先进先出"的线性表

〖B〗队列是"先进后出"的线性表

〖C〗循环队列是非线性结构

〖D〗有序线性表既可以采用顺序存储结构, 也可以采用链式存储结构

栈是先进后出的线性表, 所以A)错误; 队列是先进先出的线性表, 所以B)错误; 循环队列是线性结构的线性表, 所以C)错误。

【2】支持子程序调用的数据结构是()。

〖A〗栈

〖B〗树

〖C〗队列

〖D〗二叉树

栈支持子程序调用。  
栈是一种只能在一端进行插入或删除的线性表, 在主程序调用子函数时要首先保存主程序当前的状态, 然后转去执行子程序, 最终把子程序的执行结果返回到主程序中调用子程序的位置, 继续向下执行, 这种调用符合栈的特点, 因此本题的答案为A)。

【3】某二叉树有5个度为2的结点, 则该二叉树中的叶子结点数是()。

〖A〗10

〖B〗8

〖C〗6

〖D〗4

根据二叉树的基本性质3:在任意一棵二叉树中, 度为0的叶子节点总是比度为2的节点多一个, 所以本题中是5 + 1 = 6个。

【4】下列排序方法中, 最坏情况下比较次数最少的是()。

〖A〗冒泡排序

〖B〗简单选择排序

〖C〗直接插入排序

〖D〗堆排序

冒泡排序与简单插入排序与简单选择排序法在最坏情况下均需要比较n(n - 1) / 2次, 而堆排序在最坏情况下需要比较的次数是



。

【5】软件按功能可以分为：应用软件、系统软件和支撑软件(或工具软件)。下面属于应用软件的是()。

〖A〗编译程序

〖B〗操作系统

〖C〗教务管理系统

〖D〗汇编程序

编译软件、操作系统、汇编程序都属于系统软件, 只有C)教务管理系统才是应用软件。

【6】下面叙述中错误的是()

〖A〗软件测试的目的是发现错误并改正错误

〖B〗对被调试的程序进行"错误定位"是程序调试的必要步骤

〖C〗程序调试通常也称为Debug

〖D〗软件测试应严格执行测试计划, 排除测试的随意性

软件测试的目的是为了发现错误而执行程序的过程, 并不涉及改正错误, 所以选项A)错误。  
程序调试的基本步骤有：错误定位、修改设计和代码, 以排除错误、进行回归测试, 防止引进新的错误。  
程序调试通常称为Debug, 即排错。  
软件测试的基本准则有：所有测试都应追溯到需求、严格执行测试计划, 排除测试的随意性、充分注意测试中的群集现象、程序员应避免检查自己的程序、穷举测试不可能、妥善保存测试计划等文件。

【7】耦合性和内聚性是对模块独立性度量的两个标准。下列叙述中正确的是()。

〖A〗提高耦合性降低内聚性有利于提高模块的独立性

〖B〗降低耦合性提高内聚性有利于提高模块的独立性

〖C〗耦合性是指一个模块内部各个元素间彼此结合的紧密程度

〖D〗内聚性是指模块间互相连接的紧密程度

模块独立性是指每个模块只完成系统要求的独立的子功能, 并且与其他模块的联系最少且接口简单。  
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, 所以A)错误, B)正确。  
耦合性是模块间互相连接的紧密程度的度量而内聚性是指一个模块内部各个元素间彼此结合的紧密程度, 所以C)与D)错误。

