[**【计算机二级C语言】卷004**](https://www.cnblogs.com/mjios/p/12446757.html)

**选择题**

**公共知识**

【1】程序流程图中带有箭头的线段表示的是()。

〖A〗图元关系

〖B〗数据流

〖C〗控制流

〖D〗调用关系

在数据流图中, 用标有名字的箭头表示数据流。  
在程序流程图中, 用标有名字的箭头表示控制流。  
所以选择C)。

【2】结构化程序设计的基本原则不包括()。

〖A〗多态性

〖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)。

【3】软件设计中模块划分应遵循的准则是()。

〖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

软件设计中模块划分应遵循的准则是高内聚低偶合、模块大小规模适当、模块的依赖关系适当等。  
模块的划分应遵循一定的要求, 以保证模块划分合理, 并进一步保证以此为依据开发出的软件系统可靠性强, 易于理解和维护。  
模块之间的耦合应尽可能的低, 模块的内聚度应尽可能的高。

【4】在软件开发中, 需求分析阶段产生的主要文档是()。

〖A〗可行性分析报告

〖B〗软件需求规格说明书

〖C〗概要设计说明书

〖D〗集成测试计划

A)错误, 可行性分析阶段产生可行性分析报告。  
C)错误, 概要设计说明书是总体设计阶段产生的文档。  
D)错误, 集成测试计划是在概要设计阶段编写的文档。  
B)正确, 需求规格说明书是后续工作如设计、编码等需要的重要参考文档。

【5】算法的有穷性是指()。

〖A〗算法程序的运行时间是有限的

〖B〗算法程序所处理的数据量是有限的

〖C〗算法程序的长度是有限的

〖D〗算法只能被有限的用户使用

算法原则上能够精确地运行, 而且人们用笔和纸做有限次运算后即可完成。  
有穷性是指算法程序的运行时间是有限的。

【6】对长度为n的线性表排序, 在最坏情况下, 比较次数不是n(n - 1) / 2的排序方法是()。

〖A〗快速排序

〖B〗冒泡排序

〖C〗直接插入排序

〖D〗堆排序

除了堆排序算法的比较次数是



, 其他的都是n(n - 1) / 2。

【7】下列关于栈的叙述正确的是()。

〖A〗栈按"先进先出"组织数据

〖B〗栈按"先进后出"组织数据

〖C〗只能在栈底插入数据

〖D〗不能删除数据

栈是按"先进后出"的原则组织数据的, 数据的插入和删除都在栈顶进行操作。

【8】在数据库设计中, 将E - R图转换成关系数据模型的过程属于()。

〖A〗需求分析阶段

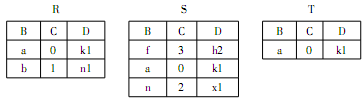
〖B〗概念设计阶段

〖C〗逻辑设计阶段

〖D〗物理设计阶段

E - R图转换成关系模型数据则是把图形分析出来的联系反映到数据库中, 即设计出表, 所以属于逻辑设计阶段。

【9】有三个关系R、S和T如下：  
由关系R和S通过运算得到关系T, 则所使用的运算为()。



〖A〗并

〖B〗自然连接

〖C〗笛卡尔积

〖D〗交

自然连接是一种特殊的等值连接, 它要求两个关系中进行比较的分量必须是相同的属性组, 并且在结果中把重复的属性列去掉, 所以B)错误。  
笛卡尔积是用R集合中元素为第一元素, S集合中元素为第二元素构成的有序对, 所以C)错误。  
根据关系T可以很明显的看出是从关系R与关系S中取得相同的关系组所以取得是交运算, 选择D)。

【10】设有表示学生选课的三张表, 学生S(学号, 姓名, 性别, 年龄, 身份证号), 课程C(课号, 课名), 选课SC(学号, 课号, 成绩), 则表SC的关键字(键或码)为()。

