# C语言题

## 第一章

1.一个用C语言编写的源程序，要在计算机上运行，应该经历哪四个过程？

【1】编辑 【2】编译 【3】连接 【4】运行

2.任何C语言程序都从\_\_\_\_开始执行。

主函数

3.一个程序由\_\_\_\_\_源程序文件组成

一个或多个

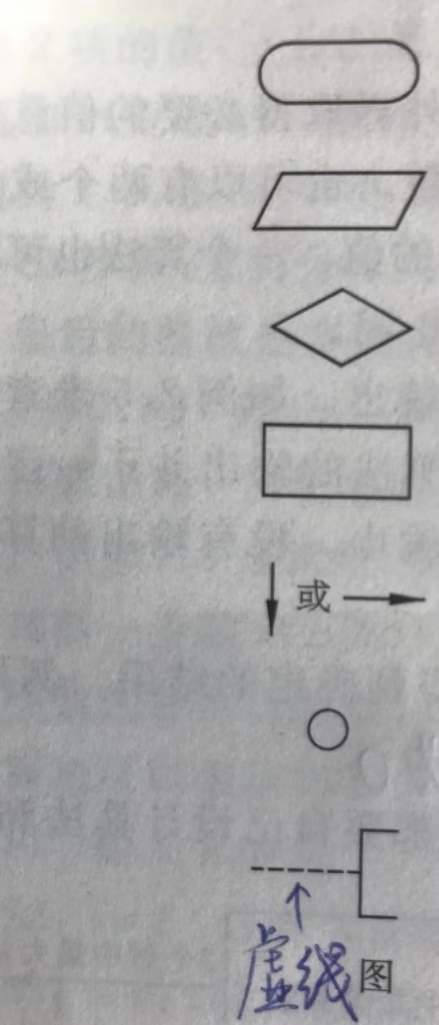
4.一个源程序包括哪三部分？

1. 预处理指令。如：#include<stdio.h>，#define PI 3.1415926
2. 全局声明。
3. 函数定义。

5.#include指令的两种形式？

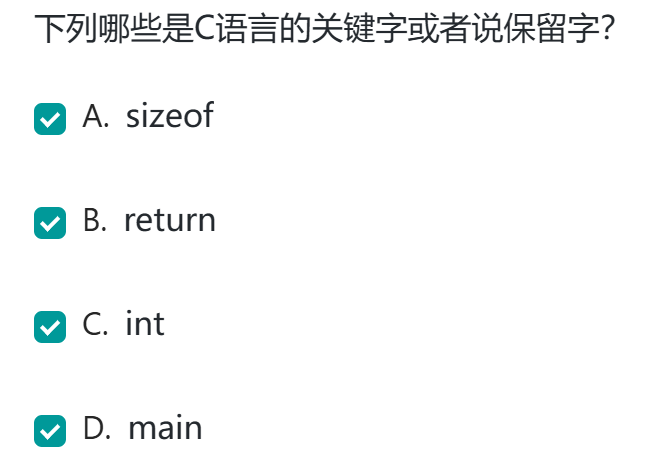
1. **尖括号**形式。（**标准方式：**在编译系统的子目录中寻找）
2. **双撇号**形式。（先在**用户**的当前目录中寻找，再按标准方式寻找）

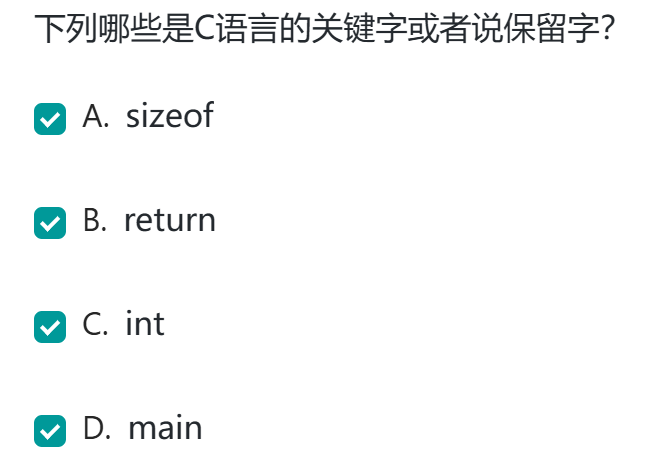
6.流程图的符号含义



起止框 输入输出框 判断框 处理框 流程线 连接点 注释框

7.





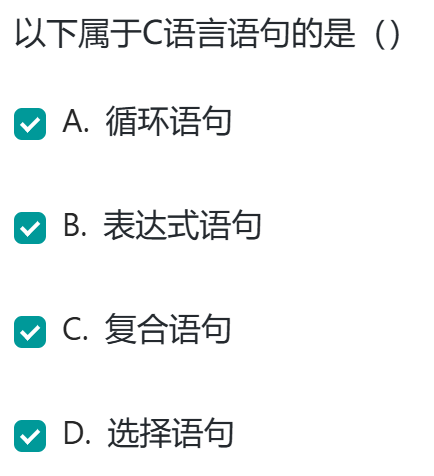
A,B,C

关键字（或称为保留字）是由**C 语言标准预定义**的，具有特殊含义的单词，

它们不能用作标识符（如变量名、函数名等）。

注意：D.main不是关键字。但它是一个**标识符。**

8.



记

## 第二章 数据类型、常量与变量

1.C语言中规定，标识符（包括：变量名、函数名、数组名、类型名、文件名等）只能由哪三种字符组成？且第一个字符必须是什么？

哪三种字符：字母、数字、下画线

第一个字符：必须是字母，或下画线。不能是数字

2.’a’ 在内存中占多少字节？“a”在内存中占多少字节？

1. 2

3.若int类型的数据长度为2字节，则int类型数据的取值范围是？unsigned int类型数据的取值范围是？

int:[-32768,32767] unsigned int:[0,65535]

计算公式：

已知：某整型数据长度为n字节，一个字节有8个二进位

则该整型数据共占(8\*n)个二进位

1. 有符号的整型：
2. 无符号的整型：

4.请选出下面的合法常量\_\_\_\_\_

(A)8e2.3

(B).e7

(C)13e-3

(D)1.23e6

(E)0.0056E2

C,D,E

指数型的常量: E(e)前必须有数字（整数或小数），E(e)后必须有**整数**（正负皆可）

5. 请选出下面合法的八进制常量\_\_\_\_\_

(A)099

(B)057

(C)0121

(D)256

(E)-0777

B,C,E

C语言中的八进制数：以0开头，后面的数字为0~7。如果是负数，则在0前面添上“-”

6. 请选出下面合法的二进制常量\_\_\_\_\_

(A)0b101

(B)0B122

(C)1001

(D)0B11

A,D

C语言中的二进制数：以0b或0B开头，后面的数字为0或1

7.请选出下面合法的十六进制常量\_\_\_\_\_

(A)0x2A

(B)0xxx

(C)0XFFFF

(D)3b3f

(E)0x1g

A,C

C语言中的十六进制数：以0x或0X开头，后面为0~9以及a~f或A~F

8.请选出下面合法的八进制转义字符\_\_\_\_\_

(A)’\123’

(B)’\890’

(C)’\7’

(D)’\12’

(E)’\1314’

(F)’165’

A,C,D

C语言中的八进制转义字符:’\o’,’\oo’,’\ooo’ (o表示一个0~7的数字)

注意：八进制转义字符中的数字不能超过3个！

9. 请选出下面合法的十六进制转义字符\_\_\_\_\_

(A)’\x131’

(B)’\xH’

(C)’\xff’

(D)’\x61’

C,D

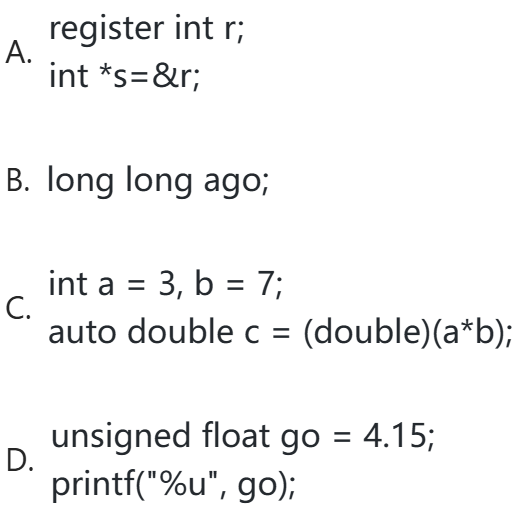
C语言中的八进制转义字符:’\xh’,’\xhh’(h表示一个0~9的数字或者a~f，A~F)

注意：除去x之外，十六进制转义字符中的数字及字母不能超过2个！

## 变量的限定符

1.





B,C

解释：

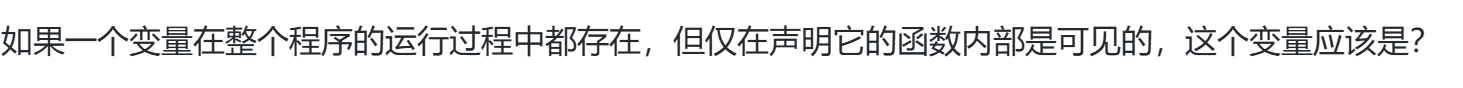
A．对于使用 register 声明的变量，不能取其地址。所以在 int \*s = &r; 这部分，r 是 register 类型的变量，不能取地址 &r，这会导致编译错误。

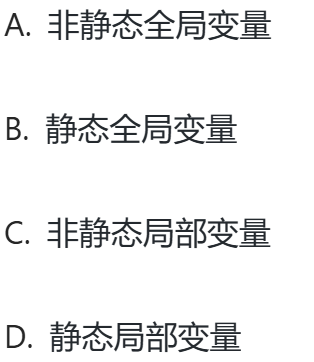
报错信息如下：



D．浮点数类型float, double，不能用unsigned修饰。因为，实际存放时，浮点数都是必定有符号的（IEE754标准）

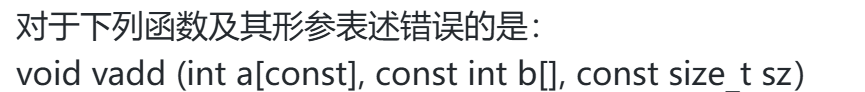
2.

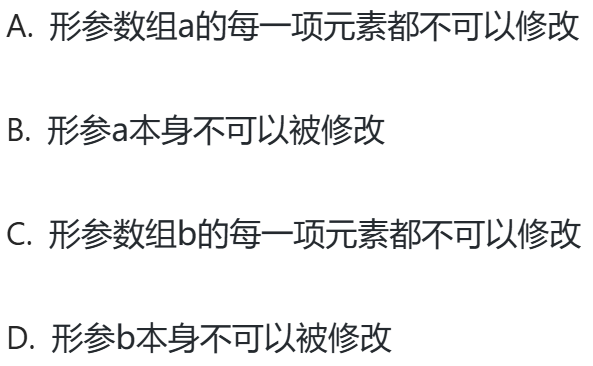




D

3.





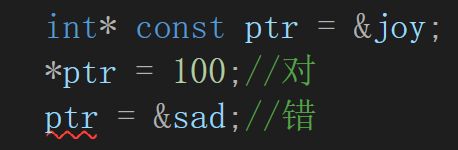
A D

const限定符

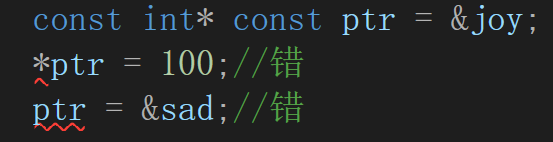
1.指针***不能***改变所指向的变量的值，但指针本身的值***可以***改变



2. 指针***能***改变所指向的变量的值，但指针本身的值***不可以***改变

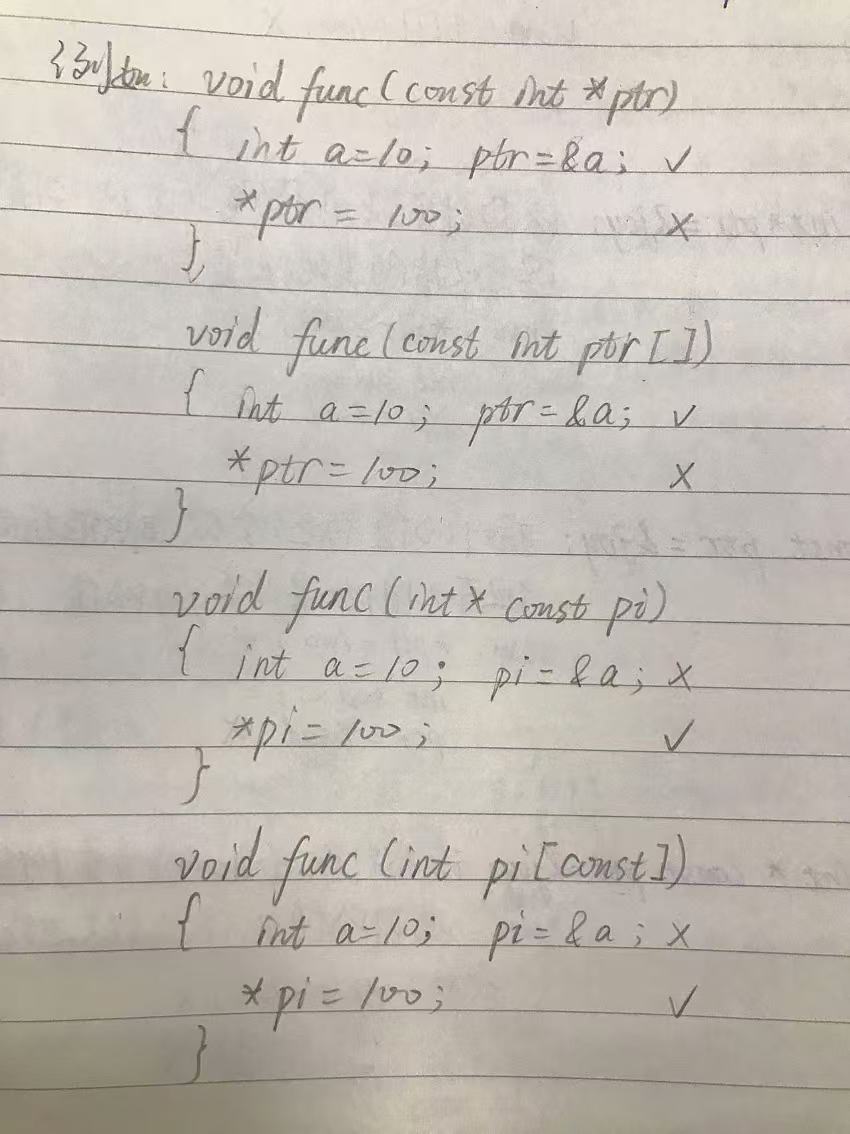


1. 都不能



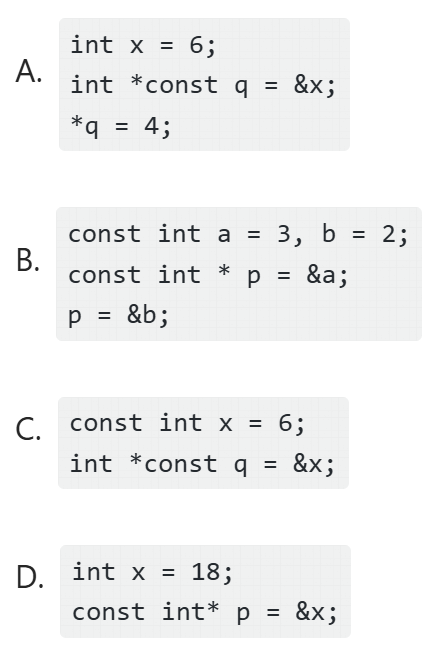
在函数中，也是同理

如下图所示



4.





C

A．int\* const q 是一个指针，指针本身的值不能修改，但是指针所指向元素的值可以修改

B．const int\* p 是一个指针，指针本身的值能修改，但是指针所指向元素的值***不***可以修改

const int b 是一个普通变量，b的值不能修改，

p=&b (p指向b)

“p所指向元素的值***不***可以修改”与“b的值不能修改”相符，所以正确

C . int\* const q 是一个指针，指针本身的值不能修改，但是指针所指向元素的值可以修改

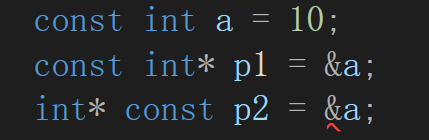
const int x 是一个普通变量，x的值不能修改

q=&x (q指向x)

“q所指向元素的值可以修改”与“x的值不能修改”相矛盾，所以错误！！！

D．正确

B，C的区别可见于下图



## 第三章 数据的运算与输入输出

1.以下正确的表达式是\_\_\_

(A)10++

(B)(x+y)++

(C)++(a1-b2)

(D)d++

D

自增自减都是对**变量**进行的，A,B,C都不是变量

A是常量

**B、C是表达式**

2.已知int a=6，则执行“a+=a-=a\*a;”语句后，a=？

-60

“x+=y”的含义是：x=x+y

因此，“a+=a-=a\*a;”语句相当于：a=a+(a=a-a\*a)

3.执行语句“printf(“The program \’s name is c:\\tools\book.txt”);”后的输出是：

（A）The program ’s name is c:\tools\book.txt

（B）The program ’s name is c:\toolook.txt

B

转义字符“\’”是输出单撇号（‘）

转义字符“\\“是输出一个反斜杠（\）

转义字符“\b”是**输出退格**，将当前位置向前移一列

4.下面程序运行后的输出是

int main()

{

int x = 100;

printf("%d\n", 10+x++);

printf("%d\n", 10+(++x));

return 0;

}

110

112

注意：不能将printf的两行分开来看，两个语句是在同一个程序中的

5．赋值表达式。

判断下列赋值表达式执行后各个变量的值

注意：以下这些赋值语句都**不能**用于定义变量的同时初始化！！！

1. a=b=c=5

a=5,b=5,c=5

1. a=(b=2\*3)

a=6,b=6

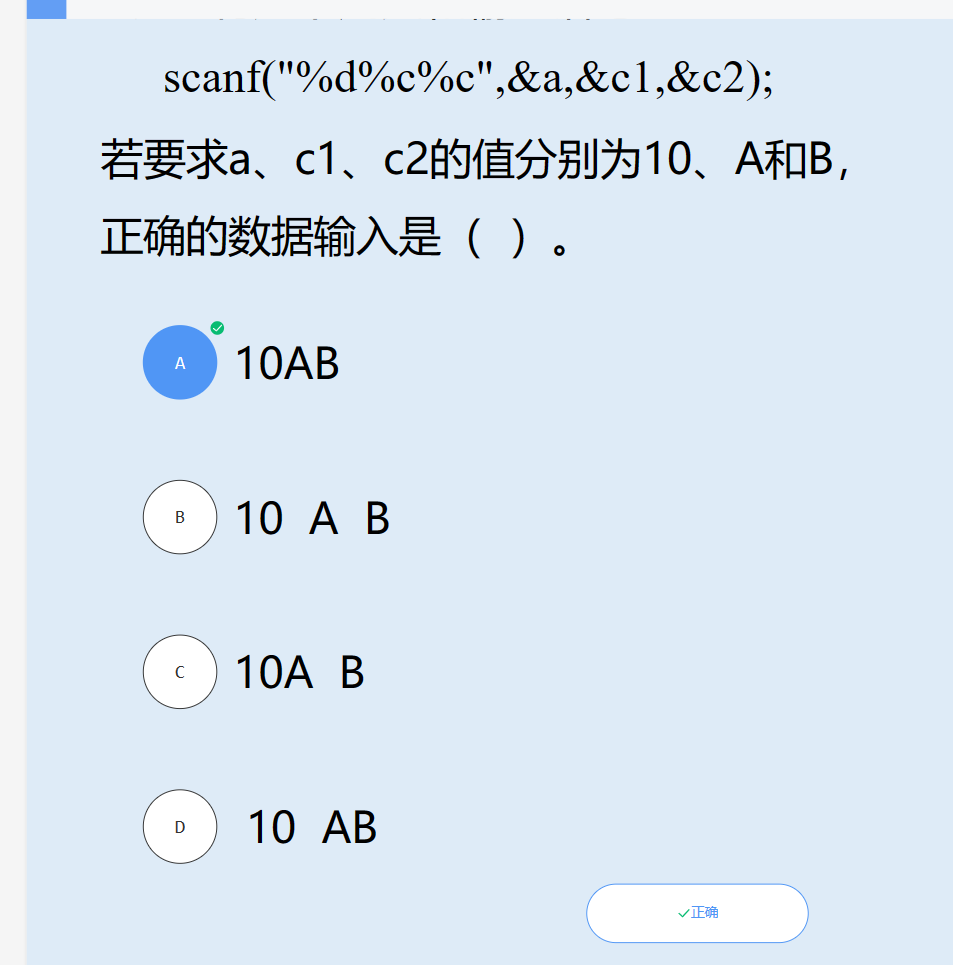
1. a=(b=4)+(c=6)

a=10,b=4,c=6

1. a=(b=10)/(c=2)

a=5,b=10,c=2

6.



