[**【计算机二级C语言】卷016**](https://www.cnblogs.com/mjios/p/12453787.html)

**选择题**

**公共知识**

【1】在长度为n的顺序表中查找一个元素, 假设需要查找的元素有一半的机会在表中, 并且如果元素在表中, 则出现在表中每个位置上的可能性是相同的。则在平均情况下需要比较的次数大约为()。

〖A〗3n / 4

〖B〗n

〖C〗n / 2

〖D〗n / 4

因为查找的元素有一半机会在表中, 所以二分之一的情况下平均比较次数为n / 2, 二分之一情况下平均比较次数为n。  
总的平均比较次数为(n / 2 + n) / 2 = 3n / 4。  
故本题答案为A选项。

【2】设一棵树的度为3, 其中度为3, 2, 1的结点个数分别为4, 1, 3。则该棵树中的叶子结点数为()。

〖A〗10

〖B〗11

〖C〗12

〖D〗不可能有这样的树

根据题意, 树的结点数为3×4 + 2×1 + 1×3 = 17, 再加1个根结点, 17 + 1 = 18, 即总结点数为18, 总结点减去度不为0的结点数即是叶子结点。  
18 - (4 + 1 + 3) = 10。  
故本题答案为A选项。

【3】设栈的存储空间为 S(1 : 50), 初始状态为 top = 0。现经过一系列正常的入栈与退栈操作后, top = 51, 则栈中的元素个数为()。

〖A〗不可能

〖B〗50

〖C〗0

〖D〗1

栈是一种特殊的线性表, 它所有的插入与删除都限定在表的同一端进行。  
入栈运算即在栈顶位置插入一个新元素, 退栈运算即取出栈顶元素赋予指定变量。  
栈为空时, 栈顶指针top = 0, 经过入栈和退栈运算, 指针始终指向栈顶元素。  
初始状态为top = 0, 当栈满时top = 50, 无法继续入栈, top值不可能为51。  
故本题答案为A选项。

【4】设顺序表的长度为n。下列算法中, 最坏情况下比较次数等于n(n - 1) / 2的是()。

〖A〗可行性分析报告

〖B〗快速排序

〖C〗顺序查找

〖D〗寻找最大项

快速排序在最坏情况下, 是整个序列都已经有序且完全倒序, 此时, 快速排序退化为冒泡排序, 要比较n(n - 1) / 2次才能完成。  
堆排序最坏情况和平均情况下比较次数都是nlog2n。  
顺序查找和寻找最大项在最坏情况下比较次数为n。  
故本题答案为B选项。

【5】下面是面向对象主要特征之一的是()。

〖A〗对象唯一性

〖B〗数据和操作(方法)无关

〖C〗对象是类的抽象

〖D〗多态性体现复用

面向对象方法的基本概念有对象、类和实例、消息、继承与多态性。  
对象的特点有标识唯一性、分类性、多态性、封装性、模块独立性。  
数据和操作(方法)等可以封装成一个对象。  
类是关于对象性质的描述, 而对象是对应类的一个实例。  
多态性指同样的消息被不同的对象接收时可导致完全不同的行为。  
故本题答案为A选项。

【6】软件测试的实施步骤是()。

〖A〗单元测试, 集成测试, 确认测试

〖B〗集成测试, 确认测试, 系统测试

〖C〗确认测试, 集成测试, 单元测试

〖D〗单元测试, 集成测试, 回归测试

软件测试过程一般按4个步骤进行, 即单元测试、集成测试、确认测试和系统测试。  
通过这些步骤的实施来验证软件是否合格, 能否交付用户使用。  
故本题答案为A选项。

【7】软件生存周期中, 解决软件"做什么"的阶段是()。

〖A〗需求分析

〖B〗软件设计

〖C〗软件实现

〖D〗可行性研究

软件需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。  
需求分析的任务是发现需求、求精、建模和定义需求的过程。  
故本题答案为A选项。

【8】数据库三级模式体系结构的划分有利于保持数据库的()。

〖A〗数据独立性

〖B〗数据安全性

〖C〗结构规范化

〖D〗操作可行性

数据库领域公认的标准结构是三级模式结构, 它包括外模式、概念模式、内模式, 有效地组织、管理数据, 提高了数据库的逻辑独立性和物理独立性。  
用户级对应外模式, 概念级对应概念模式, 物理级对应内模式, 使不同级别的用户对数据库形成不同的视图。  
所谓视图, 就是指观察、认识和理解数据的范围、角度和方法, 是数据库在用户"眼中"的反映, 很显然, 不同层次(级别)用户所"看到"的数据库是不相同的。  
故本题答案为A选项。

