[**【计算机二级C语言】卷014**](https://www.cnblogs.com/mjios/p/12452928.html)

**选择题**

**公共知识**

【1】下列叙述中正确的是

〖A〗一个算法的空间复杂度大, 则其时间复杂度也必定大

〖B〗一个算法的空间复杂度大, 则其时间复杂度必定小

〖C〗一个算法的时间复杂度大, 则其空间复杂度必定小

〖D〗算法的时间复杂度与空间复杂度没有直接关系

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, 算法的时间复杂度, 是指执行算法所需要的计算工作量, 两者之间并没有直接关系, 答案为D。

【2】下列叙述中正确的是

〖A〗循环队列中的元素个数随队头指针与队尾指针的变化而动态变化

〖B〗循环队列中的元素个数随队头指针的变化而动态变化

〖C〗循环队列中的元素个数随队尾指针的变化而动态变化

〖D〗以上说法都不对

在循环队列中, 用队尾指针rear指向队列中的队尾元素, 用排头指针front指向排头元素的前一个位置。  
因此, 从排头指针front指向的后一个位置直到队尾指针rear指向的位置之间所有的元素均为队列中的元素。  
所以循环队列中的元素个数与队头指针和队尾指针的变化而变化, A正确。

【3】一棵二叉树中共有80个叶子结点与70个度为1的结点, 则该二叉树中的总结点数为

〖A〗219

〖B〗229

〖C〗230

〖D〗231

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, 则n2 = 79, 总结点数为n0 + n1 + n2 = 80 + 70 + 79 = 229, 答案为B。

【4】对长度为10的线性表进行冒泡排序, 最坏情况下需要比较的次数为

〖A〗9

〖B〗10

〖C〗45

〖D〗90

冒泡法是在扫描过程中逐次比较相邻两个元素的大小, 最坏的情况是每次比较都要将相邻的两个元素互换, 需要互换的次数为9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 45, 选C。

【5】构成计算机软件的是

〖A〗源代码

〖B〗程序和数据

〖C〗程序和文档

〖D〗程序、数据及相关文档

软件指的是计算机系统中与硬件相互依赖的另一部分, 包括程序、数据和有关的文档, 选D。

【6】软件生命周期可分为定义阶段、开发阶段和维护阶段, 下面不属于开发阶段任务的是

〖A〗测试

〖B〗设计

〖C〗可行性研究

〖D〗实现

开发阶段包括分析、设计和实施两类任务。  
其中分析、设计包括需求分析、总体设计和详细设计3个阶段, 实施则包括编码和测试两个阶段, C不属于开发阶段。

【7】下面不能作为结构化方法软件需求分析工具的是

〖A〗系统结构图

〖B〗数据字典(DD)

〖C〗数据流程图(DFD图)

〖D〗判定表

结构化方法软件需求分析工具主要有数据流图、数据字典、判定树和判定表。

【8】在关系模型中, 每一个二维表称为一个

〖A〗关系

〖B〗属性

〖C〗元组

〖D〗主码(键)

关系模型采用二维表来表示, 即每个二维表称为一个关系。

【9】若实体A和B是一对多的联系, 实体B和C是一对一的联系, 则实体A和C的联系是

〖A〗一对一

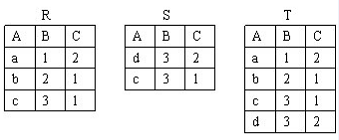
〖B〗一对多

〖C〗多对一

〖D〗多对多

A和B为一对多的联系, 则对于A中的每一个实体, B中有多个实体与之联系, 而B与C为一对一联系, 则对于B中的每一个实体, C中之多有一个实体与之联系, 则可推出对于A中的每一个实体, C中有多个实体与联系, 所以为一对多联系。

