C语言题目

2021年2月19日 17:28

(A)

第1题（单选题）

题目名称:

能把函数处理结果的二个数据返回给主调函数，在下面的方法中不正确的是：（）

题目内容:

A.return 这二个数
B.形参用数组
C.形参用二个指针
D.用二个全局变量

(C)

题目名称:

关于函数调用说法不正确的是：（）

题目内容:

A.函数可以传值调用，传值调用的时候形参是实参的一份临时拷贝
B.函数可以传址调用，传址调用的时候，可以通过形参操作实参
C.函数可以嵌套定义，但是不能嵌套调用
D.函数可以嵌套调用，但是不能嵌套定义

(B)

题目名称:

在函数调用时，以下说法正确的是：（）

题目内容:

A.函数调用后必须带回返回值
B.实际参数和形式参数可以同名
C.函数间的数据传递不可以使用全局变量
D.主调函数和被调函数总是在同一个文件里

(B)

题目名称:

关于函数的声明和定义说法正确的是：（）

题目内容:

A.函数的定义必须放在函数的使用之前
B.函数必须保证先声明后使用
C.函数定义在使用之后，也可以不声明
D.函数的声明就是说明函数是怎么实现的

(C)

题目名称:

关于实参和形参描述错误的是：（）

题目内容:

A.形参是实参的一份临时拷贝
B.形参是在函数调用的时候才实例化，才开辟内存空间
C.改变形参就是改变实参
D.函数调用如果采用传值调用，改变形参不影响实参

(B)

(v1,v2）的结果是v2

题目名称:

函数调用exec((vl，v2)，(v3，v4)，v5，v6);中，实参的个数是：（）

题目内容:

A.3
B.4
C.5
D.6

(B)

题目名称:

以下关于函数设计不正确的说法是：（）

题目内容:

A.函数设计应该追求高内聚低耦合
B.要尽可能多的使用全局变量
C.函数参数不易过多
D.设计函数时，尽量做到谁申请的资源就由谁来释放

(C)

题目名称:

关于C语言函数描述正确的是：（）

题目内容:

A.函数必须有参数和返回值
B.函数的实参只能是变量
C.库函数的使用必须要包含对应的头文件
D.有了库函数就不需要自定函数了

(C)

题目名称:

C语言规定，在一个源程序中，main函数的位置（）

题目内容:

A.必须在最开始
B.必须在库函数的后面
C.可以任意
D.必须在最后

(D)

题目名称:

以下叙述中不正确的是：（）

题目内容:

A.在不同的函数中可以使用相同名字的变量
B.函数中的形式参数是在栈中保存
C.在一个函数内定义的变量只在本函数范围内有效
D.在一个函数内复合语句中定义的变量在本函数范围内有效（复合语句指函数中的成对括号构成的代码）

(C)

题目名称:

关于一维数组初始化，下面哪个定义是错误的？（）

题目内容:

A.int arr[10] = {1,2,3,4,5,6};
B.int arr[] = {1,2,3,4,5,6};
C.int arr[] = (1,2,3,4,5,6);
D.int arr[10] = {0};

(B)

题目名称:

以下能对二维数组a进行正确初始化的语句是：（）

题目内容:

A.int ta[2][]={{0,1,2},{3,4,5}};
B.int ta[][3]={{0,1,2},{3,4,5}};
C.int ta[2][4]={{0,1,2},{3,4},{5}};
D.int ta[][3]={{0,,2},{},{3,4,5}};

(C)

题目名称:

定义了一维 int 型数组 a[10] 后，下面错误的引用是：（）

题目内容:

A.a[0] = 1;
B.a[0] = 5*2;
C.a[10] = 2;
D.a[1] = a[2] * a[0];

(B)

题目名称:

若定义int a[2][3]={1,2,3,4,5,6};则值为4的数组元素是( )

题目内容:

A.a[0][0]
B.a[1][0]
C.a[1][1]
D.a[2][1]

(B)

题目名称:

下面代码的结果是：( )

```
#include <stdio.h>
int main()
{
      int arr[] = {1,2,(3,4),5};
      printf("%d\n", sizeof(arr));
      return 0;
}
```

题目内容:

A.4
B.16
C.20
D.5

（A)

\n算一个字符的大小

题目名称:

下面代码的结果是：( )

```
#include <stdio.h>
int main()
{
      char str[] = "hello bit";
      printf("%d %d\n", sizeof(str), strlen(str));
        return 0;
}
```

题目内容:

A.10 9
B.9 9
C.10 10
D.9 10

（C)

题目名称:

