[**【计算机二级C语言】卷002**](https://www.cnblogs.com/mjios/p/12403848.html)

**选择题**

**公共知识**

【1】在最坏情况下比较次数相同的是()。

〖A〗冒泡排序与快速排序

〖B〗简单插入排序与希尔排序

〖C〗简单选择排序与堆排序

〖D〗快速排序与希尔排序

冒泡排序、快速排序、简单插入排序、简单选择排序在最坏情况下比较次数均为n的r次方(1 < r < 2)。

【2】设二叉树的中序序列为BCDA, 前序序列为ABCD, 则后序序列为()。

〖A〗CBDA

〖B〗DCBA

〖C〗BCDA

〖D〗ACDB

二叉树的前序序列为ABCD, 由于前序遍历首先访问根节点, 可以确定该二叉树的根节点是A。  
再由中序序列为BCDA(与前序序列相反), 可知以A为根的该二叉树只存在左子树, 不存在右子树, 故后序序列为中序序列为BCDA。

【3】树的度为3, 且有9个度为3的结点, 5个度为1的结点, 但没有度为2的结点。则该树中的叶子结点数为()。

〖A〗18

〖B〗33

〖C〗19

〖D〗32

设叶子结点数为n, 则该树的节点数为n + 9 + 5 = n + 14, 根据树中的结点数=树中所有结点的度之和+1, 得9×3 + 0×2 + 5×1 + n×0 + 1 = n + 14, 则n = 19。

【4】下列叙述中错误的是()。

〖A〗向量属于线性结构

〖B〗二叉链表是二叉树的存储结构

〖C〗栈和队列是线性表

〖D〗循环链表是循环队列的链式存储结构

循环链表是线性表的一种链式存储结构, 循环队列是队列的一种顺序存储结构。  
因此D选项叙述错误。

【5】下面对软件特点描述错误的是()。

〖A〗软件的使用存在老化问题

〖B〗软件的复杂性高

〖C〗软件是逻辑实体具有抽象性

〖D〗软件的运行对计算机系统具有依赖性

软件具有以下特点。

①软件是一种逻辑实体, 具有抽象性。

②软件没有明显的制作过程。

③软件在使用期间不存在磨损、老化问题。

④对硬件和环境具有依赖性。

⑤软件复杂性高, 成本昂贵。

⑤软件开发涉及诸多的社会因素

【6】数据流图(DFD)的作用是()。

〖A〗描述软件系统的控制流

〖B〗支持软件系统功能建模

〖C〗支持软件系统的面向对象分析

〖D〗描述软件系统的数据结构

数据流图是系统逻辑模型的图形表示, 从数据传递和加工的角度, 来刻画数据流从输入到输出的移动变化过程, 它直接支持系统的功能建模。

【7】结构化程序的三种基本控制结构是()。

〖A〗递归、堆栈和队列

〖B〗过程、子程序和函数

〖C〗顺序、选择和重复

〖D〗调用、返回和转移

1966年Boehm和Jacopini证明了程序设计语言仅仅使用顺序、选择和重复三种基本控制结构就足以表达出各种其他形式结构的程序设计方法。

【8】同一个关系模型的任意两个元组值()。

〖A〗可以全相同

〖B〗不能全相同

〖C〗必须全相同

〖D〗以上都不对

关系具有以下7条性质：①元组个数有限性：二维表中元组的个数是有限的。

②元组的唯一性：二维表中任意两个元组不能完全相同。

③元组的次序无关性：二维表中元组的次序, 即行的次序可以任意交换。

④元组分量的原子性：二维表中元组的分量是不可分割的基本数据项。

⑤属性名唯一性：二维表中不同的属性要有不同的属性名。

⑥属性的次序无关性：二维表中属性的次序可以任意交换。

⑦分量值域的同一性：二维表属性的分量具有与该属性相同的值域, 或者说列是同质的。

满足以上7个性质的二维表称为关系, 以二维表为基本结构所建立的模型称为关系模型。

【9】在银行业务中, 实体客户和实体银行之间的联系是()。

〖A〗一对一

〖B〗一对多

〖C〗多对一

〖D〗多对多

一个客户可以在多家银行办理业务, 一家银行也有多个客户办理业务, 因此, 实体客户和实体银行之间的联系是多对多。

【10】定义学生选修课程的关系模式如下：

复制代码

C

SC(S#, Sn, C#, Cn, G, Cr)(其属性分别为学号、姓名、课程号、课程名、成绩、学分)