【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〗选择

〖B〗投影

〖C〗交

〖D〗并

关系T中的元素与关系R和关系S中不同元素的总和, 因此为并操作。

**专业知识**

【11】我们所写的每条C语句, 经过编译最终都将转换成二进制的机器指令。关于转换以下说法错误的是

〖A〗一条C语句可能会被转换成零条机器指令

〖B〗一条C语句可能会被转换成多条机器指令

〖C〗一条C语句对应转换成一条机器指令

〖D〗某种类型和格式的C语句被转换成机器指令的条数是固定的

一个C语句经过编译后产生若干条机器指令。  
C语言的注释语句不会转换成二进制的机器指令。  
所以选择C选项。

【12】关于"while(条件表达式)循环体", 以下叙述正确的是

〖A〗循环体的执行次数总是比条件表达式的执行次数多一次

〖B〗条件表达式的执行次数总是比循环体的执行次数多一次

〖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

while循环是先判断条件, 满足条件后执行循环体, 执行完后接着判断条件, 执行, 直到最后一次判断条件后不成立, 跳出循环, 所以最后一次执行了条件表达式, 但是没有执行循环体, 所以选择B选项。

【13】关于C语言的符号常量, 以下叙述中正确的是

〖A〗符号常量的符号名是标识符, 但必须大写

〖B〗符号常量是指在程序中通过宏定义用一个符号名来代表一个常量

〖C〗符号常量在整个程序中其值都不能再被重新定义

〖D〗符号常量的符号名必须是常量

在C语言中, 可以用一个标识符来代表一个常量, 称为符号常量。  
这个标识符必须在程序中进行特别的"指定", 并符合标识符的命名规则。  
用作符号常量的标识符通常采用大写字母表示, 在主函数中其值不能再被定义。  
所以选择B选项。

【14】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗1 2

〖B〗-1 2

〖C〗-1 -2

〖D〗1 -2

先计算b %= 4, 即b = b % 4, 结果为2, 然后计算a %= 2, 结果为-1, 最后a的值为-1, b的值为2, 所以选择B选项。

【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 -= ++a), (a += b, b += 4); |
|  | printf("%d,%d,%d\n", a, b, c); |
|  | } |

则程序的输出结果是

〖A〗0,4,0

〖B〗0,4,4

〖C〗1,4,1

〖D〗1,4,4

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, 所以先计算c = (a -= ++a)这部分。  
其中, 赋值运算符从右往左计算, 先执行++a, a自增1后再赋值, 所以a的值为1, 执行a = a - 1, 即a = 1 - 1, a的值为0, 并将0赋给变量c。  
第二部分(a += b, b += 4), 先执行a = a + b, a = 0 + 0, 即a的值为0, 然后执行b += 4, b = 0 + 4, 即b的值为4。  
所以打印结果为0, 4, 0, 选择A选项。

【16】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a = 0, b = 0, c = 0, d; |
|  | c = (a += b, , b += a); |
|  | /\*第4行\*/ |
|  | d = c; |
|  | ; |
|  | /\*第5行\*/; |
|  | /\*第6行\*/; |
|  | printf("%d,%d,%d\n", a, b, c); |
|  | /\*第7行\*/ |
|  | } |

编译时出现错误, 你认为出错的是

〖A〗第4行

〖B〗第5行

〖C〗第6行

〖D〗第7行

第4行逗号表达式中间的第二个表达式为空, 是不合法的, 可以去掉写成a += b, b += a, 也可以在里面补一个表达式, 如a += b, a, b += a。  
所以选择A选项。

【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〗1,2,3,4

〖B〗2,1,2,-1

〖C〗5,1,2,-1

〖D〗2,2,2,4

第一个if语句的判断条件中是逻辑或表达式, 所以只要有一个运算对象为真结果就为真, 且如果前面的已经为真, 后面的表达式不再进行执行判断。  
执行第一个if判断条件时a的值为2, 整个表达式的值为真, b的值不变, c的值为2。  
判断第二个if条件为逻辑与表达式, 只有两个运算对象均非零, 才为真, 第一个表达式c == 3, 不成立, 不再判断第二个运算对象, 所以不执行d = -1操作和a = 5操作。  
所以结果为选项D。

【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; |
|  | else |
|  | ; |
|  | if (c != 3) |
|  | r += 3; |
|  | else |
|  | ; |
|  | if (d == 4) |
|  | r += 4; |
|  | printf("%d\n", r); |
|  | } |