【9】数据库概念设计阶段得到的结果是()。

〖A〗E - R模型

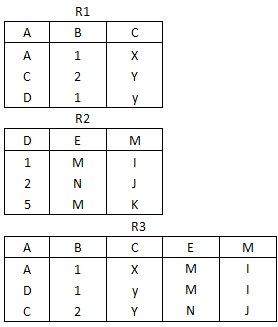
〖B〗数据字典

〖C〗关系模型

〖D〗物理模型

数据库设计有需求分析设计、概念设计、逻辑设计、物理设计、编码、测试、运行、进一步修改等几个阶段。  
在需求分析设计阶段形成需求说明书, 概念设计阶段形成概念数据模型(作为进一步设计数据库的依据, 概念设计通常采用E - R模型), 逻辑设计阶段形成逻辑数据模型(从ER图向关系模式转换、关系视图设计、模式规范化), 物理设计阶段形成数据库内部模型(此时涉及具体软件硬件环境)。  
故本题答案为A选项。

【10】由关系R1和R2得到关系R3的操作是()。



〖A〗等值连接

〖B〗并

〖C〗笛卡尔积

〖D〗交

在连接运算中, 如果连接条件为"=", 就称次连接为等值连接, 否则称为不等值连接。  
观察关系R1的B列和关系R2的D列可以发现, 两列中有相等的值, 而关系R3中的B列为等值连接的结果。  
故本题答案为A选项。

**专业知识**

【11】C语言程序的模块化通过以下哪个选项来实现

〖A〗变量

〖B〗函数

〖C〗程序行

〖D〗语句

C语言程序的模块化通过函数来体现, 所以选择B。

【12】以下选项中不属于C语言标识符的是

〖A〗常量

〖B〗用户标识符

〖C〗关键字

〖D〗预定义标识符

常量是指在程序运行过程中其值不能被改变的量, 如5、1.0、字符'a'等。  
C语言的标识符分为3类：关键字、预定义标识符和用户标识符。  
常量不属于标识符, 所以选择A。

【13】以下选项中不属于C语言程序运算符的是

〖A〗sizeof

〖B〗<>

〖C〗()

〖D〗&&

C语言中的不等于符号用"!="表示, 没有符号"< >", 所以选择B。

【14】若变量已正确定义并赋值, 以下不能构成C语句的选项是

〖A〗B++;

〖B〗A = a + b;

〖C〗a = a + b

〖D〗A ? a : b;

C语言规定, C语句必须以分号";"结束。  
C选项中没有结束符号, 所以错误。

【15】若有以下程序段

复制代码

C

|  |  |
| --- | --- |
|  | double x = 5.16894; |
|  | printf("%f\n", (int) (x \* 1000 + 0.5) / (double) 1000); |

则程序段的输出结果是

〖A〗5.170000

〖B〗5.175000

〖C〗5.169000

〖D〗5.168000

%f的输出格式是以带小数点的数学形式输出浮点数。  
首先计算x \* 1000, 结果为5168.94然后+0.5, 结果为5169.44, 然后进行强制类型转换, 转换成整型为5169, 然后除以双精度数据1000, 结果也为双精度数据5.169, 输出时按照%f的格式输出, 所以输出结果为5.169000。

【16】设有定义：

复制代码

C

double a, b, c;

若要求通过输入分别给a、b、c输入1、2、3, 输入形式如下(注：此处□代表一个空格)  
□□1.0□□2.0□□3.0<回车>  
则能进行正确输入的语句是

〖A〗scanf("%lf%lf%lf", a, b, c);

〖B〗scanf("%lf%lf%lf", &a, &b, &c);

〖C〗scanf("%f%f%f", &a, &b, &c);

〖D〗scanf("%5.1lf%5.1lf%5.1lf", &a, &b, &c);

scanf函数的一般形式为scanf(格式控制, 变量1地址, 变量2地址, …), 其中, 格式字符串要与数据类型保持一致, 所以格式字符串中应为%lf, 后面应该为变量地址, 所以需要加上符号"&", 所以选择B。

【17】若有定义语句：char c = '\101';  
则变量c在内存中占

〖A〗1个字节

〖B〗2个字节

〖C〗3个字节

〖D〗4个字节

本题考查字符变量的存储空间, 每个字符变量被分配一个字节的内存空间。  
因此A选项正确。

【18】若有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | char c1, c2; |
|  | c1 = 'C' + '8' - '3'; |
|  | c2 = '9' - '0'; |
|  | printf("%c %d\n", c1, c2); |
|  | } |