给出以下定义：

```
char acX[] = "abcdefg";
char acY[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g' };
```

以下说法正确的是( )

题目内容:

A.数组acX和数组acY等价
B.数组acX和数组acY的长度相同
C.数组acX的长度大于数组acY的长度
D.数组acX的长度小于数组acY的长度

（D)

题目名称:

关于一维数组描述不正确的是:( )

题目内容:

A.数组的下标是从0开始的
B.数组在内存中是连续存放的
C.数组名表示首元素的地址
D.随着数组下标的由小到大，地址由高到低

(D)

题目名称:

关于表达式求值说法不正确的是: ( )

题目内容:

A.表达式求值先看是否存在整形提升或算术转换，再进行计算
B.表达式真正计算的时候先看相邻操作符的优先级决定先算谁
C.相邻操作符的优先级相同的情况下，看操作符的结合性决定计算顺序
D.只要有了优先级和结核性，表达式就能求出唯一值

(D)

题目名称:

下面代码的结果是: ( )

```
#include <stdio.h>
int main()
{
      int i = 1;
      int ret = (++i)+(++i)+(++i);
      printf("ret = %d\n", ret);
        return 0;
}
```

题目内容:

A.10
B.12
C.9
D.程序错误

(C)

题目名称:

关于指针的概念，错误的是:( )

题目内容:

A.指针是变量，用来存放地址
B.指针变量中存的有效地址可以唯一指向内存中的一块区域
C.野指针也可以正常使用
D.局部指针变量不初始化就是野指针

(C)

题目名称：

以下系统中，int类型占几个字节，指针占几个字节，操作系统可以使用的最大内存空间是多大：( )

题目内容：

A.32位下：4,4,2^32 64位下：8,8,2^64
B.32位下：4,4,不限制 64位下：4,8,不限制
C.32位下：4,4,2^32 64位下：4,8,2^64
D.32位下：4,4,2^32 64位下：4,4,2^64

（B）

short是两个字节，i每加2才算往后了一个下标

下面代码的结果是：（）

```
#include <stdio.h>
int main()
{
    int arr[] = {1,2,3,4,5};
    short *p = (short*)arr;
    int i = 0;
    for(i=0; i<4; i++)
    {
        *(p+i) = 0;
    }

    for(i=0; i<5; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

题目内容：

A.1 2 3 4 5
B.0 0 3 4 5
C.0 0 0 0 5
D.1 0 0 0 0

(C)

题目名称：

下面代码输出的结果是：（）

```
#include <stdio.h>
int main()
{
        int a = 0x11223344;
    char *pc = (char*)&a;
    *pc = 0;
    printf("%x\n", a);
    return 0;
}
```

题目内容：

A.00223344
B.0
C.11223300
D.112233

（A）

函数sizeof()返回的数字是无符号数，有符号数-1转成无符号数很大，比4大

题目名称:

下面代码的结果是:

```
#include <stdio.h>
int i;
int main()
{
	i--;
	if (i > sizeof(i))
	{
		printf(">\n");
	}
	else
	{
		printf("<\n");
	}
	return 0;
}
```

题目内容:

A.>
B.<
C.不输出
D.程序有问题

(B)

=优先级大于,

+=优先级低于+

++a是a先+1再运算

a++是a先运算在+1

下面代码的结果是:()

```
#include <stdio.h>
int main()
{
	int a, b, c;
	a = 5;
	c = ++a;
	b = ++c, c++, ++a, a++;
	b += a++ + c;
	printf("a = %d b = %d c = %d\n:", a, b, c);
	return 0;
}
```

题目内容:

A.a = 8 b = 23 c = 8
B.a = 9 b= 23 c = 8
C.a = 9 b = 25 c = 8
D.a = 9 b = 24 c = 8

(A)

下面哪个是位操作符: ()

题目内容:

A.&
B.&&
C.||
D.!

(D)

题目名称:

根据下面递归函数：调用函数Fun(2)，返回值是多少（D）

```
int Fun(int n)
{
    if(n==5)
        return 2;
    else
        return 2*Fun(n+1);
}
```

题目内容:

A.2
B.4
C.8
D.16

(C)

题目名称:

关于递归的描述错误的是：（）

题目内容:

A.存在限制条件，当满足这个限制条件的时候，递归便不再继续
B.每次递归调用之后越来越接近这个限制条件
C.递归可以无限递归下去
D.递归层次太深，会出现栈溢出现象

(C)

题目名称:

下列程序段的输出结果为（）