则程序的输出结果是

〖A〗10

〖B〗7

〖C〗6

〖D〗3

else 总是和与之最近的上面的if配对, 执行第一个if, 判断条件不成立, 执行else r = 1; 语句。  
再执行第二个if语句中的判断条件, 成立, 左移执行r += 2, r的值为3, 再执行第三个if的判断条件, 不成立执行第7行else后面的空语句";"。  
再执行第四个if的判断条件d == 4成立, 执行r += 4, r的值为7, 然后打印输出。

【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; |
|  | case 2: |
|  | s += 2; |
|  | case 3: |
|  | s += 3; |
|  | } |
|  | } |
|  | printf("%d\n", s); |
|  | } |

则程序的输出结果是

〖A〗6

〖B〗18

〖C〗10

〖D〗24

第一次for循环, n的值为0, 所以从default后面的语句开始执行, s += 4, s += 1, s += 2, s += 3, s的值为10。  
在进入第二次for循环, n的值为1, 所以执行s += 1, s += 2, s += 3, s的值为16。  
在进入第三次for循环, n的值为2, 所以执行s += 2, s += 3, s的值为21。  
在进入第四次for循环, n的值为3, 所以执行s += 3, s的值为24。

【20】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗1,2

〖B〗0,2

〖C〗1,3

〖D〗2,3

第一次while判断条件中a++, 先用后加, 为-2不为0条件为真, 执行a的自加与++b操作, a的值变为-1, b的值变为1。  
第二次while判断后, a的值变为0, b的值变为2。  
第三次while判断条件为假, 但是仍要执行自加操作, 即值为1, 跳出循环, 打印1和2。

【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 += a; |
|  | a -= ++c; |
|  | } |
|  | printf("%d,%d,%d\n", a, b, c); |
|  | } |

则程序的输出结果是

〖A〗1,14,3

〖B〗0,14,3

〖C〗0,18,3

〖D〗0,14,6

本循环的条件判断时只要a为0就退出循环体, 初始值a的值为6, 满足条件执行完循环体, b的值为6, a -= ++c即a = a - (++c), 所以a的值为5, c的值为1。  
第二次循环后b的值为11, a的值为3, c的值为2。  
第三次也是最后循环后, a的值为0, b的值14, c的值为3。

【22】以下选项中非法的C语言字符常量是

〖A〗'\007'

〖B〗'\b'

〖C〗'aa'

〖D〗'\xaa'

C选项中是字符串常量, 应该用双引号表示即"aa"?所以选择C选项。

【23】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | char f(char x) { |
|  | return x \* x + 'a'; |
|  | } |
|  | void main() { |
|  | char a, b = 0; |
|  | for (a = 0; a < 4; a += 1) { |
|  | b = f(a); |
|  | putchar(b); |
|  | } |
|  | } |

则程序的输出结果是

〖A〗abcd

〖B〗ABEJ

〖C〗abej

〖D〗ABCD

该题目首先初始化变量a和b, 通过for循环语句使a取值为0、1、2、3, f函数的功能是将a \* a + 'a'的值返回给b, 即b为a、b、e、j。  
最后通过字符输出函数putchar()将b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 = &b; |
|  | \*a = \*a \* 2; |
|  | printf("%d,", \*a); |
|  | } |
|  | void main() { |
|  | int k = 3, \*p = &k; |
|  | sp(p); |
|  | printf("%d,%d\n", k, \*p); |
|  | } |

则程序的输出结果是

〖A〗4,3,3

〖B〗4,3,4

〖C〗6,3,6

〖D〗6,6,6

首先在主函数中给整型变量k赋值为3, 将变量k的地址赋给指针变量p; 调用sp函数, 将实参p的值传给形参a, 在sp函数中, 指针变量a又获得变量b的地址, 通过运算后, 输出a的值为4; 返回主函数, 输出k的值和p的值为3和3。  
因此A选项正确。

【25】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int k = 7; |
|  | void f(int \*\*s) { |
|  | int \*t = &k; |
|  | \*s = t; |
|  | 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〗3,3,3,7,7,7