【8】数据库应用系统中的核心问题是()。

〖A〗数据库设计

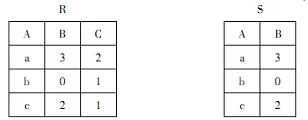
〖B〗数据库系统设计

〖C〗数据库维护

〖D〗数据库管理员培训

数据库应用系统中的核心问题是数据库的设计。

【9】有两个关系R, S如下：  
由关系R通过运算得到关系S, 则所使用的运算为()。



〖A〗选择

〖B〗投影

〖C〗插入

〖D〗连接

投影运算是指对于关系内的域指定可引入新的运算。  
本题中S是在原有关系R的内部进行的, 是由R中原有的那些域的列所组成的关系。  
所以选择B)。

【10】将E - R图转换为关系模式时, 实体和联系都可以表示为()。

〖A〗属性

〖B〗键

〖C〗关系

〖D〗域

从E - R图到关系模式的转换是比较直接的, 实体与联系都可以表示成关系, E - R图中属性也可以转换成关系的属性。

**专业知识**

【11】以下叙述中错误的是()。

〖A〗使用三种基本结构构成的程序只能解决简单问题

〖B〗结构化程序由顺序、分支、循环三种基本结构组成

〖C〗C语言是一种结构化程序设计语言

〖D〗结构化程序设计提倡模块化的设计方法

使用顺序, 选择(分支), 循环三种基本结构构成的程序可以解决所有问题, 而不只是解决简单问题, 所以A)错误。

【12】以下四个程序中, 完全正确的是()。

〖A〗#include < stdio.h > void main(); { /\*/ programming / \*/ printf("programming!\n"); }

〖B〗#include < stdio.h > void main() { /\*programming\*/ printf("programming!\n"); }

〖C〗#include < stdio.h > void main() { /\*/\*programming\*/\*/ printf("programming!\n"); }

〖D〗include < stdio.h > void main() { /\*programming\*/ printf("programming!\n"); }

C语言中注释语句的注释方法是：/\*注释内容\*/ 或//注释一行。  
所以A)与C)错误, D)选项中预编译命令include < stdio.h>前丢掉了"#"号。  
所以选择B)。

【13】C源程序中不能表示的数制是()。

〖A〗十六进制

〖B〗八进制

〖C〗十进制

〖D〗二进制

在C语言中整型常量可以用十进制、八进制和十六进制等形式表示, 但不包括二进制, 所以选择D)。

【14】以下选项中, 能用作用户标识符的是()。

〖A〗\_0\_

〖B〗8\_8

〖C〗void

〖D〗unsigned

C语言中标识符由字母、下划线、数字组成, 且开头必须是字母或下划线。  
另外, 关键字不能作为标识符。  
B)中以数字8开头, 所以错误。  
C)与D)中用的是关键字void与unsigned, 所以错误。

【15】若有定义语句：int x = 10;, 则表达式x -= x + x的值为()。

〖A〗0

〖B〗-20

〖C〗-10

〖D〗10

算术运算符+的优先级高于-=, 且-=的结合方向为自右向左, 所以表达式x -= x + x可以表示成x = x - (x + x) = 10 - (10 + 10) = -10, 选择C)。

【16】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 1, b = 0; |
|  | printf("%d,", b = a + b); |
|  | printf("%d\n", a = 2 \* b); |
|  | } |

程序运行后的输出结果是()。

〖A〗1,2

〖B〗1,0

〖C〗3,2

〖D〗0,0

首先打印b = a + b = 1 + 0 = 1的值1, 此时已给b赋值为1。  
然后打印a = 2 \*b = 2 \*1 = 2的值2。  
所以结果是1, 2.

【17】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a1, a2; |
|  | char c1, c2; |
|  | scanf("%d%c%d%c", &a1, &c1, &a2, &c2); |
|  | printf("%d,%c,%d,%c", a1, c1, a2, c2); |
|  | } |

若想通过键盘输入, 使得a1的值为12, a2的值为34, c1的值为字符a, c2的值为字符b, 程序输出结果是：12, a, 34, b 则正确的输入格式是(以下□代表空格, 代表回车)()。

〖A〗12□a34□b

〖B〗12□a□34□b

〖C〗12,a,34,b

〖D〗12a34b

在输入多个数据时, 若格式控制串中无非格式字符, 则认为所有输入的字符均为有效字符。  
所以应按选项D)的顺序输入数据。

【18】若变量已正确定义, 在if(W)printf("%d\n", k); 中, 以下不可替代W的是()。

〖A〗a<>b + c

〖B〗ch = getchar()

〖C〗a == b + c

〖D〗a++

选项A)是非法的表达式, C语言中没有<>运算符。

【19】有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int a, b, c; |
|  | a = 10; |
|  | b = 50; |
|  | c = 30; |
|  | if (a > b) |
|  | a = b, b = c; |
|  | c = a; |
|  | printf("a=%d b=%d c=%d\n", a, b, c); |

