[**【计算机二级C语言】卷015**](https://www.cnblogs.com/mjios/p/12452929.html)

**选择题**

**公共知识**

【1】下列叙述中正确的是

〖A〗算法的效率只与问题的规模有关, 而与数据的存储结构无关

〖B〗算法的时间复杂度是指执行算法所需要的计算工作量

〖C〗数据的逻辑结构与存储结构是一一对应的

〖D〗算法的时间复杂度与空间复杂度一定相关

算法的效率与问题的规模和数据的存储结构都有关, A错误。  
算法的时间复杂度, 是指执行算法所需要的计算工作量, B正确。  
由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同, 因此数据的逻辑结构和存储结构不是一一对应的, C错误。  
算法的时间复杂度和空间复杂度没有直接的联系, D错误。

【2】下列叙述中正确的是

〖A〗线性表链式存储结构的存储空间一般要少于顺序存储结构

〖B〗线性表链式存储结构与顺序存储结构的存储空间都是连续的

〖C〗线性表链式存储结构的存储空间可以是连续的, 也可以是不连续的

〖D〗以上说法均错误

线性表的顺序存储结构具备如下两个基本特征：(1)线性表中的所有元素所占的存储空间是连续的; (2)线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。  
用一组任意的存储单元来依次存放线性表的结点, 这组存储单元既可以是连续的, 也可以是不连续的, 甚至是零散分布在内存中的任意位置上的。  
因此C正确。

【3】某二叉树共有12个结点, 其中叶子结点只有1个。则该二叉树的深度为(根结点在第1层)

〖A〗3

〖B〗6

〖C〗8

〖D〗12

二叉树中, 度为0的节点数等于度为2的节点数加1, 即n2 = n0 - 1, 叶子节点即度为0, n0 = 1, 则n2 = 0, 总节点数为12 = n0 + n1 + n2 = 1 + n1 + 0, 则度为1的节点数n1 = 11, 故深度为12, 选D。

【4】对长度为n的线性表作快速排序, 在最坏情况下, 比较次数为

〖A〗n

〖B〗n - 1

〖C〗n(n - 1)

〖D〗n(n - 1) / 2

快速排序最坏情况就是每次选的基准数都和其他数做过比较, 共需比较(n - 1) + (n - 2)+…+1 = n(n - 1) / 2, 选D。

【5】结构化程序设计中, 下面对goto语句使用描述正确的是

〖A〗禁止使用goto语句

〖B〗使用goto语句程序效率高

〖C〗应避免滥用goto语句

〖D〗以上说法均错误

结构化程序设计中, 要注意尽量避免goto语句的使用, 故选C。

【6】下面不属于软件测试实施步骤的是

〖A〗集成测试

〖B〗回归测试

〖C〗确认测试

〖D〗单元测试

软件测试主要包括单元测试、集成测试、确认测试和系统测试。

【7】下面不属于软件需求分析阶段主要工作的是

〖A〗需求变更申请

〖B〗需求分析

〖C〗需求评审

〖D〗需求获取

需求分析阶段的工作可概括为４个方面：①需求获取。  
②需求分析。  
③编写需求规格说明书。  
④需求审评。

【8】一般情况下, 当对关系R和S进行自然连接时, 要求R和S含有一个或者多个共有的

〖A〗记录

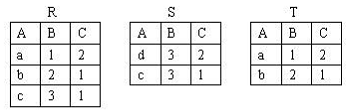
〖B〗行

〖C〗属性

〖D〗元组

自然连接是一种特殊的等值连接, 它满足下面的条件：①两关系间有公共域; ②通过公共域的等值进行连接, 选C。

【9】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〗选择

〖B〗差

〖C〗交

〖D〗并

关系T是关系R的一部分, 并且是关系R去掉R和S相同的元素, 符合差操作。

【10】有表示公司和职员及工作的三张表, 职员可在多家公司兼职。其中公司C(公司号, 公司名, 地址, 注册资本, 法人代表, 员工数), 职员S(职员号, 姓名, 性别, 年龄, 学历), 工作W(公司号, 职员号, 工资), 则表W的键(码)为