〖B〗3,7,7,7,7,7

〖C〗7,7,7,3,3,3

〖D〗7,7,7,3,7,7

在主函数中分别给整型变量i, 指针型变量p和指向指针的指针变量r赋初值; 调用f函数, 并将实参变量r的值传递给形参变量s, 在f函数中, 通过printf("%d,%d,%d,", k, \*t, \*\*s); 语句输出7, 7, 7; 返回主函数, 通过printf("%d,%d,%d\n", i, \*p, \*\*r); 输出3, 7, 7。  
因此D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〗3,5,5,3,5

〖B〗3,5,3,5,5

〖C〗5,3,5,3,5

〖D〗5,3,3,5,5

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3, 5, 3, 5, 5。  
因此B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[i][0]; |
|  | for (j = 1; j < N; j++) |
|  | if (flag ? (b[i] > a[i][j]) : (b[i] < a[i][j])) |
|  | b[i] = a[i][j]; |
|  | } |
|  | } |
|  | 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〗1,5,9,13,4,8,12,16,

〖B〗4,8,12,16,1,5,9,13,

〖C〗1,2,3,4,13,14,15,16,

〖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列的数值; 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列的数值。  
因此A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 = 1; 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〗2,9,4,7,6,5,8,3,10,1,

〖D〗10,1,8,3,6,5,4,7,2,9,

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偶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!"; |
|  | for (i = 0; a[i]; i++) |
|  | if (a[i] != ' ') |
|  | a[j++] = a[i]; |
|  | a[j] = '\0'; |
|  | printf("%s\n", a); |
|  | } |

则程序的输出结果是

〖A〗Hay!

〖B〗Howareyou

〖C〗Howareyou!

〖D〗How are you!

主函数中首先定义字符数组a[] = "How are you!"; 执行for循环语句, 语句if(a[i] != ' ')a[j++] = a[i]; 其功能是将字符数组中的空格去掉。  
因此C选项正确。

【30】若有以下程序

复制代码

C

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

则程序的输出结果是

〖A〗China!

〖B〗to China!

〖C〗me you to China!

〖D〗you to 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D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 < 7; i++) |
|  | scanf("%c", &b[i]); |
|  | gets(a); |
|  | printf("%s%s\n", a, b); |
|  | } |

执行时若输入：(其中表示回车符)  
Fig flower is red.  
则输出结果是

〖A〗wer is red.Fig flo is blue.

〖B〗wer is red.Fig flo

〖C〗wer is red.The sky is blue.

〖D〗Fig flower is red.The sky is blue.

程序首先初始化字符数组b[], 执行for循环语句, 循环变量i的取值范围从0到6。  
在for循环语句中通过scanf函数将从键盘上输入的数据输入到b[]中, 即b的值为Fig flo is blue.。  
退出for循环语句, 执行语句gets(a);, gets()函数的调用形式为：gets(str\_adr), 其中str\_adr是存放输入字符串的起始地址, 可以是字符数组名。  
字符数组元素的地址或字符指针变量。  
gets函数用来从终端键盘读入字符串(包括空格符), 直到读入一个换行符为止, 即a的值为wer is red.。  
因此A选项正确。

【32】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char w[20], a[5][10] = { "abcdef", "ghijkl", "mnopq", "rstuv", "wxyz" }; |
|  | int p[6][2] = { { 0, 1 }, { 1, 5 }, { 0, 0 }, { 0, 2 }, { 1, 4 }, { 4, 4 } }, i; |
|  | for (i = 0; i < 6; i++) |
|  | w[i] = a[p[i][0]][p[i][1]]; |
|  | puts(w); |
|  | } |

则程序的输出结果是

〖A〗catis

〖B〗ababj

〖C〗ckbdl

〖D〗black

程序首先初始化字符数组a[5][10]和二维数组p[6][2]。  
执行for循环语句, 循环变量i从0取值到5, 得到字符数组w[]的值, 并通过字符串输出函数 puts进行输出操作。  
w[0] = a[0][1]、w[1] = a[1][5]、w[2] = a[0][0]、w[3] = a[0][2]、w[4] = a[1][4]、w[5] = a[4][4]; 因此最后字符数组w的结果为black。  
因此D选项正确。