```
unsigned long pulArray[] = {6,7,8,9,10};
unsigned long *pulPtr;
pulPtr = pulArray;
*(pulPtr + 3) += 3;
printf("%d,%d\n",*pulPtr, *(pulPtr + 3));
```

题目内容:

A.9,12
B.6,9
C.6,12
D.6,10

(B)

题目名称:

关于二级指针描述描述正确的是：（）

题目内容:

A.二级指针也是指针，只不过比一级指针更大
B.二级指针也是指针，是用来保存一级指针的地址
C.二级指针是用来存放数组的地址
D.二级指针的大小是4个字节

(C)

题目名称:

下面关于指针运算说法正确的是：（）

题目内容:

A.整形指针+1，向后偏移一个字节
B.指针-指针得到是指针和指针之间的字节个数
C.整形指针解引用操作访问4个字节
D.指针不能比较大小

(A)

题目名称:

下面哪个是指针数组：（）

题目内容:

A. int* arr[10];
B.int * arr[];
C.int **arr;
D.int (*arr)[10];

(D)

题目名称:

如有以下代码:

```
struct student
{
    int num;
    char name[32];
    float score;
}stu;
```

则下面的叙述不正确的是：( )

题目内容:

A.struct 是结构体类型的关键字
B.struct student 是用户定义的结构体类型
C.num, score 都是结构体成员名
D.stu 是用户定义的结构体类型名

(B)

题目名称:

下面程序要求输出结构体中成员a的数据,以下不能填入横线处的内容是( )

```
#include < stdio.h >
struct S
{
    int a;
    int b;
};
int main( )
{
    struct S a, *p=&a;
    a.a = 99;
    printf( "%d\n", _____ );
    return 0;
}
```

题目内容:

A.a.a
B.*p.a
C.p->a
D.(*p).a

(C)

下面程序的输出结果是:

```
struct stu
{
    int num;
    char name[10];
    int age;
};

void fun(struct stu *p)
{
    printf("%s\n", (*p).name);
    return;
}

int main()
{
    struct stu students[3] = { { 9801, "zhang", 20 },
                               { 9802, "wang", 19 },
                               { 9803, "zhao", 18 }
                             };
    fun(students + 1);
    return 0;
}
```

题目内容:

A.zhang
B.zhao
C.wang
D.18

(D)

题目名称:

结构体访问成员的操作符不包含: ( )

题目内容:

A.. 操作符
B.-> 操作符
C.* 解引用操作符
D.sizeof

(C)

栈溢出是具体错误不是一个分类

题目名称:

C程序常见的错误分类不包含: ( )

题目内容:

A.编译错误
B.链接错误
C.栈溢出
D.运行时错误

(A)

Ctrl+F5才是开始执行不调试

F5是开始调试

题目名称:

关于VS调试快捷键说法错误的是: ( )

题目内容:

A.F5-是开始执行，不调试
B.F10-是逐过程调试，遇到函数不进入函数
C.F11-是逐语句调试，可以观察调试的每个细节
D.F9是设置断点和取消断点

(D)

Release版本不可以调试

题目名称:

关于Debug和Release的区别说法错误的是: ( )

题目内容:

A.Debug被称为调试版本，程序调试找bug的版本
B.Release被称为发布版本，测试人员测试的就是Release版本
C.Debug版本包含调试信息，不做优化。
D.Release版本也可以调试，只是往往会优化，程序大小和运行速度上效果最优

(C)

题目名称:

以下关于指针的说法,正确的是( )

题目内容:

A.int *const p`与`int const *p等价
B.const int *p`与`int *const p等价
C.const int *p`与`int const *p等价
D.int *p[10]`与`int (*p)[10]等价

(C)

题目名称:

语言中哪一种形式声明了一个指向char类型变量的指针p，p的值不可修改，但p指向的变量值可修改？( )

题目内容:

A.const char *p
B.char const *p
C.char*const p
D.const char *const p

数组下标溢出了，数组大小是固定的，循环的时候下标超出数组范围了

题目名称:

程序死循环解释

题目内容:

VS开发环境调试下面的代码，画图解释下面代码的问题

```
#include <stdio.h>
int main()
{
	int i = 0;
	int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
	for(i=0; i<=12; i++)
	{
		arr[i] = 0;
		printf("hello bit\n");
	}
	return 0;
}
```

(D)

题目名称:

原码、反码、补码说法错误的是（）

题目内容:

A.一个数的原码是这个数直接转换成二进制
B.反码是原码的二进制符号位不变，其他位按位取反
C.补码是反码的二进制加1
D.原码、反码、补码的最高位是0表示负数，最高位是1表示正数

(B)

题目名称:

关于大小端字节序的描述正确的是（）

题目内容:

A.大小端字节序指的是数据在电脑上存储的二进制位顺序
B.大小端字节序指的是数据在电脑上存储的字节顺序
C.大端字节序是把数据的高字节内容存放到高地址，低字节内容存放在低地址处
D.小段字节序是把数据的高字节内容存放到低地址，低字节内容存放在高地址处

（C）

没怎么懂

题目名称:

程序的执行结果为（）

