[**【计算机二级C语言】卷012**](https://www.cnblogs.com/mjios/p/12452917.html)

**选择题**

**公共知识**

【1】下列链表中, 其逻辑结构属于非线性结构的是

〖A〗二叉链表

〖B〗循环链表

〖C〗双向链表

〖D〗带链的栈

在定义的链表中, 若只含有一个指针域来存放下一个元素地址, 称这样的链表为单链表或线性链表。  
带链的栈可以用来收集计算机存储空间中所有空闲的存储结点, 是线性表。  
在单链表中的结点中增加一个指针域指向它的直接前件, 这样的链表, 就称为双向链表(一个结点中含有两个指针), 也是线性链表。  
循环链表具有单链表的特征, 但又不需要增加额外的存贮空间, 仅对表的链接方式稍做改变, 使得对表的处理更加方便灵活, 属于线性链表。  
二叉链表是二叉树的物理实现, 是一种存储结构, 不属于线性结构。  
答案为A选项。

【2】设循环队列的存储空间为Q(1 : 35), 初始状态为front = rear = 35。现经过一系列入队与退队运算后, front = 15, rear = 15, 则循环队列中的元素个数为

〖A〗15

〖B〗16

〖C〗20

〖D〗0或35

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。  
在循环队列中进行出队、入队操作时, 头尾指针仍要加1, 朝前移动。  
只不过当头尾指针指向向量上界时, 其加1操作的结果是指向向量的下界0。  
由于入队时尾指针向前追赶头指针, 出队时头指针向前追赶尾指针, 故队空和队满时, 头尾指针均相等。  
答案为D选项。

【3】下列关于栈的叙述中, 正确的是()。

〖A〗栈底元素一定是最后入栈的元素

〖B〗栈顶元素一定是最先入栈的元素

〖C〗栈操作遵循先进后出的原则

〖D〗以上说法均错误

栈顶元素总是后被插入的元素, 从而也是最先被删除的元素; 栈底元素总是最先被插入的元素, 从而也是最后才能被删除的元素。  
栈的修改是按后进先出的原则进行的。  
因此, 栈称为先进后出表, 或"后进先出"表, 所以选择C。

【4】在关系数据库中, 用来表示实体间联系的是

〖A〗属性

〖B〗二维表

〖C〗网状结构

〖D〗树状结构

关系模型实体间的联系采用二维表来表示, 简称表。  
选项C为网状模型实体间的联系, 选项D为层次模型实体间的联系, 选项A属性刻画了实体。

【5】公司中有多个部门和多名职员, 每个职员只能属于一个部门, 一个部门可以有多名职员。则实体部门和职员间的联系是

〖A〗1 : 1联系

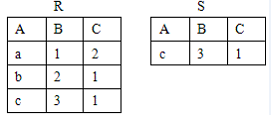
〖B〗m : 1联系

〖C〗1 : m联系

〖D〗m : n联系

两个实体集间的联系实际上是实体集间的函数关系, 主要有一对一联系(1 : 1)、一对多联系(1 : m)、多对一联系(m : 1)、多对多联系(m : n)。  
对于每一个实体部门, 都有多名职员, 则其对应的联系为一对多联系(1 : m), 答案选C。

【6】有两个关系R和S如下：



则由关系R得到关系S的操作是

〖A〗选择

〖B〗投影

〖C〗自然连接

〖D〗并

由关系R到关系S为一元运算, 排除C和D。  
关系S是关系R的一部分, 是通过选择之后的结果, 因此选A。

【7】数据字典(DD)所定义的对象都包含于

〖A〗数据流图(DFD图)

〖B〗程序流程图

〖C〗软件结构图

〖D〗方框图

在数据流图中, 对所有元素都进行了命名, 所有名字的定义集中起来就构成了数据字典。  
因此选A, 而B, C, D都不符合。

【8】软件需求规格说明书的作用不包括

〖A〗软件验收的依据

〖B〗用户与开发人员对软件要做什么的共同理解

〖C〗软件设计的依据

〖D〗软件可行性研究的依据

软件需求规格说明书是需求分析阶段的最后成果, 是软件开发的重要文档之一。

软件需求规格说明书有以下几个方面的作用。

①便于用户、开发人员进行理解和交流, B正确;  
②反映出用户问题的结构, 可以作为软件开发工作的基础和依据, C正确;  
③作为确认测试和验收的依据, A正确。