〖A〗课号, 成绩

〖B〗学号, 成绩

〖C〗学号, 课号

〖D〗学号, 姓名, 成绩

学号是学生表S的主键, 课号是课程表C的主键, 所以选课表SC的关键字就应该是与前两个表能够直接联系且能唯一定义的学号和课号, 所以选择C)。

**专业知识**

【11】以下叙述中错误的是()。

〖A〗C语言中的每条可执行语句和非执行语句最终都将被转换成二进制的机器指令

〖B〗C程序经过编译、连接步骤之后才能形成一个真正可执行的二进制机器指令文件

〖C〗用C语言编写的程序称为源程序, 它以ASCII代码形式存放在一个文本文件中

〖D〗C语言源程序经编译后生成后缀为.obj的目标程序

C语言中的非执行语句不会被编译, 不会生成二进制的机器指令, 所以A)错误。  
由C语言构成的指令序列称C源程序, C源程序经过C语言编译程序编译之后生成一个后缀为.OBJ的二进制文件(称为目标文件); 最后要由"连接程序"把此.OBJ文件与C语言提供的各种库函数连接起来生成一个后缀为.EXE的可执行文件。

【12】以下选项中, 合法的一组C语言数值常量是()。

〖A〗12.0Xa23 4.5e0

〖B〗028.5e - 3 - 0xf

〖C〗.177 4e1.5 0abc

〖D〗0x8A 10, 000 3.e5

C语言中八进制整型常量的开头是数字0, 十六进制整型常量的开头数字是0x。  
C语言中的实型常量有两种表示形式：小数形式, 小数形式表示的实型常量必须要有小数点; 指数形式, 以"e"或"E"后跟一个整数来表示以10为底数的幂数, 且规定字母e或E之前必须要有数字, 且e或E后面的指数必须为整数。  
B)选项中 028错误, 八进制中没有数字8。  
C)选项中4e1.5中e后面的指数不是整数。  
D)选项中10, 000 数字中不能有逗号。

【13】以下选项中不合法的标识符是()。

〖A〗&a

〖B〗FOR

〖C〗print

〖D〗\_00

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。  
因为C语言中区分大小写, 所以B)选项中的"FOR"可以作为标识符来用。  
A)选项中含有非法字符&, 所以选择A)。

【14】若有代数式子



(其中e仅代表自然对数的底数, 不是变量), 则以下能够正确表示该代数式的C语言表达式是()。

〖A〗sqrt(fabs(pow(n, x) + exp(x)))

〖B〗sqrt(fabs(pow(n, x) + pow(x, e)))

〖C〗sqrt(abs(n ^ x + e ^ x))

〖D〗sqrt(fabs(pow(x, n) + exp(x)))

B)选项中函数pow(x, e)错误应该直接使用exp(x)函数。  
C)选项中函数abs(n ^ x + e ^ x)错误, 应该使用fabs()返回浮点数的绝对值。  
D)选项中pow(x, n)参数顺序错误。

【15】若有定义：double a = 22; int i = 0, k = 18;, 则不符合C语言规定的赋值语句是()

〖A〗i = (a + k) <= (i + k);

〖B〗i = a % 11;

〖C〗a = a++, i++

〖D〗i = !a;

取模运算符"%", 二元运算符, 具有左结合性, 参与运算的量均为整型。  
选项 B中的a变量是double实型, 所以B)不符合规定。

【16】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int s, t, A = 10; |
|  | double B = 6; |
|  | s = sizeof(A); |
|  | t = sizeof(B); |
|  | printf("%d,%d\n", s, t); |
|  | } |

在VC平台上编译运行, 程序运行后的输出结果是()。

〖A〗10,6

〖B〗4,4

〖C〗2,4

〖D〗4,8

C语言中利用sizeof()函数判断数据类型长度, 在VC平台中, 整型int占有4个字节, double型数据占有8个字节。

【17】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char a, b, c, d; |
|  | scanf("%c%c", &a, &b); |
|  | c = getchar(); |
|  | d = getchar(); |
|  | printf("%c%c%c%c\n", a, b, c, d); |
|  | } |

当执行程序时, 按下列方式输入数据(从第1列开始, 代表回车, 注意：回车也是一个字符)  
12  
34  
则输出结果是()。

〖A〗 12 34

〖B〗 12

〖C〗 1234

〖D〗 12 3

按照从键盘输入的数据可以判断字符1给了变量a, 字符2给了变量b, 字符即回车给了变量c, 字符3给了变量d。  
所以打印输出的结果为D)选项。

【18】下关于逻辑运算符两侧运算对象的叙述中正确的是()。

〖A〗可以是任意合法的表达式

〖B〗只能是整数0或非0整数

〖C〗可以是结构体类型的数据

〖D〗只能是整数0或1

C语言的逻辑运算符比较特别, 它的操作数没有明确的数据类型, 可以是任意合法的表达式, 所以选择A)。

【19】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 0, b = 0, c = 0, d = 0; |
|  | if (a = 1) |
|  | b = 1; |
|  | c = 2; |
|  | else |
|  | d = 3; |
|  | printf("%d,%d,%d,%d\n", a, b, c, d); |
|  | } |