A．10 A B

B．10AB

C．10 AB

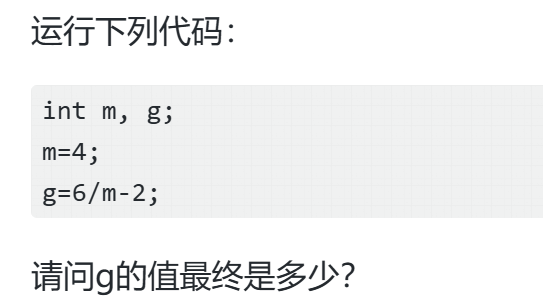
D．10A B

B

【10与A之间】在输入数值数据时，输入空格、回车、Tab键或非法字符（不属于数值的字符），系统会认为该数据结束

【A与B之间】%c只会吸收一个字符，所以第一个%c吸收了’A’之后，系统就会执行第二个%c，吸收下一个字符（’B’）。因此，如果在A与B之间插入空格（’ ’），则空格也会被当成字符被%c吸收

7.



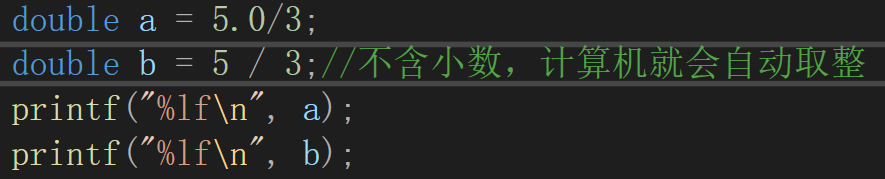
答案：最终：g=-1

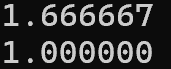
解释:

6/m=6/4=1,这里 整数除以整数 会自动取整

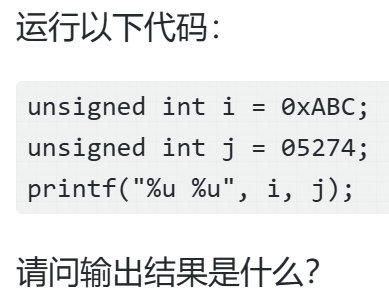
然后1-2=-1

例如：





8.



2748 2748

%u代表无符号整型

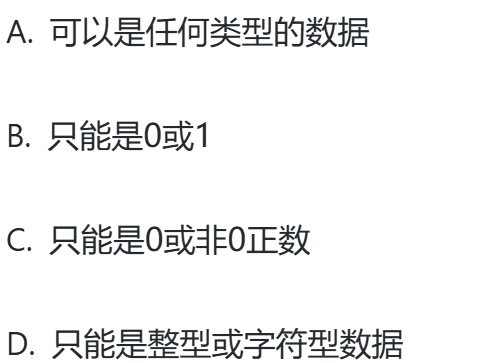
在数字前面的：0x表示十六进制，0表示八进制

ABC(16进制)=2748(十进制)

5274（八进制）=2748(十进制)

9.





A

逻辑运算符两侧任何类型的数据都可，可以是数字，变量，表达式等。

10.



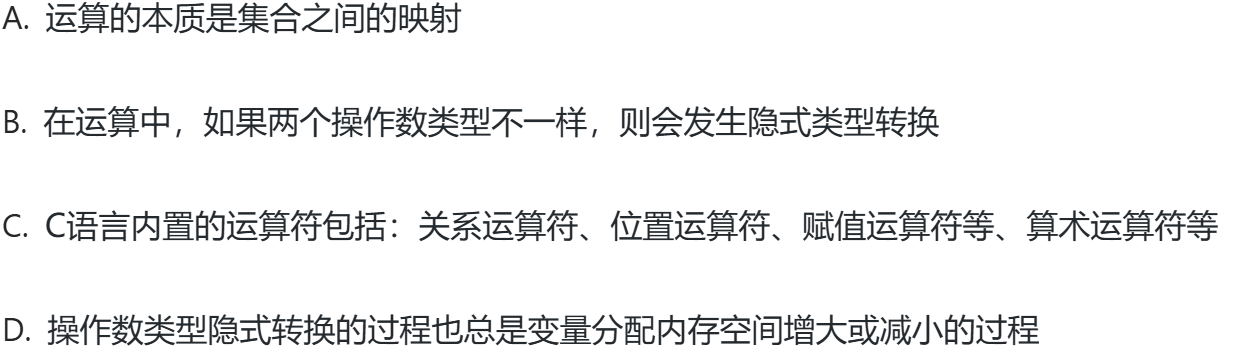
%u表示无符号整型十进制（32位）

i\*-1得到-3（有符号）

-3的32位二进制补码：1111 1111 1111 1111 1111 1111 1111 1101，对应的十进制数为：4294967293

11.





C，D

C．没有“位置运算符”，只有“位运算符”

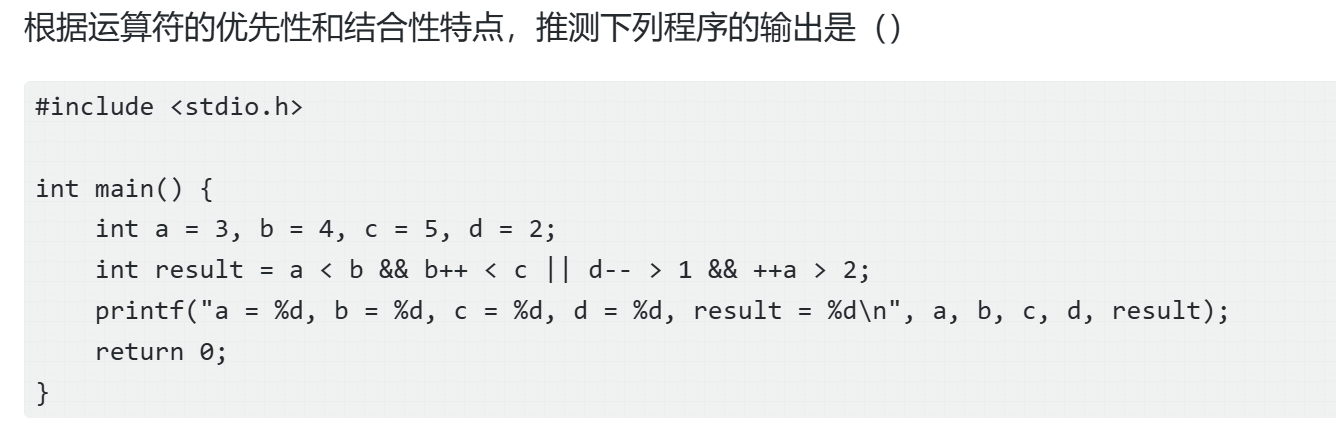
D．并不是“总是”，

例如：unsigned int与int的隐式转换，二者都是4字节，所以二者隐式转换时 内存空间不会变化,

float与int的隐式转换，也是内存不变。

但是要注意： float与int的隐式转换，虽然二者都是4字节（32位），但是float的有效位只有24位，所以int 转化为float有可能会导致精度缺失

12.



a=3,b=5,c=5,d=2,result=1

&& 和 || 两个运算符优先级相等，所以从左至右依次计算

a<b,为真。

继续执行b++<c

b<c, 为真。然后b=b+1=5

根据|| 的“偷懒性”：前面有真，整体必定为真，则不执行后面的

所以后面的都不执行了，result=1

所以a=3,b=5,c=5,d=2,result=1

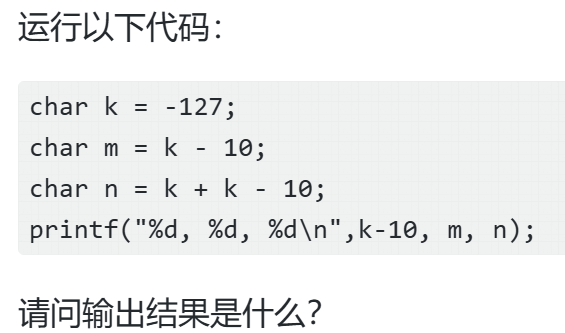
注意：&& 和 || 两个运算符的“偷懒性”

&& 的“偷懒性”：前面有假，整体必定为假，则不执行后面的

|| 的“偷懒性”：前面有真，整体必定为真，则不执行后面的

## 溢出类问题

1.



答案：-137，119，-8

解释：

char类型占1字节，即8位

char类型默认是有符号的，

所以，char类型范围：，即-128~127

当溢出时，会以滚动式循环-128~127这些数字

比如：

（-128）-1=127

127+1=-128

底层原因：

1. C语言忽略溢出

（“忽略溢出“指：即使溢出了，CPU也不会发生异常，仍会继续执行，

但是，编译器仍会发出警告）

1. 二进制补码的计算：

（-128）的二进制补码：1000 0000

127的二进制补码：0111 1111

所以（-128）-1对应的二进制补码为：1000 0000-1=0111 1111，正好就是127的二进制补码；

127+1对应的二进制补码为：0111 1111+1=1000 0000，正好就是（-128）的二进制补码。

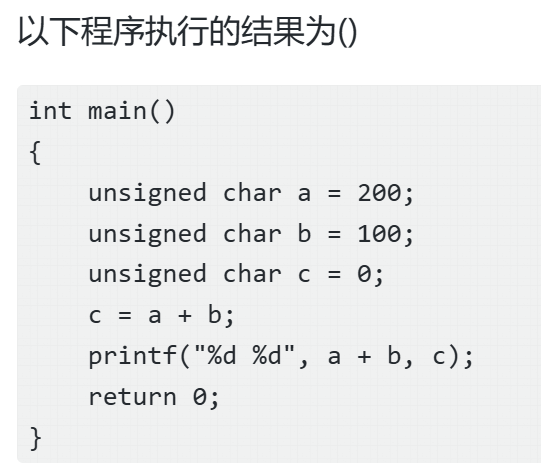
因此，题目中：

m=k-10=(-127)-10=(-128)-9=[(-128)-1]-8]=127-8=119

n=k+ k-10=(-127) +(-127)-10=(-127)+ (-128)-9=(-127)+ [(-128)-1]-8]=(-127)+127-8=-8

而直接输出的k-10比较特殊，由于它没有再存放于char类型中，所以不受溢出的限制，所以输出-137

2.



300 44

直接输出的a+b, 由于它没有再存放于char类型中，所以不受溢出的限制，所以输出300

unsigned char 占1个字节，即8位，是无符号的

所以unsigned char类型的范围是：，即0~255

与上一题相似，当溢出时，会以滚动式循环0~255这些数字

所以：c=a+b=200+100=200+55+1+44=(255+1)+44=0+44=44

## 第四章（三种基本结构）

1.C语言对嵌套if语句的规定是：else总与（ ）配对。

A．缩进位置相同的if

B．其之前最近且尚未配对的if

B．

注意：可能有多个缩进位置相同的if，因此A的表述不够准确

2.下面程序运行后的输出是

#include<stdio.h>

void main()

{

int a = 0, b = 0, c;

if (a < b) c = 1;

else if (a = b) c = 0;//第六行

else c = -1;

printf("%d\n", c);

}

-1

注意：第六行的“a=b”是赋值运算，先把b的值赋给a，再判断a的真假

易错：小心：等于号（==）与赋值运算符（=）

3. 下面程序运行后的输出是

void for\_while\_do()

{

int a, b, c, d, i, j, k;

a = 10;

b = c = d = 5;

i = j = k = 0;

for (; a > b; ++b) i++;

while (a > ++c) j++;

do k++;

while (a > d++);

printf("i=%d,j=%d,k=%d\n", i, j, k);

}

i=5,j=4,k=6

4. 下面程序运行后的输出是

#include<stdio.h>

int main()

{

int x = 2;

while (x--);

printf("x=%d\n", x);

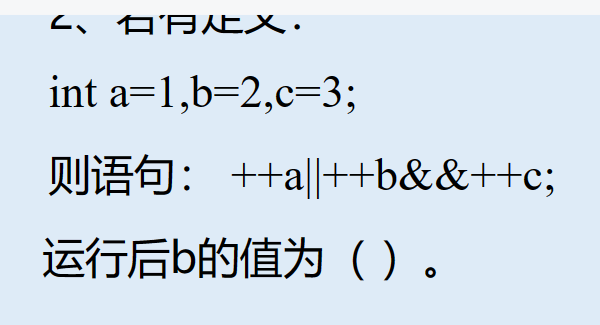
return 0;

}

x=-1

注意：无论while（x--）是否为真，判断**后**x都会-1

5．若有以下程序



A.3

B.1

C.2

C

或(||)，与(&&)这两个运算具有“偷懒性”

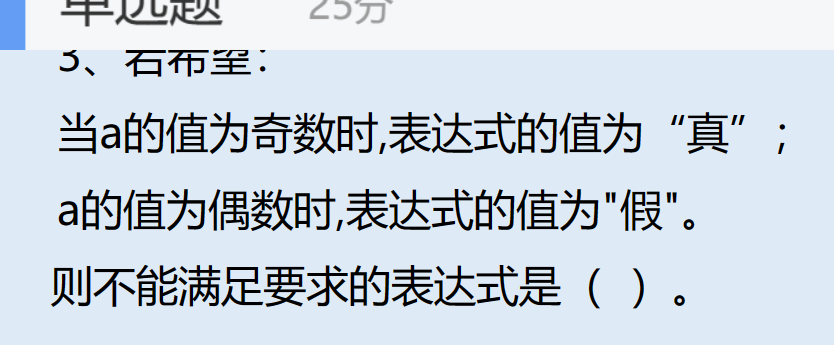
或(||)，前一个为真，则不执行后一个，直接得出结果“1”

与(&&)，前一个为假，则不执行后一个，直接得出结果“0”

此题中，“++a”为真，因此不执行“++b”，直接得出结果“1”，然后再执行“&&++c”。

因此b的值不变，仍是2

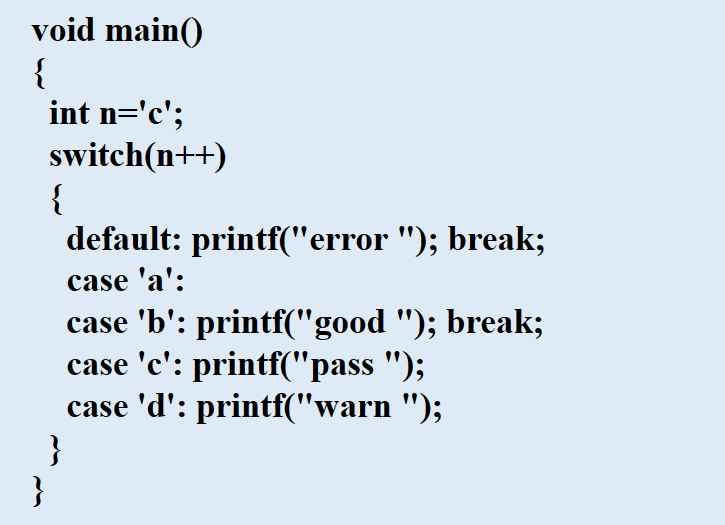
6.



1. a%2==1
2. !(a%2==0)
3. a%2
4. !(a%2)

D

7．以下程序运行后的结果是（）



1. error
2. pass
3. warn
4. pass warn

D

switch语句：

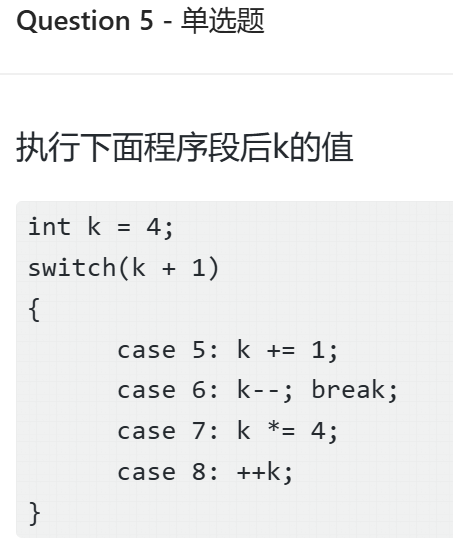
系统会将switch（）括号内的表达式与 case后的常量 **逐行对比**

遇到相同的，则从这个case开始执行。

若这个case中的语句没有“break;”,则逐行向下执行，直至遇到“break;”

若与所有case都不符合，才执行default中的语句。因此，无论default的位置在哪里，都不影响执行的顺序。

8.



答案：4

k+1=5

所以执行case 5:k+=1;

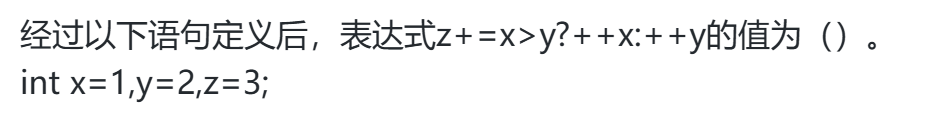
得到k=5

然后执行case 6:k--;break;

得到k=4

遇到break;退出

9.



答案：6

注意：先执行(x>y?++x:++y),再执行z+=

由于x=1,y=2,所以x>y不成立；

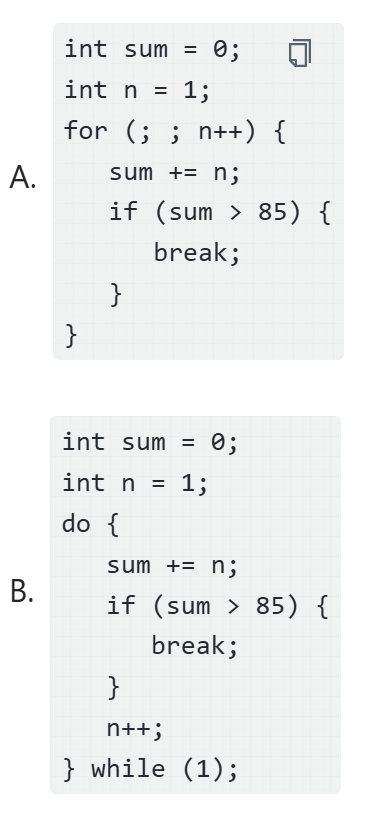
所以，选择++y；

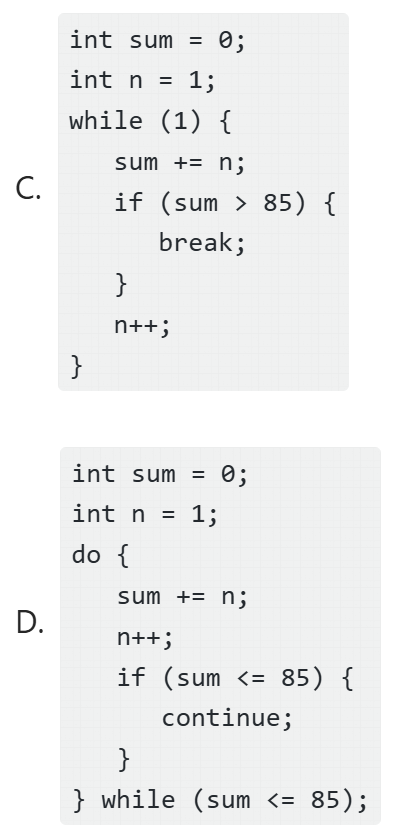
再执行z+=++y（y先加1，后使用）；

y=2+1=3;

所以z=3+y=6

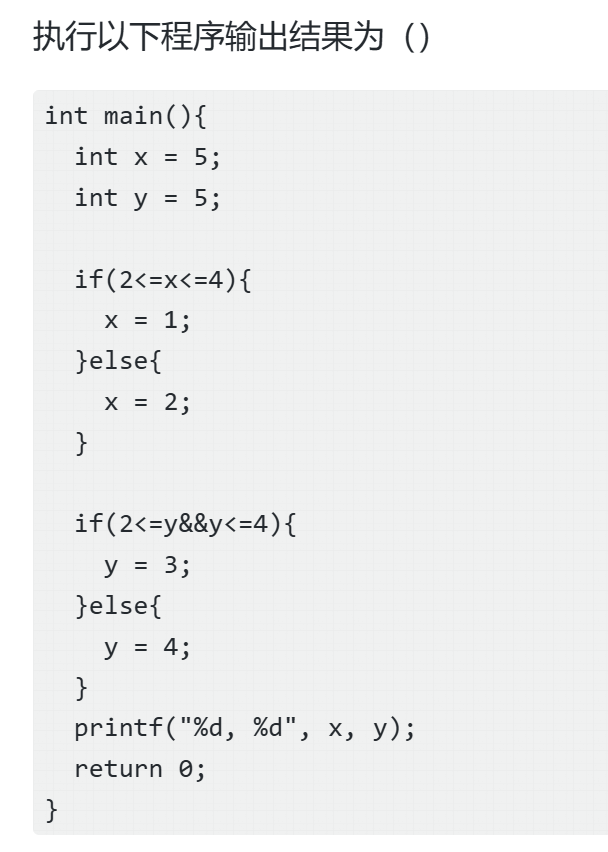
10.（多选题）以下C语言代码中，哪一个或者那几个输出是：能够满足条件1+2+……+n>85 的n的最小值？





ABC

11.