【9】下面属于黑盒测试方法的是

〖A〗语句覆盖

〖B〗逻辑覆盖

〖C〗边界值分析

〖D〗路径覆盖

黑盒测试不关心程序内部的逻辑, 只是根据程序的功能说明来设计测试用例。  
在使用黑盒测试法时, 手头只需要有程序功能说明就可以了。  
黑盒测试法分等价类划分法、边界值分析法和错误推测法, 答案为C。  
而A、B、D均为白盒测试方法。

【10】下面不属于软件设计阶段任务的是

〖A〗软件总体设计

〖B〗算法设计

〖C〗制定软件确认测试计划

〖D〗数据库设计

从技术观点上看, 软件设计包括软件结构设计、数据设计、接口设计、过程设计。  
所以A、B、D正确, C为软件测试阶段的任务。

**专业知识**

【11】下列叙述中错误的是

〖A〗C程序可以由多个程序文件组成

〖B〗一个C语言程序只能实现一种算法

〖C〗C程序可以由一个或多个函数组成

〖D〗一个C函数可以单独作为一个C程序文件存在

一个C语言程序可以实现多种算法, 可以由多个程序文件共同构成。  
一个C语言程序可以由一个或多个函数组成, 一个C函数可以单独作为一个C程序文件存在, 被包含到其他程序中。  
所以选择B选项。

【12】C语言源程序名的后缀是

〖A〗.C

〖B〗.exe

〖C〗.obj

〖D〗.cp

C语言源程序名的后缀是.C, 目标文件的扩展文件是.obj, 可执行文件的扩展名是.exe。

【13】以下选项中不能用作C程序合法常量的是

〖A〗123

〖B〗'\123'

〖C〗1, 234

〖D〗"\x7D"

C选项中不能含有逗号, 所以"1,234"不能用作C程序的合法常量。  
B选项中表示的是八进制的常量, D选项中表示的是十六进制的常量。

【14】表达式 a += a -= a = 9 的值是

〖A〗18

〖B〗-9

〖C〗0

〖D〗9

首先计算a = 9, 然后计算a -= 9, 即a = a - 9, 结果为0, 然后计算a += 0, 即a = a + 0, 所以最终结果为0。

【15】若有定义语句：

复制代码

C

int x = 12, y = 8, z;

在其后执行语句  
z = 0.9 + x / y;  
则z的值为

〖A〗1

〖B〗1.9

〖C〗2

〖D〗2.4

先计算x / y, 结果为1, 然后计算0.9 + 1, 结果为1.9, 但由于变量z为整型, 所以取值1, 所以z的值为1。

【16】阅读以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int case; |
|  | float printF; |
|  | printf("请输入2个数:"); |
|  | scanf("%d %f", &case, &printF); |
|  | printf("%d %f\n", case, printF); |
|  | } |

该程序在编译时产生错误, 其出错原因是

〖A〗定义语句出错, case是关键字, 不能用作用户自定义标识符

〖B〗定义语句出错, printF不能用作用户自定义标识符

〖C〗定义语句无错, scanf不能作为输入函数使用

〖D〗定义语句无错, printf不能输出case的值

C语言中标识符的定义中区分大小写, 所以printF与printf不一样, 可以作为标识符。  
而case是C语言中的关键字, 不能作为变量名。  
scanf是标准的输入函数, printf是标准的输出函数, 所以选择A选项。

【17】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char c1, c2, c3, c4, c5, c6; |
|  | scanf("%c%c%c%c", &c1, &c2, &c3, &c4); |
|  | c5 = getchar(); |
|  | c6 = getchar(); |
|  | putchar(c1); |
|  | putchar(c2); |
|  | printf("%c%c\n", c5, c6); |
|  | } |