程序输出()。

〖A〗1, 1, 2, 0

〖B〗0, 0, 0, 3

〖C〗编译有错

〖D〗0, 1, 2, 0

本题中if(a = 1)b = 1; 与else d = 3; 之间多了语句c = 2; 所以会出现else语句的位置错误的编译失败提示。

【20】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int x = 1, y = 0, a = 0, b = 0; |
|  | switch (x) { |
|  | case 1: |
|  | switch (y) { |
|  | case 0: |
|  | a++; |
|  | break; |
|  | case 1: |
|  | b++; |
|  | break; |
|  | } |
|  | case 2: |
|  | a++; |
|  | b++; |
|  | break; |
|  | case 3: |
|  | a++; |
|  | b++; |
|  | } |
|  | printf("a=%d,b=%d\n", a, b); |
|  | } |

程序的运行结果是()。

〖A〗a = 2, b = 2

〖B〗a = 2, b = 1

〖C〗a = 1, b = 1

〖D〗a = 1, b = 0

case常量表达式只是起语句标号作用, 并不是该处进行条件判断。  
在执行switch语句时, 根据switch的表达式, 找到与之匹配的case语句, 就从此case子句执行下去, 不在进行判断, 直到碰到break或函数结束为止。  
所以执行内层switch(y)时只执行了a++, 此时a的值为1。  
然后执行外层case 2语句的a++; b++; a为2, b为1。  
所以结果为B)。

【21】以下程序段中的变量已正确定义：

复制代码

C

|  |  |
| --- | --- |
|  | for (i = 0; i < 4; i++, i++) |
|  | for (k = 1; k < 3; k++); |
|  | printf("\*"); |

程序段的输出结果是()。

〖A〗\*\*

〖B〗\*\*\*\*

〖C〗\*

〖D〗\*\*\*\*\*\*\*\*

由于内层循环for(k = 1; k < 3; k++)后面直接跟了空语句";"所以在循环内部什么操作也不做, 跳出外层循环后执行打印语句, 所以打印了一个"\*", 选择C)。

【22】有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i, n; |
|  | for (i = 0; i < 8; i++) { |
|  | n = rand() % 5; |
|  | switch (n) { |
|  | case 1: |
|  | case 3: |
|  | printf("%d \n", n); |
|  | break; |
|  | case 2: |
|  | case 4: |
|  | printf("%d \n", n); |
|  | continue; |
|  | case 0: |
|  | exit(0); |
|  | } |
|  | printf("%d \n", n); |
|  | } |
|  | } |

以下关于程序段执行情况的叙述, 正确的是()。

〖A〗for循环语句固定执行8次

〖B〗当产生的随机数n为4时结束循环操作

〖C〗当产生的随机数n为1和2时不做任何操作

〖D〗当产生的随机数n为0时结束程序运行

case常量表达式只是起语句标号作用, 并不是该处进行条件判断。  
在执行switch语句时, 根据switch的表达式, 找到与之匹配的case语句, 就从此case子句执行下去, 不再进行判断, 直到碰到break或函数结束为止。  
简单的说break是结束整个循环体, 而continue是结束单次循环。  
B)选项中当产生的随机数n为4时要执行打印操作。  
C)选项中当产生的随机数为1和2时分别执行case3与case4后面语句的内容。  
for循环不是固定执行8次, 执行次数与产生的随机数n有关系。

【23】有以下程序：

复制代码

C

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

程序运行后的输出结果是()。

〖A〗8

〖B〗2

〖C〗4

〖D〗1

第一次调用m = f(f(f(1))), 第二次为m = f(f(2)), 第三次为m = f(4), 即返回值为8。

【24】以下叙述中错误的是()。

〖A〗可以给指针变量赋一个整数作为地址值

〖B〗函数可以返回地址值

〖C〗改变函数形参的值, 不会改变对应实参的值

〖D〗当在程序的开头包含头文件stdio.h时, 可以给指针变量赋NULL

不能将一个整数直接赋给指针变量作为地址, 所以A)是错误的。  
函数的返回值可以是地址, 即指针。  
函数调用中形参值的变化不会传递给实参。

【25】设已有定义：float x;, 则以下对指针变量p进行定义且赋初值的语句中正确的是()。

〖A〗int \*p = (float) x;