〖A〗公司号, 职员号

〖B〗职员号, 工资

〖C〗职员号

〖D〗公司号, 职员号, 工资

由于职员可以再多加公司兼职, 表W的键(码)应为公司关系和职员关系的主码, 即公司号和职员号。

**专业知识**

【11】关于算法, 以下叙述中错误的是

〖A〗某个算法可能会没有输入

〖B〗同一个算法对于相同的输入必须得出相同的结果

〖C〗一个算法对于某个输入的循环次数是可以事先估计出来的

〖D〗任何算法都能转换成计算机高级语言的程序, 并在有限时间内运行完毕

算法的特征：①有穷性：一个算法(对任何合法的输入)在执行有穷步后能够结束, 并且在有限的时间内完成。

②确定性：算法中的每一步都有确切的含义。

③可行性：算法中的操作能够用已经实现的基本运算执行有限次来实现。

④输入：一个算法有零个或者多个输入, 零个输入就是算法本身确定了初始条件。

⑤输出：一个算法有一个或者多个输出, 以反映出数据加工的结果。

所以选择D选项。

【12】关于C语言的变量, 以下叙述中错误的是

〖A〗变量所占的存储单元地址可以随时改变

〖B〗所谓变量是指在程序运行过程中其值可以被改变的量

〖C〗程序中用到的所有变量都必须先定义后才能使用

〖D〗由三条下划线构成的符号名是合法的变量名

一个变量实质上是代表了内存中的某个存储单元, 必须先定义后使用, 且在定义时为之分配存储单元, 不能随时改变, 所以选择A选项。

【13】关于：do 循环体 while (条件表达式);  
以下叙述中正确的是

〖A〗条件表达式的执行次数总是比循环体的执行次数多一次

〖B〗循环体的执行次数总是比条件表达式的执行次数多一次

〖C〗条件表达式的执行次数与循环体的执行次数一样

〖D〗条件表达式的执行次数与循环体的执行次数无关

do…while循环的执行过程是先执行一遍循环体后再执行条件表达式判断条件, 直到条件不成立跳出循环, 所以循环体和条件表达式的执行次数是一致的, 所以选择C。

【14】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = -11, b = 10; |
|  | a /= b /= -4; |
|  | printf("%d %d\n", a, b); |
|  | } |

则程序的输出结果是

〖A〗4 -3

〖B〗-1 -2

〖C〗5 -2

〖D〗5 -3

先执行b /= -4, 即b = b / -4, 结果为-2, 然后执行a /= -2, 即a = a / -2, 结果为5。  
所以选择C。

【15】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 0, b = 0, c = 0; |
|  | c = (a += ++b, b += 4); |
|  | printf("%d,%d,%d\n", a, b, c); |
|  | } |

则程序的输出结果是

〖A〗1,5,1

〖B〗1,5,5

〖C〗-1,4,4

〖D〗-1,4,-1

逗号运算符的结合性从左到右, 因此逗号表达式将从左到右进行计算, 且逗号运算符的优先级最低。  
先计算a += ++b, 结果a为1, b为1, 然后计算b += 4, b的值为5; 逗号表达式的值为其中最后一个表达式的值, 所以将5赋给变量c, 即a为1, b为5, c为5。  
所以选择B。

【16】若有定义

复制代码

C

|  |  |
| --- | --- |
|  | int a; |
|  | float b; |
|  | double c; |

程序运行时输入：  
3 4 5<回车>  
能把值3输入给变量a、4输入给变量b、5输入给变量c的语句是

〖A〗scanf("%d%f%lf", &a, &b, &c);

〖B〗scanf("%d%lf%lf", &a, &b, &c);