答案：1，4

注意：C语言中“2<=x<=4”并不是数学中的意思

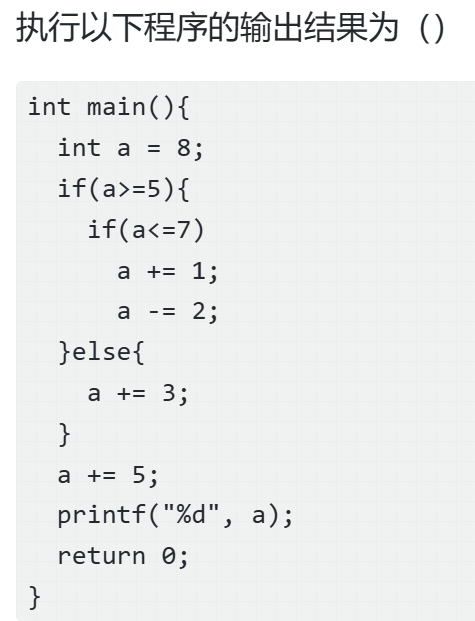
而是：从左至右的逻辑表达式

先是：“2<=x”，由于x=5,所以成立，值为1

再是：“1<=4”，成立，所以值为1

所以进入if语句里，执行x=1

12.



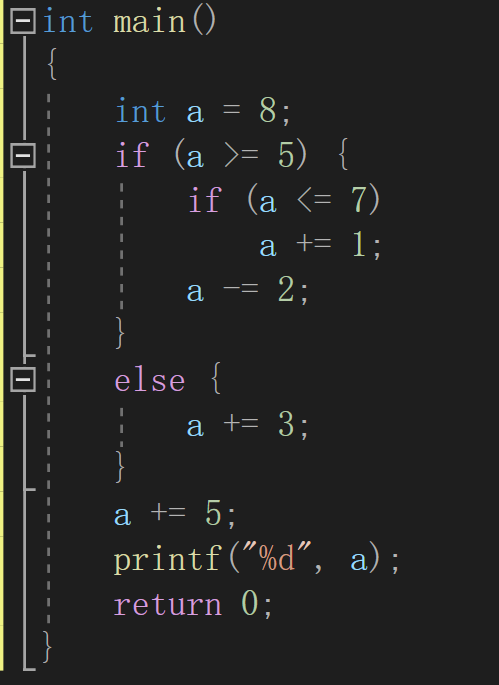
答案：11

注意：

1.题目中的代码缩进不正确，要自己调整一下缩进

2.对于if,else,while,for等语句，如果没有花括号，则后面最多只有1句代码是受这些语句的控制的

正确的缩进格式如下



因为a=8,8>=5成立,所以进入第1个if语句

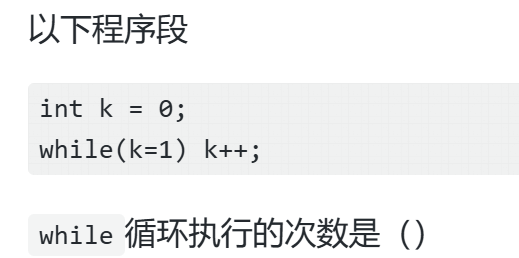
但是8<=7不成立，所以不进入第2个if语句

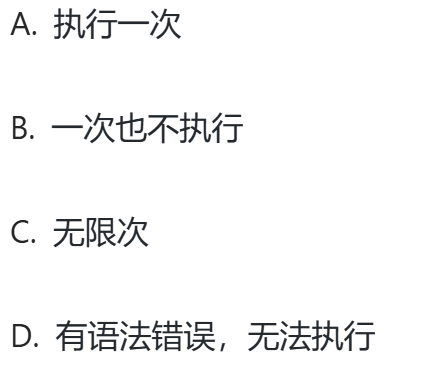
所以只会执行：a-=2

a+=5

得到:11

13.



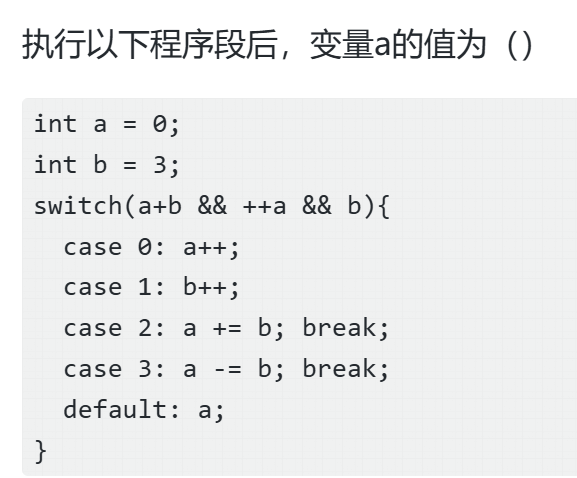


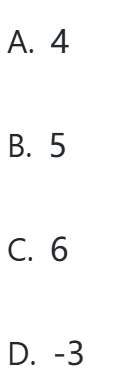
C

注意：k=1是赋值语句，而不是判断相等（判断相等用==）

因此，k每次都被赋值为1，所以while语句永远成立，所以会循环无限次

14.





答案 B

注意：

由于（a+b && ++a && b） 为真，所以值为1

## 第五章 函数

1.以下函数调用语句中含有（）个实参

fun(exp1,(exp2,exp3,exp4));

A.1

B.2

C.3

D.4

B

2.如下函数是什么类型（）

fun(float a)

{ printf(“%f\n”,a\*a);}

1. 无法确定
2. int
3. float
4. void

B

函数的类型符号省略时，默认函数是int型

3.宏定义#define G 9.8 中的宏名G代表（）

A．一个单精度实数

B．一个双精度实数

C．一个字符串

D．不确定类型的数

C

4，以下关于文件的叙述**不正确**的是（）

1. 一个include命令只能指定一个被包含的文件
2. 一个include命令能指定多个被包含的文件
3. 文件包含是可以嵌套的
4. 在#include命令中，文件名可以用双引号或尖括号括起来

B

注意：CD都对！！！

5. 以下程序的运行结果是（）

void fun()

{

int a = 3;

static int b = 4;

a = a + 1;

b = b + 1;

printf("a=%d,b=%d\n", a, b);

}

int main()

{

fun();

fun();

}

a=4,b=5

a=4,b=6

注意：static表示静态变量

静态局部变量：（这题中的b就是静态局部变量）【前加static】

1.作用域为函数内部

2.除了第一次之外，以后每次使用该函数时会沿用上次使用该函数后的值

静态全局变量：【前加static】

1.作用域为当前文件，从定义处到该文件末尾

动态全局变量：【前加auto，也可以不加】

1.在所有函数（包括主函数）之外定义

2.作用域：作用域为整个项目，即最终编译成可执行文件的所有文件中均可以使用动态全局变量。外部文件可以通过extern关键字来声明另一个文件中存在的全局变量

动态局部变量：【前加auto，也可以不加】

1.在函数内部定义，或复合语句内部定义。

2.作用域：函数内部，从定义处到该函数末尾

6.以下程序的运行结果是（）

#include<stdio.h>

int x = 1, y = 2, z = 3;//第2行

void main()

{

void sub();

void add(int x, int y, int z);

int x = 2, y = 3, z = 0; //第7行

printf("1st:x=%d,y=%d,z=%d\n", x, y, z);

add(x, y, z);

printf("3rd:x=%d,y=%d,z=%d\n", x, y, z);

sub();

}

void add(int x, int y, int z)

{

z = x + y;

printf("2nd:x=%d,y=%d,z=%d\n", x, y, z);

}

void sub()

{

z = x + y;

printf("4ed:x=%d,y=%d,z=%d\n", x, y, z);

}

1st:x=2,y=3,z=0

2nd:x=2,y=3,z=5

3rd:x=2,y=3,z=0

4ed:x=1,y=2,z=3

第2行定义的x,y,z是全局变量

第7行定义的x,y,z是局部变量

因为 sub（）是无参函数，

所以 它所用的x,y,z不是从main函数传递来的，而是第2行定义的全局变量

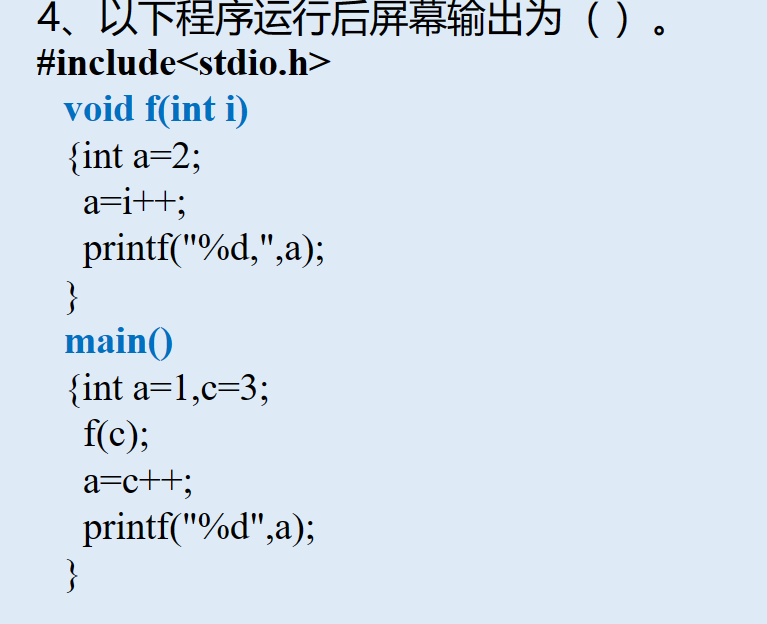
因为 add(int x, int y, int z)是有参函数，

所以 它所用的x,y,z是从main函数传递来的，

又因为：当变量名相同时，在局部用局部变量

所以 add函数用的是第7行定义的局部变量，不是第2行定义的全局变量

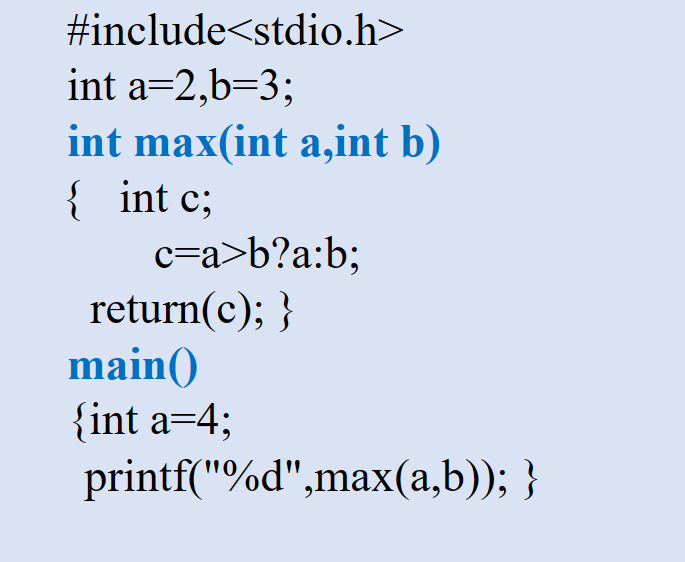
7.以下程序运行的结果是（）



1. 4,3
2. 3,3
3. 2,3
4. 3,1

B

8.以下程序运行后的结果是（）



A．2

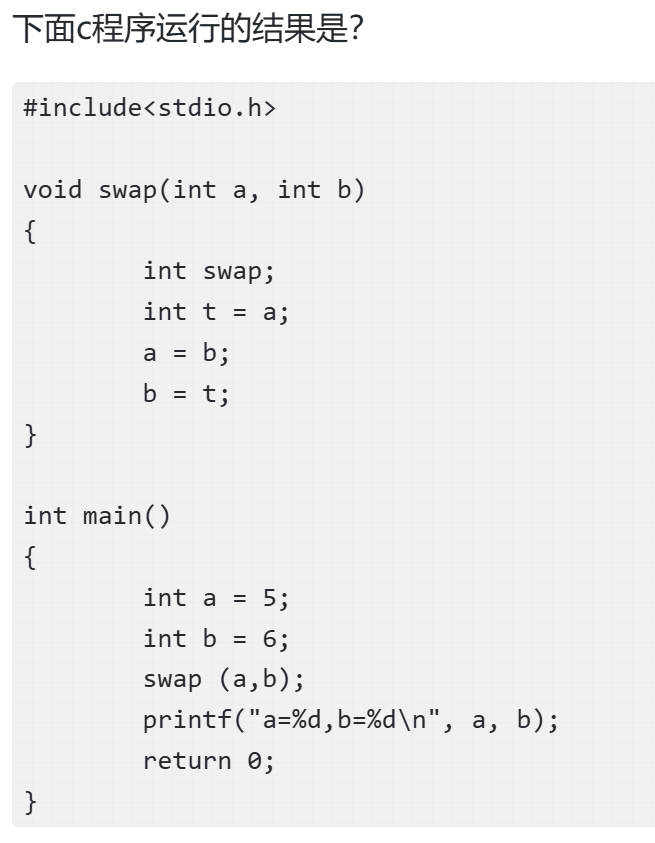
B．3

C．4

C

max函数中的a从main函数中传递而来，而b从全局变量处的b传递而来

9.

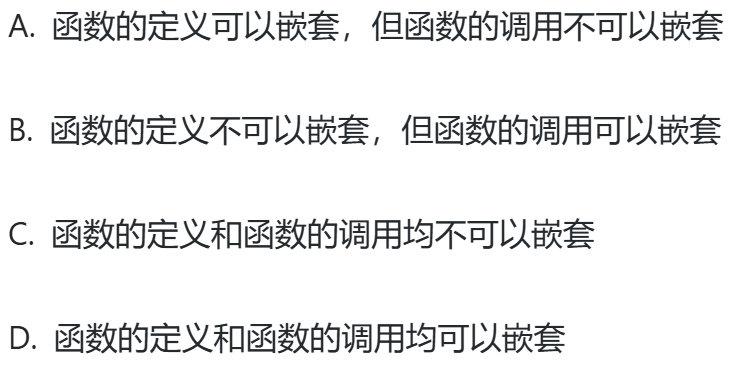


答案：a=5,b=6

注意：函数形参与实参直接是***值传递***

10.



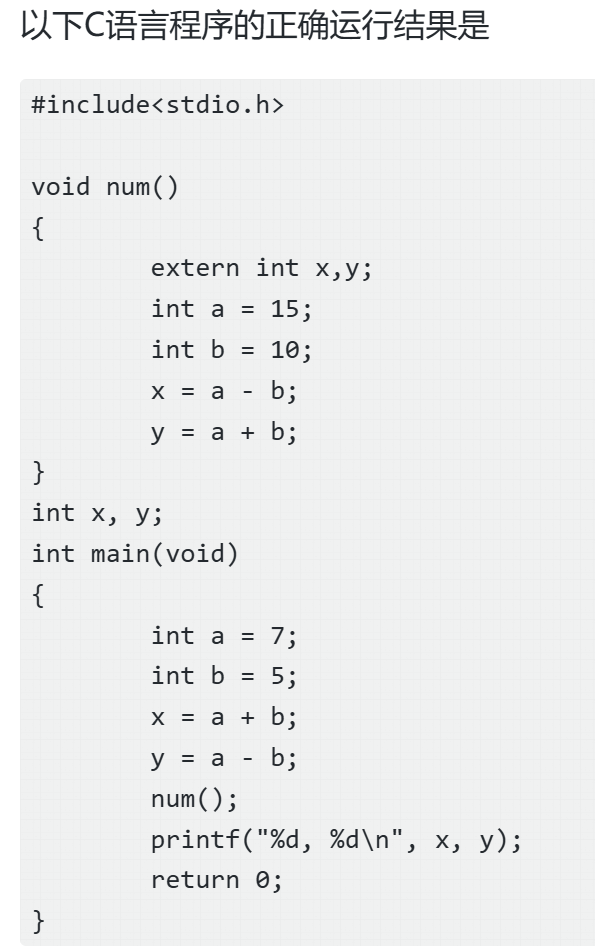


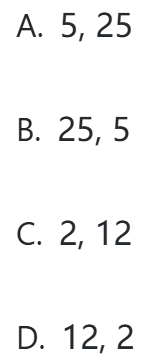
B

不可以在一个函数内部定义一个函数

但是可以在一个函数内部调用一个函数

11.