〖B〗float \*p = &x;

〖C〗float p = &x;

〖D〗float \*p = 1024;

指针是用来存放地址的变量, 用(类型名\*指针变量名)的形式定义。  
赋值时应将某个变量地址即&x 赋给指针变量, 所以选择B)。

【26】以下数组定义中错误的是()。

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

〖B〗int x[][3] = { 0 };

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

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

D)选项中x[2][3]定义的是一个两行三列的二维数组, 而在给数组元素赋值时却赋成了三行, 所以错误。

【27】有以下程序：

复制代码

C

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

程序的运行结果是()。

〖A〗321678

〖B〗876543

〖C〗1098765

〖D〗345678

本题中的函数fun()的功能是将数组k中前5个元素倒序, 所以返回后数组k中的元素排列是5, 4, 3, 2, 1, 6, 7, 8, 9, 10。  
所以打印输出k[2]到k[7]元素的值, 即321678, 所以选择A。

【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] - a[i][N - 1 - i]; |
|  | } |
|  | void main() { |
|  | int x[N][N] = { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 }, { 13, 14, 15, 16 } }, y[N], i; |
|  | fun(x, y); |
|  | for (i = 0; i < N; i++) |
|  | printf("%d,", y[i]); |
|  | printf("\n"); |
|  | } |

程序运行后的输出结果是()。

〖A〗-3,-1,1,3,

〖B〗-12,-3,0,0,

〖C〗0,1,2,3,

〖D〗-3,-3,-3,-3,

本题中由fun函数可知, b[0] = a[0][0] - a[0][3] = 1 - 4 = -3, b[1] = a[1][1] - [1][2] = 6 - 7 = -1, b[2] = a[2][2] - [2][1] = 11 - 10 = 1, b[3] = a[3][3] - [3][0] = 16 - 13 = 3。  
所以主函数中打印y数组元素的值为A)选项。

【29】设有定义：char s[81]; int i = 0;, 以下不能将一行(不超过80个字符)带有空格的字符串正确读入的语句或语句组是()。

〖A〗gets(s);

〖B〗while ((s[i++] = getchar()) != '\n'); s[i] = '\0';

〖C〗scanf("%s", s);

〖D〗do { scanf("%c", &s[i]); } while (s[i++] != '\n'); s[i] = '\0';

用%s格式符输入字符串时, 空格和回车符都作为数据的分隔符而不能被读入, 这里应该使用gets(s)。

【30】设有定义：char p[] = { '1', '2', '3' }, \*q = p;, 以下不能计算出一个char型数据所占字节数的表达式是()。

〖A〗sizeof(p)

〖B〗sizeof(char)

〖C〗sizeof(\*q)

〖D〗sizeof(p[0])

根据题目中的定义可以知道sizeof(p), 计算的是数组p中所有元素所占用的字节数, 而不是char型数据所占字节数。

【31】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | void main() { |
|  | char str[][20] = { "One\*World", "One\*Dream!" }, \*p = str[1]; |
|  | printf("%d,", strlen(p)); |
|  | printf("%s\n", p); |
|  | } |

程序运行后的输出结果是()。

〖A〗10,One\*Dream!

〖B〗9,One\*Dream!

〖C〗9,One\*World

〖D〗10,One\*World

p是指向二维字符数组第二行One \* Dream !的数组指针, 所以长度是10, 打印输出的也是该字符串。

【32】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char s[] = "012xy\08s34f4w2"; |
|  | int i, n = 0; |
|  | for (i = 0; s[i] != 0; i++) |
|  | if (s[i] >= '0' && s[i] <= '9') |
|  | n++; |
|  | printf("%d\n", n); |
|  | } |

程序运行后的输出结果是()。

〖A〗0

〖B〗3

〖C〗7

〖D〗8

本题中遇到"\"字符循环结束,所以只统计"\"之前的数字字符, 注意数字0与字符串结束标记的区别, 所以为3。

【33】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | fun(int x, int y) { |
|  | static int m = 0, i = 2; |
|  | i += m + 1; |
|  | m = i + x + y; |
|  | return m; |
|  | } |
|  | void main() { |
|  | int j = 1, m = 1, k; |
|  | k = fun(j, m); |
|  | printf("%d,", k); |
|  | k = fun(j, m); |
|  | printf("%d\n", k); |
|  | } |