程序的输出结果是()。

〖A〗a=10 b=50 c=30

〖B〗a=10 b=50 c=10

〖C〗a=10 b=30 c=10

〖D〗a=50 b=30 c=50

本题中a > b的条件不满足, 所以不执行逗号表达式a = b, b = c; 的操作, 而是执行c = a操作, 即c的值为10。

【20】下列叙述中正确的是()。

〖A〗在switch语句中, 不一定使用break语句

〖B〗在switch语句中必须使用default

〖C〗break语句必须与switch语句中的case配对使用

〖D〗break语句只能用于switch语句

default语句在switch语句中可以省略, 所以B)错误; switch语句中并非每个case后都需要使用break语句, 所以C)错误; break语句还可以用于for等循环结构中, 所以D)错误。

【21】以下不构成无限循环的语句或语句组是()。

〖A〗n = 0; do { ++n; } while (n <= 0);

〖B〗n = 0; while (1) { n++; }

〖C〗n = 10; while (n); { n--; }

〖D〗for (n = 0, i = 1; ; i++) n += i;

选项A)中do后面的语句只执行了一次便结束了循环; B)选项中条件while(1)永远成立, 所以是死循环; C)选项中n的值为10, 而循环体为空语句, 所以while(n)永远为真, 进入死循环; D)选项中for语句第二个表达式为空, 所以没有判别条件, 进入死循环。

【22】有以下程序：

复制代码

C

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

程序运行后的输出结果是()。

〖A〗7

〖B〗5

〖C〗3

〖D〗9

向switch语句块传送参数后, 编译器会先寻找匹配的case语句块, 找到后就执行该语句块, 遇到break跳出; 如果没有匹配的语句块, 则执行default语句块。  
case与default没有顺序之分。  
所以第一次循环k的值为1, 执行c += k, c的值为1, 再执行case 2 后的语句c++, c的值为2, 遇到break语句跳出循环; 第二次循环k的值为2, 执行case 2 后面的语句c++, c的值为3, 跳出循环。

【23】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int x, int y) { |
|  | return ((y - x) \* x); |
|  | } |
|  | void main() { |
|  | int a = 3, b = 4, c = 5, d; |
|  | d = f(f(a, b), f(a, c)); |
|  | printf("%d\n", d); |
|  | } |

程序运行后的输出结果是()。

〖A〗7

〖B〗10

〖C〗8

〖D〗9

调用f(a, b)函数返回3, 调用f(a, c)函数返回6, 所以外层调用f(f(a, b), f(a, c)); 即调用f(3, 6)函数返回9。

【24】若有定义语句：double a, \*p = &a; 以下叙述中错误的是()。

〖A〗定义语句中的\*号是一个间址运算符

〖B〗定义语句中的\*号是一个说明符

〖C〗定义语句中的p只能存放double类型变量的地址

〖D〗定义语句中, \*p = &a把变量a的地址作为初值赋给指针变量p

在变量定义double a, \*p = &a; 中, \*号是一个指针运算符, 而非间址运算符, 所以A)错误。

【25】若有定义语句：double x, y, \*px, \*py; 执行了px = &x; py = &y; 之后, 正确的输入语句是()。

〖A〗scanf("%lf %le", px, py);

〖B〗scanf("%f %f" & x, &y);

〖C〗scanf("%f %f", x, y);

〖D〗scanf("%lf %lf", x, y);

因为x, y都是double型数据, 所以输入时的格式字符应为%lf, 所以B)与C)错误。  
D)选项中scanf("%lf %lf", x, y); 应为scanf("%lf %lf", &x, &y); 。

【26】以下定义数组的语句中错误的是()。

〖A〗int num[][3] = { { 1, 2 }, 3, 4, 5, 6 };

〖B〗int num[2][4] = { { 1, 2 }, { 3, 4 }, { 5, 6 } };

〖C〗int num[] = { 1, 2, 3, 4, 5, 6 };

〖D〗int num[][4] = { 1, 2, 3, 4, 5, 6 };

B)选项中定义的数组为2行4列, 而赋值时赋成了3行所以出错。

【27】有以下程序：

复制代码

C

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

程序的运行结果是()。

〖A〗1, 2, 3, 4, 5, 6, 7, 8, 9, 0,

〖B〗2, 1, 4, 3, 6, 5, 8, 7, 0, 9,

〖C〗0, 9, 8, 7, 6, 5, 4, 3, 2, 1,

〖D〗0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

函数调用中发生的数据传送是单向的。  
即只能把实参的值传送给形参, 而不能把形参的值反向地传送给实参。  
因此在函数调用过程中, 形参的值发生改变, 而实参中的值不会变化, 所以数组c中的元素的值并没有变化, 选择A)。