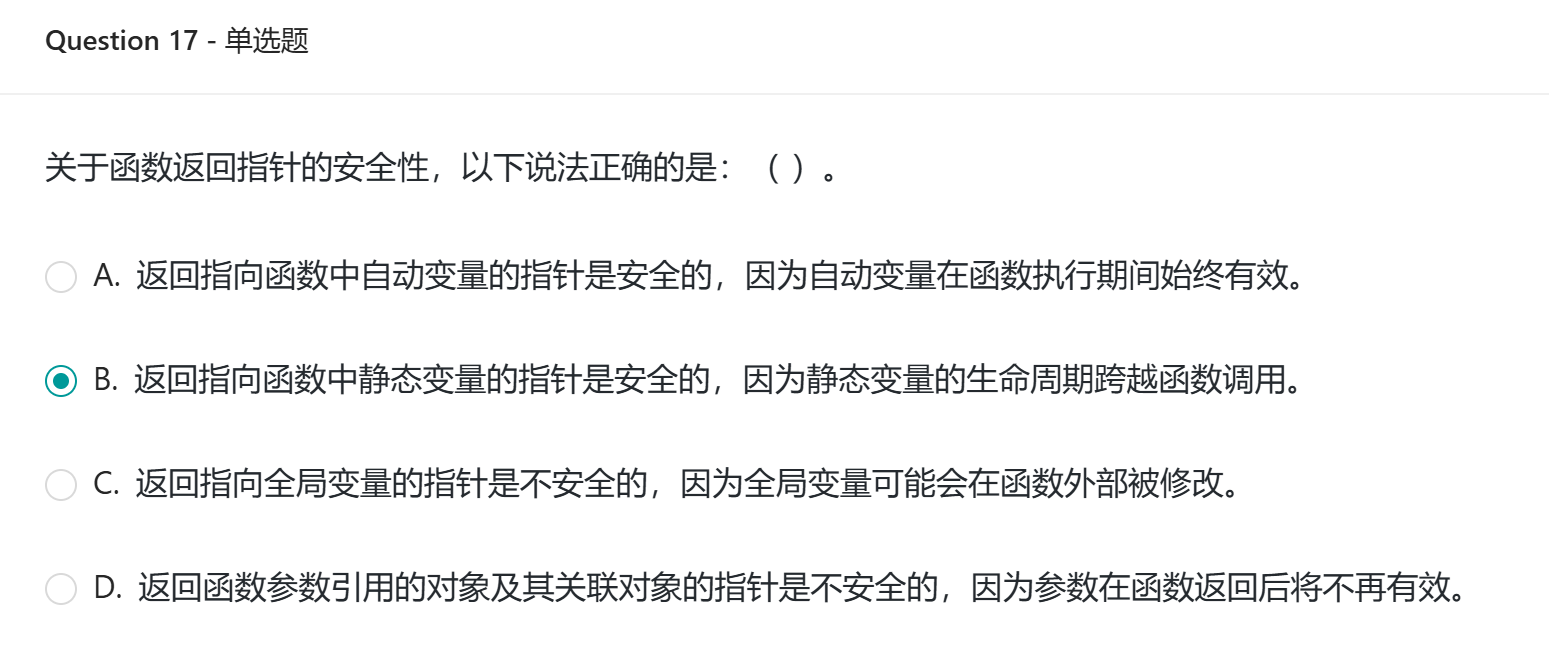


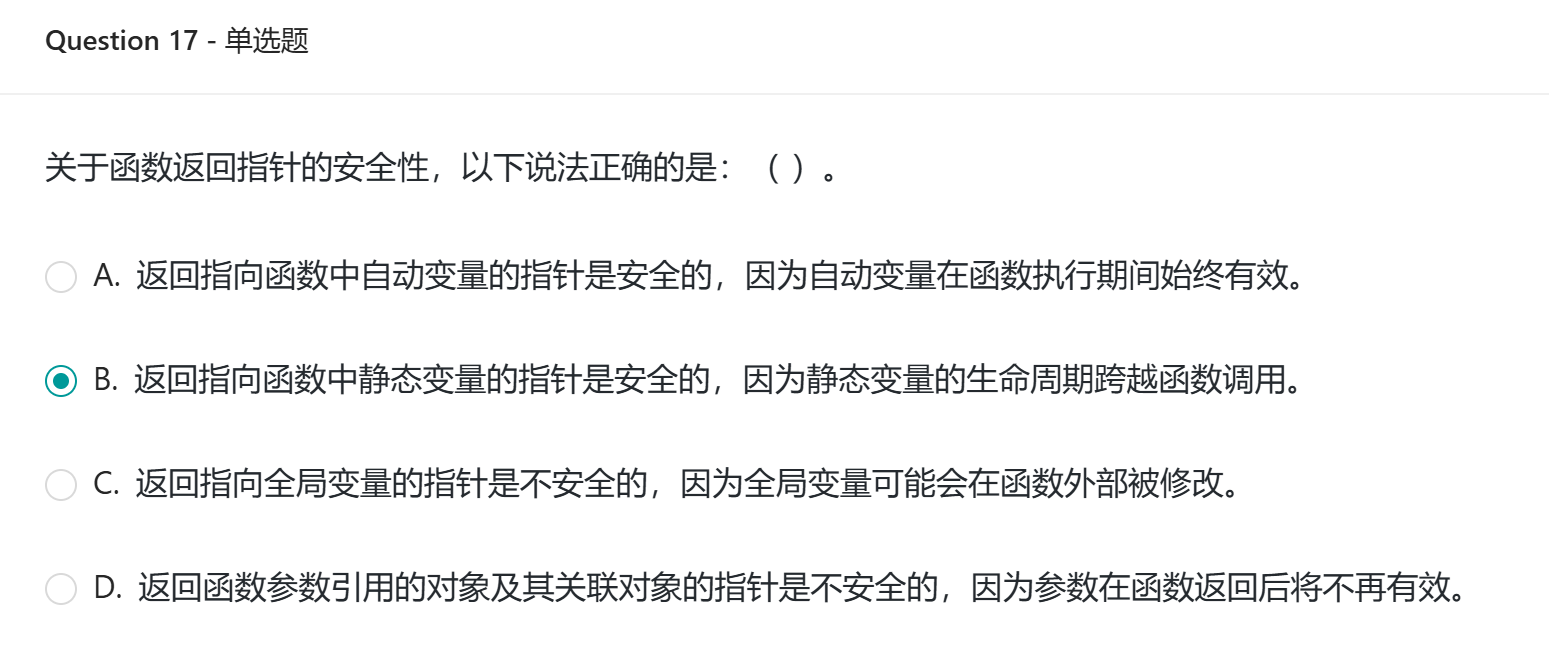
答案：A

注意extern 外部变量声明符

x，y是全局变量

12.





1. 自动变量的指针：不安全。因为自动变量在离开函数后会***自动被销毁***
2. 静态变量的指针：安全
3. 全局变量的指针：安全
4. 返回函数参数引用的对象及其关联对象的指针，如指向传入数组的元素，或

指向传入结构体指针的成员：安全

1. 返回动态申请内存的指针：安全

## 第六章 数组

1.下列操作正确的是（）

A. char name[10] = "asdf";

B. char name[10]; name[10] = "asdf";

C. char name[10]; strcpy(name, "asdf");

D. char\* name="asdf";

E. char\* name; name="asdf";

ACDE

易错：

B，E对比：字符数组 与 指向字符数组的指针 的赋值方式不同！

E.char\* name;

name="asdf";//地址变量名name前无\*

2.下面对含有10个元素的一维数组初始化正确的是（）

1. int a[10] = { };
2. int a[10] = {10\*1 };
3. int a[] = {0 };
4. int a[10] = (0,0,0);

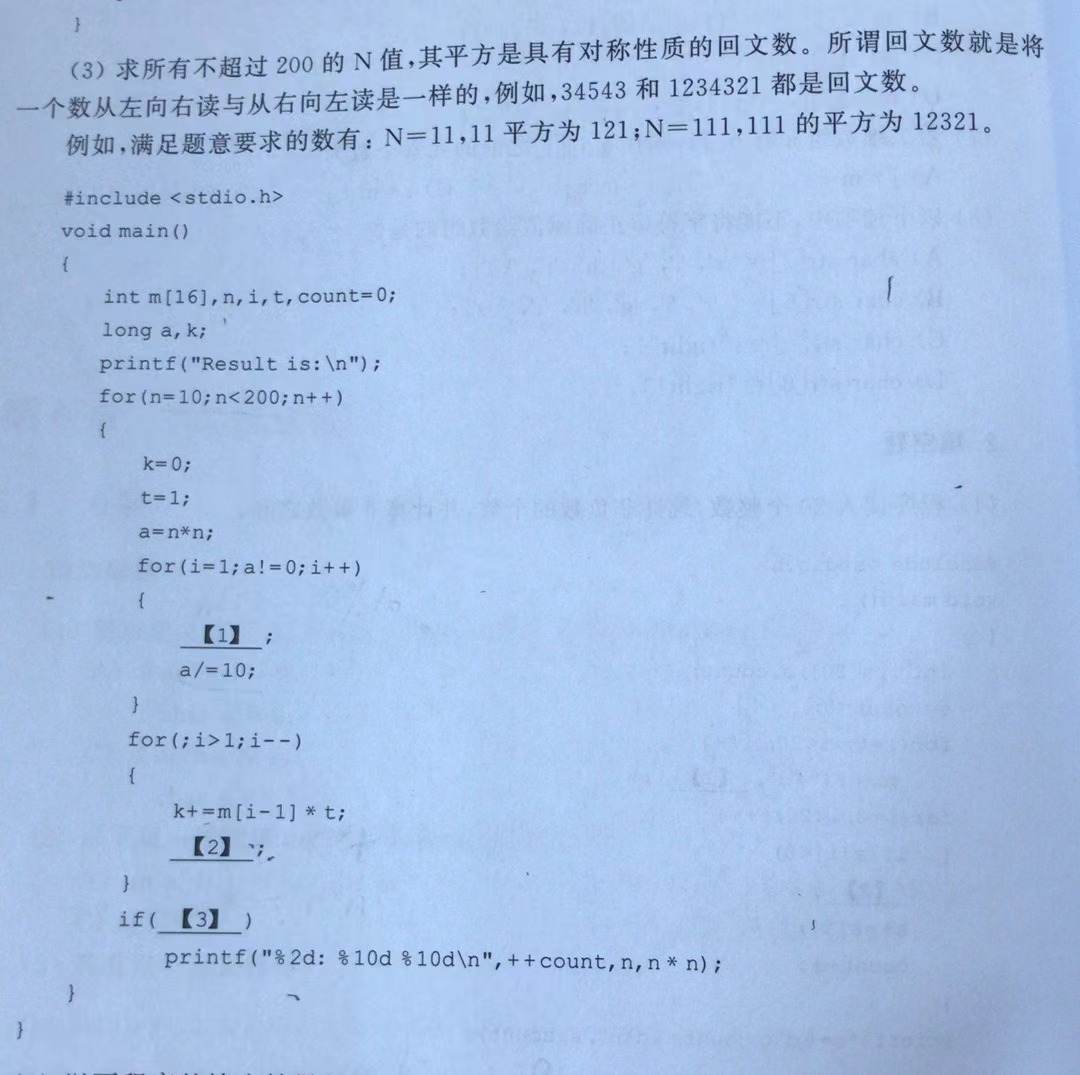
B

A．花括号内至少要有一个数，如果想全部赋值0，则填1个0

C．未指定数组长度，系统会默认根据花括号内数字的个数，将其定为1

D．应该用花括号，而不是圆括号

3.

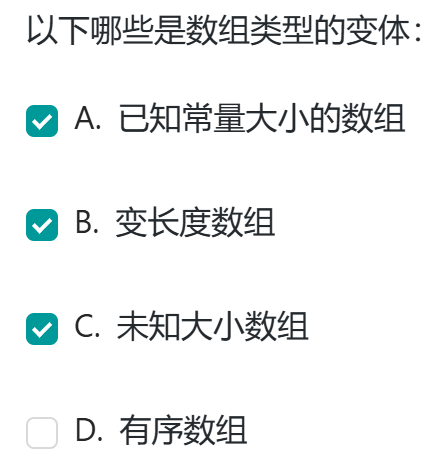


【1】m[i]=a%10

【2】t=10\*t

【3】k==n\*n

4.



记！

A,B,C

A. 已知常量大小的数组：这是标准的数组类型，在定义时其大小是已知且不变的。例如，int a[10]; 定义了一个有 10 个整数的数组。

B. 变长度数组（Variable Length Array, VLA）：在 C99 标准中引入。这种数组的长度不是在编译时确定的，而是在运行时确定。例如，int n; scanf("%d", &n); int a[n]; 这里 a 就是一个变长度数组。

C. 未知大小数组：在某些情况下，可以声明一个未知大小的数组，特别是在结构体中作为最后一个成员时。这通常用于所谓的“柔性数组成员”，其长度可以在运行时确定。

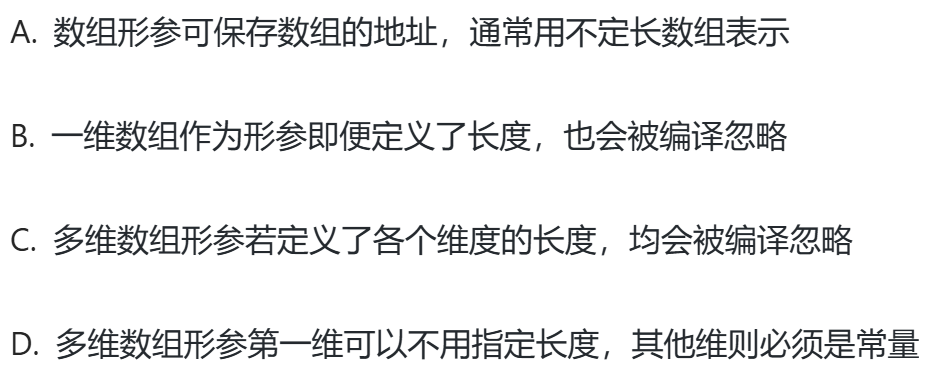
D. 有序数组：这不是一个由 C 语言本身定义的数组类型，而是对数组内容的描述。一个数组是否有序，取决于它的内容和如何被程序处理，而不是数组的类型。

综上所述，A（已知常量大小的数组）、B（变长度数组）和C（未知大小数组）可以被视为数组类型的变体，而D（有序数组）是关于数组内容的描述，而非类型的变体。

因此，正确的选项是 A、B 和 C。

5.





C

A 题目表述的意思是：下图的这种情况，是对的

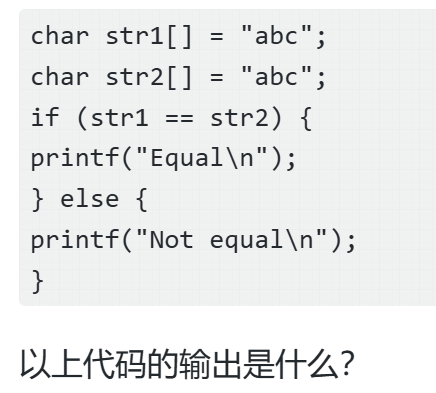


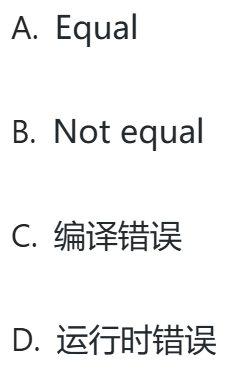
B正确，编译器不会检查数组第一维度的长度，因此是否定义都可以

C D 编译器只会忽略数组第一维度的长度

## 字符数组

1.





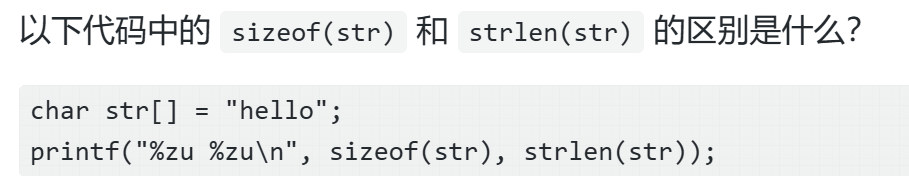
B

str1，str2是两个字符数组的数组名，代表的是数组首元素的地址，

两个数组的首元素肯定放在不同位置，所以它们的地址肯定不同。

所以str1!=str2

2.



答案：6 5

sizeof 会返回整个字符数组所占的字节数，

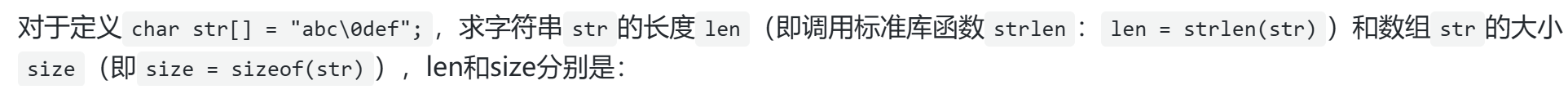
注意：系统会自动在字符数组的最后面加上一个空字符’\0’,

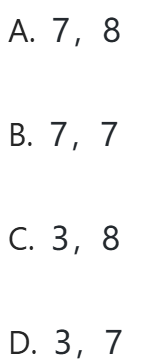
所以，sizeof(str)=6

strlen 返回字符数组中的字符个数，以空字符’\0’作为结束标志，且不包含这个空字符’\0’

所以，strlen(str)=5

3.





C

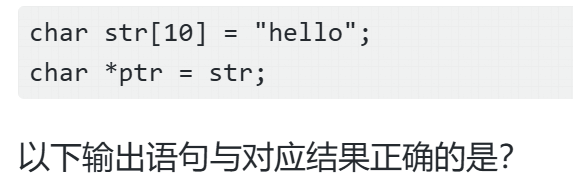
strlen 返回字符数组中的字符个数，以空字符’\0’作为结束标志，且不包含这个空字符’\0’

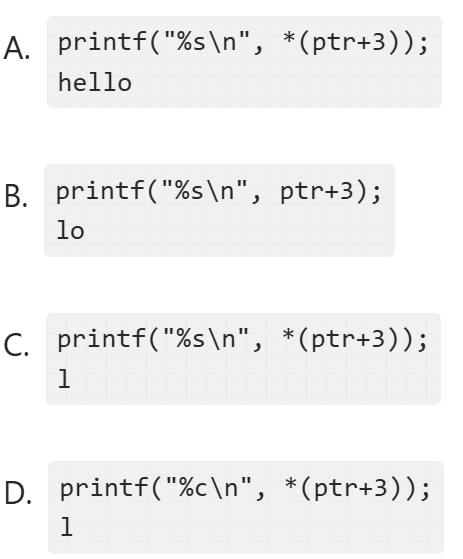
所以，strlen(str)=3

注意：系统会自动在字符数组的最后面加上一个空字符’\0’,

所以，sizeof(str)=8

4.





B,D

注意：

%s -- 对应使用：字符数组的**地址**

%c -- 对应使用：**字符**（字符变量或字符常量均可）

B选项中，指针变量ptr+3表示地址

D选项中，\*( ptr+3)表示字符变量 ‘l’

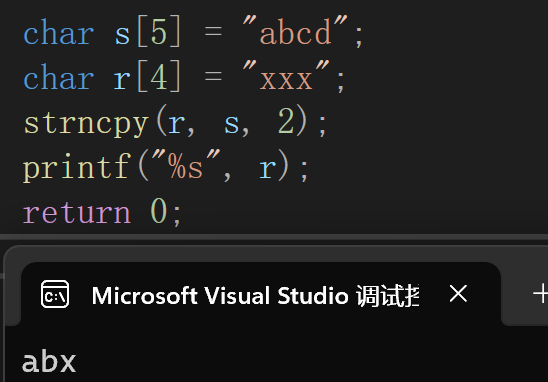
5.





答案：B，D

1. 这是strcpy()函数的功能
2. 正确
3. 错，任何字符都会复制
4. 正确，例如以下的代码



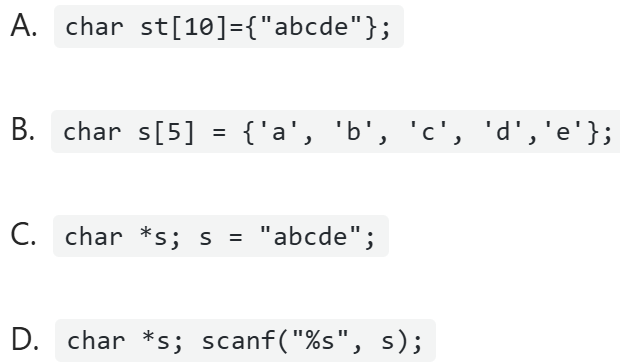
源字符串s的长度为4，大于n=2

所以只把ab复制到了目标字符串r中，末尾没有‘\0’，

因此最终字符串r为：abx

6.



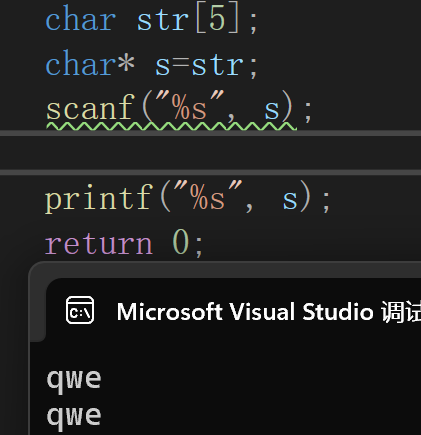


D

B 虽然 字符数组末尾不会自动加上空字符’\0’，但是不会报错，是正确的

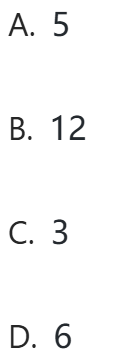
D 错因是char\* s没有指向某个变量或数组。

改成下面这样则对



7.





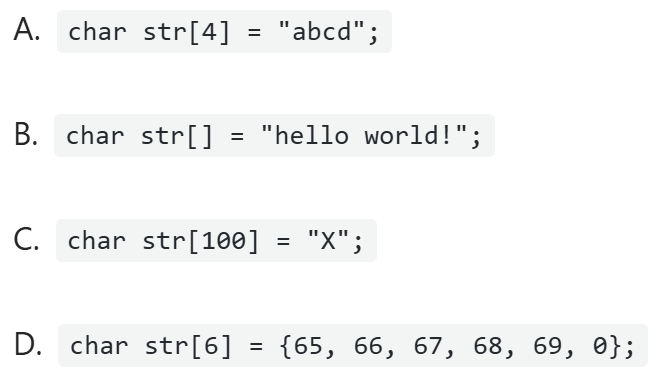
答案：A

注意：

1. 转义字符 算是 1个字符
2. \065是一个整体，是八进制转义字符，表示‘A’。以为0后面有65，所以这里的\0 不是空字符，

8.