执行后的输出结果是()。

〖A〗5,11

〖B〗5,5

〖C〗11,11

〖D〗11,5

声明静态局部变量：函数调用结束后, 其占用的存储单元不释放, 在下次该函数调用时, 该变量保留上一次函数调用结束时的值。  
本题子函数fun中的变量 i和m均为静态局部变量。  
所以第一次调用fun函数, 返回m的值为5, 第二次再调用fun函数时, i的值为3, m的值已经是5了, 所以执行i += m + 1, i的值变为9, m = i + x + y = 9 + 1 + 1 = 11。

【34】在C语言中, 只有在使用时才占用内存单元的变量, 其存储类型是()。

〖A〗auto和static

〖B〗extern和register

〖C〗auto和register

〖D〗static和register

auto:函数中的局部变量, 动态地分配存储空间, 数据存储在动态存储区中, 在调用该函数时系统会给它们分配存储空间, 在函数调用结束时就自动释放这些存储空间。

register:为了提高效率, C语言允许将局部变量的值放在CPU中的寄存器中, 这种变量叫"寄存器变量", 只有局部自动变量和形式参数可以作为寄存器变量。

extern:外部变量(即全局变量)是在函数的外部定义的, 它的作用域为从变量定义处开始, 到本程序文件的末尾。  
如果外部变量不在文件的开头定义, 其有效的作用范围只限于定义处到文件终了。

static:静态局部变量属于静态存储类别, 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。

【35】下面结构体的定义语句中, 错误的是()。

〖A〗struct ord { int x; int y; int z; } struct ord a;

〖B〗struct ord { int x; int y; int z; }; struct ord a;

〖C〗struct ord { int x; int y; int z; } a;

〖D〗struct { int x; int y; int z; } a;

A)选项struct ord { int x; int y; int z } struct ord a; 错误, 不能在定义结构体的同时, 又用结构体类型名定义变量。  
应该写成B)选项或者D)选项的格式。

【36】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | typedef struct { |
|  | int b, p; |
|  | } A; |
|  | void f(A c) /\*注意:c是结构变量名\*/ { |
|  | int j; |
|  | c.b += 1; |
|  | c.p += 2; |
|  | } |
|  | void main() { |
|  | int i; |
|  | A a = { 1, 2 }; |
|  | f(a); |
|  | printf("%d,%d\n", a.b, a.p); |
|  | } |

程序运行后的输出结果是()。

〖A〗2,4

〖B〗1,2

〖C〗1,4

〖D〗2,3

结构体变量可以作为函数的参数和返回值。  
作为函数的实参时, 可以实现函数的传值调用。  
当使用结构体变量作为函数的形参时, 实参也应该是结构体变量名以实现传值调用, 实参将拷贝副本给形参, 在被调用函数中改变形参值对于调用函数中的实参没有影响。  
所以选择B)。

【37】以下叙述中正确的是()。

〖A〗在C语言中, 预处理命令行都以"#"开头

〖B〗预处理命令行必须位于C源程序的起始位置

〖C〗#include < stdio.h>必须放在C程序的开头

〖D〗C语言的预处理不能实现宏定义和条件编译的功能

预处理命令是以"#"号开头的命令, 它们不是C语言的可执行命令, 这些命令应该在函数之外书写, 一般在源文件的最前面书写, 但不是必须在起始位置书写, 所以B), C)错误。  
C)语言的预处理能够实现宏定义和条件编译等功能, 所以D)错误。

【38】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | int fun(int n) { |
|  | int \*p; |
|  | p = (int \*) malloc(sizeof(int)); |
|  | \*p = n; |
|  | return \*p; |
|  | } |
|  | void main() { |
|  | int a; |
|  | a = fun(10); |
|  | printf("%d\n", a + fun(10)); |
|  | } |

程序的运行结果是()。

〖A〗0

〖B〗10

〖C〗20

〖D〗出错

malloc(sizeof(int))的作用是开辟一个长度为sizeof(int)存储空间, 并通过强制类型转换(int \*)将此存储空间的地址赋给了一个整型的指针变量p。  
然后执行语句\*p = n, 使得\*p的值为10, 并通过return 返回此值, 即a的值为10。  
然后在主函数中输出a + fun(10) = 10 + 10 = 20。