则对主属性部分依赖的是()。

〖A〗C#→Cn

〖B〗(S#, C#)→G

〖C〗(S#, C#)→S#

〖D〗(S#, C#)→C#

关系SC中的主键是(S#, C#), 但C#(课程号)单独就可以决定Cn(课程名), 存在着对主键的部分依赖。

**专业知识**

【11】一个算法应当具有五个特性, 以下叙述中正确的是()。

〖A〗有穷性、确定性、复杂性、有零个或多个输入、有一个或多个输出

〖B〗有穷性、确定性、可行性、有零个或多个输入、有一个或多个输出

〖C〗有穷性、确定性、可行性、必须要有一个以上的输入、有一个或多个输出

〖D〗有穷性、确定性、复杂性、有零个或多个输入、必须要多个输出

算法是指为解决某个特定问题而采取的确定且有限的步骤, 一个算法应当具有五个特征：有穷性、确定性、可行性、有零个或多个输入、有一个或多个输出。  
本题答案B

【12】以下不能定义为用户标识符的是()。

〖A〗Void

〖B〗scanf

〖C〗int

〖D〗\_3com\_

标识符的命名可以由字母、数字或下划线组成, 并且第一个字符必须为字母或下划线, 另外用户标识符不能使用关键字, 选项A的Void可以定义为用户标识符, 因为C语言对大小写敏感, Void与关键字void属于不同的标识符; 选项B中scanf是库函数名, 属于预定义标识符, 它也可以作为用户标识符使用, 不过通常不建议这么使用; 选项C中的int属于关键字, 错误; 选项D符合标识符的命名规则, 也不属于关键字, 可以作为标识符使用; 本题答案：C

【13】以下不能作为合法常量的是()。

〖A〗'cd'

〖B〗1.234e04

〖C〗"\a"

〖D〗'\011'

字符常量是使用单引号括起来的单个字符, 选项A错误; 选项B属于浮点数常量, 正确; 选项C属于转义字符常量, 正确; 选项D属于转义字符, 代表八进制数011的ASCII码值的字符, 正确; 本题答案A。

【14】若有定义：int a = 1, b = 2, c = 3, d = 4, m = 2, n = 2;, 则执行(m = a > b) && (n = c > d)后n的值是()。

〖A〗4

〖B〗3

〖C〗2

〖D〗1

对于表达式：(m = a > b) && (n = c > d), 首先执行m = a > b, 由于a、b的值分别为：1, 2, 所以a > b的值为0, m的值为0, 又由于逻辑与运算符&&有短路原则, 第一个表达式的值为假, 所以整个表达式的值已经确定为假, 第二个表达式不会被执行, n的值依然是2, 本题答案：C

【15】有说明语句：int a, b;, 如果输入111222333, 使得a的值为111, b的值为333, 则以下正确的语句是()。

〖A〗scanf("%3d%\*3d%3d", &a, &b);

〖B〗scanf("%\*3d%3d%3d", &a, &b);

〖C〗scanf("%3d%3d%\*3d", &a, &b);

〖D〗scanf("%3d%\*2d%3d", &a, &b);

根据题意, 要是a的值为111, b的值为333, 必须在读入时指定a的读入宽度为3, b的读入宽度为3, 且a和b的控制字符之间必须额外增加%\*控制符, 用于跳过中间的三位输入数字, 选项A正确; 本题答案A

【16】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | double x = 2.0, y; |
|  | if (x < 0.0) |
|  | y = 0.0; |
|  | else if (x < 10.0) |
|  | y = 1.0 / x; |
|  | else |
|  | y = 1.0; |
|  | printf("%f\n", y); |
|  | } |