〖C〗scanf("%d%f%f", &a, &b, &c);

〖D〗scanf("%lf%lf%lf", &a, &b, &c);

scanf()函数的格式串中, 必须含有与输入项一一对应的格式转换说明符, 整型数据、float型数据、double型数据对应的格式转换说明符分别为%d、%f、%lf。  
所以选择A。

【17】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 1, b = 2, c = 3, d = 4; |
|  | if ((a = 2) && (b = 1))c = 2; if((c == 3) || (d = -1)) |
|  | a = 5; |
|  | printf("%d,%d,%d,%d\n", a, b, c, d); |
|  | } |

则程序的输出结果是

〖A〗2,2,2,4

〖B〗2,1,2,-1

〖C〗5,1,2,-1

〖D〗1,2,3,4

第一个if语句的判断条件中是逻辑与表达式, 只有运算对象均非零结果才为真, 执行第一个if判断条件, 均为赋值表达式, 第一个运算对象a = 2非零, 判断第二个表达式b = 1也非零, 所以条件成立执行c = 2的赋值操作。  
执行第二个if判断条件中的逻辑或表达式, 只要有一个对象非零结果就是真, 本条件中c == 3运算对象为假, 所以需要判断第二个运算对象d = -1, 结果为-1非零, 所以条件成立, 执行a = 5, 的操作, 所以a值为5, b值为1, c值为2, d值为-1。

【18】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 1, b = 2, c = 3, d = 4, r = 0; |
|  | if (a != 1); |
|  | else |
|  | r = 1; |
|  | if (b == 2) |
|  | r += 2; |
|  | if (c != 3); |
|  | r += 3; |
|  | if (d == 4) |
|  | r += 4; |
|  | printf("%d\n", r); |
|  | } |

则程序的输出结果是

〖A〗3

〖B〗7

〖C〗6

〖D〗10

第一个if语句a != 1条件为假, 所以执行else后的语句r = 1。  
第二个if语句b == 2条件成立, 执行r += 2, r的值变为3, 第三个if语句c != 3条件为假, 所以不做任何操作。  
执行下面的r += 3操作, r的值变为6。  
判断第四个if条件, d == 4条件成立, 执行r += 4操作, 结果为10。

【19】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗10

〖B〗11

〖C〗13

〖D〗15

break语句的作用是终止正在执行的switch流程, 跳出switch结构或者强制终止当前循环, 从当前执行的循环中跳出。  
题干中第一次循环n值为0, 执行default语句后的s += 4, s的值变为4, 执行case1语句后的s += 1, s的值变为5, 遇到break语句跳出switch语句, 进入第二次循环。  
第二次循环时n的值为1, 执行case1后的s += 1, s的值变为6, 遇到break语句跳出switch语句, 进入第三次循环。  
第三次循环时n的值为2, 执行case2后的s += 2, s的值变为8, 遇到break语句跳出switch语句, 进入第四次循环。  
第四次循环时n的值为3, 执行case3后的s += 3, s的值变为11。  
再判断循环条件为假, 退出循环打印s的值11。

【20】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗1,3

〖B〗0,2

〖C〗1,2

〖D〗2,3

本题重点考查do - while语句及do - while 的循环语句保证会执行一次(表达式的真值在每次循环结束后检查), ++b即变量b自增1后再参与运算, 而a++则表示a参与运算后, 再自增1。  
所以第一次循环结束后, b = 1, a = -1。  
第二次循环后, b = 2, a = 0。  
第三次循环后, b = 3, a = 1。  
因此A选项正确。

【21】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 6, b = 0, c = 0; |
|  | for (; a && (b == 0); ) { |
|  | b += a; |
|  | a -= c++; |
|  | } |
|  | printf("%d,%d,%d\n", a, b, c); |
|  | } |