【39】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | unsigned char a = 8, c; |
|  | c = a >> 3; |
|  | printf("%d\n", c); |
|  | } |

程序运行后的输出结果是()。

〖A〗16

〖B〗1

〖C〗32

〖D〗0

无符号整型变量a的值为8, 二进制表示为00001000, 右移3位后为00000001, 即十进制的1, 所以输出1。

【40】读取二进制文件的函数调用形式为：fread(buffer, size, count, fp);, 其中buffer代表的是()。

〖A〗一个内存块的字节数

〖B〗一个整型变量, 代表待读取的数据的字节数

〖C〗一个文件指针, 指向待读取的文件

〖D〗一个内存块的首地址, 代表读入数据存放的地址

fread(void \*buffer, size\_t size, size\_t count, FILE \*stream); 功能是从一个文件流中读数据, 读取count个元素, 每个元素size字节, 如果调用成功返回count。

buffer:用于接收数据的内存地址, 大小至少是 size \* count 字节;  
size:单个元素的大小, 单位是字节;  
count:元素的个数, 每个元素是size字节;  
stream:输入流。

**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。程序通过定义学生结构体变量, 存储了学生的学号、姓名和三门课的成绩。所有学生数据均以二进制方式输出到文件中。函数fun的功能是重写形参filename所指文件中最后一个学生的数据, 即用新的学生数据覆盖该学生原来的数据, 其他学生的数据不变。  
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) filename |
|  | (2) fp |
|  | (3) fp |

填空1:本空是对文本文件的打开, filename所指的文件中的最后数据要进行重写, 因此首先是要打开, 因而本空填写filename。  
填空2 : fseek 功能是重定位流上的文件指针。  
用法：int fseek(FILE \* stream, long offset, int fromwhere); 本空应该填写fp。  
填空3:因为题目中要对所有学生数据均以二进制方式输出到文件中, 因此本空填写fp。  
知识点讲解：  
函数名：fseek  
功 能：重定位流上的文件指针。  
用 法：int fseek(FILE \* stream, long offset, int fromwhere)。  
描 述：函数设置文件指针stream的位置。  
如果执行成功, stream将指向以fromwhere为基准, 偏移offset个字节的位置。  
如果执行失败(比如offset超过文件自身大小), 则不改变stream指向的位置。  
返回值：成功, 返回0; 否则返回非0值。

【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Creatlink的功能是：创建带头结点的单向链表, 并为各结点数据域赋0到m - 1的值。  
请改正函数Creatlink中指定部位的错误, 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) p = (NODE \*) malloc(sizeof(NODE)); |
|  | (2) return h; |

本题考查：变量赋初值; for循环语句; 函数返回值。  
(1) 由变量定义可知p为指向结点的指针。  
指向刚分配的结构指针, 所以应改为p = (NODE \*) malloc(sizeof(NODE))。  
(2) 在动态分配内存的下一行语句是, 使用临时结构指针变量h保存p指针的初始位置, 最后返回不能使用p, 是因为p的位置已经发生了变化, 所以应改为return h。

【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：统计一行字符串中单词的个数, 作为函数值返回。字符串在主函数中输入, 规定所有单词由小写字母组成, 单词之间有若干个空格隔开, 一行的开始没有空格。  
注意：部分源程序在文件prog1.c中。  
请勿改动main函数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

|  |  |
| --- | --- |
|  | int fun(char \*s) { |
|  | int i, j = 0; |
|  | for (i = 0; s[i] != '\0'; i++) |
|  | if (s[i] != ' ' && (s[i + 1] == ' ' || s[i + 1] == '\0')) |
|  | /\*如果一个字母的下一个字符为空格或者结束标记, 则表示一个单词结束\*/ |
|  | j++; |
|  | return j; |
|  | /\*返回单词个数\*/ |
|  | } |

要判断单词的个数, 首先想到的是程序怎样识别出一个单词, 如果一个字母的下一个字符为空格或者结束标记, 则表示一个单词结束, 因此程序使用for循环语句遍历整个字符串, 用if条件语句判断当前字符是否表示一个单词的结束, 如果当前字符表示一个单词结束, 则存放单词个数的变量加1, 最后返回单词的个数。