程序运行后的输出结果是()。

〖A〗1.000000

〖B〗0.000000

〖C〗0.250000

〖D〗0.500000

程序定义double变量x, y, 给x赋初值2.0, if语句判断, 当x小于0.0时, 给y赋值0.0; 否则当x小于10.0时, y的值为1.0 / x; 当x大于等于10.0时, y的值为1.0, 题意中x的值为2.0, 所以y的值为1.0 / x, 即0.500000, 本题答案D

【17】有以下程序

复制代码

C

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

程序运行后的输出结果是()。

〖A〗20

〖B〗13

〖C〗10

〖D〗31

程序首先定义整型变量s和i, 对s赋初值为0, for循环中i的取值为1, 2, 3, 4, 另外在switch语句中, 执行各个分支后, 若没有break语句, 会继续执行后续分支。

当i = 1时, switch语句执行case 1, case 2和default, 将s自增3后再自增5, 此时s的值为8

当i = 2时, switch语句执行case 2, default, 将s自增3后再自增5, 此时s的值为16;  
当i = 3时, switch语句执行case 3, case 1, case 2, default, 将s逐步自增2, 自增3, 自增5, 此时s的值为26

当i = 4时, switch语句执行default语句, 将s自增5, 此时s的值为31

本题答案：D

【18】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int w = 4, x = 3, y = 2, z = 1; |
|  | printf("%d\n", (w < x ? w : z < y ? z : x)); |
|  | } |

程序运行后的输出结果是()。

〖A〗4

〖B〗2

〖C〗3

〖D〗1

条件表达式形式：表达式1?表达式2:表达式3, 当表达式1的值为非零时, 整个表达式的值是表达式2的值; 当表达式1的值为零值时, 整个表达式的值时表达式3的值; 题意中的表达式是嵌套的条件表达式：w < x ? w : z < y ? z : x, 等价于w < x ? w : (z < y ? z : x), 由于w取值为4, x取值为3, 所以w < x值为零值, 整个表达式的值为z < y ? z : x, z取值为1, y取值为2, 所以z < y的值为真, 整个表达式的值为z的值1, 本题答案D。

【19】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int x, i; |
|  | for (i = 1; i <= 100; i++) { |
|  | x = i; |
|  | if (++x % 2 == 0) |
|  | if (++x % 3 == 0) |
|  | if (++x % 7 == 0) |
|  | printf("%d ", x); |
|  | } |
|  | printf("\n"); |
|  | } |

程序运行后的输出结果是()。

〖A〗42 84

〖B〗28 70

〖C〗26 68

〖D〗39 81

for循环中, i取值为1到100, 循环体中, 三个if语句必须同时满足才会输出x, 输出的x取值为i + 3, 所以x的取值只能是4到104; x必须满足以下条件：1、x % 7 == 0(x必须是7的倍数)2、(x - 1) % 3 == 0(x必须是3的倍数+1)3、(x - 2) % 2 == 0(x必须是2的倍数)满足条件3的x取值为：7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98

又满足条件2的x的取值为：7, 28, 49, 70, 91

又满足条件1的x的取值为：28, 70

本题答案：B

【20】以下叙述中正确的是()。

〖A〗在switch语句中, 不一定使用break语句

〖B〗break语句只能用于switch语句

〖C〗break语句必须与switch语句中的case配对使用

〖D〗在switch语句中必须使用default

switch语句中不一定使用break语句, 选项A正确; break语句除了用于switch语句, 还可以用于循环语句中, 选项B、C错误; switch语句不一定需要使用default语句, 选项D错误; 本题答案A。

【21】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | char fun(char ch) { |
|  | if (ch >= 'A' && ch <= 'Z') |
|  | ch = ch - 'A' + 'a'; |
|  | return ch; |
|  | } |
|  | void main() { |
|  | char s[] = "ABC+abc=defDEF", \*p = s; |
|  | while (\*p) { |
|  | \*p = fun(\*p); |
|  | p++; |
|  | } |
|  | printf("%s\n", s); |
|  | } |