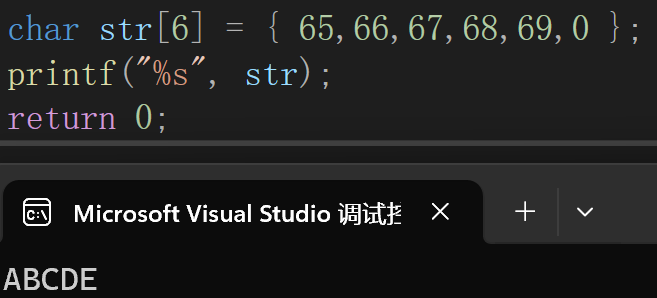




A

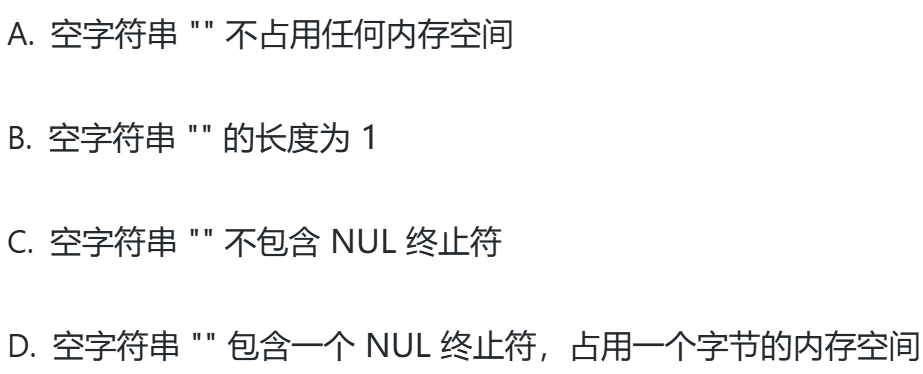
A．字符数组 没有位置给系统自动添加上空字符‘\0’，所以会报错

D．前5个数在ASCII码中分别对应 ABCDE，最后1个0在ASCII码中对应空字符‘\0’，所以正确。如下图所示。



9.



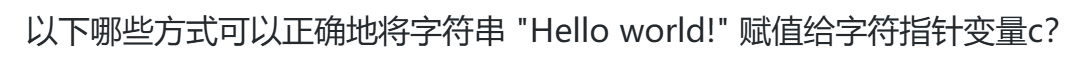


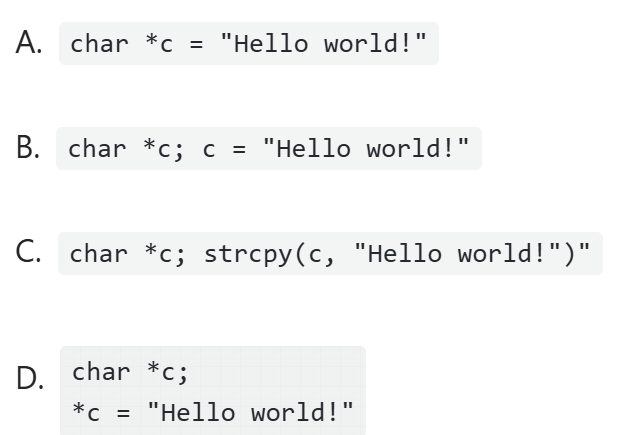
D

这里的NUL终止符就是 空字符‘\0’

空字符串中包含一个NUL终止符（空字符‘\0’），所以占用1个字节的内存空间，但长度为0

10.



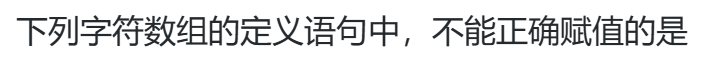


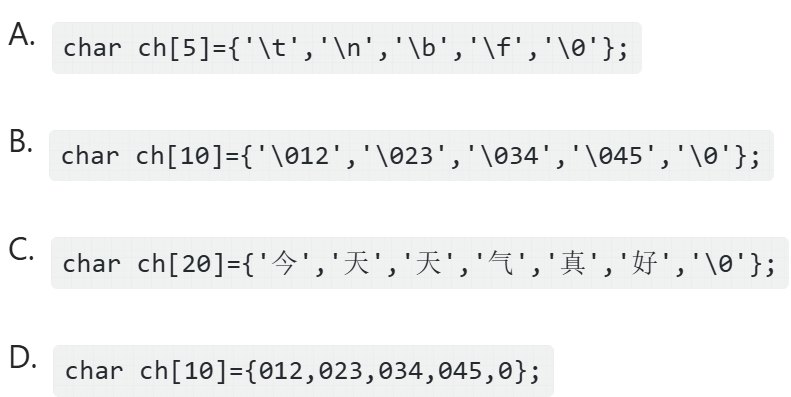
1. B
2. B显然正确

C．指针c没有被赋某个变量的地址，所以报错

D．“ \*c=”Hello world!” ”中的 “ \*c ”错误，直接写“c”即可

11.





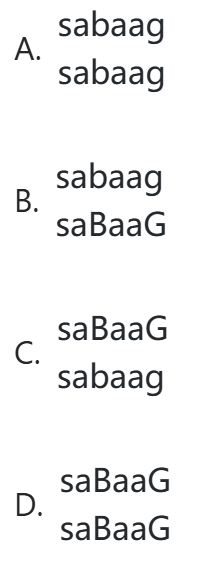
C

每个中文字符通常占用 2 或 3 个字节

而单引号（‘’）中只能放1个字节

12.





A

注意：

strlwr(str1)会把str1的全部英文字符变成小写，并且作为返回值

所以，最后str1,str2都是全小写的字符串

13.



答案：10

每个char类型占1字节

转义字符’\t’ 占1字节，

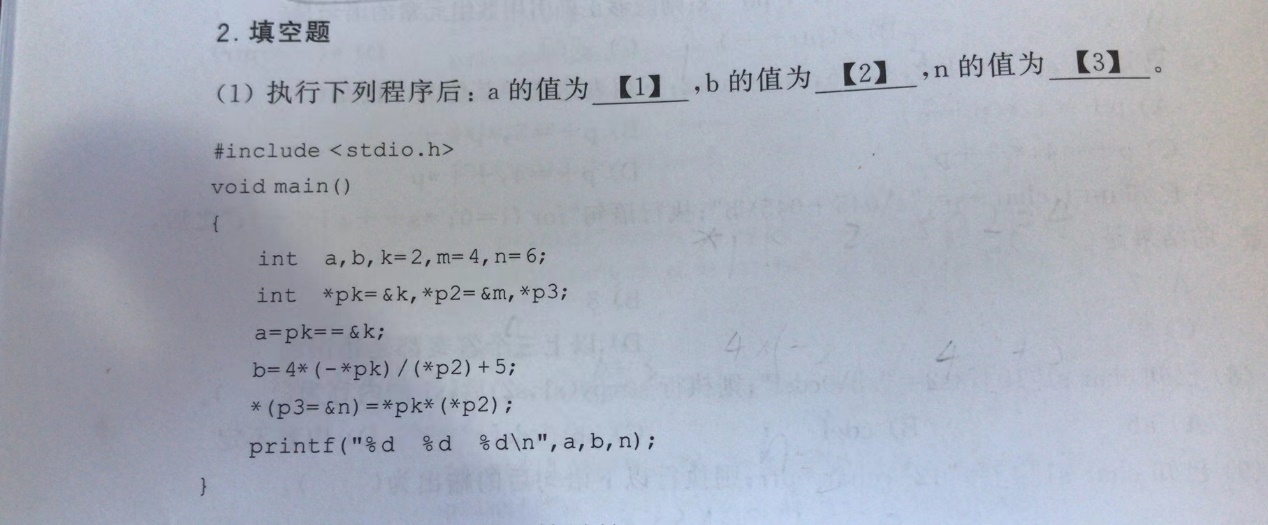
易错：系统会在字符串末尾自动地加上一个空字符’\0’,也占1字节！

所以，字符数组s共有10个字符，共占10字节

所以，sizeof(s)=10

## 第七章 指针

1.



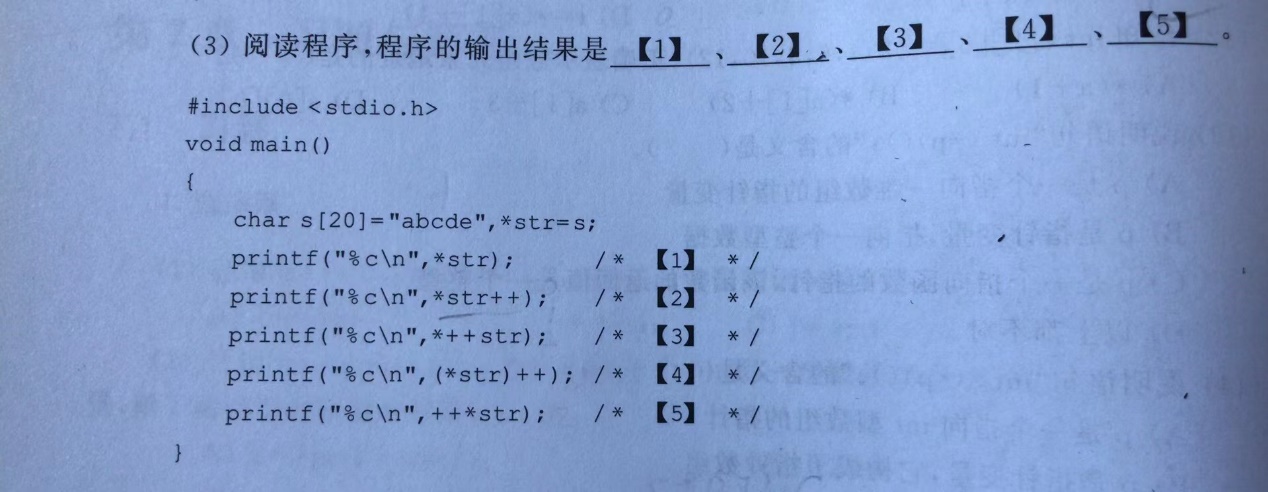
【1】1

【2】3

【3】8

特别注意【3】：\*(p3=&n)=……是将n的地址给了p3，然后将“……”处运算后的值给\*p3

2.



1. a
2. a
3. c
4. c
5. e

注意：

在【4】处：(\*str)++

先把地址变量str当前地址对应的字符c输出，

然后使字符c的值+1，变成d，

（也就是此时的字符串s变成了“abdde”）

接着，在【5】处：++\*str

先把地址变量str当前地址对应的字符d的值+1，变成e

（也就是此时的字符串s变成了“abede”）

然后再输出地址变量str当前地址对应的字符e

3.以下程序运行后的结果是



值不确定

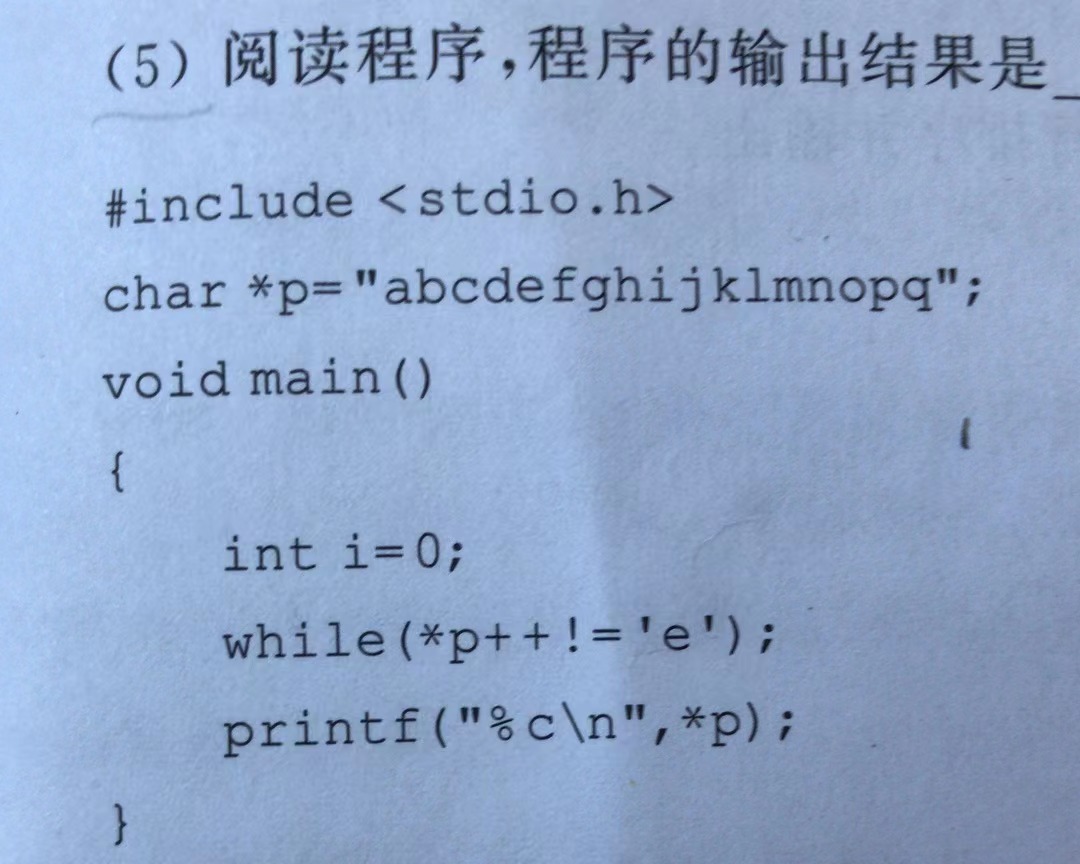
因为：

1.main函数中的地址变量p没有存放某个地址

2.在fun函数中对形参s进行了赋值运算（因为地址变量的值就是地址），而形参s的值是不能传递回给实参的。

在函数中，地址变量作为形参只能对 地址变量所指向的变量 的值 进行运算！！！

4.

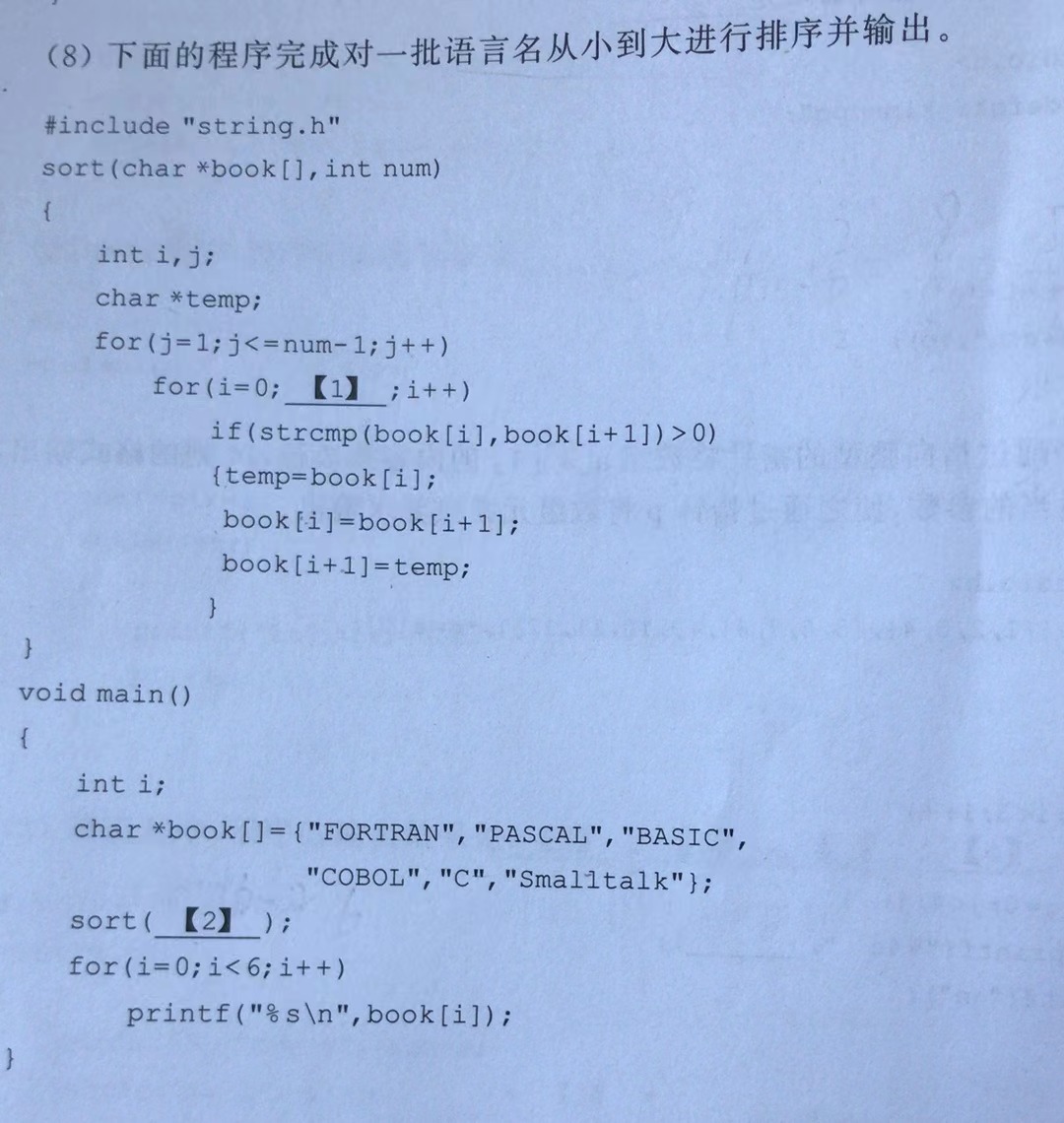


f

注意：while语句是（）内的语句成立时，执行{ }内的语句，直至（）内的语句不成立。

如果{ }内无语句，则表示不执行任何命令，直至（）内的语句不成立。

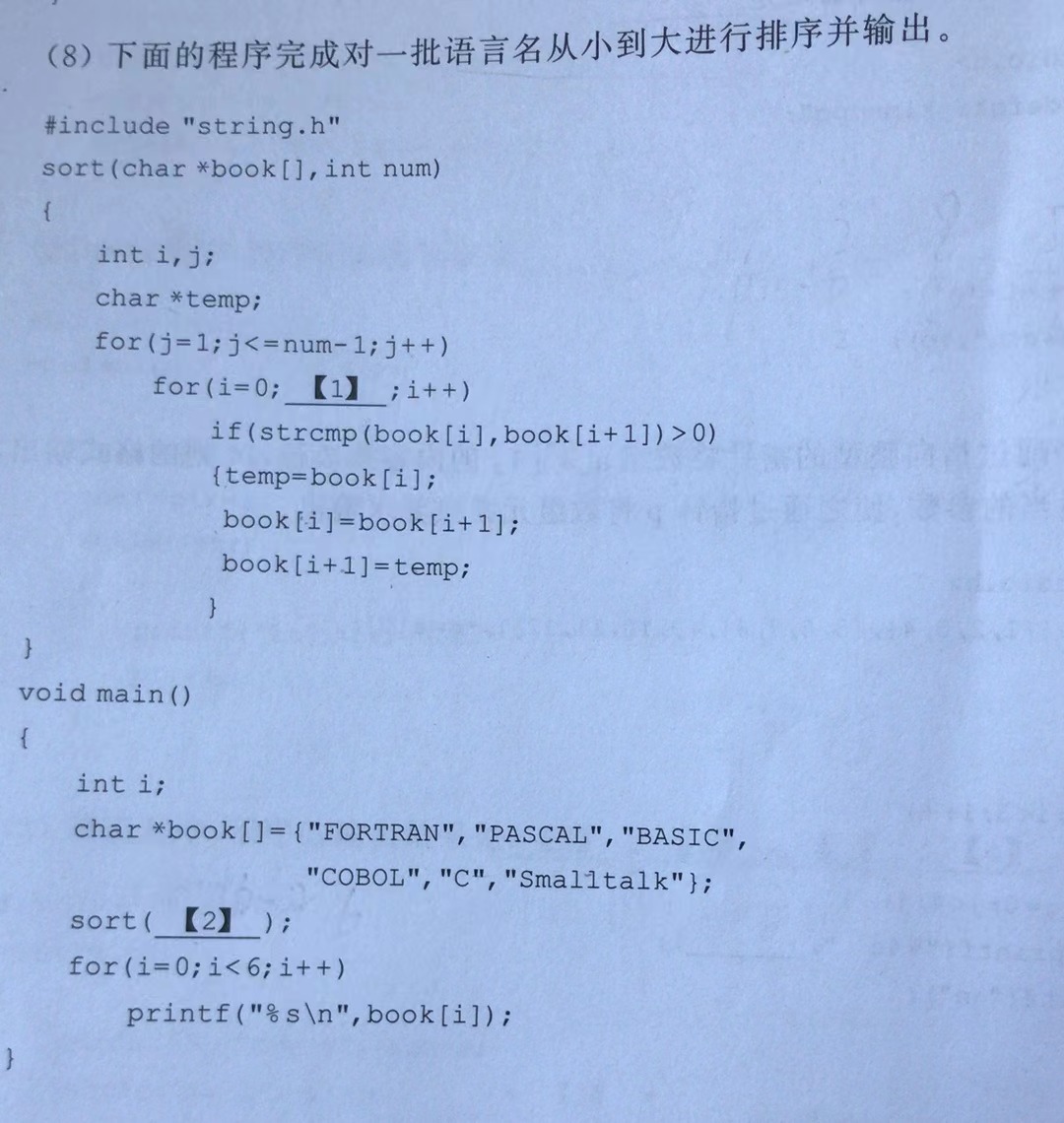
5.



