[**【计算机二级C语言】卷019**](https://www.cnblogs.com/mjios/p/12453818.html)

**选择题**

**公共知识**

【1】下列算法中均以比较作为基本运算, 则平均情况与最坏情况下的时间复杂度相同的是()。

〖A〗在顺序存储的线性表中寻找最大项

〖B〗在顺序存储的线性表中进行顺序查找

〖C〗在顺序存储的有序表中进行对分查找

〖D〗在链式存储的有序表中进行查找

在顺序存储的线性表中查找最大项时, 最坏情况下要比较所有元素直到最后一个最大值, 比较次数为n, 顺序查找的平均情况时间复杂度为O(n)。  
可见最坏情况与平均情况下的时间复杂度相同。  
因此A选项正确。  
故本题答案为A选项。

【2】在具有2n个结点的完全二叉树中, 叶子结点个数为()。

〖A〗n

〖B〗n + 1

〖C〗n - 1

〖D〗n / 2

完全二叉树是指除最后一层外, 每一层上的结点数均达到最大值, 在最后一层上只缺少右边的若干结点。  
根据完全二叉树性质, 如果共2n个结点, 从根结点开始按层序用自然数1, 2, …, 2n给结点编号, 则编号为n的结点左子结点编号为2n, 因此叶子结点编号为n + 1, n + 2, …, 2n。  
故叶子结点个数为n, 本题答案为A选项。

【3】下列叙述中正确的是()。

〖A〗在栈中, 栈顶指针的动态变化决定栈中元素的个数

〖B〗在循环队列中, 队尾指针的动态变化决定队列的长度

〖C〗在循环链表中, 头指针和链尾指针的动态变化决定链表的长度

〖D〗在线性链表中, 头指针和链尾指针的动态变化决定链表的长度

在栈中, 栈底指针保持不变, 有元素入栈, 栈顶指名增加, 有元素出栈, 栈顶指针减少。  
在循环队列中, 队头指针和队尾指针的动态变化决定队列的长度。  
在循环链表中, 前一个结点指向后一个结点, 而最后一个结点指向头结点, 只有头结点是固定的。  
线性链表中, 由于前一个结点包含下一个结点的指针, 尾结点指针为空, 要插入或删除元素, 只需要改变相应位置的结点指针即可, 头指针和尾指针无法决定链表长度。  
故本题答案为A选项。

【4】循环队列的存储空间为 Q(1 : 40), 初始状态为 front = rear = 40。经过一系列正常的入队与退队操作后, front = rear = 15, 此后又退出一个元素, 则循环队列中的元素个数为()。

〖A〗39, 或0且产生下溢错误

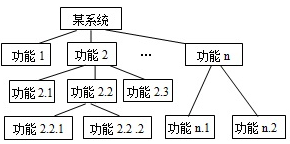
〖B〗14

〖C〗40

〖D〗15

循环队列是队列的一种顺序存储结构, 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。  
入队运算时, 队尾指针进1(即rear + 1), 然后在rear指针指向的位置插入新元素。  
退队运算时, 排头指针进1(即front + 1), 然后删除front指针指向的位置上的元素。  
当front = rear = 15时可知队列空或者队列满, 此后又退出一个元素, 如果之前队列为空, 退出操作会产生错误, 队列里有0个元素; 如果退出之前队列已满(40个元素), 执行退出后, 队列里还有39个元素。  
故本题答案为A选项。