程序运行后的输出结果是()。

〖A〗abc+abc=defdef

〖B〗abc+ABC=DEFdef

〖C〗abcABCDEFdef

〖D〗abcabcdefdef

题意中, 函数fun的功能是：判断形参ch是否大写字母, 若是大写字母改写成小写字母, 其他字符不变; main函数中, 通过while循环, 调用fun函数, 将字符数组s中的各个字符传入, 将s中的大写字母改成小写字母, 程序输出：abc + abc = defdef, 本题答案A。

【22】若要判断char型变量c中存放的是否为小写字母, 以下正确的表达式是()。

〖A〗'a' <= c <= 'z'

〖B〗(c >= 'a') && (c <= 'z')

〖C〗(c >= 'a') || (c <= 'z')

〖D〗('a' <= c)AND('z' >= c)

C语言中, char型变量的值是其对应字符的ASCII码值, 可以作比较运算, 由于小写字母的ASCII码值按字母表的顺序连续递增, 所以判断char型变量c是否是小写字母时, 判断c的ASCII码值是否在'a'和'z'之间, 即：(c >= 'a') && (c <= 'z'), 本题答案B。

【23】以下叙述中错误的是()。

〖A〗在一个函数内的复合语句中定义的变量在本函数范围内有效

〖B〗在一个函数内定义的变量只在本函数范围内有效

〖C〗在不同的函数中可以定义相同名字的变量

〖D〗函数的形参是局部变量

在一个函数内的复合语句中定义的变量在本复合语句块范围内有效, 选项A错误, 其他选项正确, 本题答案A。

【24】以下叙述中错误的是()。

〖A〗形参可以是常量、变量或表达式

〖B〗实参可以是常量、变量或表达式

〖C〗实参的类型应与形参的类型赋值兼容

〖D〗实参的个数应与形参的个数一致

函数中, 形参必须是变量, 实参可以是常量, 变量或表达式, 选项A错误, 选项B正确; 实参的个数和类型要与形参一致, 选项C、D正确; 本题答案：A

【25】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int k = 2, m = 4, n = 6, \*pk = &k, \*pm = &m, \*p; |
|  | \*(p = &n) = \*pk \* (\*pm); |
|  | printf("%d\n", n); |
|  | } |

程序运行后的输出结果是()。

〖A〗6

〖B〗10

〖C〗8

〖D〗4

题意中, 整型指针变量pk指向k, pm指向m, 所以表达式\*pk \* (\*pm)的值为k \* m, 即2 \*4 = 8; 左边表达式\*(p = &n)先将变量n的地址赋给p, 然后对p解引用, 引用到n, 对n赋值为8, 本题答案为C。

【26】若有定义int \* p[3];, 则以下叙述中正确的是()。

〖A〗定义了一个指针数组p, 该数组含有三个元素, 每个元素都是基类型为int的指针

〖B〗定义了一个基类型为int的指针变量p, 该变量具有三个指针

〖C〗定义了一个名为\*p的整型数组, 该数组含有三个int类型元素

〖D〗定义了一个可指向一维数组的指针变量p, 所指一维数组应具有三个int类型元素

int \*p[3]含义是定义一个指针数组p, 数组中包含3个元素, 每个元素都是int\*类型的指针, 本题答案A。

【27】以下不能对二维数组a进行正确初始化的语句是()。

〖A〗int a[2][3] = { 0 };

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

〖C〗int a[][3] = { { 1, 2 }, { 0 } };

〖D〗int a[][3] = { 1, 2, 3, 4, 5, 6 };

选项A中, 对二维数组a的6个元素都赋值为0, 正确; 选项B中, 由于a包含2个元素, 每个元素都是包含3个元素的一维数组, 初始化列表中包含3个元素, 每个元素是包含2个元素的数组, 错误; 选项C中a的每个元素是包含3个元素的一维数组, 初始化列表对a[0]的三个元素初始化为：1, 2, 0, 对a[1]的三个元素初始化为0, 0, 0, 正确; 选项D中对a[0]初始化为1, 2, 3, 对a[1]初始化为4, 5, 6, 正确; 本题答案：B