【28】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int x[3][2] = { 0 }, i; |
|  | for (i = 0; i < 3; i++) |
|  | scanf("%d", x[i]); |
|  | printf("%3d%3d%3d\n", x[0][0], x[0][1], x[1][0]); |
|  | } |

若运行时输入：2 4 6<回车>, 则输出结果为()。

〖A〗2 0 4

〖B〗2 0 0

〖C〗2 4 0

〖D〗2 4 6

本题中输入的3个数据2, 4, 6分别赋值给了x[0][0], x[1][0], x[2][0]。  
x[0][1]仍为初始时的0, 所以打印输出时的结果为A)选项。

【29】有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int j; |
|  | float y; |
|  | char name[50]; |
|  | scanf("%2d%3f%s", &j, &y, name); |

当执行上述程序段, 从键盘上输入55566 7777abc后, y的值为()。

〖A〗566.0

〖B〗55566.0

〖C〗7777.0

〖D〗566777.0

它是格式输入函数, 即按用户指定的格式从键盘上把数据输入到指定的变量之中。  
其中的格式命令可以说明最大域宽。  
在百分号(%)与格式码之间的整数用于限制从对应域读入的最大字符数。  
所以j的值为55, y的值为566.0, 字符数组name为7777abc。

【30】下列语句组中, 正确的是()。

〖A〗char \*s; s = "Olympic";

〖B〗char s[7]; s = "Olympic";

〖C〗char \*s; s = { "Olympic" };

〖D〗char s[7]; s = { "Olympic" };

字符型指针变量可以用选项A)的赋值方法：char \* s; s = "Olympic", 选项C)的写法：char \* s, s = { "Olympic" }; 是错误的。  
字符数组可以在定义的时候初始化：char s[] = { "Olympic" }; ?或者char s[] = "Olympic", 都是正确的。  
但是不可以在定义字符数组后, 对数组名赋值。  
(数组名是常量, 代表数组首地址)所以选项B)和选项D)都是错误的。  
对于本例, 选项B)、D)中字符数组s的大小至少为8, 才能存放下字符串。  
(字符串的末尾都有结束标志"\0")。

【31】有以下函数：

复制代码

C

|  |  |
| --- | --- |
|  | int fun(char \*s) { |
|  | char \*t = s; |
|  | while (\*t++); |
|  | t--; |
|  | return (t - s); |
|  | } |

该函数的功能是()

〖A〗计算s所指字符串的长度

〖B〗比较两个字符串的大小

〖C〗计算s所指字符串占用内存字节的个数

〖D〗将s所指字符串复制到字符串t中

首先 char \* s 接受一个字符型数组的首地址并将这个首地址赋给另一个字符型指针 char \* t, while (\*t++) 不断循环直到\*t 为'\0', 再将 t - 1, 这时字符指针 t 指向字符串的最后一个字符, 又因为 s 指向字符数组的首地址即字符串的首地址所以 return (t - s)便是返回字符数组中字符串的长度。

【32】有以下程序(注：字符a的ASCII码值为97):

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char \*s = { "abc" }; |
|  | do { |
|  | printf("%d", \*s % 10); |
|  | ++s; |
|  | } while (\*s); |
|  | } |

程序运行后的输出结果是()。

〖A〗789

〖B〗abc

〖C〗7890

〖D〗979899

因为小写字符a, b, c的ASCII的值分别为97, 98, 99, 而在do while循环语句中, 每次对字符的ASCII的值取余数并输出, 所以分别输出7, 8, 9。

【33】设有如下函数定义：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun(int k) { |
|  | if (k < 1) |
|  | return 0; |
|  | else if (k == 1) |
|  | return 1; |
|  | else |
|  | return fun(k - 1) + 1; |
|  | } |

若执行调用语句：n = fun(3);, 则函数fun总共被调用的次数是()。

〖A〗2

〖B〗3

〖C〗4

〖D〗5

首先 n = fun(3), 3被当作参数传递进去, 这就进行了一次调用, 3被当做参数传进去后, 程序会执行这句 else return fun(k - 1) + 1; 这就调用了第二次, 而参数是3 - 1也就是2。  
2被当做参数传进去后, 程序会执行这句 else return fun(k - 1) + 1; 这就调用了第三次, 而参数是2 - 1也就是1。  
1被当做参数传进去后, 程序会执行这句 else if (k == 1) return 1; 不再递归调用, 所以最终结果为3次。