则程序的输出结果是

〖A〗5,6,0

〖B〗6,0,0

〖C〗6,6,1

〖D〗5,6,1

本题重点考查C语言中for语句和逻辑表达式, for语句的一般形式为：for (表达式1; 表达式2; 表达式3) 语句

其循环中的"表达式1(循环变量赋初值)"、"表达式2(循环条件)"和"表达式3(循环变量增量)"都是选择项, 即可以缺省, 但";"不能缺省。  
该题目中省略了"表达式1(循环变量赋初值)"和"表达式3(循环变量增量)"。

for (; a && (b == 0); ) {  
b += a;  
a -= c++;  
}  
相当于

while (a && (b == 0)) {  
b += a;  
a -= c++;  
}  
。

因此C选项正确。

【22】以下选项中非法的C语言字符常量是

〖A〗'\x9d'

〖B〗'9'

〖C〗'\x09'

〖D〗'\09'

本题考查字符常量, 字符常量是用单引号括起来的一个字符。  
例如：'a'、'='、'+'等。  
转义字符是一种特殊的字符常量。  
转义字符以反斜线"\"开头, 后跟一个或几个字符。  
D选项中'\09', \0表示后面的字符是八进制数, 又出现9所以是非法的, 故选D选项。

【23】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | char f(char x) { |
|  | return x \* x % 10 + '0'; |
|  | } |
|  | void main() { |
|  | char a; |
|  | int b = 0; |
|  | for (a = 0; a < 5; a += 1) { |
|  | b = f(a); |
|  | putchar(b); |
|  | } |
|  | } |

则程序的输出结果是

〖A〗014916

〖B〗01234

〖C〗01496

〖D〗00000

该题目中x \* x % 10即x和x相乘后, 结果再对10求余操作。  
主函数中通过for语句使变量a从0到4进行取值; 当a = 0时, 调用函数f后, 返回值为0赋给b, 并且进行字符输出; 当a = 1时, 返回值为1赋给b, 并且进行字符输出; 当a = 2时, 返回值为4赋给b, 并且进行字符输出; 当a = 3时, 返回值为9赋给b, 并且进行字符输出; 当a = 4时, 返回值为6赋给b, 并且进行字符输出。  
因此C选项正确。

【24】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗6,12,12

〖B〗6,4,3

〖C〗6,4,6

〖D〗9,4,9

首先在主函数中给整型变量k赋值为3, 调用sp函数, 将变量k的地址作为实参传递给形参指针变量a; 在sp函数中, \*a = \*a \* 2; 即结果为6, 通过printf()函数输出。  
整型变量b的地址赋给指针变量a, 再执行\*a = \*a \* 2;, 即结果为4, 通过printf()函数输出。  
返回主函数, 输出k的值6。  
因此C选项正确。

【25】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int k = 7, m = 5; |
|  | void f(int \*\*s) { |
|  | int \*t = &k; |
|  | s = &t; |
|  | \*s = &m; |
|  | printf("%d,%d,%d,", k, \*t, \*\*s); |
|  | } |
|  | void main() { |
|  | int i = 3, \*p = &i, \*\*r = &p; |
|  | f(r); |
|  | printf("%d,%d,%d\n", i, \*p, \*\*r); |
|  | } |

则程序的输出结果是

〖A〗7,5,7,3,5,7,

〖B〗7,5,7,3,3,3,

〖C〗7,5,5,3,3,3,

〖D〗7,7,5,3,3,3,

在主函数中首先初始化整型变量i, 将i的地址赋给指针变量p, 再将指针变量p的地址赋给指向指针的指针变量r; 调用f函数, 将变量r的值传递给变量s; 由于k和m分别赋值7和5, printf("%d,%d,%d,", k, \*t, \*\*s);, 即结果为7, 5, 5; 返回主函数, printf("%d,%d,%d\n", i, \*p, \*\*r);, 即结果为3, 3, 3。  
因此C选项正确。