程序运行后, 若从键盘输入(从第1列开始)  
123<回车>  
45678<回车>  
则输出结果是

〖A〗1245

〖B〗1256

〖C〗1278

〖D〗1267

Scanf按照格式字符进行输入, 所以123与回车的值分别送入了c1、c2、c3、c4变量的存储空间。  
getchar()函数的功能是从键盘输入的字符串中读入一个字符, 所以4送入了变量c5的存储空间, 5送入了变量c6的存储空间。  
所以打印时结果为1245。

【18】以下选项中, 当x为大于1的奇数时, 值为0的表达式是

〖A〗x / 2

〖B〗x % 2 == 0

〖C〗x % 2 != 0

〖D〗x % 2 == 1

A选项的值为不定值, 但绝对不是0。  
算术运算符的优先级高于关系运算符的优先级, 所以当x为大于1的奇数时, B选项的值为假, 即0。  
C选项的值为真, 即1。  
D选项的值为真, 即1。  
选项B正确。

【19】以下选项中与  
if (a == 1) a = b;  
else a++;  
语句功能不同的switch语句是

〖A〗switch (a == 1) { case 0: a = b; break; case 1: a++; }

〖B〗switch (a) { case 1: a = b; break; default: a++; }

〖C〗switch (a) { default: a++; break; case 1: a = b; }

〖D〗switch (a == 1) { case 1: a = b; break; case 0: a++; }

if语句的含义是如果a的值为1, 将b的值赋给a, 否则将a的值自增1。  
Swich(a == 1)中表达式a == 1的结果如果为真即为1, 则从case1开始执行, 直到遇到break语句跳出switch语句, 如果a == 1的结果如果为假即0, 则从case0开始执行, 所以与题干含义执行操作相反的是选项A中的语句。

【20】若变量已正确定义, 有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | i = 0; |
|  | do printf("%d,", i); |
|  | while (i++); |
|  | printf("%d\n", i); |

其输出结果是

〖A〗0,1

〖B〗0,0

〖C〗1,1

〖D〗程序进入无限循环

第一次首先先执行循环体打印i的值0, 然后判断while的条件i++, 先用后自增, 所以判断时条件为假, 跳出循环, 但是仍执行了i的自增操作, i的值为1。  
在接下来的打印语句中进行打印。

【21】有以下程序：

复制代码

C

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

程序运行后的输出结果是

〖A〗1,3,2

〖B〗1,3,3

〖C〗1,2,3

〖D〗3,2,1

else 总是和最近的if配对, 所以进入第一个if语句中条件判断时, 因为是逻辑与操作需要两边运算对象的值均为非零值, 才为真, 所以需要逐个执行判断的结果为1, 不再执行第二个case语句中的操作, 而是进入第二个if语句条件判断, 因为b != 2条件成立所以整个条件表达式的值为真, 所以不再执行逻辑或的第二个运算对象(c-- != 3)c的值不变, 也不再执行第一个else语句?打印a的值1, b的值3, c的值3。

【22】有以下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | 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

当产生的随机数为4时, 对5取余结果为4, 执行case4后面语句, 见到continue结束本次循环, 而非结束程序运行, 所以B选项错误。  
当产生的随机数为1时, 执行case1后面的所有语句, 直到遇到break语句跳出switch结构; 当随机数为2时, 执行case2后面的所有语句, 直到遇到continue语句跳出本次循环, 所以C选项错误。  
由于循环体中含有exit(0)语句, 该语句的功能是终止程序的执行, 所以不是固定执行8次, A选项错误。

【23】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun1(double a) { |
|  | return (int) (a \*= a); |
|  | } |
|  | int fun2(double x, double y) { |
|  | double a = 0, b = 0; |
|  | a = fun1(x); |
|  | b = fun1(y); |
|  | return (int) (a + b); |
|  | } |
|  | void main() { |
|  | double w; |
|  | w = fun2(1.1, 2.0); |
|  | printf("%4.2f", w); |
|  | } |