1. i<=num-1-j
2. book,6

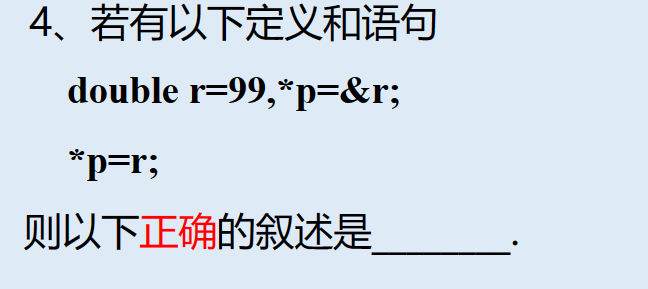
注意：【1】处用的是冒泡法，而不是选择排序法！

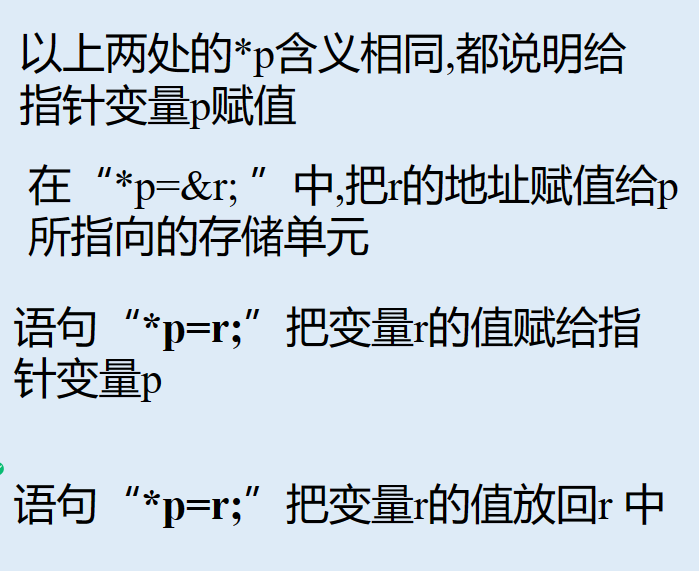
冒泡法的特征语句：



（前后两项比较）

6.若有以下定义和语句





A．

B．

C．

D．

D

1. 两处的“\*p”含义不同，第一处是定义指针变量p，第二处是给指针变量p当前地址所对应的变量赋值
2. 应该是：

把r的地址赋给指针变量p

1. 应该是：

 把变量r的值赋值给指针变量p所指向的存储单元

7.以下语句正确的是（）



A．

B．

C．

D．

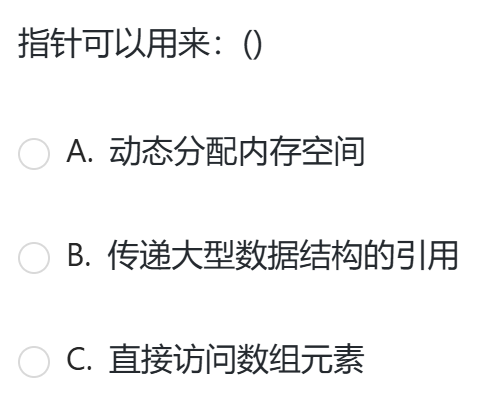
B

1. 指针变量p还没有被赋予地址

C．数组名不能进行运算，因此不能写“str++”。指针变量才能进行运算

D．系统会自动定义数组str的长度为5，不会自动在最后加上’\0’。因此，用printf %s法输出时，由于最后没有’\0’，会额外输出其他字符。

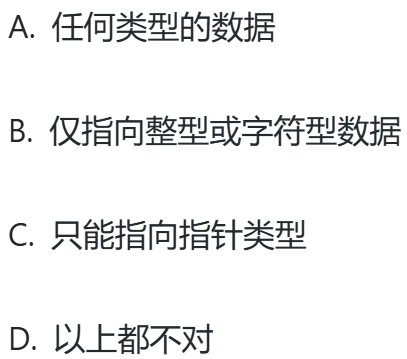
8.



A,B,C

9.

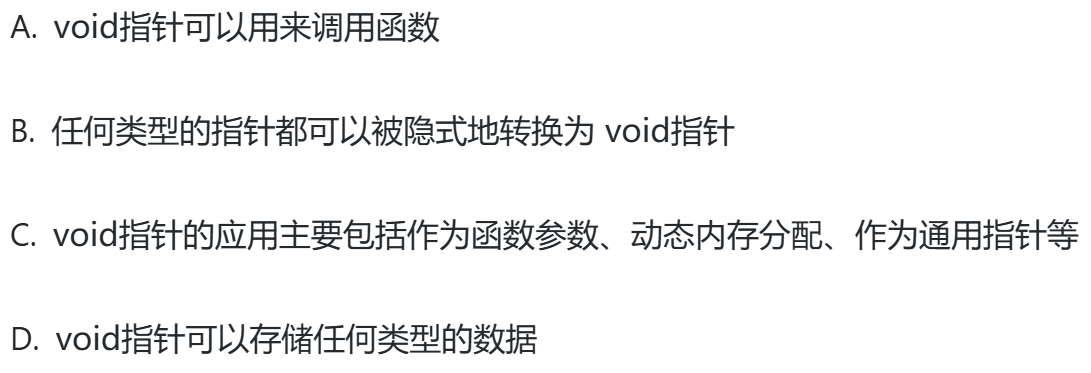




A

10.（多选题）





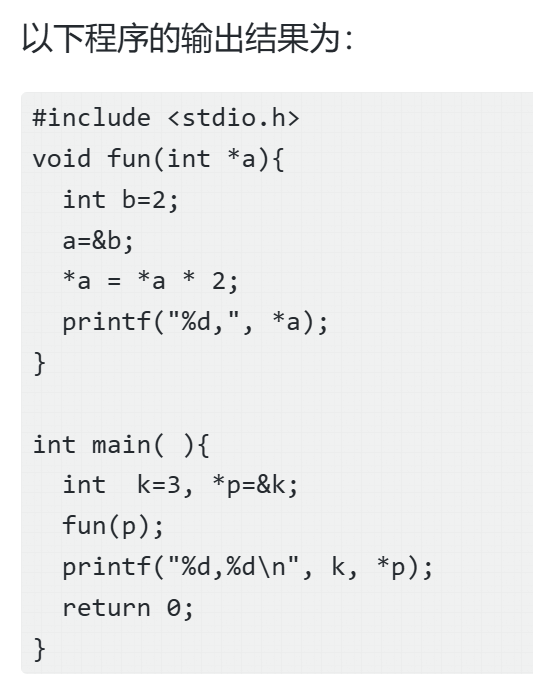
B,C

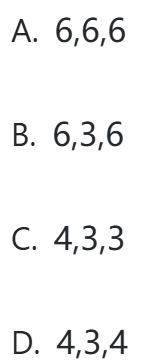
A. 错因：调用函数的指针应该要与该函数**有相同的函数类型、参数类型**

D. 错因：void 指针本身并不直接存储数据，它只是一个指向未知类型数据的指针。为了使用 void 指针，必须将其***转换为具体类型的指针才能访问数据***。

但是，void指针可以**指向**任何类型的数据

11.





C

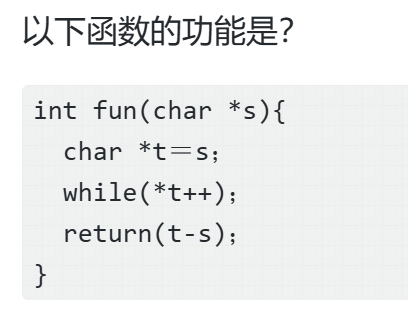
注意：指针变量（地址变量）的值是地址，

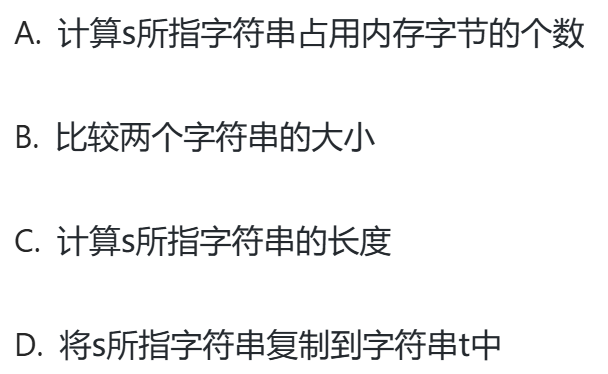
在fun函数中，“a=&b“改变了a的值，

但是在函数中，实参与形参是单向的值传递。

所以在main函数中，指针变量p的值并没有改变，仍然是&k

12.





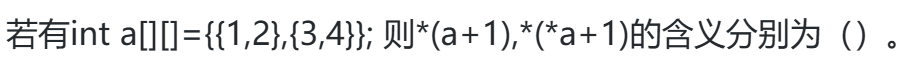
A

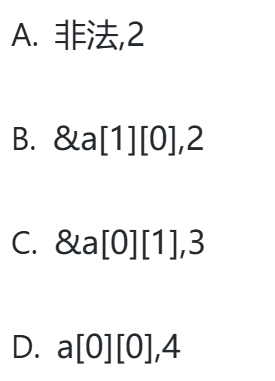
while(\*t++); 执行后t指向字符数组最后的空字符，

t-s得到首元素与末元素之间地址的差距，即为字符串占用的内存字节数

## 复杂指针

1.





答案：B

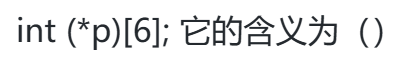
从内到外看

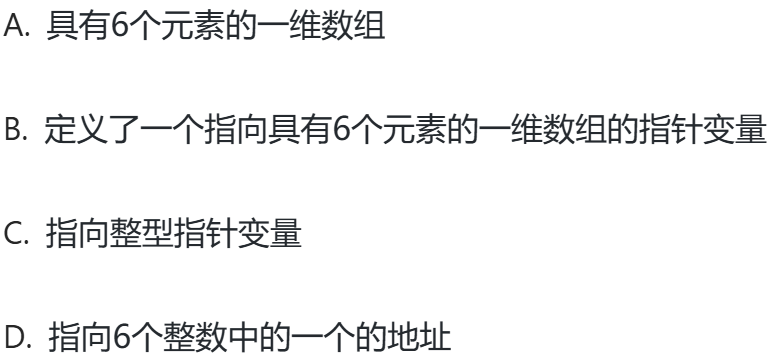
内层的\*是 取第一维度

外层的\*是 取出第二维度

备注：int a[ ][ ] 相当于 int \*\*p

2.

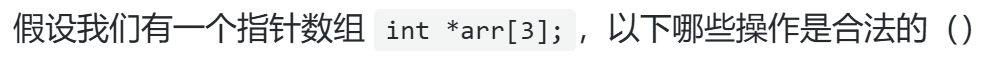


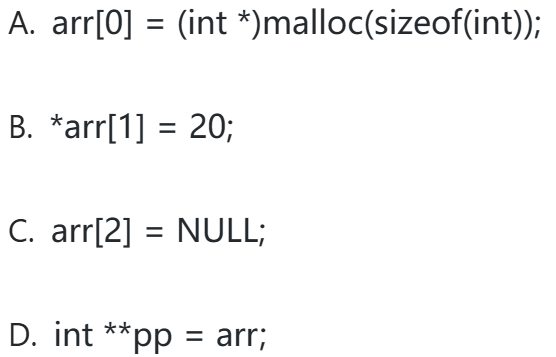


B

含义：一个指向 由6个元素构成的一维数组的 一个指针 ，称为：数组的指针

3.





A,B,C,D

int \*arr[3];

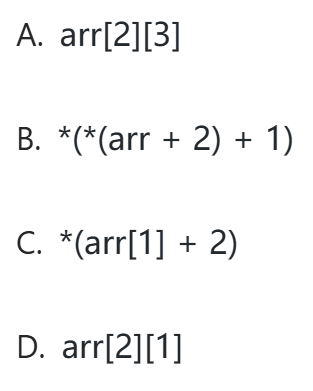
的含义是：含有3个指针变量的一个数组，称为：指针的数组

其中的每一个元素arr[0]，arr[1]，arr[2]都是指针变量

1. 正确，创建了一个内存空间，将其指针返回给arr[0]
2. 正确，\* arr[1]表示： arr[1]对应地址上的值，将20赋给它
3. 正确，将空指针NULL赋给arr[2]
4. 正确，arr表示指针数组名，将它赋给指针的指针pp

4.





C

方法有：

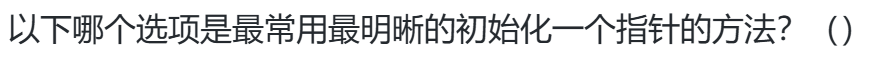
数组法：arr[1][2]

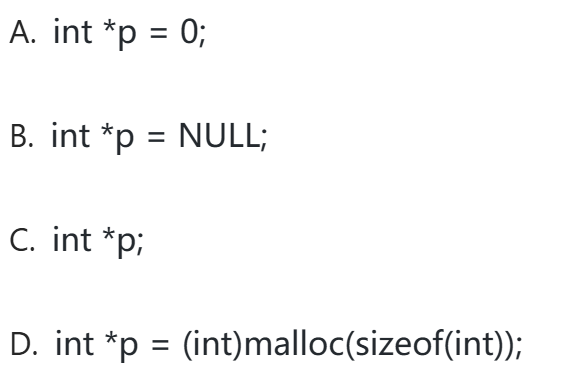
指针法：\*(\*(arr+1)+2)

混合法：\*(arr[1]+2)

所以选C

5．

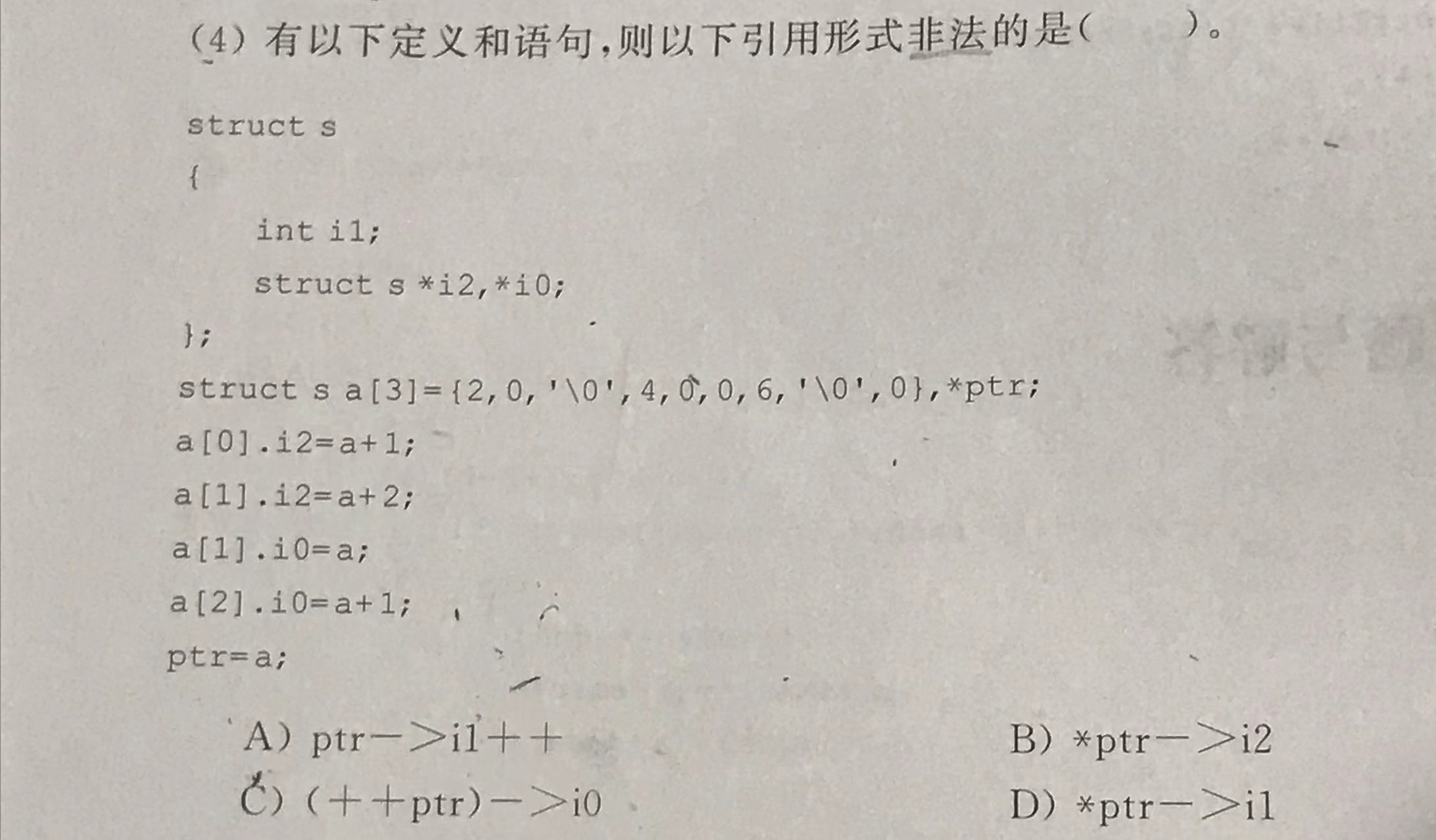




**B**.

用了NULL，防止p随意指向某个存储单元

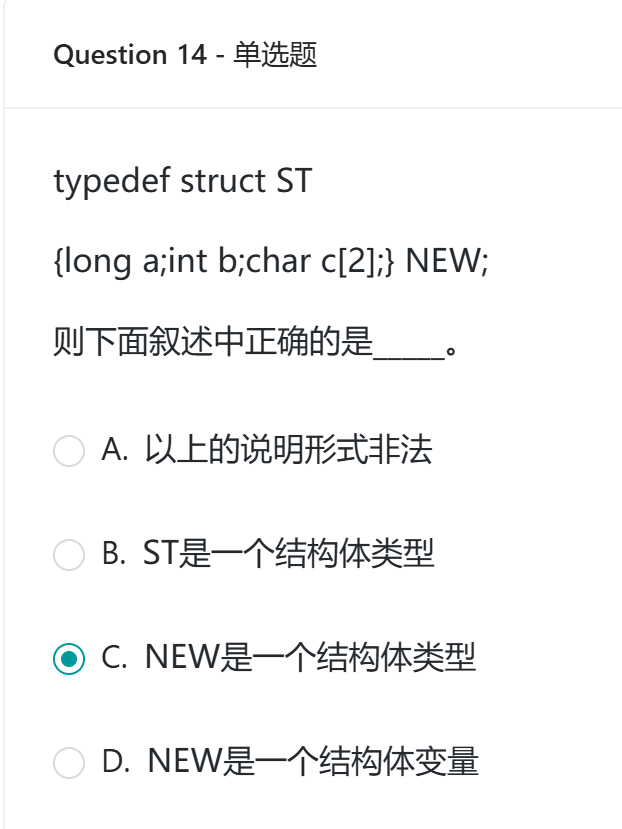
## 第八章 结构体

1. 