【28】若有以下说明和定义

复制代码

C

|  |  |
| --- | --- |
|  | int fun(int \*c) { |
|  | ... |
|  | } |
|  | void main() { |
|  | int (\*a)(int \*) = fun, \*b(), x[10], c; |
|  | ... |
|  | } |

则对函数fun的正确调用语句是()。

〖A〗(\*a)(&c);

〖B〗a = a(x);

〖C〗b = \*b(x);

〖D〗fun(b);

题意中函数fun接收一个整型指针参数, 返回值为int类型, main函数首先定义一个函数指针a, 将函数fun的地址赋给a, 所以a是指向函数fun的指针, 可以通过a调用函数fun; 选项A中, 通过a调用函数fun, 可以使用(\*a), 接收的参数是整型变量c的地址, 正确; 选项B中参数x是一个数组, 错误; 选项C中调用b函数, 由于程序没有给出函数b的定义, 所以这里调用b是错误的, 而且函数b是没有参数的, 这里调用b的时候传入了参数, 所以C错误; 选项D中, 由于b是一个函数, 不能作为整型指针变量传给fun函数, 所以D错误; 本题答案A。

【29】有以下函数

复制代码

C

|  |  |
| --- | --- |
|  | int fun(char \*p, char \*q) { |
|  | while ((\*p != '\0') && (\*q != '\0') && (\*p == \*q)) { |
|  | p++; |
|  | q++; |
|  | } |
|  | return (\*p - \*q); |
|  | } |

此函数的功能是()。

〖A〗比较p和q所指字符串的大小

〖B〗计算p和q所指字符串的长度差

〖C〗将q所指字符串连接到p所指字符串后面

〖D〗将q所指字符串复制到p所指字符串中

函数fun接收两个整型指针变量作为参数, 通过while循环, 比较p和q对应位上的各个字符, 如果字符相同, 继续向后比较; 否则循环结束, 返回第一次对应不同字符的ASCII码差值, 所以函数fun是对p和q指向的字符串进行比较, 比较的大小是按第一个对应位置上不同字符的ASCII码值; 本题答案A。

【30】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(char \*a, char \*b) { |
|  | a = b; |
|  | (\*a)++; |
|  | } |
|  | void main() { |
|  | char ch1 = 'A', ch2 = 'a', \*p1 = &ch1, \*p2 = &ch2; |
|  | fun(p1, p2); |
|  | printf("%c%c\n", ch1, ch2); |
|  | } |

程序运行后的输出结果是()。

〖A〗Aa

〖B〗Ab

〖C〗ab

〖D〗Ba

函数fun将形参b赋给形参a, 使得a和b都指向原b所指向的地址, 然后对该地址的值执行自增1; main函数中p1指向ch1, p2指向ch2, 通过fun函数的调用, 将ch2的值完成自增1, 字符'a'自增1后变成字符'b', 所以程序输出：Ab, 本题答案：B。

【31】有以下程序

复制代码

C

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

程序运行后的输出结果是()。

〖A〗34

〖B〗55

〖C〗10

〖D〗18

题意中, 函数fun的功能是通过递归, 将数组a中下标为0到n - 1位置的元素累加, 作为函数返回值返回, main函数调用fun时, 传入的a + 2作为数组参数, 传入n的值是4, 所以函数返回值是元素a[2], a[3], a[4], a[5]的和, 程序输出18, 本题答案：D

【32】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(int n, int \*s) { |
|  | int f1, f2; |
|  | if (n == 1 || n == 2) |
|  | \*s = 1; |
|  | else { |
|  | fun(n - 1, &f1); |
|  | fun(n - 2, &f2); |
|  | \*s = f1 + f2; |
|  | } |
|  | } |
|  | void main() { |
|  | int x; |
|  | fun(6, &x); |
|  | printf("%d\n", x); |
|  | } |

程序运行后的输出结果是()。

〖A〗5

〖B〗2

〖C〗3

〖D〗8

题意中, 函数fun的第二个参数通过指针作为函数返回值, 它的功能是通过递归, 求得fun(n)的值为fun(n - 1) + fun(n - 2), 由于当n取值为1, 2时, 值为1, 所以可知整个数列如下：1, 1, 2, 3, 5, 8..., 即某一项是前两项之和, 所以当n取值为6时, fun(6, \*s)返回s的值为3 + 5 = 8, 本题答案D。

