参考答案

【2.1】参考答案： D

注释：程序中除法运算的两个操作数均是整型，运算结果也是整型。

【2.2】参考答案： B

注释： C 语言允许在程序块 (分程序)中说明变量。

【2.3】参考答案： C

注释：变量 i 中的负号传送给变量 n 后，因 n 是无符号数，已不作为负号处理。

【2.4】参考答案： D

注释：对变量 x 的 —— 操作是后缀形式，变量 x 的减 1 操作要在执行完 printf 函数之后才进行，所以变量 x 的值在输出的时候仍然保持原值 10。

【2.5】参考答案： B

注释： C 语言在执行 printf() 时，对函数中的表达式表列的处理顺序是从后向前， 即先处理 n- -，再处理 n++，最后处理 n，而且每一个表达式作为一个处理单元，也就是说在不同的表达式中自增自减运算是单独考虑的。

【2.6】参考答案： A

注释：变量 x 和变量 y 做按位与，结果为 0x0200，右移 4 位为 0x0020，再与 0x005f 做按位或，最后结果为 0x007f。

【2.7】参考答案： A

注释：逗号表达式的结果是用逗号分开的最后一个表达式的值，此题由于 c=='A' 的值是 0,所以逗号表达式的值为 0。

【2.8】参考答案 ：B

【2.9】参考答案 ：A

【2.10】参考答案 ：C

注释： 在输出格式描述 "%m.ns" 中， m 是输出总长度， n 是实际字符的个数， 这里 m 没有给出， 则输出总长度就是实际输出字符的个数。

【2.11】参考答案 ：C

【2.12】参考答案： B

【2.13】参考答案： C

【2.14】参考答案： B

【2.15】参考答案： D

【2.16】参考答案： A

【2.17】参考答案 ：C

【2.18】参考答案： A

【2.19】参考答案 ：C

注释：在 switch 语句中， case本身仅起到语句标号的作用，不会改变语句的流程， 执行 break 语句才能退出当前的 switch 语句。

【2.20】参考答案 ：D

注释： siwtch 语句的表达式中，变量 c 是后缀的增一运算，第一次执行 do-while 循环时，执行 case 'A'后面的语句。

【2.21】参考答案 ：D

【2.22】参考答案 ：B

【2.23】参考答案 ：B

注释： fabs()是浮点数绝对值函数。

【2.24】参考答案 ：A

【2.25】参考答案 ：C

注释： C 语言允许在程序块（分程序）内说明变量，如果在程序块内说明的变量和程序块外的变量同名，在块外说明的变量在块内是不可见的。可将此题和【 2.11】进行比较，加深理解。

【2.26】参考答案 ：C

【2.27】参考答案 ：B

【2.28】参考答案： ① D ② A

【2.29】参考答案： D

【2.30】参考答案 ：B

注释：输出结果为字符串长度。

【2.31】参考答案 ：D

注释：字符串拷贝函数 strcpy()要求的两个参数都是字符串首地址。本题中第二个参数是字符串常量，接受这个字符串的第一个参量不是直接给出字符数组名，而是进行了地址运算

后的结果。由于 str 字符串的长度是 13，除 2 取整后是 6，第一个参数给出的地址是字符数组 str 的首地址加 6，也就是原来字符串中第二个空格的位置，把"es she"从该处放入，字符串 str 变为 "How does she"。

【2.32】参考答案 ：C

注释： main 函数调用 func 函数时，第一个实参使用的是逗号表达式的值，也就是 x+y 的结果。由于对变量 x、y、z 进行的是后缀运算，所以函数 func 的参数值是 13 和 8。

【2.33】参考答案 ：C

【2.34】参考答案 ：① C ② A ③ C

【2.35】参考答案 ：C

【2.36】参考答案 ：B

注释：函数 fun 进行了递归调用，实际进行的运算是 5×4×3×2×1×3×10。主函数内说明的局部变量 w 屏蔽了外部变量 w，所以在主函数中外部变量 w 是不可见的，在调用 printf 函数时表达式 "fun(5)*w" 中 w 的值是 10。