【5】某系统结构图如下图所示



该系统结构图的深度是()。

〖A〗4

〖B〗3

〖C〗2

〖D〗1

树的根结点所在层次为1, 其他结点所在的层次等于它的父结点所在层次加1, 树的最大层次就是树的深度。  
因此本系统结构图的深度为4, 本题答案为A选项。

【6】软件测试的目的是()。

〖A〗发现程序中的错误

〖B〗发现并改正程序中的错误

〖C〗设计和执行测试用例

〖D〗诊断程序中的错误

软件测试是为了发现错误而执行程序的过程。  
测试要以查找错误为中心, 而不是为了演示软件的正确功能, 不是为了评估软件或改正错误。  
故本题答案为A选项。

【7】需求分析阶段产生的主要文档是()。

〖A〗需求规格说明书

〖B〗初步的用户手册

〖C〗确认测试计划

〖D〗数据流图和数据字典

需求分析阶段的工作可以概括为：需求获取、需求分析、编写需求规格说明书、需求评审四个方面。  
软件需求规格说明书是描述需求中的重要文档, 是软件需求分析的主要成果。  
故本题答案为A选项。

【8】下列关于数据库特点的说法中正确的是()。

〖A〗数据能共享且独立性高

〖B〗数据能共享但数据冗余很高

〖C〗能保证数据完整性但降低了安全性

〖D〗数据独立性高但不能实现共享

数据库管理系统的特点：共享性高, 冗余度小; 具有高度的物理独立性和逻辑独立性; 整体结构化, 用数据模型描述; 由数据库管理系统提供数据安全性、完整性、并发控制和恢复能力。  
故本题答案为A选项。

【9】在关系数据库设计中, 关系模式设计属于()。

〖A〗逻辑设计

〖B〗概念设计

〖C〗物理设计

〖D〗需求分析

关系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), 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)。  
因此关系模式设计属于逻辑设计。  
故本题答案为A选项。

【10】定义学生、教师和课程的关系模式：S(S#, Sn, Sd, SA)(属性分别为学号、姓名、所在系、年龄); C(C#, Cn, P#)(属性分别为课程号、课程名、先修课); SC(S#, C#, G)(属性分别为学号、课程号和成绩)。则该关系为()。

〖A〗第三范式

〖B〗第一范式

〖C〗第二范式

〖D〗BCNF范式

范式是符合某一种级别的关系模式的集合。  
关系数据库中的关系必须满足一定的要求, 满足不同程度要求的为不同范式。  
目前关系数据库有六种范式：第一范式(1NF)、第二范式(2NF)、第三范式(3NF)、Boyce - Codd范式(BCNF)、第四范式(4NF)和第五范式(5NF)。  
满足最低要求的范式是第一范式(1NF)。  
在第一范式的基础上进一步满足更多要求的称为第二范式(2NF), 其余范式以次类推。  
一般说来, 数据库只需满足第三范式(3NF)就行了。

第一范式：主属性(主键)不为空且不重复, 字段不可再分(存在非主属性对主属性的部分依赖)。

第二范式：如果关系模式是第一范式, 每个非主属性都没有对主键的部分依赖。

第三范式：如果关系模式是第二范式, 没有非主属性对主键的传递依赖和部分依赖。

BCNF范式：所有属性都不传递依赖于关系的任何候选键。

题目中关系模式没有非主属性对主键的传递依赖和部分依赖, 满足第三范式, 但不满足BCNF范式。  
故本题答案为A选项。

**专业知识**

【11】以下叙述中正确的是

〖A〗程序必须包含所有三种基本结构才能成为一种算法

〖B〗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

〖C〗如果算法非常复杂, 则需要使用三种基本结构之外的语句结构, 才能准确表达

〖D〗只有简单算法才能在有限的操作步骤之后结束

C语言程序可以不包含三种基本结构, 也可以包含其中的一种或多种, 所以A错误。  
三种基本结构可以表示任何复杂的算法, 所以C错误。  
正确的算法, 不管是简单算法还是复杂算法都可以在有限的操作步骤之后结束, 这是算法的有穷性, 所以D错误。

【12】以下叙述中正确的是

〖A〗在算法设计时, 可以把复杂任务分解成一些简单的子任务

〖B〗在C语言程序设计中, 所有函数必须保存在一个源文件中

〖C〗只要包含了三种基本结构的算法就是结构化程序

〖D〗结构化程序必须包含所有的三种基本结构, 缺一不可

C语言的程序中, 函数不一定都放在同一个源文件中, 可以分别放在不同源文件中, 通过#include命令来引用, 所以B错误。  
结构化程序可以包含三种基本结构中的一种或几种, 所以C、D错误。