程序执行后输出结果是

〖A〗0.0

〖B〗5

〖C〗5.21

〖D〗5.00

题中变量w的定义为double型, 函数fun2()的定义为int型, 按照各类数值型数据间的混合运算, 整型数据被转换为实型数据。  
因此D选项正确。

【24】设有定义语句：int(\*f)(int);  
则以下叙述正确的是

〖A〗f是基类型为int的指针变量

〖B〗f是指向函数的指针变量, 该函数具有一个int类型的形参

〖C〗f是指向int类型一维数组的指针变量

〖D〗f是函数名, 该函数的返回值是基类型为int类型的地址

int (\*f)(int ); 为指向函数的指针变量的定义方法, 其中f为指向函数的指针变量, 第一个int为函数返回值类型, 第二个int为函数的形参类型。  
因此B选项正确。

【25】有以下程序：

复制代码

C

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

程序运行后的输出结果是

〖A〗3,5,3,5

〖B〗3,5,5,3

〖C〗5,3,3,5

〖D〗5,3,5,3

fun函数的功能是交换形式参数的值, 即交换指针变量a和b的值, 但是fun函数并不能够交换实参的值, 因此fun(p, q)不能交换p和q的值, 所以第一个printf语句的输出为3, 5。  
第二个fun函数对x和y的地址进行了操作, 同样不能交换x和y的值, 并不能影响p和q指针指向的数据, 因此第二个printf语句的输出也是3, 5。  
因此A选项正确。

【26】若有定义：

复制代码

C

int w[3][5];

则以下不能正确表示该数组元素的表达式是

〖A〗\*(&w[0][0] + 1)

〖B〗\*(\*w + 3)

〖C〗\*(\*(w + 1))

〖D〗\*(w + 1)[4]

A选项中\*(&w[0][0] + 1)表示w[0][1]; B选项中\*(\*w + 3)表示w[0][3]; C选项中\*(\*(w + 1))表示w[0][1]; 而D选项中\*(w + 1)[4], 其中w + 1表示行指针移动到第二行, 此时[4]表示行指针在原来的基础上继续移动4个单位, 但是定义的数组只有3行, 因此D选项不能正确表示该数组中的元素; 选择D选项。

【27】有以下程序：

复制代码

C

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

程序运行后的输出结果是

〖A〗1,2,3,4,5,6,7,8,9,10,

〖B〗1,2,6,8,10,12,7,8,9,10,

〖C〗1,2,3,4,10,12,14,16,9,10,

〖D〗1,2,6,8,10,12,14,16,9,10,

数组名是常量, 表示的是数组首元素的地址。  
当执行f(a)的时候, 由于传递的是首地址, 相当于直接对数组a进行操作, 所以从数组a的第三个元素a[2]到元素a[5], 每个元素值扩大两倍。  
因此B选项正确。

【28】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun(char s[]) { |
|  | int n = 0; |
|  | while (\*s <= '9' && \*s >= '0') { |
|  | n = 10 \* n + \*s - '0'; |
|  | s++; |
|  | } |
|  | return (n); |
|  | } |
|  | void main() { |
|  | char s[10] = { '6', '1', '\*', '4', '\*', '9', '\*', '0', '\*' }; |
|  | printf("%d\n", fun(s)); |
|  | } |

程序的运行结果是

〖A〗5

〖B〗9

〖C〗61

〖D〗61490

在fun函数中, while循环的功能是：逐个取字符数组s的字符判断其是否是数字。  
若是则将其作为个位数字保存到变量n中, n的原数据的各个数位将相应左移一个10进制位。  
当指针s指向数组的第3位时, 循环条件不成立, 循环结束, 返回n的值, 输出n的值为61。  
因此C选项正确。

【29】若要求从键盘读入含有空格字符的字符串, 应使用函数

〖A〗getchar()

〖B〗getc()

〖C〗gets()

〖D〗scanf()

scanf()语句中用"空格"间隔不同的字符串, 空格将被全部忽略掉, 所以用scanf()函数不能输入空格; getchar()函数用于输入字符, 其调用格式为：ch = getchar(), getchar()函数从终端读入一个字符作为函数值, 把读入的字符赋给变量ch。  
在输入时, 空格、回车符都将作为字符读入, 而且只有在用户敲入回车键时, 读入才开始执行。  
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。  
getc()函数的调用形式为：ch = getc(pf)其中pf是文件指针, 函数的功能是从pf指定的文件中读入一个字符, 并把它作为函数值返回。  
因此C选项正确。