【33】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void f(int x) { |
|  | if (x >= 10) { |
|  | printf("%d-", x % 10); |
|  | f(x / 10); |
|  | } else |
|  | printf("%d", x); |
|  | } |
|  | void main() { |
|  | int z = 123456; |
|  | f(z); |
|  | } |

则程序的输出结果是

〖A〗6-5-4-3-2-1-

〖B〗6-5-4-3-2-1

〖C〗1-2-3-4-5-6

〖D〗1-2-3-4-5-6-

本题重点考查函数的递归调用, 程序首先初始化整型变量z为123456, 调用函数f。  
f函数中首先通过if条件语句判断x是否大于等于10, 如果条件成立, 求x除以10的余数并输出, 同时将变量x进行x / 10运算, 同时调用函数f。  
如果条件不成立, 直接输出变量x。  
因此第1次调用函数, 变量x为123456, 条件成立, 输出余数6; 第2次调用函数, 变量x为12345, 条件成立, 输出余数5; 第3次调用函数, 变量x为1234, 条件成立, 输出余数4; 第4次调用函数, 变量x为123, 条件成立, 输出余数3; 第5次调用函数, 变量x为12, 条件成立, 输出余数2; 第6次调用函数, 变量x为12345, 条件不成立, 输出1。  
因此B选项正确。

【34】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int a = 2; |
|  | int f(int m) { |
|  | static int n; |
|  | n = 0; |
|  | 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〗4,5,6,7,

〖B〗4,7,10,13,

〖C〗4,6,8,10,

〖D〗3,5,7,9,

主函数中变量k作为for语句的循环变量, 取值0、1、2、3, 同时也作为实参通过调用函数f传递给形参变量m。  
由于变量a是全局变量, 变量n是static静态变量, 静态变量n在定义时没有赋初始值, 在程序运行时重新赋值为0, 每次程序执行都被赋值为0, 因此k = 0时, f函数中, 语句return n + m + a; 即1 + 0 + 3 = 4; k = 1时, f函数中, 语句return n + m + a; 即1 + 1 + 4 = 6; k = 2时, f函数中, 语句return n + m + a; 即1 + 2 + 5 = 8; k = 3时, f函数中, 语句return n + m + a; 即1 + 3 + 6 = 10。  
因此C选项正确。

【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) |
|  | void main() { |
|  | int k = 5, j = 2; |
|  | printf("%d,%d\n", S(k + j), T(k + j)); |
|  | } |

则程序的输出结果是

〖A〗17,289

〖B〗49,2401

〖C〗17,37

〖D〗49,289

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 = 17, T(k + j) = S(k + j) \* S(k + j) = k + j \* k + j \* k + j \* k + j = 37, 选项C正确。

【36】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | struct stu { |
|  | char \*name, gender; |
|  | int score; |
|  | }; |
|  | void main() { |
|  | struct stu a = { NULL, 'm', 290 }, b; |
|  | a.name = (char \*) malloc(10); |
|  | strcpy(a.name, "Zhao"); |
|  | b = a; |
|  | b.gender = 'f'; |
|  | b.score = 350; |
|  | strcpy(b.name, "Qian"); |
|  | 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,Zhao,f,350

〖B〗Zhao,m,290,Qian,f,350

〖C〗Qian,f,350,Qian,f,350

〖D〗Qian,m,290,Qian,f,350

本题中考查结构体中变量的引用, a结构体中name被修改为Qian, 所以本题答案为D。

【37】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | typedef struct stu { |
|  | char \*name, gender; |
|  | int score; |
|  | } STU; |
|  | void f(char \*p) { |
|  | p = (char \*) malloc(10); |
|  | strcpy(p, "Qian"); |
|  | } |
|  | void main() { |
|  | STU a = { NULL, 'm', 290 }, b; |
|  | a.name = (char \*) malloc(10); |
|  | strcpy(a.name, "Zhao"); |
|  | 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,f,350