【13】以下叙述中正确的是

〖A〗只能在函数体内定义变量, 其他地方不允许定义变量

〖B〗常量的类型不能从字面形式上区分, 需要根据类型名来决定

〖C〗预定义的标识符是C语言关键字的一种, 不能另作它用

〖D〗整型常量和实型常量都是数值型常量

变量可以定义在函数体外也可以定义在函数体内, 所以A错误。  
常量的类型可以从字面形式上区分, 比如1为整型常量, 1.0为实型常量, a为字符型常量, 所以B错误。  
预定义的标识符不是C语言的关键字, 所以C错误。

【14】以下叙述中正确的是

〖A〗在C程序中的八进制和十六进制, 可以是浮点数

〖B〗整型变量可以分为int型、short型、long型和unsigned型四种

〖C〗八进制数的开头要使用英文字母o, 否则不能与十进制区分开

〖D〗英文大写字母X和英文小写字母x都可以作为二进制数字的开头字符

C程序中的实型常量只有十进制一种数制, 所以A错误。  
八进制数以0打头, 所以C错误。  
C语言中整数只有十进制、八进制、十六进制三种, 没有二进制数制, 所以D错误。

【15】以下叙述中正确的是

〖A〗在scanf函数中的格式控制字符串是为了输入数据用的, 不会输出到屏幕上

〖B〗在使用scanf函数输入整数或实数时, 输入数据之间只能用空格来分隔

〖C〗在printf函数中, 各个输出项只能是变量

〖D〗使用printf函数无法输出百分号%

scanf函数中的格式控制字符串是为了输入数据用的, 无论其中有什么字符, 也不会输出到屏幕上, 所以A正确。  
scanf()的格式控制串可以使用其他非空白字符, 如逗号, 但在输入时必须输入这些字符, 以保证匹配就可以, 所以B错误。  
Printf函数可以输出常量也可以输出变量, 所以C错误。  
Prinf函数可以用"\%"来输出百分号%, 所以D错误。

【16】以下叙述中正确的是

〖A〗在scanf函数的格式串中, 必须有与输入项一一对应的格式转换说明符

〖B〗只能在printf函数中指定输入数据的宽度, 而不能在scanf函数中指定输入数据占的宽度

〖C〗scanf函数中的字符串, 是提示程序员的, 输入数据时不必管它

〖D〗复合语句也被称为语句块, 它至少要包含两条语句

在printf和scanf函数中都可以指定数据的宽度, 所以B错误。  
scanf()的格式控制串可以使用其他非空白字符, 如逗号, 但在输入时必须输入这些字符, 以保证匹配, 所以C错误。  
复合语句可以由任意多条语句构成, 可以使一条也可以没有, 所以D错误。

【17】以下叙述中正确的是

〖A〗逻辑"或"(即运算符||)的运算级别比算术运算要高

〖B〗C语言的关系表达式：0 < x < 10完全等价于：(0 < x) && (x < 10)

〖C〗逻辑"非"(即运算符 !)的运算级别是最低的

〖D〗由&&构成的逻辑表达式与由||构成的逻辑表达式都有"短路"现象

B选项中, (x > 0) && (x < 10)的值是1或0, 条件是x > 0 并且x < 10。  
但是 0 < x < 10这种写法的结果任何时候都是1; C选项中, 逻辑"非"(即运算符 !)的运算级别是最高的; A选项中, 算术运算符优先级较高, 关系和逻辑运算符优先级较低。  
因此D选项正确。

【18】以下叙述中正确的是

〖A〗if语句只能嵌套一层

〖B〗if子句和else子句中可以是任意的合法的C语句

〖C〗不能在else子句中再嵌套if语句

〖D〗改变if - else 语句的缩进格式, 会改变程序的执行流程

A选项中, if语句的嵌套没有规定层数; C选项中, 可以在else子句中再嵌套if语句; D选项中, 改变if - else 语句的缩进格式, 不会改变程序的执行流程, 需要注意if和else的配对问题。  
因此B选项正确。