则程序的输出结果是

〖A〗H '9'

〖B〗H 9

〖C〗F '9'

〖D〗表达式不合法输出无定值

本题考查字符变量以及printf()函数相关知识, 字符变量c1被赋值为'C' + '8' - '3', 即ASSCII码的运算, 67 + 54 - 49 = 72, 即H; 字符变量c2被赋值为'9' - '0', 但输出时, 需要注意的是c1以字符变量输出, 而c2是以十进制整型变量输出。  
因此B选项正确。

【19】表示关系式x≤y≤z的C语言表达式的是

〖A〗(x <= y) && (y <= z)

〖B〗(x <= y) || (y <= x)

〖C〗(x <= y <= z)

〖D〗x <= y) !(y <= x)

本题考查关系表达式和逻辑表达式, y大于等于x同时y小于等于z, 因此x <= y和y <= z是逻辑与的关系。  
因此A选项正确。

【20】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int x = 1, y = 0, a = 0, b = 0; |
|  | switch (x) { |
|  | case 1: |
|  | switch (y) { |
|  | case 0: |
|  | a++; |
|  | break; |
|  | case 1: |
|  | b++; |
|  | break; |
|  | } |
|  | case 2: |
|  | a++; |
|  | b++; |
|  | break; |
|  | } |
|  | printf("a=%d,b=%d\n", a, b); |
|  | } |

程序运行后的输出结果是

〖A〗a=1,b=0

〖B〗a=1,b=1

〖C〗a=2,b=1

〖D〗a=2,b=2

本题重点考查switch语句, switch语句中case分支后面如果没有break, 那么就会顺序执行下面的case, 直到遇到break跳出。  
由于变量a和变量b均赋初值为0, 当程序执行完语句case 0 : a++; break; 后, a的值为1, 继续执行语句case 2 : a++; b++; break; a的值为2, b的值为1。  
因此C选项正确。

【21】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int k, j, s; |
|  | for (k = 2; k < 6; k++, k++) { |
|  | s = 1; |
|  | for (j = k; j < 6; j++) |
|  | s += j; |
|  | } |
|  | printf("%d\n", s); |
|  | } |

程序运行后的输出结果是

〖A〗6

〖B〗10

〖C〗24

〖D〗40

本题重点考查for语句循环嵌套, 首先k = 2, 进行一次循环后, k自增两次, 变成4, 这时s为1。  
即当k = 4的时候, s = 1, j = 4, 执行内层循环, s = 1 + 4, j自增1后等于5, s = 5 + 5, j再自增1后即为6, 跳出内层循环。  
k自增两次后等于6, 不成立, 跳出外层循环。  
最后输出s的值, 即为10。  
因此B选项正确。

【22】由以下while构成的循环, 循环体执行的次数是  
int k = 0;  
while (k = 1) k++;

〖A〗有语法错, 不能执行

〖B〗一次也不执行

〖C〗执行一次

〖D〗无限次

本题考查while语句, while语句的语义是：计算表达式的值, 当值为真(非0)时, 执行循环体语句。  
k = 1为赋值语句, 即k 的值等于1。  
while (k = 1) = while (1) , 是死循环, 执行无限次。

【23】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun(int a, intB) { |
|  | return a + b; |
|  | } |
|  | void main() { |
|  | int x = 6, y = 7, z = 8, r; |
|  | r = fun(fun(x, y), z--); |
|  | printf(" %d\n", r); |
|  | } |

程序运行后的输出结果是

〖A〗21

〖B〗15

〖C〗20

〖D〗31

函数fun的功能是返回两个形参的和。  
调用函数fun(x, y)时, 将变量x = 6和y = 7 的值分别传送变量a和b, 并且通过语句return a + b; 将13返回给主函数。  
继续调用函数fun(13, z--), 这里z--指将z的值参与运算后, 再将z的值减1操作; 将13和8分别传送变量a和b, 且通过语句return a + b; 将21返回给主函数。  
因此A选项正确。

【24】以下选项中函数形参不是指针的是

〖A〗fun(int \*a) { … }

〖B〗fun(int a[10]) { … }

〖C〗fun(int & p) { … }

〖D〗fun(int p[]) { … }

B选项和D选项是将数组作为函数参数; A选项是将指针作为函数参数; 因此C选项正确。

【25】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int y = 1, x, a[] = { 2, 4, 6, 8, 10 }, \*p; |
|  | p = &a[1]; |
|  | for (x = 0; x < 3; x++) |
|  | y = y + \*(p + x); |
|  | printf("%d\n", y); |
|  | } |