【26】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int \*f(int \*s, int \*t) { |
|  | int k; |
|  | if (\*s < \*t) { |
|  | k = \*s; |
|  | \*s = \*t; |
|  | \*t = k; |
|  | } |
|  | return s; |
|  | } |
|  | void main() { |
|  | int i = 3, j = 5, \*p = &i, \*q = &j, \*r; |
|  | r = f(p, q); |
|  | printf("%d,%d,%d,%d,%d\n", i, j, \*p, \*q, \*r); |
|  | } |

则程序的输出结果是

〖A〗5,3,3,5,5

〖B〗3,5,5,3,5

〖C〗3,5,3,5,5

〖D〗5,3,5,3,5

在主函数中分别给整型变量i、j和指针型变量p、q赋初值, 并声明指针变量r; 调用f函数, 并将实参变量p和q的值传递给形参变量s和t, 并且f函数是指针型函数, 即函数的返回值将是一个地址。  
在f函数中, 如果条件成立, 则将指针变量s和t的值互换, 并且将指针s的地址返回主函数。  
最后输出i, j, \*p, \*q, \*r的值, 即5, 3, 5, 3, 5。  
因此D选项正确。

【27】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #define N 4 |
|  | void fun(int a[][N], int b[], int flag) { |
|  | int i, j; |
|  | for (i = 0; i < N; i++) { |
|  | b[i] = a[0][i]; |
|  | for (j = 1; j < N; j++) |
|  | if (flag ? (b[i] > a[j][i]) : (b[i] < a[j][i])) |
|  | b[i] = a[j][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, 1); |
|  | for (i = 0; i < N; i++) |
|  | printf("%d,", y[i]); |
|  | fun(x, y, 0); |
|  | for (i = 0; i < N; i++) |
|  | printf("%d,", y[i]); |
|  | printf("\n"); |
|  | } |

则程序的输出结果是

〖A〗4,8,12,16,1,5,9,13,

〖B〗1,2,3,4,13,14,15,16,

〖C〗1,5,9,13,4,8,12,16,

〖D〗13,14,15,16,1,2,3,4,

该题首先初始化二维数组, if (flag ? (b[i] > a[i][j]) : (b[i] < a[i][j])) 条件语句的条件表达式使用了条件运算符构成的选择结构, 即flag为真时, 以(b[i] > a[i][j])作为条件表达式的值, 否则以(b[i] < a[i][j])作为条件表达式的值, fun函数功能是给一维数组赋值。  
fun(x, y, 1); 该函数调用后, 即当flag为真时, 使一维数组获得二维数组第1行的数值; fun(x, y, 0); 该函数调用后, 即当flag为假时, 使一维数组获得二维数组第4行的数值; 因此B选项正确。

【28】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗1,10,3,8,5,6,7,4,9,2,

〖B〗10,9,8,7,6,5,4,3,2,1,

〖C〗10,1,8,3,6,5,4,7,2,9,

〖D〗2,9,4,7,6,5,8,3,10,1,

该题首先对一维数组进行初始化, c[0]到c[9]依次赋值为10到1; fun(c, 10); 语句调用fun函数, fun函数的功能是将一维数组奇数位元素的数值由小到大排序; 最后将排好序的新的一维数组进行输出。  
因此D选项正确。

【29】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i, j = 0; |
|  | char a[] = "How are you!", b[10]; |
|  | for (i = 0; a[i]; i++) |
|  | if (a[i] == ' ') |
|  | b[j++] = a[i + 1]; |
|  | b[j] = '\0'; |
|  | printf("%s\n", b); |
|  | } |

则程序的输出结果是

〖A〗Howareyou!

〖B〗Howareyou

〖C〗Hay!