【34】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int n); |
|  | void main() { |
|  | int a = 3, s; |
|  | s = f(a); |
|  | s = s + f(a); |
|  | printf("%d\n", s); |
|  | } |
|  | int f(int n) { |
|  | static int a = 1; |
|  | n += a++; |
|  | return n; |
|  | } |

程序运行后的输出结果是()。

〖A〗9

〖B〗8

〖C〗7

〖D〗10

题目中静态局部变量a, 在静态存储区内分配存储单元, 在程序整个运行期间都不释放。  
所以第一次调用函数执行n += a++; 时a先与n相加在再进行自增。  
n的值为4, a的值为2, 且a变量执行完后空间没有释放。  
再执行 s = s + f(a)时, s的值为4, 调用f(a)函数时n的返回值为 n = 3 + 2 = 5, 且此时a的值为3了。  
所以s的值为9。

【35】设有定义：

复制代码

C

|  |  |
| --- | --- |
|  | struct complex { |
|  | int real, unreal; |
|  | } data1 = { 1, 8 }, data2; |

则以下赋值语句中错误的是()。

〖A〗data2 = (2, 6);

〖B〗data2 = data1;

〖C〗data2.real = data1.real;

〖D〗data2.real = data1.unreal;

A)选项中可以在声明变量的同时为data2赋值, 但是 data2 = (2, 6); 应写作data2 = { 2, 6 } 。  
所以选择A)。

【36】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | struct S { |
|  | int n; |
|  | int a[20]; |
|  | }; |
|  | void f(struct S \*p) { |
|  | int i, j, t; |
|  | for (i = 0; i < p->n - 1; i++) |
|  | for (j = i + 1; j < p->n; j++) |
|  | if (p->a[i] > p->a[j]) { |
|  | t = p->a[i]; |
|  | p->a[i] = p->a[j]; |
|  | p->a[j] = t; |
|  | } |
|  | } |
|  | void main() { |
|  | int i; |
|  | struct S s = { 10, { 2, 3, 1, 6, 8, 7, 5, 4, 10, 9 } }; |
|  | f(&s); |
|  | for (i = 0; i < s.n; i++) |
|  | printf("%d,", s.a[i]); |
|  | } |

程序运行后的输出结果是()。

〖A〗2,3,1,6,8,7,5,4,10,9,

〖B〗10,9,8,7,6,5,4,3,2,1,

〖C〗1,2,3,4,5,6,7,8,9,10,

〖D〗10,9,8,7,6,1,2,3,4,5,

本题的子函数f的功能是对结构体变量s中第二个成员数组中所有的数据进行从小到大的冒泡排序, 所以结果是C)。

【37】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | typedef struct { |
|  | char name[9]; |
|  | char sex; |
|  | int score[2]; |
|  | } STU; |
|  | STU f(STU a) { |
|  | STU b = { "Zhao", 'm', 85, 90 }; |
|  | int i; |
|  | strcpy(a.name, b.name); |
|  | a.sex = b.sex; |
|  | for (i = 0; i < 2; i++) |
|  | a.score[i] = b.score[i]; |
|  | return a; |
|  | } |
|  | void main() { |
|  | STU c = { "Qian", 'f', 95, 92 }, d; |
|  | d = f(c); |
|  | printf("%s,%c,%d,%d,", d.name, d.sex, d.score[0], d.score[1]); |
|  | printf("%s,%c,%d,%d\n", c.name, c.sex, c.score[0], c.score[1]); |
|  | } |

程序运行后的输出结果是()。

〖A〗Zhao,m,85,90, Qian,f,95,92

〖B〗Zhao,m,85,90, Zhao,m,85,90

〖C〗Qian,f,95,92,Qian,f,95,92

〖D〗Qian,f,95,92,Zhao,m,85,90

本题考查的是函数调用时的参数传递问题。  
程序在调用函数f时, 传给函数f的参数只是结构变量c在栈中的一个拷贝, 函数f所做所有操作只是针对这个数据拷贝进行的修改, 这些都不会影响变量c的值。