【19】以下叙述中正确的是

〖A〗只要适当地修改代码, 就可以将do - while与while相互转换

〖B〗对于"for(表达式1;表达式2;表达式3) 循环体"首先要计算表达式2的值, 以便决定是否开始循环

〖C〗对于"for(表达式1;表达式2;表达式3) 循环体", 只在个别情况下才能转换成while语句

〖D〗如果根据算法需要使用无限循环(即通常所称的"死循环"), 则只能使用while语句

B选项中, for语句中需要先求解先求解表达式1; C选项中, for语句使用最为灵活, 它完全可以取代 while 语句; D选项中, 对于死循环, 也可以使用for语句。  
因此A选项正确。

【20】以下叙述中正确的是

〖A〗只能在循环体内和switch语句体内使用break语句

〖B〗当break出现在循环体中的switch语句体内时, 其作用是跳出该switch语句体, 并中止循环体的执行

〖C〗continue语句的作用是：在执行完本次循环体中剩余语句后, 中止循环

〖D〗在while语句和do - while语句中无法使用continue语句

break语句通常用在循环语句和开关语句中。  
当break用于开关语句switch中时, 可使程序跳出switch而执行switch以后的语句; 当break语句用于do - while、for、while循环语句中时, 可使程序终止循环而执行循环后面的语句, 即满足条件时便跳出循环。  
continue语句的作用是跳过循环体中剩余的语句而强行执行下一次循环。  
B、C和D三个选项中均有错误。  
因此A选项正确。

【21】以下叙述中正确的是

〖A〗'\0'表示字符0

〖B〗"a"表示一个字符常量

〖C〗表达式：'a' > 'b'的结果是"假"

〖D〗'\"'是非法的

A选项中, '\0'表示结束; B选项中, 字符常量是用单引号括起来的一个字符; D选项中, '\"'是双引号符。  
因此C选项正确。

【22】以下叙述中正确的是

〖A〗字符变量在定义时不能赋初值

〖B〗字符常量可以参与任何整数运算

〖C〗同一英文字母的大写和小写形式代表的是同一个字符常量

〖D〗转义字符用@符号开头

A选项中, 字符变量在定义时可以赋初值; C选项中, 同一英文字母的大写和小写形式代表的是不同的字符常量; D选项中, 转义字符以反斜线"\"开头的。  
因此B选项正确。

【23】以下叙述中正确的是

〖A〗函数名允许用数字开头

〖B〗函数调用时, 不必区分函数名称的大小写

〖C〗调用函数时, 函数名必须与被调用的函数名完全一致

〖D〗在函数体中只能出现一次return 语句

B选项中, 函数调用时, 函数名称是需要区分大小写的; A选项中, 函数名不允许以数字开头; D选项中, 在函数中允许有多个return 语句, 但每次调用只能有一个return 语句被执行。  
因此C选项正确。

【24】以下叙述中正确的是

〖A〗如果p是指针变量, 则&p是不合法的表达式

〖B〗如果p是指针变量, 则\*p表示变量p的地址值

〖C〗在对指针进行加、减算术运算时, 数字1表示1个存储单元的长度

〖D〗如果p是指针变量, 则\*p + 1和\*(p + 1)的效果是一样的

B选项中, 如果p是指针变量, 则\*p表示变量p所指向的地址的值; A选项中, 如果p是指针变量, 则&p表示变量p的地址; D选项中, 如果p是指针变量, \*p + 1表示将p所指的值加上1, 而\*(p + 1)表示的是先将指针右移一位再取所指向变量的值。  
因此C选项正确。