【30】设有如下程序段：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | char s[20] = "Beijing", \*p; |
|  | p = s; |

则执行p = s; 语句后, 以下叙述正确的是

〖A〗s和p都是指针变量

〖B〗s数组中元素的个数和p所指字符串长度相等

〖C〗可以用\*p表示s[0]

〖D〗数组s中的内容和指针变量p中的内容相同

程序定义了字符数组s和字符型指针变量p, 并且使得p指向数组, 执行p = s; 语句后, \*p即表示s[0]; s数组中元素的个数和 p所指字符串长度不相等, 因为s数组中元素的个数包括字符串结束标志"\0", 而p所指字符串长度不包括"\0"; s是字符型数组, p是指针变量; 数组s中的内容和指针变量p中的内容不相同, 数组中存放的是字符串, 而指针变量p存放的是数组的首地址。  
因此C选项正确。

【31】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char s[] = "abcde"; |
|  | s += 2; |
|  | printf("%d\n", s[0]); |
|  | } |

执行后的结果是

〖A〗输出字符c的ASCII码

〖B〗程序出错

〖C〗输出字符c

〖D〗输出字符a的ASCII码

本题重点考查数组名的概念。  
在C语言中, 数组名类似于一个指向数组首地址的指针常量, 一旦定义就不能修改其内容。  
所以本题中的s += 2; 语句让数组名s的内容加2是错误的, 编译无法通过。  
因此B选项正确。

【32】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | void fun(char \*s[], int n) { |
|  | char \*t; |
|  | int i, j; |
|  | for (i = 0; i < n - 1; i++) |
|  | for (j = i + 1; j < n; j++) |
|  | if (strlen(s[i]) > strlen(s[j])) { |
|  | t = s[i]; |
|  | s[i] = s[j]; |
|  | s[j] = t; |
|  | } |
|  | } |
|  | void main() { |
|  | char \*ss[] = { "bcc", "bbcc", "xy", "aaaacc", "aabcc" }; |
|  | fun(ss, 5); |
|  | printf("%s,%s\n", ss[0], ss[4]); |
|  | } |

程序的运行结果是

〖A〗aaaacc, xy

〖B〗xy, aaaacc

〖C〗bcc, aabcc

〖D〗aabcc, bcc

函数fun(char s[], int n)的功能是对字符串数组的元素按照字符串的长度从小到大排序, 在主函数中执行fun(ss, 5)语句后, \*ss[] = { "xy", "bcc", "bbcc", "aabcc", "aaaacc" }, ss[0], ss[4]的输出结果为xy, aaaacc。  
因此B选项正确。

【33】有以下程序：

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int a = 4; |
|  | int f(int n) { |
|  | int t = 0; |
|  | static int a = 5; |
|  | if (n % 2) { |
|  | int a = 6; |
|  | t += a++; |
|  | } else { |
|  | int a = 7; |
|  | t += a++; |
|  | } |
|  | return t + a++; |
|  | } |
|  | void main() { |
|  | int s = a, i = 0; |
|  | for (; i < 2; i++) |
|  | s += f(i); |
|  | printf("%d\n", s); |
|  | } |

程序运行后的输出结果是

〖A〗36

〖B〗24

〖C〗32

〖D〗28

在主函数main()中定义了两个变量s和i, 同时给s赋初值。  
a = 4, i赋初值0。  
当i = 0时。  
执行"s+=f(i);"语句, 调用f()并将i的初值 0传递给形参n。  
首先执行if语句中内条件：n % 2, 若条件为假。  
则执行else下的语句, a = 7, t = 7 + 0 = 7, 使用return 返回t, t = 7 + (a++) = 7 + 5 = 12, 此时a运算完后自增1, 变为6; 返回主函数中, s = 4 + 12 = 16。  
当i = 1时, 执行"s+=f(i);"语句, 调用f()并将i的初值1传递给形参n。  
首先执行if语句中的条件：n % 2, 条件为真, 执行if下面的语句, t = 0, a = 6 t = 0 + 6 = 6, 使用return 返回t, t = 6 + 6 = 12, 返回主函数中, s = 16 + 12 = 28。  
最后输出的结果为28。  
因此D选项正确。