程序运行后的输出结果是

〖A〗17

〖B〗19

〖C〗18

〖D〗20

本题通过语句"p = &a[1]将指针变量p指向了存储单元a[1], 即使得p[0]的值为4。  
然后通过3次for循环, x = 0时, \*(p + x) = \*p = a[1] = 4所以y = 1 + 4 = 5; x = 1时, \*(p + x) = \*(p + 1) = a[2] = 6, 所以y = 5 + 6 = 11; x = 2时, \*(p + x) = \*(p + 2) = a[3] = 8, 所以y = 11 + 8 = 19。  
因此B选项正确。

【26】以下程序调用fun函数求数组中最大值所在元素的下标。

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(int \*s, int n, int \*k) { |
|  | int i; |
|  | for (i = 0, \*k = i; i < n; i++) |
|  | if (s[i] > s[\*k]) |
|  | \_\_\_\_\_\_\_\_\_\_; |
|  | } |
|  | void main() { |
|  | int a[5] = { 1, 6, 2, 8, 0 }, k; |
|  | fun(a, 5, &k); |
|  | printf("%d %d\n", k, a[k]); |
|  | } |

在横线处应填入的内容是

〖A〗\*k = i

〖B〗k = i

〖C〗\*k = i - s

〖D〗k = i - s

fun函数的功能是求数组中最大值所在元素的下标。  
通过for循环语句, 每次将最大的数给\*k, 即\*k = i。  
因此A选项正确。

【27】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int a[3][4] = { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 }, (\*p)[4] = a, i, j, k = 0; |
|  | for (i = 0; i < 3; i++) |
|  | for (j = 0; j < 2; j++) |
|  | k = k + \*(\*(p + i) + j); |
|  | printf("%d\n", k); |
|  | } |

程序运行后的输出结果是

〖A〗108

〖B〗68

〖C〗99

〖D〗60

该题首先给二维数组赋值a[3][4] = { { 1, 3, 5, 7 } { 9, 11, 13, 15 } { 17, 19, 21, 23 } }; (\*p)[4] = a指针p指向二维数组a的首地址, 接下来执行for循环, 首先 i = 0, j = 0时k = k + \*(\*(p + 0) + 0) = 1(a[0][0]); 然后 i = 0, j = 1时k = k + \*(\*(p + 0) + 1) = 4(a[0][1]); 然后j = 2时, 跳出j的循环, 执行i的循环i = 1, j = 0 时k = k + \*(\*(p + 1) + 0) = 13(a[1][0]); 依次类推一直到i = 3时跳出全部循环。  
循环语句在这里的功能是累加二维数组第1列和第2列的元素, 累加结果为60。

【28】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int funa(int a, intB) { |
|  | return a + b; |
|  | } |
|  | int funb(int a, intB) { |
|  | return a - b; |
|  | } |
|  | int sss(int(\*t)(), int x, int y) { |
|  | return ((\*t)(x, y)); |
|  | } |
|  | void main() { |
|  | int x; |
|  | x = sss(funa, 9, 3); |
|  | x += sss(funb, 8, 3); |
|  | printf("%d\n", x); |
|  | } |

程序运行后的输出结果是

〖A〗24

〖B〗23

〖C〗17

〖D〗22

由x = sss(funa, 9, 3); 可得到x = 9 + 3 = 12, 而后直接传入函数funb, x = x + 8 - 3 = 12 + 5 = 17。  
因此C选项正确。

【29】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | int fun(int x) { |
|  | int p; |
|  | if (x == 0 || x == 1) |
|  | return (3); |
|  | p = x - fun(x - 2); |
|  | return (p); |
|  | } |
|  | void main() { |
|  | printf("%d\n", fun(9)); |
|  | } |

程序运行后的输出结果是

〖A〗4

〖B〗5

〖C〗9

〖D〗7

本题考查函数的递归调用, 执行f(9)后, f(9) = 9 - (7 - (5 - (3 - f(1)))) = 7, 所以答案为D选项。

【30】以下选项中, 能正确进行字符串赋值的是

〖A〗char \*s = "ABCDE";

〖B〗char s[5] = { 'A', 'B', 'C', 'D', 'E' };

〖C〗char s[4][5] = { "ABCDE" };