D

因为ptr->i1是整型变量2，不能进行\*运算

2.



 **A. 以上的说明形式非法**：这段代码是合法的，typedef 允许为结构体类型定义别名。

 **B. ST是一个结构体类型**：ST 只是结构体标签的名称，在代码中并没有定义为类型名。结构体类型的别名是 NEW。

 **C. NEW是一个结构体类型**：正确。通过 typedef，NEW 成为了结构体的类型别名。

 **D. NEW是一个结构体变量**：错。NEW 是类型别名，不是变量。

注意区分struct ST，ST，NEW：

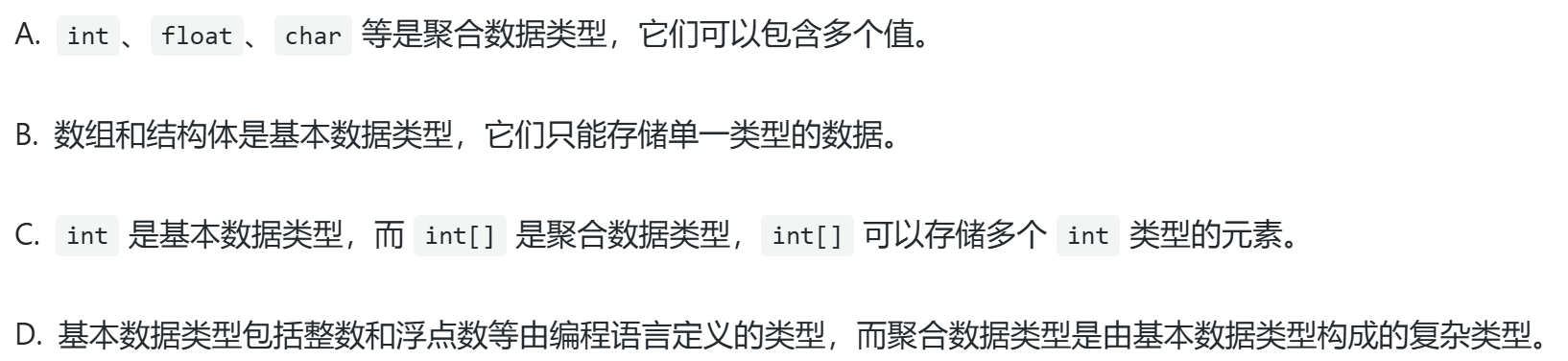
ST是“结构体标签的名称”，或者叫“结构体名”

“struct + 结构体名”是结构体类型，比如“struct ST”

NEW是结构体的类型别名，其实也是结构体类型

3.



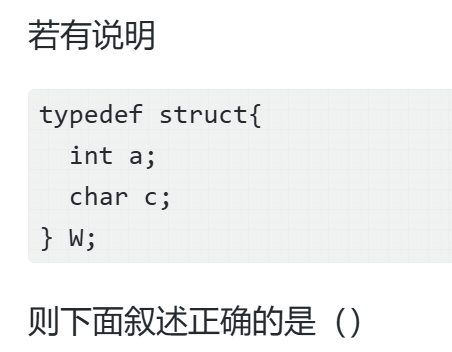


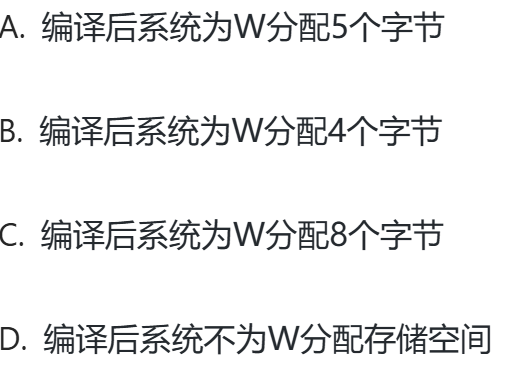
C,D

1. int float,char等是基本数据类型，只能包含一个值

B，C，D 数组、结构体都是聚合数据类型，可以存储多个基本数据类型

4.





D

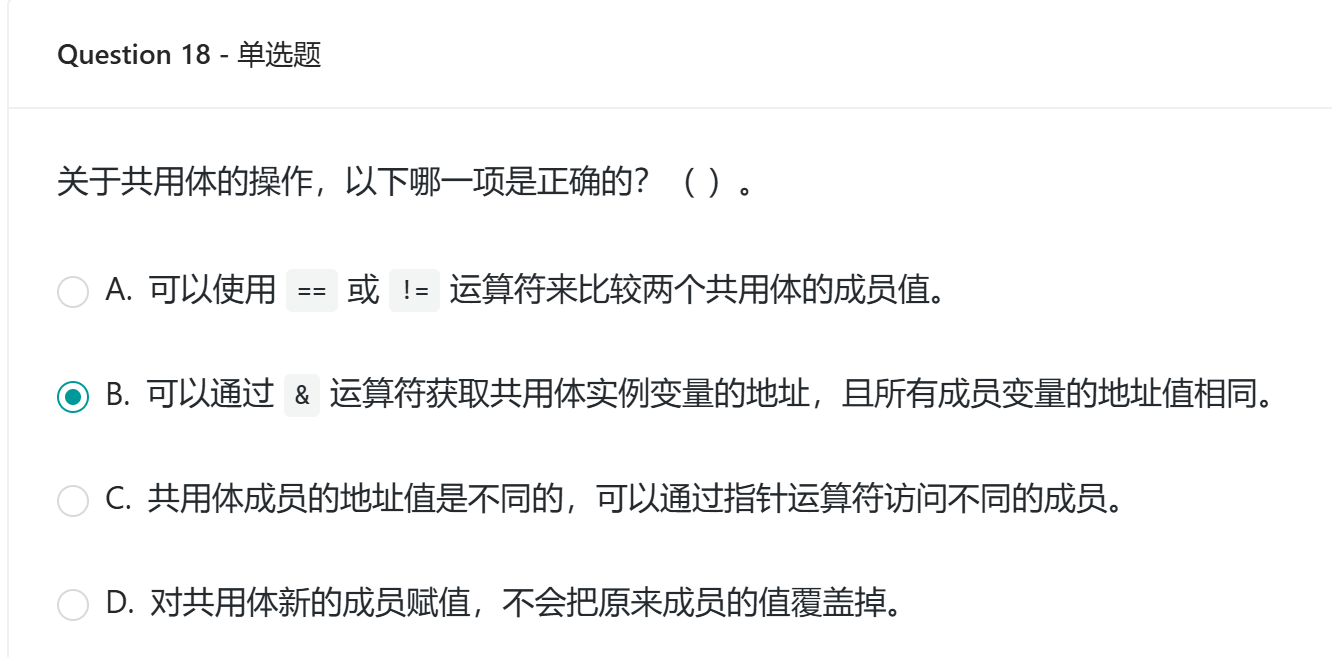
易错：

声明结构体时，这只是一个定义结构体的模板。所以，编译器***不会给它分配存储空间***

编译器只会在定义结构体变量时，给***结构体变量***分配存储空间

这题中是声明结构体。所以，编译器***不会给它分配存储空间***

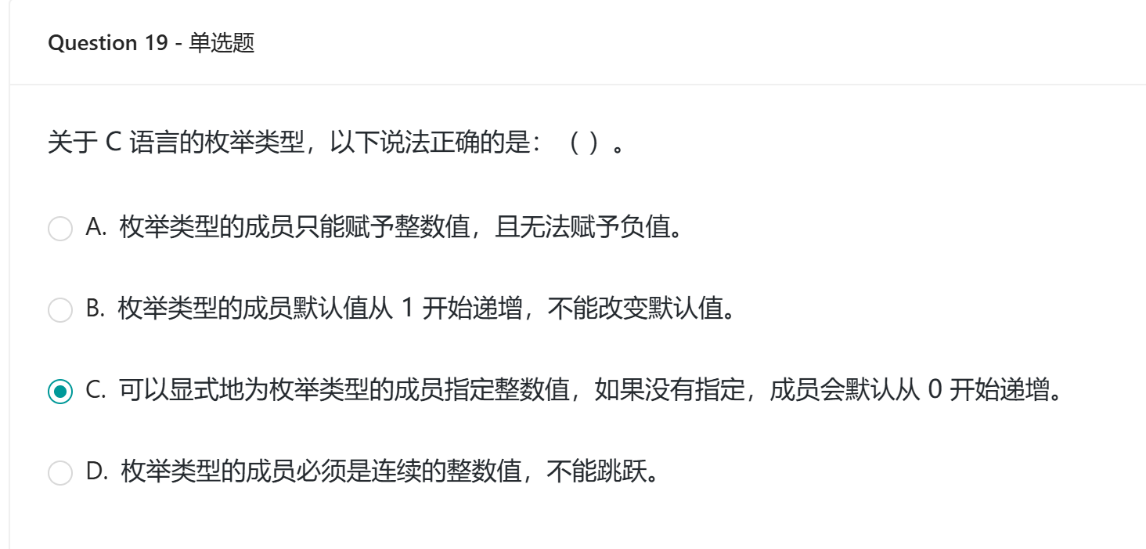
共用体



1. 不能。两个共用体的成员之间不能比较
2. 对
3. 共用体所有成员的地址值相同
4. 会覆盖

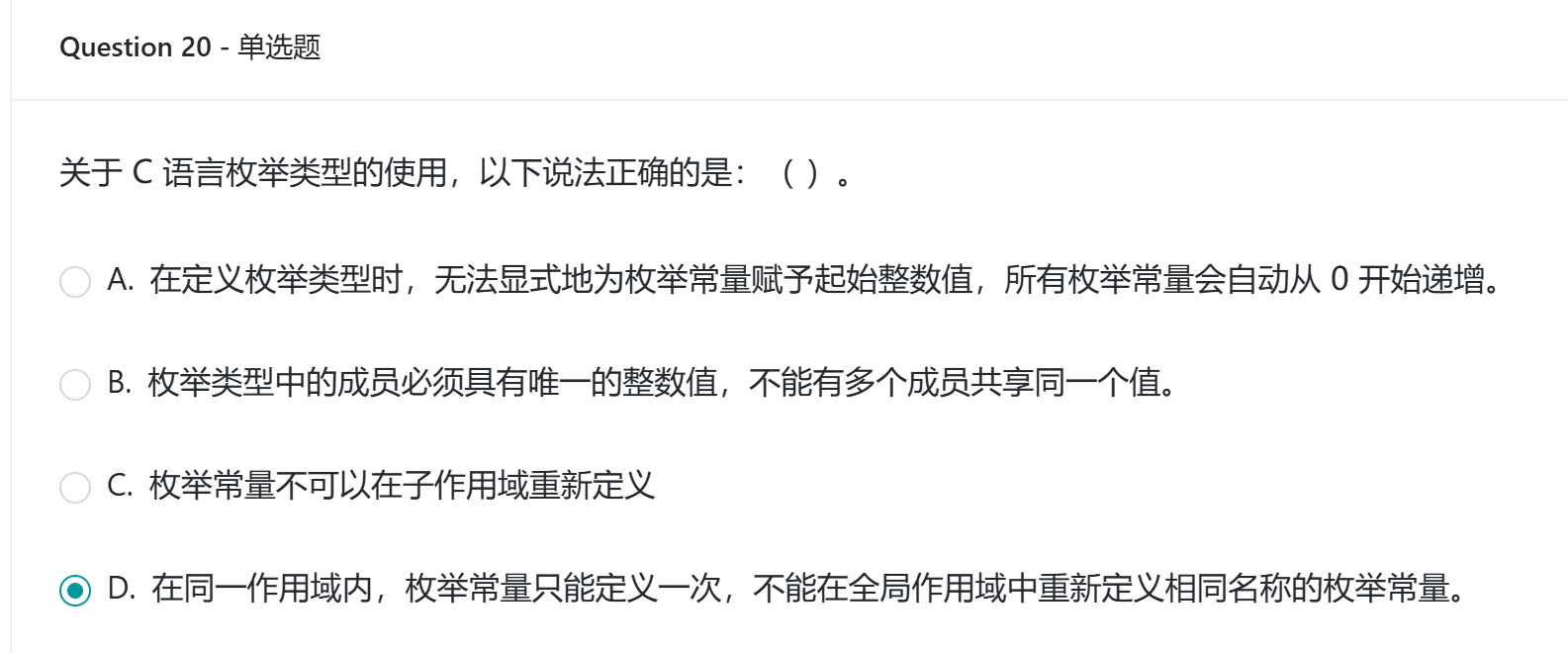
## 枚举类型

1.

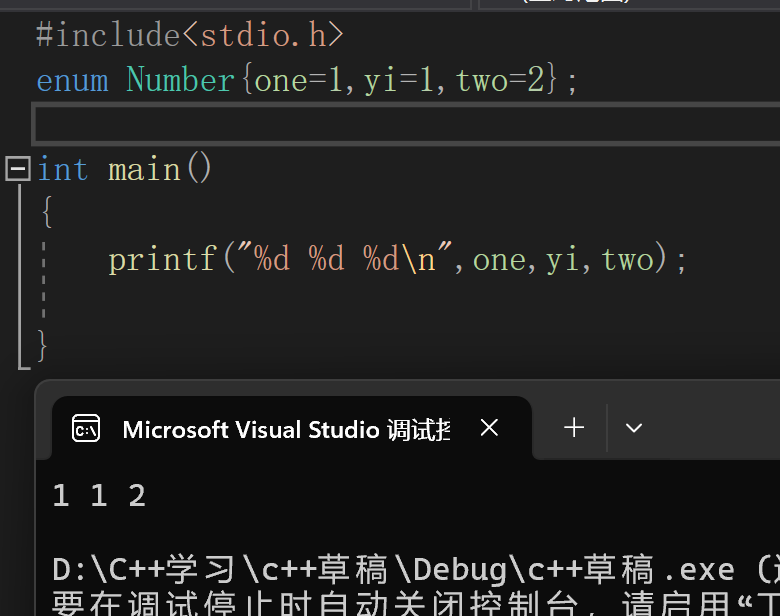


1. 可以是负值
2. 默认从0开始
3. 正确
4. 可以跳跃，不要求连续。如果要跳跃，则要人为指定，否则系统会按顺序赋予连续的整数值

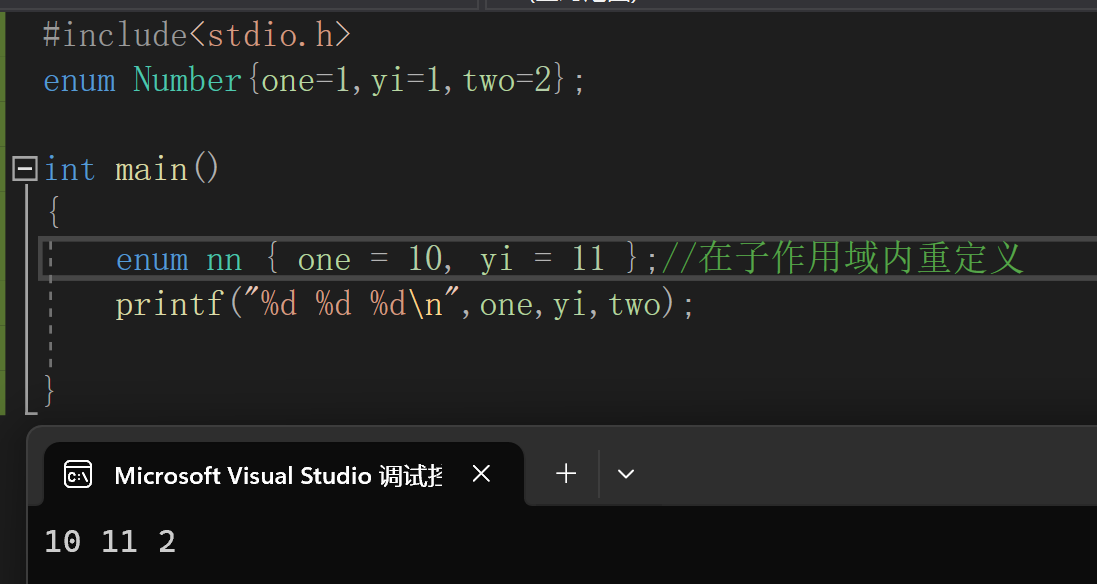
2.



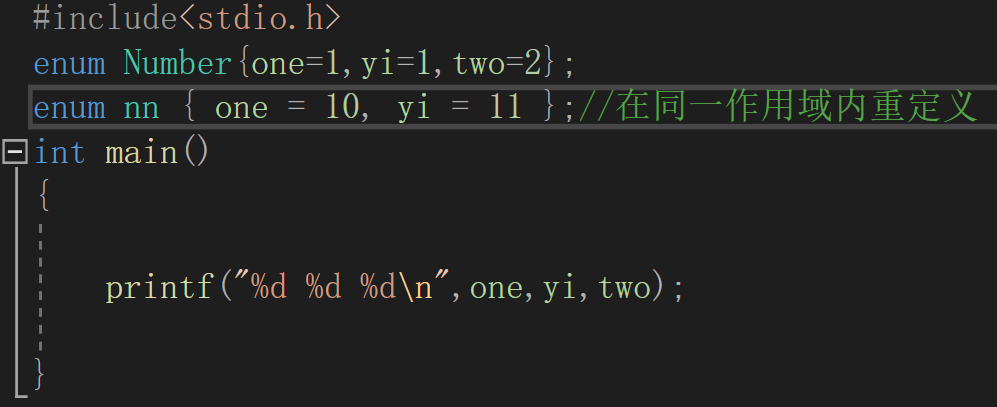
1. 可以显式赋值。若不显式赋值，则默认从0开始赋值
2. 可以有多个成员共享一个值，例如：



1. 可以在子作用域内重定义，例如：

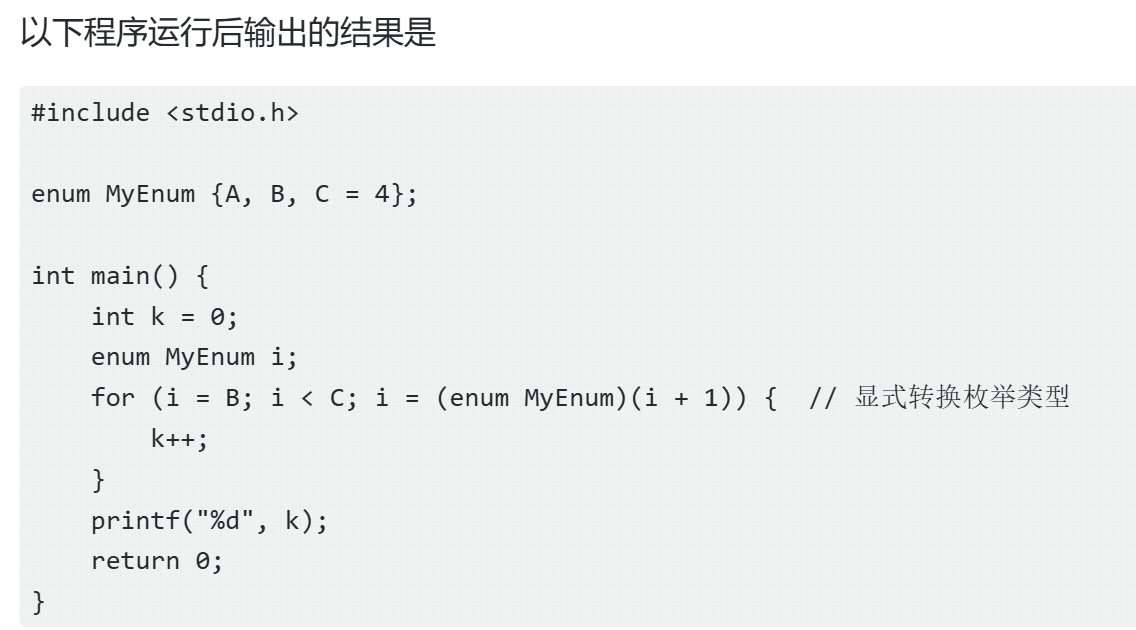


1. 不可以在同一作用域内重定义，例如：



会报错！

3.