```
int main()
{
    unsigned char a = 200;
    unsigned char b = 100;
    unsigned char c = 0;
    c = a + b;
    printf("%d %d", a+b,c);
    return 0;
}
```

题目内容:

A.300 300
B.44 44
C.300 44
D.44 300

（A）

0x1234存储

小端34 12 00 00

大端00 00 12 34

强转成char，取一个字节00

题目名称:

```
unsigned int a= 0x1234; unsigned char b=*(unsigned char *)&a;
```

在32位大端模式处理器上变量b等于（）

题目内容:

A.0x00
B.0x12
C.0x34
D.0x1234

（C）

-1,-2.....-128,127,126.....3,2,1,0,-1,-2.....

strlen到\0结束,所以是255

题目名称:

下面代码的结果是（）

```
int main()
{
    char a[1000] = {0};
    int i=0;
    for(i=0; i<1000; i++)
    {
    a[i] = -1-i;
    }
    printf("%d",strlen(a));
    return 0;
}
```

题目内容:

A.1000
B.999
C.255
D.256

```
//烧香问题
//有一个种香，材质不均匀,但是每一根这样的香，燃烧完恰好是1个小时
//给你2跟香，帮我确定一个15分钟的时间段
```

香1两头都点燃，香2点燃一头

等香1烧完，就是过了半小时，此时香2还有半小时的量

这时点燃香2另一头，剩下的时间就是半小时

```
//赛马问题：有36匹马，6个跑道，没有计时器，请赛马确定，36匹马中的前三名。
//请问最少比赛几次?
```

先比六次，六次的各个第一名再比一次，第七次的第一名的第一场比赛的前三名+第七次第二名的第一场比赛的前两名+第七次的第三名，比第八场，前三就是这场比赛的前三名

（A）

题目名称:

下面关于"指针"的描述不正确的是：（）

题目内容:

A.当使用free释放掉一个指针内容后,指针变量的值被置为NULL
B.32位系统下任何类型指针的长度都是4个字节
C.指针的数据类型声明的是指针实际指向内容的数据类型
D.野指针是指向未分配或者已经释放的内存地址

（C）

题目名称:

关于下面代码描述正确的是: ( )

```
char* p = "hello bit";
```

题目内容:

A.把字符串hello bit存放在p变量中
B.把字符串hello bit的第一个字符存放在p变量中
C.把字符串hello bit的第一个字符的地址存放在p变量中
D.*p等价于hello bit

(C)

题目名称:

关于数组指针的描述正确的是: ( )

题目内容:

A.数组指针是一种数组
B.数组指针是一种存放数组的指针
C.数组指针是一种指针
D.指针数组是一种指向数组的指针

(C)

题目名称:

下面哪个是数组指针 ( )

题目内容:

A.int** arr[10]
B.int (*arr[10])
C.char *(*arr)[10]
D.char(*)arr[10]

(D)

&arr是数组地址

题目名称:

下面哪个代码是错误的? ( )

```
#include <stdio.h>
int main()
{
    int *p = NULL;
    int arr[10] = {0};
    return 0;
}
```

题目内容:

A.p = arr;
B.int (*ptr)[10] = &arr;
C.p = &arr[0];
D.p = &arr;

(A)

题目名称:

下面代码关于数组名描述不正确的是（）

```
int main()
{
    int arr[10] = {0};
    return 0;
}
```

题目内容:

A.数组名arr和&arr是一样的

B.sizeof(arr)，arr表示整个数组

C.&arr，arr表示整个数组

D.除了sizeof(arr)和&arr中的数组名，其他地方出现的数组名arr，都是数组首元素的地址。

(C)

题目名称:

如何定义一个int类型的指针数组，数组元素个数为10个：()

题目内容:

A.int a[10]

B.int (*a)[10]

C.int *a[10];

D.int (*a[10])(int);

(C)

题目名称:

下面代码的执行结果是（）