〖D〗char \*s; gets(s);

本题考查字符数组的初始化, B选项中赋值号右侧是赋值给一个字符数组, 而字符串需要'\0'作为结束标记, 所以错误。  
C选项中字符串包含6个字符, 所以不能赋值在左边只能放5个字符的字符串数组中。  
D选项gets函数, 指针s需要初始化。  
所以选项A正确。

【31】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(char \*\*p) { |
|  | int i; |
|  | for (i = 0; i < 4; i++) |
|  | printf("%s", p[i]); |
|  | } |
|  | void main() { |
|  | char \*s[6] = { "ABCD", "EFGH", "IJKL", "MNOP", "QRST", "UVWX" }; |
|  | fun(s); |
|  | printf("\n"); |
|  | } |

程序运行后的输出结果是

〖A〗ABCDEFGHIJKL

〖B〗ABCD

〖C〗AEIM

〖D〗ABCDEFGHIJKLMNOP

本题考查字符串数组, s表示指向字符数组的指针, s指向了第一个字符串, s++指向了第二个字符串, 所以最后输出结果为D选项。

【32】以下fun函数的功能是将形参s所指字符串内容颠倒过来

复制代码

C

|  |  |
| --- | --- |
|  | void fun(char \*s) { |
|  | int i, j, k; |
|  | for (i = 0, j = strlen(s)\_\_\_\_\_\_\_\_\_\_; i < j; i++, j--) { |
|  | k = s[i]; |
|  | s[i] = s[j]; |
|  | s[j] = k; |
|  | } |
|  | } |

在横线处应填入的内容是

〖A〗+1

〖B〗-1

〖C〗+0

〖D〗, k = 0

本题考查字符串处理函数strlen, strlen计算一个字符串的长度, 不包含结尾符, 所以本题中j的初始化应该是从最后一个字符开始, 因为数组下标从0开始, 所以应该将得到的字符串长度减1, 选项B正确。

【33】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void fun(char \*p, int n) { |
|  | char b[6] = "abcde"; |
|  | int i; |
|  | for (i = 0, p = b; i < n; i++) |
|  | p[i] = b[i]; |
|  | } |
|  | void main() { |
|  | char a[6] = "ABCDE"; |
|  | fun(a, 5); |
|  | printf("%s\n", a); |
|  | } |

程序运行后的输出结果是

〖A〗abcde

〖B〗ABCDE

〖C〗edcba

〖D〗EDCBA

本题考查数组名作为函数参数, 执行f函数时, 传进去的a指针被重新指向了b, 所以原本a数组的地址内容不变, 所以输出结果为ABCDE, 答案为B选项。

【34】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | fun(int a, int b) { |
|  | int static m = 0, i = 2; |
|  | i = i + m + 1; |
|  | m = i + a + b; |
|  | return m; |
|  | } |
|  | void main() { |
|  | int k = 4, m = 1, p; |
|  | p = fun(k, m); |
|  | printf("%d,", p); |
|  | p = fun(k, m); |
|  | printf("%d\n", p); |
|  | } |

程序运行后的输出结果是

〖A〗8,17

〖B〗7,16

〖C〗8,8

〖D〗7,17

本题考查静态局部变量, 本题中先执行fun(4, 1)得到返回值为8, 此时静态局部变量m为8, i为3, 那么再次执行fun(4, 1)得到的值为17, 所以选项A正确。

【35】为了建立如图所示的存储结构(即每个结点含两个域, data是数据域, next是指向结点的指针域), 则在【 】 处应填入的选项是  
Struct link { char data; 【 】 } node;



〖A〗link next;

〖B〗struct link \*next;

〖C〗link \*next;

〖D〗struct link next;

本题主要考查链表中节点的概念, 属于基础知识, 其中指针域应该是指向下个节点, 所以答案为B。

【36】以下结构体说明和变量定义中, 正确的是

〖A〗typedef struct abc { int n; double m; } ABC; ABC x, y;

〖B〗struct abc { int n; double m }; struct abc x, y;

〖C〗struct ABC { int n; double m; } struct ABC x, y;

〖D〗struct abc { int n; double m; }; abc x, y;

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A中ABC为新的类型别名, 定义正确。

【37】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | void fun(int \*p1, int \*p2, int \*s) { |
|  | s = (int \*) malloc(sizeof(int)); |
|  | \*s = \*p1 + \*(p2++); |
|  | } |
|  | void main() { |
|  | int a[2] = { 1, 2 }, b[2] = { 10, 20 }, \*s = a; |
|  | fun(a, b, s); |
|  | printf("%d\n", \*s); |
|  | } |