【33】以下叙述中错误的是()。

〖A〗将函数内的局部变量说明为static存储类是为了限制其他编译单位的引用

〖B〗一个变量作用域的开始位置完全取决于变量定义语句的位置

〖C〗全局变量可以在函数以外的任何部位进行定义

〖D〗局部变量的"生存期"只限于本次函数调用, 因此不能将局部变量的运算结果保存至下一次调用

将函数内的局部变量说明为static存储类, 是为了只在第一次调用该函数时才对其初始化, 后续调用时使用上一次调用结束后的值; 函数体内的局部变量无论是否声明为static, 外部编译单位都不能引用, 选项A错误, 其他选项正确; 本题答案A。

【34】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | void fun(int \*\*s, int p[2][3]) { |
|  | \*\*s = p[1][1]; |
|  | } |
|  | void main() { |
|  | int a[2][3] = { 1, 3, 5, 7, 9, 11 }, \*p; |
|  | p = (int \*) malloc(sizeof(int)); |
|  | fun(&p, a); |
|  | printf("%d\n", \*p); |
|  | } |

程序运行后的输出结果是()。

〖A〗1

〖B〗11

〖C〗7

〖D〗9

题意中函数fun的功能是将二维数组p的行下标为1, 列下标为1的元素(p[1][1])的值, 赋给二维地址s指向的\*s所指向的存储单元中, main函数中定义了整型指针p, 动态分配了整型长度的内存空间, 调用函数fun将数组元素a[1][1], 即9赋给p所指向的空间, 本题答案D。

【35】以下叙述中正确的是()。

〖A〗宏替换不占用程序的运行时间

〖B〗预处理命令行必须位于源文件的开头

〖C〗在源文件的一行上可以有多条预处理命令

〖D〗宏名必须用大写字母表示

宏替换是在编译阶段前的预处理阶段, 对程序中的宏完成文本替换, 因此宏替换不占运行时间, 选项A正确; 预处理命令行无需在源文件的开头, 它可以出现在程序的任何一行的开始部位, 其作用一直持续到源文件的末尾, 选项B错误; 在源文件的一行上至多只能有一条预处理命令, 选项C错误; 宏名通常使用大写字母表示, 这并不是语法规定, 只是一种习惯, 选项D错误; 本题答案A。

【36】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | struct NODE { |
|  | int k; |
|  | struct NODE \*next; |
|  | }; |
|  | void main() { |
|  | struct NODE m[5], \*p = m, \*q = m + 4; |
|  | int i = 0; |
|  | while (p != q) { |
|  | p->k = ++i; |
|  | p++; |
|  | q->k = i++; |
|  | q--; |
|  | } |
|  | q->k = i; |
|  | for (i = 0; i < 5; i++) |
|  | printf("%d", m[i].k); |
|  | printf("\n"); |
|  | } |

程序运行后的输出结果是()。

〖A〗13442

〖B〗13431

〖C〗01234

〖D〗02431

main函数中定义包含5个元素的数组m, 每个元素都是NODE类型, 指针p指向数组第一个元素, 指针q指向数组最后一个元素; while循环使用p, q从首尾向中间遍历, 遍历的同时为各个元素赋值, 所以第一轮循环, i的值为0, 先执行++i的值为1, 后执行i++的值也为1, m[0].k和m[4].k的值都为1; 接着第二轮循环, i的值为2, 先执行++i的值为3, 后执行i++的值也为3, m[1].k和m[3].k的值都为3; 第三轮循环时, p和q指向的都是m[2]元素, 指针相同, 循环结束, 此时i的值为4, 即m[2].k赋值为4, 综上, 程序输出：13431, 本题答案：B