【34】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int f(int n) { |
|  | int t = 0, a = 5; |
|  | if (n / 2) { |
|  | int a = 6; |
|  | t += a++; |
|  | } else { |
|  | int a = 7; |
|  | t += a++; |
|  | } |
|  | return t + a++; |
|  | } |
|  | void main() { |
|  | int s = 0, i = 0; |
|  | for (; i < 2; i++) |
|  | s += f(i); |
|  | printf("%d\n", s); |
|  | } |

程序运行后的输出结果是

〖A〗28

〖B〗24

〖C〗32

〖D〗36

在主函数main()中定义了两个变量s和i, 同时给s和i赋初值0。  
当i = 0时。  
执行"s+=f(i);"语句, 调用f()并将i的初值 0传递给形参n。  
首先执行if语句中内条件：n / 2, 条件为假。  
则执行else下的语句, a = 7, t = 7 + 0 = 7, 使用return 返回t, t = 7 + (a++) = 7 + 5 = 12; 返回主函数中, s = 0 + 12 = 12。  
当i = 1时循环继续, 执行"s+=f(i);"语句, 调用f()并将i的初值1传递给形参n。  
变量t和a再一次赋初值, 首先执行if语句中的条件：n / 2, 条件仍为假, 执行if下面的语句, t = 0, a = 7 t = 0 + 7 = 7, 使用return 返回t, t = 7 + 5 = 12, 返回主函数中, s = 12 + 12 = 24。  
最后输出的结果为24。  
因此B选项正确。

【35】有以下定义和语句

复制代码

C

|  |  |
| --- | --- |
|  | struct workers { |
|  | int num; |
|  | char name[20]; |
|  | char c; |
|  | struct { |
|  | int day; |
|  | int month; |
|  | int year; |
|  | } s; |
|  | }; |
|  | struct workers w, \*pw; |
|  | pw = &w; |

能给w中year成员赋1980的语句是

〖A〗pw->year = 1980;

〖B〗w.year = 1980;

〖C〗w.s.year = 1980;

〖D〗\*pw.year = 1980;

本题考查结构体变量的引用, 题目中定义了一个结构体, 其中结构体中的变量又是一个结构体, w为外层结构体, 如果给内层结构体赋值, C先要得到内层结构体变量, 即w.s, 若要给year赋值, 表示为w.s.year即可, 选项C正确。

【36】以下叙述中错误的是

〖A〗可以用typedef将已存在的类型用一个新的名字来代表

〖B〗可以通过typedef增加新的类型

〖C〗用typedef定义新的类型名后, 原有类型名仍有效

〖D〗用typedef可以为各种类型起别名, 但不能为变量起别名

本题考查typedef的用法, typedef并不是增加一种新的类型, 而是对已存在的类型用一个新的名字来代表, 所以B选项错误。

【37】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | struct tt { |
|  | int x; |
|  | struct tt \*y; |
|  | } \* p; |
|  | struct tt a[4] = { 20, a + 1, 15, a + 2, 30, a + 3, 17, a }; |
|  | void main() { |
|  | int i; |
|  | p = a; |
|  | for (i = 1; i <= 2; i++) { |
|  | printf("%d,", p->x); |
|  | p = p->y; |
|  | } |
|  | } |

程序的运行结果是

〖A〗20, 30,

〖B〗30, 17

〖C〗15, 30,

〖D〗20, 15,

本题考查结构体变量的引用以及结构体数组, p指向a数组的第一个元素, 所以p->x为20, 然后p = p->y后, p指向数组a的第二个元素, 所以输出15, 选项D正确。