【25】以下叙述中正确的是

〖A〗基类型不同的指针变量可以相互混用

〖B〗函数的类型不能是指针类型

〖C〗函数的形参类型不能是指针类型

〖D〗设有指针变量为 double \* p, 则 p + 1 将指针p移动8个字节

B选项中, 所谓函数类型是指函数返回值的类型。  
在Ｃ语言中允许一个函数的返回值是一个指针(即地址), 这种返回指针值的函数称为指针型函数; C选项中, 函数的参数不仅可以是整型、实型、字符型等数据, 还可以是指针类型; A选项中, 虽然不同基类型的指针变量占用字节数是相同的, 但是不能混用。  
因此D选项正确。

【26】如果定义：float a[10], x;  
则以下叙述中正确的是

〖A〗语句 a = &x; 是非法的

〖B〗表达式 a + 1 是非法的

〖C〗三个表达式 a[1]、\*(a + 1)、\*&a[1]表示的意思完全不同

〖D〗表达式\*&a[1]是非法的, 应该写成\*(&(a[1]))

B选项中, 表达式a + 1不是非法的; C选项中, \*(a + 1)即指a[1]; D选项中, \*&a[1]不是非法的。  
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[] = { 2, 4, 6, 8 }, \*p = a, i; |
|  | for (i = 0; i < 4; i++) |
|  | a[i] = \*p++; |
|  | printf("%d\n", a[2]); |
|  | } |

程序的输出结果是

〖A〗2

〖B〗8

〖C〗4

〖D〗6

本题通过语句"\*p=a"将一维数组a的首地址的值赋值给指针变量p, 即\*p的值为2。  
执行for循环, a[i] = \*p++, 将\*p的值赋给a[i]后, 指针p再进行++操作, 即指针p向后移动一个元素。  
当i = 0时a[0] = \*p, 然后p++;, 移动后p指针指向元素a[1]; 当i = 1时, 即a[1] = \*p; p++; 此时p指针指向元素a[2]; 以此类推, 可知for循环中语句"a[i]=\*p++"并没有改变数组a的值, 因此a[2]的值是6。  
故本题答案为D选项。

【28】以下叙述中正确的是

〖A〗语句 int a[4][3] = { { 1, 2 }, { 4, 5 } }; 是错误的初始化形式

〖B〗语句 int a[4][3] = { 1, 2, 4, 5 }; 是错误的初始化形式

〖C〗语句 int a[][3] = { 1, 2, 4, 5 }; 是错误的初始化形式

〖D〗在逻辑上, 可以把二维数组看成是一个具有行和列的表格或矩阵

二维数组在初始化时, 可以省略第1维, 但是不可以省略第2维。  
可以为部分数组元素初始化。  
B选项、C选项和A选项内的初始化都没有错误。  
因此D选项正确。

【29】以下叙述中正确的是

〖A〗字符串常量"Hello"会被隐含处理成一个无名字符型数组, 它有5个元素

〖B〗不能用字符串常量对字符数组名进行整体赋值操作

〖C〗char str[7] = "string!"; 在语法上是合法的, 运行也是安全的

〖D〗char \*str; str = "Hello"; 与char str[]; str = "Hello"; 效果是一样的

字符串中的字符依次存储在内存中一块连续的区域内, 并且把空字符' \0'自动附加到字符串的尾部作为字符串的结束标志。  
故字符个数为n的字符串在内存中应占(n + 1)个字节。  
选项A中有6个元素, 其中包含结尾字符, C选项中字符串常量包含8个字符, 所以不能赋值在str[7]中, D选项后者赋值错误。

【30】以下叙述中正确的是

〖A〗在语句char str[] = "string!"; 中, 数组str的大小等于字符串的长度

〖B〗语句 char str[10] = "string!"; 和 char str[10] = { "string!" }; 并不等价

〖C〗对于一维字符数组, 不能使用字符串常量来赋初值

〖D〗对于字符串常量 "string!", 系统已自动在最后加入了'\0'字符, 表示串结尾

字符串中的字符依次存储在内存中一块连续的区域内, 并且把空字符' \0'自动附加到字符串的尾部作为字符串的结束标志。  
故字符个数为n的字符串在内存中应占(n + 1)个字节。  
B选项中两者等价, C选项中可以使用字符串常量来给一维字符数组赋值, 就像B选项中一样, A选项中数组长度比字符串长度小一个字节, 字符串中包含隐含的结尾符。