【37】以下叙述中正确的是()。

〖A〗结构体变量中的成员可以是简单变量、数组或指针变量

〖B〗不同结构体的成员名不能相同

〖C〗结构体定义时, 其成员的数据类型可以是本结构体类型

〖D〗结构体定义时, 类型不同的成员项之间可以用逗号隔开

结构体变量中的成员可以是简单变量、数组、指针变量或者是结构体变量, 选项A正确; 不同结构体成员名可以相同, 选项B错误; 结构体定义时, 其成员的数据类型不能是本结构体类型, 选项C错误; 结构体定义时, 类型不同的成员项之间使用分号隔开, 选项D错误; 本题答案：A

【38】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int ch = 020; |
|  | printf("%d\n", ch = ch >> 1); |
|  | } |

程序运行后的输出结果是()。

〖A〗10

〖B〗40

〖C〗32

〖D〗8

整型变量ch使用八进制数020初始化, 二进制数为10000, 右移一位结果为：1000, 使用%d输出十进制数为：8, 本题答案D

【39】标准库函数fgets(str, n, fp)的功能是()。

〖A〗从fp所指的文件中读取长度不超过n - 1的字符串存入指针str所指的内存

〖B〗从fp所指的文件中读取长度为n的字符串存入指针str所指的内存

〖C〗从fp所指的文件中读取n个字符串存入指针str所指的内存

〖D〗从fp所指的文件中读取长度为n - 1的字符串存入指针str所指的内存

fgets函数的功能是从fp所指文件中读入n - 1个字符放入以str为起始地址的空间内, 读取长度不超过n - 1, 读入结束后, 自动在最后添加\0, 选项A正确, 本题答案A。

【40】函数fread(buffer, size, count, fp)中buffer代表的是()。

〖A〗一个存储区, 存放要读的数据项

〖B〗一个整数, 代表要读入的数据项总数

〖C〗一个文件指针, 指向要读的文件

〖D〗一个指针, 指向读入数据要存放的地址

fread函数用来读二进制文件, 其中buffer是数据块的指针, 它是内存块的首地址, 输入的数据存入此内存中; size表示每个数据块的字节数; count用来指定每读一次, 读入的数据块个数, fp是文件指针, 指向要读的文件, 选项A、B、C错误, 本题答案D

**编程题**

[编程题代码下载](https://github.com/CoderMJLee/NCRE)

【41】给定程序中, 函数fun的功能是：根据所给的一组学生的成绩, 由公式：



计算出标准差(Standard Deviation)σ并返回。公式中μ为平均成绩, N为学生人数, xi为学生成绩。  
例如, 如有成绩为：50, 60, 70, 80, 90, 100, 55, 65, 75, 85, 95, 99, 则运行结果应为：16.527754  
请在程序的下划线处填入正确的内容并把下划线删除, 使程序得出正确的结果。  
注意：源程序存放在文件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) x[i] |
|  | (2) avg |
|  | (3) sqrt(sum / n) |

(1) 根据题意公式, 首先需要计算N个学生成绩的平均值, fun()函数通过第一个for循环, 将x数组中学生成绩累加到变量avg中, 所以第一空填写：x[i]。  
(2) 由(1)可知avg是n个学生的成绩之和, 所以求平均成绩需要再除以n, 第二空填写：avg。  
(3) 按照公式, 求得平均成绩avg后, 再次遍历数组x, 对每个学生成绩与平均成绩求差平方, 累加到sum变量, 最后对sum除以n后再开方, sqrt()是求开平方函数, 第三空填写：sqrt(sum / n)。

【42】给定程序MODI1.C中, 函数fun的功能是：逐个判断字符串s2中的字符是否在字符串s1中出现过, 并把出现过的字符按照原来在s2中的顺序形成新字符串s3, 不去除重复字符。  
例如, 当s1为："abcdefghijk", s2为："zbcyedmnpggn" 时, 则s3应该是："bcedgg"  
请改正函数fun中指定部位的错误, 使它能得出正确的结果。  
注意：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) if (s2[i] == s1[j]) |
|  | (2) if (j < strlen(s1)) |
|  | (3) s3[k++] = s2[i]; |