〖D〗ay

程序首先初始化字符数组a[]为"How are you!"。  
语句if(a[i] == ' ')b[j++] = a[i + 1]; 条件是判断字符数组a[]中是否是空格字符, 如果是空格的话, 将空格后的字符赋给数组b。  
并将b输出。  
因此D选项正确。

【30】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | char \*a = "you", b[] = "welcome#you#to#China!"; |
|  | void main() { |
|  | int i, j = 0; |
|  | char \*p; |
|  | for (i = 0; b[i] != '\0'; i++) { |
|  | if (\*a == b[i]) { |
|  | p = &b[i]; |
|  | for (j = 0; a[j] != '\0'; j++) { |
|  | if (a[j] != \*p) |
|  | break; |
|  | p++; |
|  | } |
|  | if (a[j] == '\0') |
|  | break; |
|  | } |
|  | } |
|  | printf("%s\n", p); |
|  | } |

则程序的输出结果是

〖A〗#to#China!

〖B〗#you#to#China!

〖C〗me#you#to#China!

〖D〗#China!

该程序首先定义\*a和b[], 并进行初始化。  
主函数中通过外层for循环语句, 遍历字符数组b[], 并且将符合if条件语句的字符赋给数组p; for内层循环语句, 遍历字符数组a[]。  
再将符合if条件语句的结果输出。  
因此A选项正确。

【31】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char a[20], b[] = "The sky is blue."; |
|  | int i; |
|  | for (i = 0; i < 10; i++) |
|  | scanf("%c", &a[i]); |
|  | a[i] = '\0'; |
|  | gets(b); |
|  | printf("%s%s\n", a, b); |
|  | } |

执行时若输入：  
Fig flower is red.<回车>  
则输出结果是

〖A〗Fig flower is red.is blue.

〖B〗Fig flower is red.

〖C〗Fig floweris red.

〖D〗Fig floweris

本题考查字符数组的赋值以及字符串的处理函数, 本题中输入字符串Fig flower is red时, 首先是将前10个字符赋值给字符数组a, 然后执行gets(b)后, 将后续字符存放到b中, 当输出时, 首先输出字符数组a, 然后输出字符串b, 所以结果为B选项?

【32】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char w[20] = "dogs", a[5][10] = { "abcdef", "ghijkl", "mnopq", "rstuv", "wxyz ." }; |
|  | int i, j, k; |
|  | for (i = 0; w[i]; i++) { |
|  | for (j = 0; j < 5; j++) { |
|  | for (k = 0; a[j][k]; k++) |
|  | if (w[i] == a[j][k]) |
|  | break; |
|  | if (w[i] == a[j][k]) |
|  | break; |
|  | } |
|  | printf("%d,%d,", j, k); |
|  | } |
|  | } |

则程序的输出结果是

〖A〗0,3,2,2,1,0,3,1,

〖B〗1,4,3,3,2,1,4,2,

〖C〗5,6,5,6,5,6,5,6,

〖D〗6,7,6,7,6,7,6,7,

本题考查二维字符数组的运用, 题目中首先定义了一个一维字符数组, 从三层for循环可以看出要找的是w数组中每个字符在二维数组中的位置, 比如d字母, 在a数组中就是0, 3 其他字符类似, 所以本题答案为A。

【33】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int a[], int n) { |
|  | if (n > 1) { |
|  | int t; |
|  | t = f(a, n - 1); |
|  | return t > a[n - 1] ? t : a[n - 1]; |
|  | } else |
|  | return a[0]; |
|  | } |
|  | void main() { |
|  | int a[] = { 8, 2, 9, 1, 3, 6, 4, 7, 5 }; |
|  | printf("%d\n", f(a, 9)); |
|  | } |

则程序的输出结果是

〖A〗9

〖B〗1

〖C〗8

〖D〗5

本题主要考查了函数的递归调用, 解题思路只需要将参数带入函数中, 首先是执行f(a, 9), 递归执行(a, 8)直到(a, 2), 得到a[0] = 8, 然后判断a[0]是否大于a[1], 本题实际上就是求出数组a元素的最大值。  
答案为9, 选项A正确。