【31】设有定义

复制代码

C

char str[] = "Hello";

则语句  
printf("%d %d", sizeof(str), strlen(str));  
的输出结果是

〖A〗5 5

〖B〗6 6

〖C〗6 5

〖D〗5 6

本题考查sizeof和strlen对字符串的处理不同之处, sizeof求出字符串的字符个数, 包括结尾符。  
Strlen求出字符串的实际字符, 不包括结尾符。  
所以答案为6, 5, C选项正确。

【32】在32位编译器上, 设有定义

复制代码

C

char \*str1 = "Hello", str2[] = "Hello";,

则以下语句  
printf("%d %d", sizeof(str1), sizeof(str2));  
的输出结果是

〖A〗5 6

〖B〗5 5

〖C〗6 6

〖D〗4 6

本题主要考查了指向字符串的指针和字符数组, str1为一个字符指针, 所以sizeof为4, str2为字符数组, 其中包含6个字符, 所以答案为4 6。

【33】以下叙述中正确的是

〖A〗任何情况下都不能用函数名作为实参

〖B〗函数既可以直接调用自己, 也可以间接调用自己

〖C〗函数的递归调用不需要额外开销, 所以效率很高

〖D〗简单递归不需要明确的结束递归的条件

在某些情况下, 函数名可以作为实参传递, 所以A选项错误。  
函数递归调用使用栈区来递归, 需要额外开销, 并且效率不高, C选项错误。  
递归调用需要明确指出递归结束条件, 否则就陷入死循环了。

【34】以下叙述中正确的是

〖A〗函数的形式参数不属于局部变量

〖B〗对于变量而言, "定义"和"说明"这两个词实际上是同一个意思

〖C〗在复合语句中不能定义变量

〖D〗全局变量的存储类别可以是静态类

声明变量不分配空间, 定义变量要分配空间, 所以B选项错误。  
符合语句中可以定义变量, 所以C选项错误。  
函数的形参属于局部变量, 所以A选项错误。

【35】以下叙述中正确的是

〖A〗在一个程序中, 允许使用任意数量的#include命令行

〖B〗在包含文件中, 不得再包含其他文件

〖C〗#include命令行不能出现在程序文件的中间

〖D〗虽然包含文件被修改了, 包含该文件的源程序也可以不重新进行编译和连接

本题考查预处理中文件包含的概念, 包含文件中可以包含其他文件, B选项错误, #include可以出现在程序文件的中间, C选项错误, 包含文件被修改了, 包含该文件的源程序必须重新进行编译和连接。

【36】以下叙述中正确的是

〖A〗函数的返回值不能是结构体类型

〖B〗在调用函数时, 可以将结构体变量作为实参传给函数

〖C〗函数的返回值不能是结构体指针类型

〖D〗结构体数组不能作为参数传给函数

本题考查结构体变量作为函数参数, 函数返回可以是结构体类型, 也可以是结构体类型的指针, 结构体数组可以作为参数传给函数。

【37】以下叙述中正确的是

〖A〗结构体数组名不能作为实参传给函数

〖B〗结构体变量的地址不能作为实参传给函数

〖C〗结构体中可以含有指向本结构体的指针成员

〖D〗即使是同类型的结构体变量, 也不能进行整体赋值

本题考查结构体变量作为函数参数, 结构体变量的地址可以作为参数传给函数, 结构体数组名就是一个地址, 可以传给函数, 如果是同类型的结构体变量, 可以整体赋值。

【38】设有如下的说明和定义

复制代码

C

|  |  |
| --- | --- |
|  | struct { |
|  | int a; |
|  | char \*s; |
|  | } x, \*p = &x; |
|  | x.a = 4; |
|  | x.s = "hello"; |