【38】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | struct STU { |
|  | char name[9]; |
|  | char sex; |
|  | double score[2]; |
|  | }; |
|  | struct STU a = { "Zhao", 'm', 85.0, 90.0 }, b = { "Qian", 'f', 95.0, 92.0 }; |
|  | b = a; |
|  | printf("%s,%c,%2.0f,%2.0f\n", b.name, b.sex, b.score[0], b.score[1]); |
|  | } |

程序的运行结果是

〖A〗Qian, m, 85, 90

〖B〗Zhao, m, 85, 90

〖C〗Zhao, f, 95, 92

〖D〗Qian, f, 95, 92

本题考查结构体初始化操作, 本题中可以直接将结构体a赋值给b, 所以输出的结果和a一样, 选项B正确。

【39】以下叙述中错误的是

〖A〗C程序对预处理命令行的处理是在程序执行的过程中进行的

〖B〗预处理命令行的最后不能以分号表示结束

〖C〗#define MAX 是合法的宏定义命令行

〖D〗在程序中凡是以"#"开始的语句行都是预处理命令行

本题考查预处理命令行, 预处理是在程序编译之前进行的, 所以A选项错误。

【40】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*f; |
|  | f = fopen("filea.txt", "w"); |
|  | fprintf(f, "abc"); |
|  | fclose(f); |
|  | } |

若文本文件filea.txt中原有内容为：hello, 则运行以上程序后, 文件filea.txt中的内容为

〖A〗abclo

〖B〗abc

〖C〗helloabc

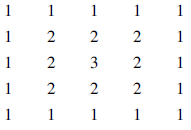
〖D〗abchello

本题考查文件操作函数, 执行fprintf(f, "abc"); 后f文件的内容就变为了abc, 所以B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的功能是建立一个N×N的矩阵。矩阵元素的构成规律是：最外层元素的值全部为1; 从外向内第2层元素的值全部为2; 第3层元素的值全部为3, ……依此类推。例如, 若N = 5, 生成的矩阵为：



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) [N] |
|  | (2) i |
|  | (3) i + 1 |

填空1:本题考查了形参的确定。  
参数传递时将实参的值赋给形参, 实参和形参是一一对应的, 因此该空应该填写[N]。  
填空2:第二重for循环中a[i][j]和a[N - i - 1][j]表示第一行和最后一行数组a[N][N]的值, 因而此空应该填写i。  
填空3:第三重for循环代表的是a[N][N]中每一列的值, 因此此空应该填写i + 1。

【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：将十进制正整数m转换成k(2≤k≤9)进制数, 并按位输出。例如, 若输入8和2, 则应输出1000(即十进制数8转换成二进制表示是1000)。  
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(int m, int k) |
|  | (2) aa[i] = m % k; |
|  | (3) printf("%d", aa[i - 1]); |

(1) 函数定义的格式错误, 不应带有";"。  
(2) 将十进制正整数转换为任意进制的数与十进制正整数转换成二进制的数的方法是一样的。  
从整数n译出它的各位k进制数值, 需采用除k取余的方法, 即求n除k的余数, 得到它的k进制的个位数, 接着将n除以k。  
在n不等于0的状况下循环, 能顺序求出n的k进制的各个位上的数。  
(3) 在进行for(i = 0; m; i++)循环结束时, i已经多加了一个1, 所以这里要减去1。

【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, 编写一个函数, 其功能是：从传入的num个字符中找出最长的一个字符串, 并通过形参指针max传回该串地址(用\*\*\*\*作为结束输入的标识)。  
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

|  |  |
| --- | --- |
|  | char \*fun(char(\*a)[81], int num, char \*max) { |
|  | int i = 0; |
|  | max = a[0]; |
|  | for (i = 0; i < num; i++) |
|  | /\*找出最长的一个字符串\*/ |
|  | if (strlen(max) < strlen(a[i])) |
|  | max = a[i]; |
|  | return max; |
|  | /\*传回最长字符串的地址\*/ |
|  | } |

解答本题之前, 首先应该明白ss是一个指向一维数组的指针变量, max是指向指针的变量, 所以引用变量时要注意加上\*。  
本程序使用循环语句遍历字符串数组, 使用条件语句判断该字符串是否最大。