【2.37】参考答案 ：D

注释： main 函数三次调用了函数 funa，在 funa 函数中的静态变量 c 仅在第一次调用时进行了初始化，再次调用时不再对静态变量赋初值。

【2.38】参考答案 ：B

注释： main 函数和 num 函数中都说明了变量 a 和 b，由于它

们是内部变量，所以它们分别在说明它们的函数内有效。外部变量 x 和 y 在函数 num 之后被说明，而在 num 函数中又要引用它们，所以在 num 函数中用关键字 "extern" 说明变量 x 和 y 是一个外部变量，也就是通知计算机这两个变量在 fun 函数以外被说明，此处不是定义两个 int 型变量。

【2.39】参考答案 ：D

注释：函数 f 中的变量 c 是静态变量，仅在第一次调用函数 f 时它被初始化为 3,第二次调用函数 f 时 c 的值是 4，第三次调用函数 f 时 c 的值是 5。

【2.40】参考答案 ：D

【2.41】参考答案 ：D

注释：程序中有三个 "x" 分别在三个不同的函数中，这三个 "x" 都是自动变量，所以三个 "x" 分别局部于三不同的函数，在三个函数中对 "x" 的操作互不影响。

【2.42】参考答案 ：A

【2.43】参考答案 ：A

注释：*(++p) 和*++p 都是指针变量值前加 1, 第一次指向 a[1]，第二次指向 a[2]；a+3 是 a[3] 的地址。

【2.44】参考答案 ：C

注释：②句没有语法错误，但是 a+6 指向数组之外，因为 a 是 a[0] 的地址，a+1 是 a[1] 的地址，a+2 是 a[2] 的地址，显然数组 a 没有 a[6] 分量。③句错误，因为 a[1]是地址常量，它

是 a[1][0] 的地址，对于地址常量是不可以进行赋值运算的。

【2.45】参考答案 ：① D ② A

注释：如果 FMT 定义为 "%x\n" ，则输出的 16 进制数据用小写字母表示。

【2.46】参考答案 ：A

注释：语句 "p=&a[0]" 表示将数组 a 中元素 a[0] 的地址赋给指针变量 p, 则 p 就是指向数组首元素 a[0] 的指针变量， "&a[0]" 是取数组首元素的地址。 对于指向数组首址的指针， p+i（或 a+i）是数组元素 a[i] 的地址， *(p+i)(或 *(a+i)) 就是 a[i] 的值。

【2.47】参考答案 ：B

【2.48】参考答案 ：D

【2.49】参考答案 ：D

【2.50】参考答案 ：A

注释：变量 g 是指向函数的指针， (*g)(a,b) 是调用指针 g 所指向的函数。

【2.51】参考答案 ：C

注释： p 是指针， pp 是指向指针的指针。

【2.52】参考答案 ：A

注释：对于指向数组的指针变量可以做下标运算， p[i] 和 alpha[i] 都是指向字符串的首地址， *p[i] 取出字符串的第一个字符。

【2.53】参考答案 ：① D ② A ③ D ④ D ⑤ B

注释： pp 是一个二维指针数组， pp+1 指向数组的第二维，*(pp+1) 是第二维的起始地址， **(pp+1) 是第二维第一个元素的地址， ***(pp+1) 是第二维第一个元素的内容，所以，①的参考答案应选 D。*(pp+1)+1 是第二维第二个元素的地址，*(*(pp+1)+1) 是第二维第二个元素， (*(*(pp+1)+1))[4] 则是第二维第二个元素所指字符串下标为 4 的元素，即是字符 w，故③应当选 D。

【2.54】参考答案 ：① B ② C

【2.55】参考答案 :D

【2.56】参考答案 :C

注释：联合体成员的取值是最后一次给成员赋的值。

【2.57】参考答案 : B

注释：整型数组 i 和字符数组 c 共用存储空间，给 i 赋值也等于给 c 赋值，所以 s->c[0]=0x39 ，所以输出 9。

【2.58】参考答案： A

注释：基本概念。 EOF 是由 C 语言在头文件 stdio.h 中定义的，用户可以直接使用。