则以下叙述中正确的是

〖A〗(p++)->a与p++->a都是合语法的表达式, 但二者不等价

〖B〗语句++p->a; 的效果是使p增1

〖C〗语句++p->a; 的效果是使成员a增1

〖D〗语句\*p->s++; 等价于(\*p)->s++;

本题考查结构体变量的引用, ++p->a的效果是使成员a增1, p++->a不合法, \*p->s++为字符e, 与(\*p)->s++不等价。

【39】下面关于位运算符的叙述, 正确的是

〖A〗||表示"按位或"的运算

〖B〗#表示"按位异或"的运算

〖C〗&表示"按位与"的运算

〖D〗~表示"按位异或"的运算

本题考查位运算。  
^为按位异或, |为按位或, ~表示按位取反。  
所以C选项正确。

【40】下面关于"EOF"的叙述, 正确的是

〖A〗EOF的值等于0

〖B〗EOF是在库函数文件中定义的符号常量

〖C〗文本文件和二进制文件都可以用EOF作为文件结束标志

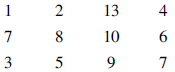
〖D〗对于文本文件, fgetc函数读入最后一个字符时, 返回值是EOF

在C语言中, 或更精确地说成C标准函数库中表示文件结束符(end of file)。  
在while循环中以EOF作为文件结束标志, 这种以EOF作为文件结束标志的文件, 必须是文本文件。  
在文本文件中, 数据都是以字符的ASCII代码值的形式存放。  
我们知道, ASCII代码值的范围是0 ~ 255, 不可能出现-1, 因此可以用EOF作为文件结束标志。

**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：在3×4的矩阵中找出在行上最大、在列上最小的那个元素, 若没有符合条件的元素则输出相应信息。  
例如, 有下列矩阵：



程序执行结果为：find : a[2][2] = 9  
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) 0 |
|  | (3) i++ |

本题是在矩阵中找出在行上最大、在列上最小的那个元素。  
填空1:找出行上最大的数, 并将该数的列数j保存在c中, 所以应填j。  
填空2:使用while循环语句和控制变量find, 如果该数不是列中的最小数, 那么把find置0, 所以应填0。  
填空3 : i是while的控制变量, 所以每做一次循环, 该数值要加1, 所以应填i++。

【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：根据整形参m的值, 计算如下公式的值。



例如, 若m中的值为5, 则应输出0.536389。  
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) for (i = 2; i <= m; i++) |
|  | (2) y -= 1.0 / (i \* i); |

(1) 使用for循环计算公式, 必须计算到m, 所以应改为for(i = 2; i <= m; i++)。  
(2) 在除法运算中, 如果除数和被除数都是整数, 那么所除结果也是整数, 因此应改为y -= 1.0 / (i \* i)。

【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, m个人的成绩存放在score数组中, 请编写函数fun, 它的功能是：将低于平均分的人数作为函数值返回, 将低于平均分的分数放在below所指的数组中。  
例如, 当score数组中的数据为10、20、30、40、50、60、70、80、90时, 函数返回的人数应该是4, below中的数据应为10、20、30、40。  
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(int score[], int m, int below[]) { |
|  | int i, j = 0; |
|  | float av = 0.0; |
|  | for (i = 0; i < m; i++) |
|  | av = av + score[i] / m; |
|  | /\*求平均值\*/ |
|  | for (i = 0; i < m; i++) |
|  | if (score[i] < av) |
|  | /\*如果分数低于平均分, 则将此分数放入below数组中\*/ |
|  | below[j++] = score[i]; |
|  | return j; |
|  | /\*返回低于平均分的人数\*/ |
|  | } |

要计算低于平均分的人数, 首先应该求出平均分, 然后通过for循环语句和if条件语句找出低于平均分的分数。  
该题第1个循环的作用是求出平均分av, 第2个循环的作用是找出低于平均分的成绩记录并存入below数组中。