【38】以下关于宏的叙述中正确的是()。

〖A〗宏替换没有数据类型限制

〖B〗宏定义必须位于源程序中所有语句之前

〖C〗宏名必须用大写字母表示

〖D〗宏调用比函数调用耗费时间

宏定义写在函数的花括号外边, 作用域为其后的程序, 通常在文件的最开头, 所以B)选项中宏定义必须位于源程序中所有语句之前是错误的。  
宏名一般用大写, 但不是必须用大写, 所以C)选项错误。  
宏展开不占运行时间, 只占编译时间, 函数调用占运行时间(分配内存、保留现场、值传递、返回值), 所以D)选项错误。

【39】设有以下语句：

复制代码

C

|  |  |
| --- | --- |
|  | int a = 1, b = 2, c; |
|  | c = a ^ (b << 2); |

执行后, c的值为()。

〖A〗7

〖B〗9

〖C〗8

〖D〗6

b为2, 二进制为00000010, 执行左移两位操作后为00001000, 然后与a 00000001做异或操作结果为00001001, 即十进制的9。

【40】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*fp; |
|  | int a[10] = { 1, 2, 3 }, i, n; |
|  | fp = fopen("d1.dat", "w"); |
|  | for (i = 0; i < 3; i++) |
|  | fprintf(fp, "%d", a[i]); |
|  | fprintf(fp, "\n"); |
|  | fclose(fp); |
|  | fp = fopen("d1.dat", "r"); |
|  | fscanf(fp, "%d", &n); |
|  | fclose(fp); |
|  | printf("%d\n", n); |
|  | } |

程序的运行结果是()。

〖A〗321

〖B〗12300

〖C〗1

〖D〗123

程序首先将数组a[10]中的元素1、2、3分别写入了文件d1.dat文件中, 然后又将d1.dat文件中的数据123, 整体写入到了变量n的空间中, 所以打印n时输出的数据为123。

**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x所指数组中N个数的平均值(规定所有数均为正数), 平均值通过形参返回给主函数, 将小于平均值且最接近平均值的数作为函数值返回, 并在主函数中输出。  
例如, 有10个正数：46、30、32、40、6、17、45、15、48、26, 平均值为30.500000。  
主函数中输出m = 30。  
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) \* av |
|  | (2) i |
|  | (3) x[j] |

填空1:从原程序中可以看出\*av代表的是平均值, 而s / N表示的就是平均值, 因此本空应该填\*av。  
填空2 : if语句来判断找最接近平均值的数, 因而此空应该填i。  
填空3:题目要求将小于平均值且最接近平均值的数作为函数返回, 而j表达的是最接近平均值的数在数组中的下标, 因而本空应该填写x[j]。

【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：根据整型形参n, 计算如下公式的值。



例如, 若n = 10, 则应输出0.617977。  
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 !

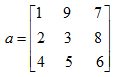
复制代码

C

|  |  |
| --- | --- |
|  | (1) float fun(int n) |
|  | (2) for (i = 2; i <= n; i++) |
|  |  |

(1) 根据函数的返回值可知, 函数应定义为float型。  
(2) 该题中函数fun的作用是计算数列前n项的和, 而数列的组成方式是：第n项的分母是1加第n - 1项的值, 分子为1, 如果循环累加按for(i = 2; i < n; i++)执行, 当输入n = 2时循环不会执行, 程序将得不到想要的结果, 因此循环变量的取值范围应包括2。

【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, 并在主函数中自动赋值。请编写函数fun(int a[][N], int m), 该函数的功能是使数组右上半三角元素中的值乘以m。  
例如, 若m的值为2, a数组中的值为：



则返回主程序后a数组中的值应为：



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的花括号中填入你编写的若干语句。

复制代码

C

|  |  |
| --- | --- |
|  | void fun(int a[][N], int m) { |
|  | int i, j; |
|  | for (j = 0; j < N; j++) |
|  | for (i = 0; i <= j; i++) |
|  | a[i][j] = a[i][j] \* m; |
|  | /\*右上半三角元素中的值乘以m\*/ |
|  | } |

本程序实现将矩阵中右上半三角元素中的值乘以m, 使用循环语句遍历数组元素, 第1个循环用于控制行坐标, 第2个循环用于控制列下标。