【34】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int m) { |
|  | static int n = 0; |
|  | int a = 2; |
|  | n++; |
|  | a++; |
|  | return n + m + a; |
|  | } |
|  | void main() { |
|  | int k; |
|  | for (k = 0; k < 4; k++) |
|  | printf("%d,", f(k)); |
|  | } |

则程序的输出结果是

〖A〗3,5,7,9,

〖B〗4,7,10,13,

〖C〗4,5,6,7,

〖D〗4,6,8,10,

本题考查静态局部变量以及自增运算符, 静态局部变量在函数调用结束后, 并不释放, 仍然保留它的值, 本题中, k为0时, return n + m + a之前, n为1, a为局部变量是3, m为0, 所以第一个返回值为4。  
k为1, 在函数返回值之前, 静态局部变量为2, 局部变量a仍然为3, m为1, 所以结果是6, 依此类推答案为D。

【35】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #define S(x) (x) \* (x) |
|  | #define T(x) S(x) / S(x) + 1 |
|  | void main() { |
|  | int k = 3, j = 2; |
|  | printf("%d,%d\n", S(k + j), T(k + j)); |
|  | } |

则程序的输出结果是

〖A〗11,2

〖B〗25,2

〖C〗11,12

〖D〗25,26

本题考查宏定义, 宏定义只是做个简单的替换, 执行S(k + j) = (k + j) \* (k + j) = 25, T(k + j) = S(k + j) / S(k + j) + 1 = (k + j) \* (k + j) / (k + j) \* (k + j) + 1 = 26, 选项D正确。

【36】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | typedef struct stu { |
|  | char name[10], gender; |
|  | int score; |
|  | } STU; |
|  | void f(STU a, STUB) { |
|  | b = a; |
|  | printf("%s,%c,%d,", b.name, b.gender, b.score); |
|  | } |
|  | void main() { |
|  | STU a = { "Zhao", 'm', 290 }, b = { "Qian", 'f', 350 }; |
|  | f(a, b); |
|  | printf("%s,%c,%d\n", b.name, b.gender, b.score); |
|  | } |

则程序的输出结果是

〖A〗Qian,f,350,Qian,f,350

〖B〗Zhao,m,290, Zhao,m,290

〖C〗Zhao,m,290,Qian,f,350

〖D〗Zhao,m,290,Zhao,f,350

本题考查结构体的相关操作以及传值、传址的区别, 该题中调用f函数后, 会首先输出被重新复制的b值, 且与a相同, 而执行完f函数后, b值并不会改变, 所以还是维持原值, 选项C正确。

【37】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | typedef struct stu { |
|  | char name[10], gender; |
|  | int score; |
|  | } STU; |
|  | void f(char \*p) { |
|  | strcpy(p, "Qian"); |
|  | } |
|  | void main() { |
|  | STU a = { "Zhao", 'm', 290 }, b; |
|  | b = a; |
|  | f(b.name); |
|  | b.gender = 'f'; |
|  | b.score = 350; |
|  | printf("%s,%c,%d,", a.name, a.gender, a.score); |
|  | printf("%s,%c,%d\n", b.name, b.gender, b.score); |
|  | } |

则程序的输出结果是

〖A〗Zhao,m,290,Qian,f,350

〖B〗Zhao,m,290, Zhao,m,290

〖C〗Qian,f,350,Qian,f,350

〖D〗Zhao,m,290,Zhao,f,350

本题考查结构体的相关操作以及传值、传址的区别, 该题中调用f函数后, b的name会重新赋值为Qian, 选项A正确。

【38】若有定义：typedef char T[10];

复制代码

C

T \*a;

上述定义中a的类型与下面选项中完全相同的是

〖A〗char a[10];

〖B〗char(\*a)[10];

〖C〗char \*a;

〖D〗char \*a[10];