(1) 根据题意, fun函数在循环中需要判断s2[i]是否在s1中出现, 所以将s2[i]与s1的每一个字符进行比较, 如果与s1[j]相等, 说明s2[i]在s1中出现了, 跳出s1的循环, 第一空修改为：if (s2[i] == s1[j]) 。  
(2) 内嵌的s1循环结束后, 需要判断j是否遍历完s1字符串, 若j小于strlen(s1), 说明在内嵌的循环中执行了break语句, 满足s2[i] == s1[j], s2[i]在s1中出现了, 需要存放到s3中; 若j不小于strlen(s1), 说明内嵌的for循环是循环条件不满足终止的, s2[i]在s1中未出现, 不需要存放到s3中; 所以第二空修改为：if (j < strlen(s1)) 。  
(3) 由(2)可知若j小于strlen(s1), 说明在内嵌的循环中执行了break语句, 满足s2[i] == s1[j], s2[i]在s1中出现了, 需要存放到s3中, 由于k初值为0, 所以k表示当前应该存放的下标, s2[i]存放到s3[k]中后, k再自增1。

【43】根据哥德巴赫猜想, 任意一个大偶数都可以分解为两个素数之和。但许多偶数分解为两个素数之和并不是唯一的。  
请编写函数fun, 其功能是：求1000(不包括1000)以内的所有恰好能分解成10组两个素数之和(5 + 109和109 + 5被认为是同一组)的偶数, 并依次存入数组a中并在屏幕上打印出来, 打印时每个数单独一行, 符合条件的个数通过函数值返回。  
例如, 114 = 5 + 109 = 7 + 107 = 11 + 103 = 13 + 101 = 17 + 97 = 31 + 83 = 41 + 73 = 43 + 71 = 47 + 67 = 53 + 61  
114恰好可以分解为10组素数之和, 因此114是我们要找的偶数。  
而 116 = 3 + 113 = 7 + 109 = 13 + 103 = 19 + 97 = 37 + 79 = 43 + 73  
120 = 7 + 113 = 11 + 109 = 13 + 107 = 17 + 103 = 19 + 101 = 23 + 97 = 31 + 89 = 37 + 83 = 41 + 79 = 47 + 73 = 53 + 67 = 59 + 61  
116可以分解为6组素数之和, 120可以分解为12组素数之和, 因此116和120都不是我们要找的偶数。  
函数prime用来判断一个数n是否为素数, 是则返回1, 否则返回0。  
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的花括号中填入所编写的若干语句。

复制代码

C

|  |  |
| --- | --- |
|  | int fun(int m, int a[]) { |
|  | int k, s, count, i = 0; |
|  | for (k = 6; k <= m; k += 2) { |
|  | count = 0; |
|  | /\*请在此处填写代码\*/ |
|  | for (s = 2; s <= k / 2; s++) { |
|  | if (prime(s) == 1 && prime(k - s) == 1) { |
|  | count++; |
|  | } |
|  | } |
|  | if (count == 10) { |
|  | printf("%d\n", k); |
|  | a[i++] = k; |
|  | } |
|  | } |
|  | return i; |
|  | } |

根据题意要求, fun()函数的功能是将m(包括m)以内的所有恰好能分解成10组两个素数之和的偶数存放到数组a中.程序首先定义变量k、s、count、i。  
k是用来遍历6 ~ m之间的所有偶数, count用来记录当前数k能分解成多少组两个素数之和。  
i表示存放数组a的下标。  
for循环遍历6 ~ m的偶数, 首先将count赋值为0, 接着通过内嵌的for循环, 使用s遍历2 ~ k / 2的数(注意：s最大取k / 2, 保证s + k - s和k - s + s被认为是同一组)。  
对于当前的s和k - s, 使用prime函数判断两个数是否为素数, 若prime(s)和prime(k - s)都为1, 说明s和k - s都是素数, 满足题意, 将count自增1, 表示找到一组两个素数之和等于k。  
内嵌的for循环执行完, count表示k可以分解成两个素数之和的组数, 若count恰好等于10, 表示k就满足题意, 将k存放到数组a[i], 并对i自增1, i最后作为函数返回值返回, 表示存放到a中的满足条件的偶数个数。