```
#include <stdio.h>
int main()
{
    char str1[] = "hello bit.";
    char str2[] = "hello bit.";
    char *str3 = "hello bit.";
    char *str4 = "hello bit.";
    if(str1 == str2)
        printf("str1 and str2 are same\n");
    else
        printf("str1 and str2 are not same\n");

    if(str3 == str4)
        printf("str3 and str4 are same\n");
    else
        printf("str3 and str4 are not same\n");

    return 0;
}
```

题目内容:

A.str1 and str2 are same str3 and str4 are same

B.str1 and str2 are same str3 and str4 are not same

C.str1 and str2 are not same str3 and str4 are same

D.str1 and str2 are not same str3 and str4 are not same

(C)

题目名称:

下面哪个是函数指针？( )

题目内容:

A.int* fun(int a, int b);

B.int(*)fun(int a, int b);

C.int (*fun)(int a, int b);

D.(int *)fun(int a, int n);

(A)

题目名称:

定义一个函数指针，指向的函数有两个int形参并且返回一个函数指针，返回的指针指向一个有一个int形参且返回int的函数？下面哪个是正确的？ ( )

题目内容:

A.int (*(*F)(int, int))(int)

B.int (*F)(int, int)

C.int (*(*F)(int, int))

D.*(*F)(int, int)(int)

(B)

题目名称:

在游戏设计中，经常会根据不同的游戏状态调用不同的函数，我们可以通过函数指针来实现这一功能，下面哪个是：一个参数为int *，返回值为int的函数指针 ( )

题目内容:

A.int (*fun)(int)

B.int (*fun)(int *)

C.int* fun(int *)

D.int* (*fun)(int *)

(C)

题目名称:

声明一个指向含有10个元素的数组的指针，其中每个元素是一个函数指针，该函数的返回值是int，参数是int*，正确的是 ( )

题目内容:

A.(int *p[10])(int*)

B.int [10]*p(int *)

C.int (*(*p)[10])(int *)

D.int ((int *)[10])*p

(B)

题目名称:

设有以下函数void fun(int n,char *s)(……),则下面对函数指针的定义和赋值均是正确的是： ( )

题目内容:

A.void (*pf)(int，char); pf=&fun;

B.void (*pf)(int n，char *s); pf=fun;

C.void *pf(); *pf=fun;

D.void *pf(); pf=fun;

(D)

题目名称:

关于回调函数描述错误的是（）

题目内容:

A.回调函数就是一个通过函数指针调用的函数
B.回调函数一般通过函数指针实现
C.回调函数一般不是函数的实现方调用，而是在特定的场景下，由另外一方调用。
D.回调函数是调用函数指针指向函数的函数。

(B，D)多选题

首元素是char*类型

首元素地址是char**

题目名称:

下面test函数设计正确的是：（）

```
char* arr[5] = {"hello", "bit"};
test(arr);
```

题目内容:

A.void test(char* arr);
B.void test(char** arr);
C.void test(char arr[5]);
D.void test(char* arr[5]);

(C)

题目名称:

下面代码中print_arr函数参数设计哪个是正确的？（）

```
int arr[3][5] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
print_arr(arr, 3, 5);
```

题目内容:

A.void print_arr(int arr[][],int row, int col);
B.void print_arr(int* arr, int row, int col);
C.void print_arr(int (*arr)[5], int row, int col);
D.void print_arr(int (*arr)[3], int row, int col);

（B）

题目名称:

下面程序的结果是：（）

```
int main()
{
    int a[5] = {5, 4, 3, 2, 1};
    int *ptr = (int *)(&a + 1);
    printf( "%d,%d", *(a + 1), *(ptr - 1));
    return 0;
}
```

题目内容:

A.5, 1
B.4, 1
C.4, 2
D.5, 2

（A）

题目名称:

下面程序的结果是: ( )

```
int main()
{
    int aa[2][5] = {10,9,8,7,6,5,4,3,2,1};
    int *ptr1 = (int *)(&aa + 1);
    int *ptr2 = (int *)(*(aa + 1));
    printf( "%d,%d", *(ptr1 - 1), *(ptr2 - 1));
    return 0;
}
```

题目内容:

A.1, 6
B.10, 5
C.10, 1
D.1, 5

(C)

题目名称:

在VS2013下，这个结构体所占的空间大小是（）字节

```
typedef struct{
    int  a;
    char  b;
    short  c;
    short  d;
}AA_t;
```

题目内容:

A.16
B.9
C.12
D.8

(C)

题目名称:

在32位系统环境，编译选项为4字节对齐，那么sizeof(A)和sizeof(B)是（）

```
struct A
{
  int a;
  short b;
  int c;
  char d;
};
struct B
{
  int a;
  short b;
  char c;
  int d;
};
```

题目内容:

A.16,16
B.13,12
C.16,12
D.11,16