本题考查typedef的使用方法, typedef对已存在的类型使用一个新的名字, 选项D正确。

【39】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int c; |
|  | c = 13 | 5; |
|  | printf("%d\n", c); |
|  | } |

则程序的输出结果是

〖A〗13

〖B〗15

〖C〗18

〖D〗5

本题考查位运算中按位或运算符, 或运算只要两个比较的位中有一个为1, 其结果是1, 否则结果为0, , 13用二进制表示为00001101, 5用二进制表示为00000101, 或后00001101, 即13, 选项A正确。

【40】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*fp; |
|  | int i, a[6] = { 1, 2, 3, 4, 5, 6 }, k; |
|  | fp = fopen("data.dat", "w+b"); |
|  | for (i = 0; i < 6; i++) { |
|  | fseek(fp, 0L, 0); |
|  | fwrite(&a[5 - i], sizeof(int), 1, fp); |
|  | } |
|  | rewind(fp); |
|  | fread(&k, sizeof(int), 1, fp); |
|  | fclose(fp); |
|  | printf("%d", k); |
|  | } |

则程序的输出结果是

〖A〗6

〖B〗1

〖C〗123456

〖D〗21

本题考查文件操作函数, fseek 用于二进制方式打开的文件, 移动文件读写指针位置。  
将文件内部的位置指针重新指向一个流(数据流/文件)的开头。  
所以最后只保留了1, 答案为B。

**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的功能是将带头结点的单向链表结点数据域中的数据从小到大排序。即若原链表结点数据域从头至尾的数据为：10、4、2、8、6, 排序后链表结点数据域从头至尾的数据为：2、4、6、8、10。  
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) h->next |
|  | (2) p->next |
|  | (3) > |

填空1:本题考查了指针变量p赋初值, 使p指向链表的头结点的下一个结点, 因此本空应该填写h->next。  
填空2:本空考查了q的作用, q代表p的下一个数, 因此次空应该填写p->next。  
填空3 : if判断句是比较链表中相邻两个数的大小, 如果p->data > q->data就交换这两个数的值, 因而此空填写>。

【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, 建立一个带头结点的单向链表, 并用随机函数为各结点数据域赋值。函数fun的作用是求出单向链表结点(不包括头结点)数据域中的最大值, 并且作为函数值返回。  
请改正函数fun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 !

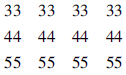
复制代码

C

|  |  |
| --- | --- |
|  | (1) p = h->next; |
|  | (2) p = p->next; |

(1) 因为链表的头结点没有数据域, 所以对指针p进行初始化时, 要将p指向头结点的下一个结点。  
(2) 通过指针p的移动遍历链表, 因此此处应改为p = p->next; 。

【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：将M行N列的二维数组中的数据, 按行的顺序依次放到一维数组中, 一维数组中数据的个数存放在形参n所指的存储单元中。  
例如, 若二维数组中的数据为：



则一维数组中的内容应该是：33 33 33 33 44 44 44 44 55 55 55 55。  
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(\*s)[10], int \*b, int \*n, int mm, int nn) { |
|  | int i, j, k = 0; |
|  | for (i = 0; i < mm; i++) |
|  | /\*将二维数组s中的数据按行的顺序依次放到一维数组b中\*/ |
|  | for (j = 0; j < nn; j++) |
|  | b[k++] = s[i][j]; |
|  | \*n = k; |
|  | /\*通过指针返回元素个数\*/ |
|  | } |

本题可以用两个循环来处理, 由于是按行的顺序取出, 所以第1个循环用于控制行下标, 第2个循环用于控制列下标; 若改成按列的顺序取出, 则循环应改成：  
for (i = 0; i < nn; i++)  
for (j = 0; j < mm; j++)  
b[k++] = s[j][i];  
要注意s[j][i]的下标, 不能用s[i][j]。  
若按行取出则列标变化较快, 若按列取出则行标变化较快, 即循环嵌套时, 越在内层的循环, 其循环变量变化就越快。