答案：3

因为

“enum MyEnum{A,B,C=4};”

所以，A的值为0，B的值为1，C 的值为4

i从B开始，每次加1 到C（i<C）

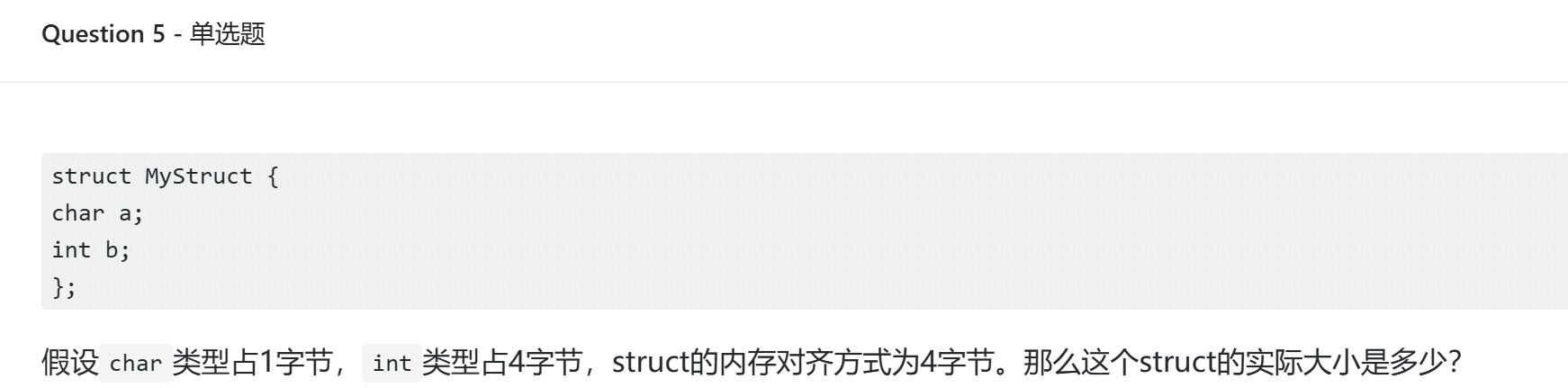
相当于i从1开始 每次加1 到4(i<4)

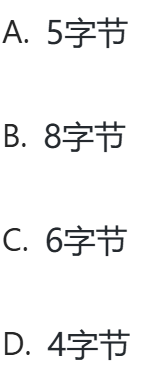
总共循环了3次

所以最终k=3

## 内存相关

1.

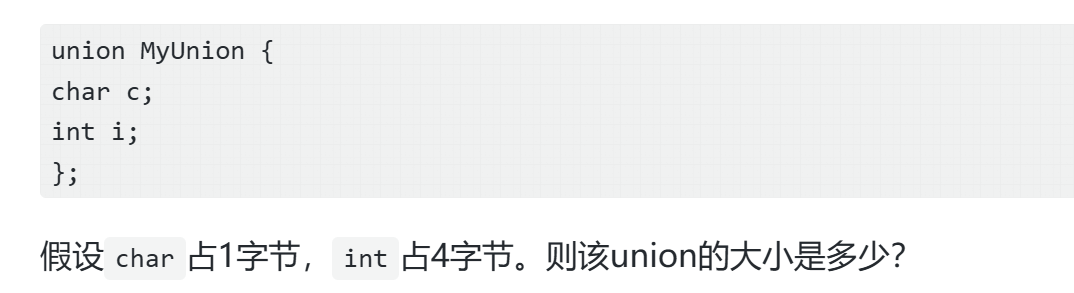




B

由于内存对齐方式是4字节，所以一共要放8字节

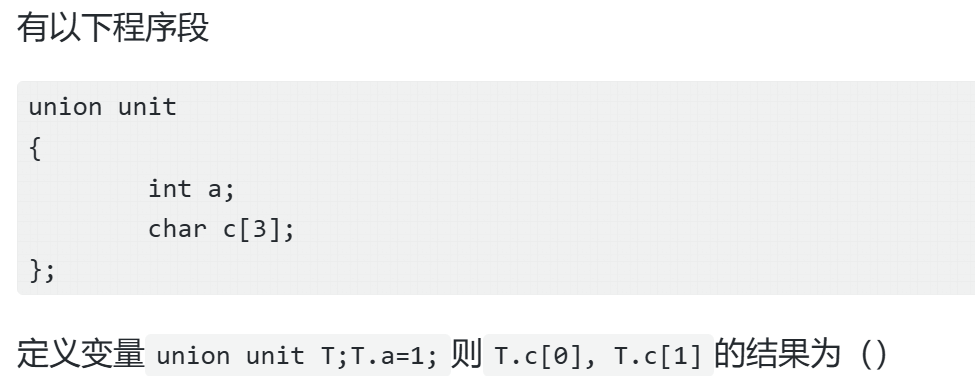
2.

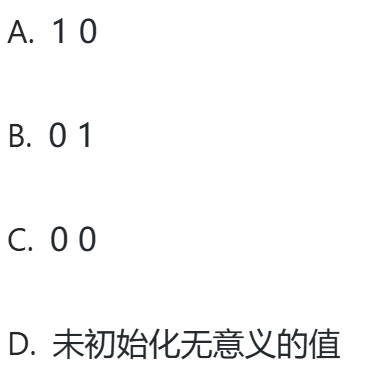


4字节

union占用的内存是所有成员的最大者，所以是4字节

3.已知内存按**小端编址，对齐方式是8字节**





A

因为内存按**小端编址，对齐方式是8字节**

在内存中数据的存放如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址（以字节为单位） | 0x00 | 0x01 | 0x02 | 0x03 |
| 数据 | 0x01 | 0x00 | 0x00 | 0x00 |
| 对应的变量 | int a | | | |
| 对应的变量 | c[0] | c[1] | c[2] | c[3] |

4. 已知内存按**小端编址，对齐方式是8字节**



答案：0

解释：

对于共用体，后面的赋值会把前面的赋值覆盖

所以内存中最后存放的数据是：e.nB=128

因为内存按**小端编址，对齐方式是8字节**

在内存中数据的存放如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址（以字节为单位） | 0x00 | 0x01 | 0x02 | 0x03 |
| 数据 | 0x80  =0b1000 000 | 0x00 | 0x00 | 0x00 |
| 对应的变量 | int nB | | | |
| 对应的变量 | char nA |  |  |  |

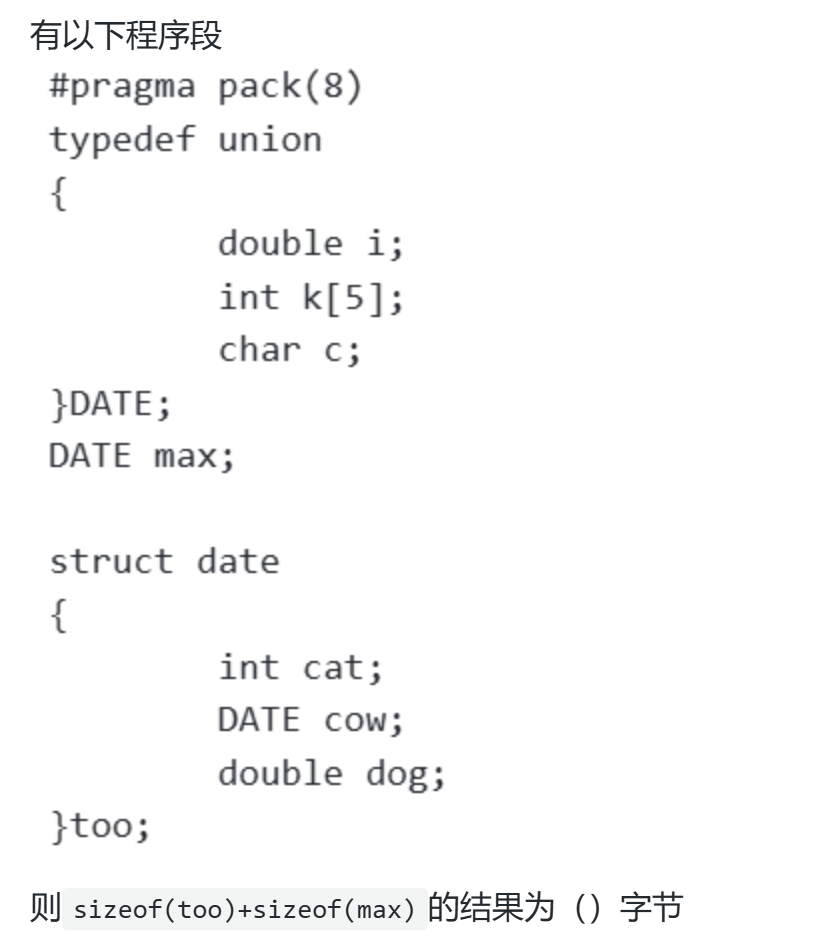
所以e.nB=128

因为char是***有符号的，占1字节***

而且0b1000 0000对应于（-128）的二进制补码，所以e.nA=-128

所以e.nB=e.nA+e.nB=(-128)+128=0

5. 已知内存按**小端编址，对齐方式是8字节**



答案：64

解释：因为 内存按小端编址，对齐方式是8字节

首先分析共用体DATE:

共用体所占的内存是 ***占用字节数最多的成员***所需的内存

其中占用字节数最多的成员是“int k[5]”,占5\*4字节=20字节

因为对齐方式是8字节，所以实际上需要24字节

内存中的分布图如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 存放的变量 | k[0] | | | | k[1] | | | |
| 地址 | 0x08 | 0x09 | 0x10 | 0x11 | 0x12 | 013 | 0x14 | 0x15 |
| 存放的变量 | k[2] | | | | k[3] | | | |
| 地址 | 0x16 | 0x17 | 0x18 | 0x19 | 0x20 | 0x21 | 0x22 | 0x23 |
| 存放的变量 | k[4] | | | | 空 | | | |

所以sizeof(max)=24

然后分析结构体too:

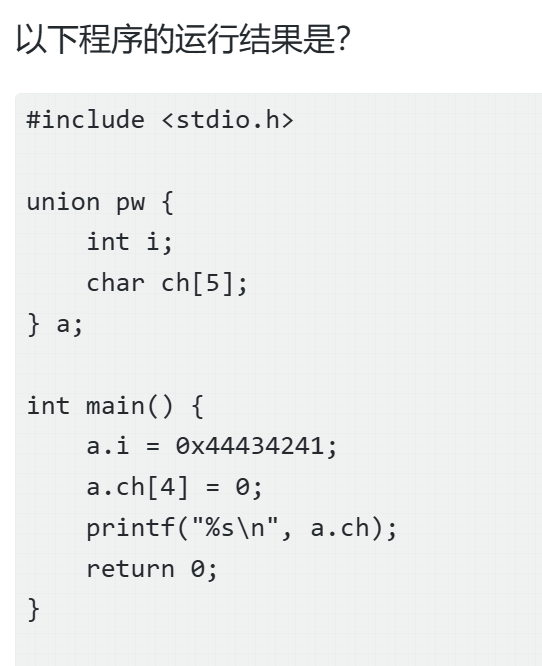
内存中的分布图如下：

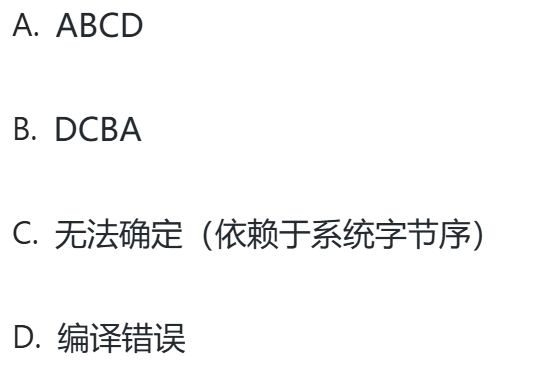
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| 存放的变量 | int cat | | | | 空 | | | |
| 地址 | 0x08 | 0x09 | 0x10 | 0x11 | 0x12 | 013 | 0x14 | 0x15 |
| 存放的变量 | k[0] | | | | k[1] | | | |
| 地址 | 0x16 | 0x17 | 0x18 | 0x19 | 0x20 | 0x21 | 0x22 | 0x23 |
| 存放的变量 | k[2] | | | | k[3] | | | |
| 地址 | 0x24 | 0x25 | 0x26 | 0x27 | 0x28 | 0x29 | 0x30 | 0x31 |
| 存放的变量 | k[4] | | | | 空 | | | |
| 地址 | 0x32 | 0x33 | 0x34 | 0x35 | 0x36 | 0x37 | 0x38 | 0x39 |
| 存放的变量 | double dog | | | | | | | |

所以，sizeof(too)=40

所以，sizeof(too)+sizeof(max)=40+24=64

6. 已知内存按**小端编址，对齐方式是8字节**





答案：A

执行

“a.i=0x44434241;

a.ch[4]=0;”之后

内存中的分布图如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 地址（以字节为单位） | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 |
| 数据 | 0x41 | 0x42 | 0x43 | 0x44 | 0x00 |
| 对应的变量 | int i | | | | 无 |
| 对应的变量 | ch[0] | ch[1] | ch[2] | ch[3] | ch[4] |

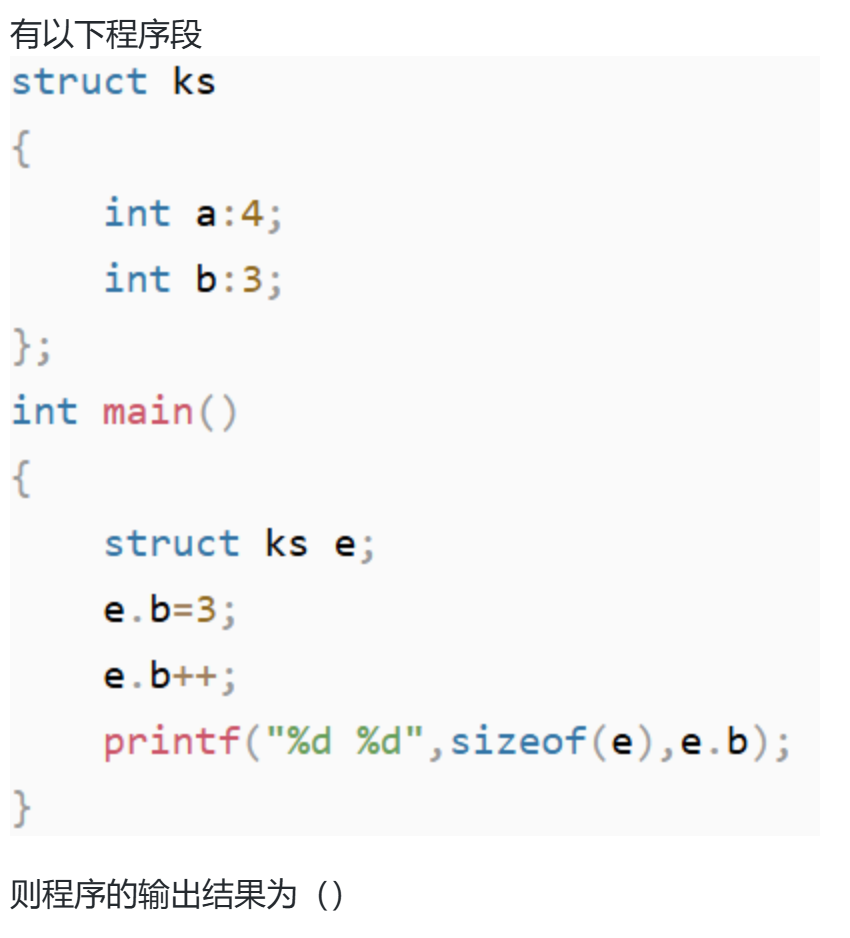
注意ch[4]=0，对应空字符

0x41~0x44分别对应于ASCII码中的A，B，C，D

所以输出结果为ABCD

## 位域

1.



答案：4 -4

结构体变量e占4个字节，所以sizeof(e)=4

e.b=3,

e.b++后，e.b应该等于4,

由于e.b只占3位，所以e.b在内存中以二进制补码表示为：100

又由于**e.b是int类型，是有符号的**，所以二进制补码表示的100对应-4

所以e.b=-4

补充：

如果**e.b是unsigned int类型，**

则是**无符号**的，

那么此时二进制补码表示的100对应4

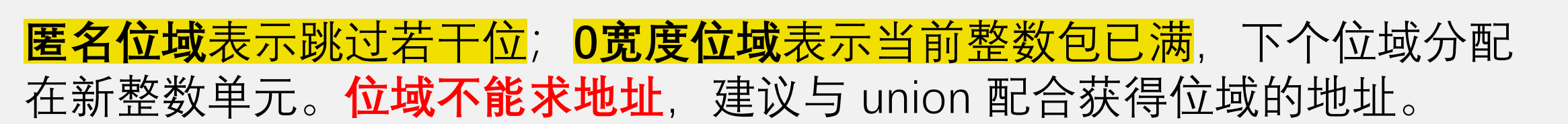
则 e.b=4

2.

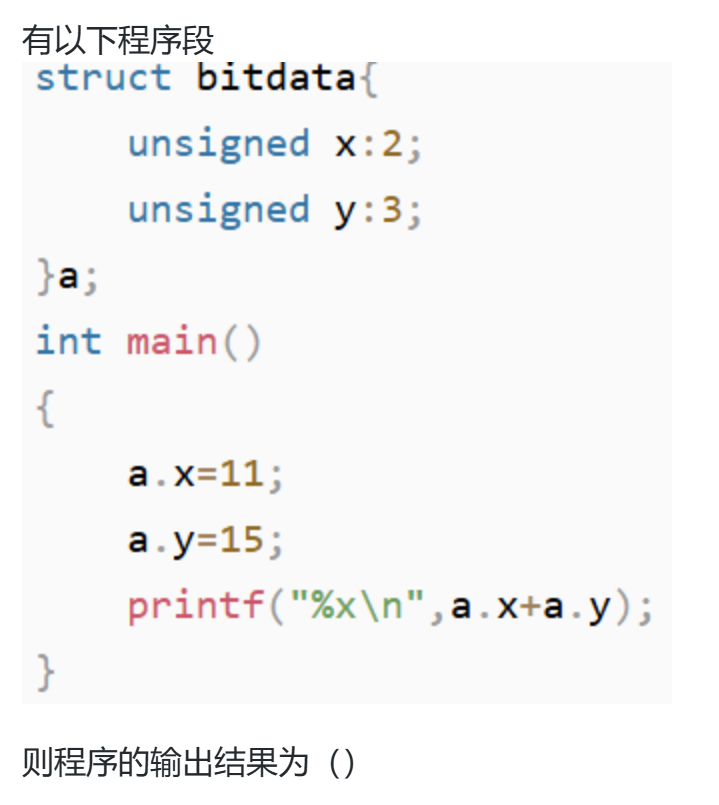


答案：8

解释：



3.



答案：a

注意：x只占2位，y只占3位

所以a.x=11执行后，a.x=3

所以a.y=15执行后，a.y=7

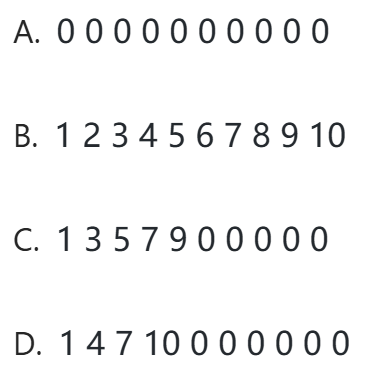
所以a.x+ a.y=10

对应的十六进制值为a

## 文件

1.





答案：D

注意：fread和fseek处的参数