程序运行后的输出结果是

〖A〗11

〖B〗10

〖C〗1

〖D〗2

本题考查把数组名作为函数参数, 执行fun函数后, s的值并没有发生变化, 仍然是指向a, 所以输出结果为1, 选项C正确。

【38】以下程序依次把从终端输入的字符存放到f文件中, 用#作为结束输入的标志, 则在横线处应填入的选项是

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | FILE \*fp; |
|  | char ch; |
|  | fp = fopen("fname", "w"); |
|  | while ((ch = getchar()) != '#') |
|  | fputc(\_\_\_\_\_\_\_\_\_\_); |
|  | fclose(fp); |
|  | } |

〖A〗ch, "fname"

〖B〗fp, ch

〖C〗ch

〖D〗ch, fp

本题考查fputc函数, 该函数将字符ch写到文件指针fp所指向的文件的当前写指针的位置。  
函数格式：int fputc(int n, File \*fp)。  
答案为D选项。

【39】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #define N 2 |
|  | #define M N + 1 |
|  | #define NUM(M + 1) \* M / 2 |
|  | void main() { |
|  | printf("%d\n", NUM); |
|  | } |

程序运行后的输出结果是

〖A〗4

〖B〗8

〖C〗9

〖D〗6

本题考查宏定义, 宏定义只是做个简单的替换, 执行NUM = (N + 1 + 1) \* N + 1 / 2 = 8, 选项B正确。

【40】有以下程序

复制代码

C

|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | void main() { |
|  | int i = 0; |
|  | i = ~i; |
|  | printf("%d\n", i); |
|  | } |

程序运行后的输出结果是

〖A〗8

〖B〗0

〖C〗1

〖D〗-1

本题考查位运算中按位取反, 0按位取反就是-1, 所以选项D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：将形参s所指字符串中的所有数字字符顺序前移, 其他字符顺序后移, 处理后新字符串的首地址作为函数值返回。  
例如, s所指字符串为：asd123fgh543df, 处理后新字符串为：123543asdfghdf。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) j |
|  | (2) k |
|  | (3) p或(p) |

填空1:函数中申请了两个内存空间, 其中p存放数字字符串, t存放非数字字符串, 根据条件可知, p依次存放数字字符串, 其位置由j来控制, 所以应填j。  
填空2:利用for循环再把t中的内容依次追加到p中, 其中t的长度为k, 所以应填k。  
填空3:处理之后的字符串存放到p中, 最后返回p的首地址即可, 所以应填p。

【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：首先把b所指字符串中的字符按逆序存放, 然后将a所指字符串中的字符和b所指字符串中的字符, 按排列的顺序交叉合并到c所指数组中, 过长的剩余字符接在c所指的数组的尾部。  
例如, 当a所指字符串中的内容为"abcdefg", b所指字符串中的内容为"1234"时, c所指数组中的内容应为"a4b3c2d1efg"; 而当a所指字符串中的内容为"1234", b所指字符串的内容为"abcdefg"时, c所指数组中的内容应该为"1g2f3e4dcba"。  
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) while (i < j) |
|  | (2) if (\*a) |

(1) 由i和j定义的初始值可知, 此处应该判断i是否小于j, 所以应改为while(i < j)。  
(2) 此处将if错写成If, 这个错误比较简单, 但往往不容易发现。

【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, 函数fun的功能是：将s所指字符串中下标为偶数同时ASCII值为奇数的字符删除, s所指串中剩余的字符形成的新串放在t所指的数组中。  
例如, 若s所指字符串中的内容为"ABCDEFG12345", 其中字符C的ASCII码值为奇数, 在数组中的下标为偶数, 因此必须删除; 而字符1的ASCII码值为奇数, 在数组中的下标为奇数, 因此不应当删除, 其他依此类推。最后t所指的数组中的内容应是"BDF12345"。  
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(char \*s, char t[]) { |
|  | int i, j = 0; |
|  | for (i = 0; i < strlen(s); i++) |
|  | if ( !((i % 2) == 0 && (s[i] % 2))) |
|  | t[j++] = s[i]; |
|  | t[j] = 0; |
|  | } |

本题是从一个字符串按要求生成另一个新的字符串。  
我们使用for循环语句来解决这个问题。  
在赋值新的字符串之前, 先对数组元素的下标和ASCII码的值进行判断, 将满足要求的元素赋给新的字符串。