〖C〗Qian,f,350,Qian,f,350

〖D〗Qian,m,290,Qian,f,350

本题考查结构体变量中的引用, 其中a的name为Zhao, b执行函数f(b.name)后b的name并没有改变, 还是Zhao, 所以答案为B选项。

【38】若有定义：typedef int \* T; T a[10];  
则a的定义与下面哪个语句等价

〖A〗int(\*a)[10];

〖B〗int \*a[10];

〖C〗int \*a;

〖D〗int 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, 其中本题中int\*用T来代替, 所以定义T a[10]; 就相当于是int \* a[10]; 选项B正确。

【39】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int c; |
|  | c = 10 ^ 5; |
|  | printf("%d\n", c); |
|  | } |

则程序的输出结果是

〖A〗5

〖B〗15

〖C〗10000

〖D〗105

本题考查位运算中按位异或运算符, 异或运算只有在两个比较的位不同时其结果是1, 否则结果为0, 10用二进制表示为00001010, 5用二进制表示为00000101, 异或后00001111, 即15, 选项B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+"); |
|  | for (i = 0; i < 6; i++) { |
|  | fseek(fp, 0L, 0); |
|  | fprintf(fp, "%d\n", a[i]); |
|  | } |
|  | rewind(fp); |
|  | fscanf(fp, "%d", &k); |
|  | fclose(fp); |
|  | printf("%d\n", k); |
|  | } |

则程序的输出结果是

〖A〗123456

〖B〗1

〖C〗6

〖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6, 答案为C。

**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：用函数指针指向要调用的函数, 并进行调用。规定在【2】处使f指向函数f1, 在【3】处使f指向函数f2。当调用正确时, 程序输出：  
x1 = 5.000000, x2 = 3.000000, x1 \*x1 + x1 \*x2 = 40.000000。  
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) double |
|  | (2) f1 |
|  | (3) f2 |

填空1 : (\*f)()代表一个指向函数的指针变量, 而本题所被调用的f1、f2两函数的返回值均为double型, 因而此空应该填写double。  
填空2:因为在给函数指针变量赋值时, 只需给出函数名而不必给出参数, 因而此空应该填写f1。  
填空3:同填空2, 此空应该填写f2。  
相关知识点：  
(1) 指向函数的指针变量的一般定义形式为：数据类型(\*指针变量名)();  
(2) 函数的调用可以通过函数名调用, 也可以通过函数指针调用。  
(3) 在给函数指针变量赋值时, 只需给出函数名而不必给出参数。

【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赋值。函数fun的功能是将单向链表结点(不包括头结点)数据域为偶数的值累加起来, 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) while (p != NULL) |
|  | (2) p = p->next; |

(1) 判断当前指针p指向的结点是否存在, 若存在则对该结点数据域进行判断操作。  
(2) 判断结束后指针指向下一个结点。

【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：判断字符串是否为回文, 若是, 则函数返回1, 主函数中输出"YES", 否则返回0, 主函数中输出"NO"。回文是指顺读和倒读都一样的字符串。  
例如, 字符串LEVEL是回文, 而字符串123312就不是回文。  
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

|  |  |
| --- | --- |
|  | int fun(char \*str) { |
|  | int i, n = 0, fg = 1; |
|  | char \*p = str; |
|  | while (\*p) |
|  | /\*将指针p置位到字符串末尾, 并统计字符数\*/ { |
|  | n++; |
|  | p++; |
|  | } |
|  | for (i = 0; i < n / 2; i++) |
|  | /\*循环比较字符\*/ |
|  | if (str[i] == str[n - 1 - i]); |
|  | /\*相同, 什么都不作\*/ |
|  | else |
|  | / \*不同, 直接跳出循环\*/ { |
|  | fg = 0; |
|  | break; |
|  | } |
|  | return fg; |
|  | } |

判断回文就是将第一个字符与最后一个字符比较, 第二个字符与倒数第二个字符进行比较, 依此类推, 直到中间的字符, 如果比较的结果都相同, 那么这个字符串就是回文, 否则不是。  
程序中循环语句用来遍历字符串, 条件语句用来判断当前字符与对应位置的字符是否相同。