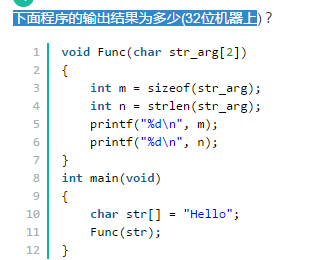
## 一、牛客网

1. 基础知识

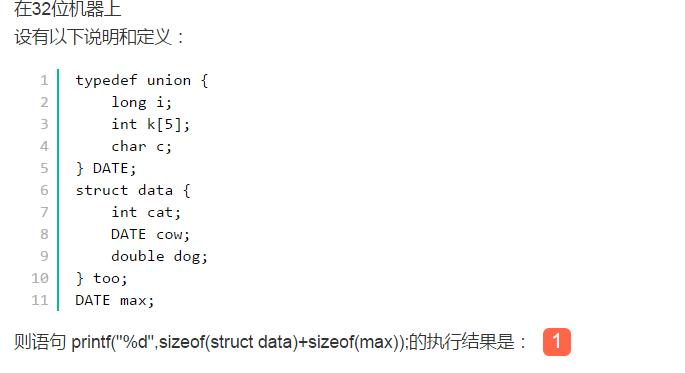
2015.8.5

1. new和malloc申请的在堆区,函数局部变量和函数参数在栈区,全局和静态变量在全局区(静态区)
2. ASCII码和十进制的对应关系0--48  9--57 A--65  Z—90 a--97   z—122 十进decimal，简称：DEC
3. Base64: base64的编码都是按字符串长度，以每3个8bit的字符为一组，然后针对每组，首先获取每个字符的ASCII编码，然后把ASCII编码转换成8bit二进制，得到24bit；再把这24bit每六个划分，高位用0补齐，得到4个8bit的字节，然后再将这4个8bit的字节转换成十进制，对照Base64编码表，得到对应编码后的字符。如果字符串的长度不能被3整出，多出来的部分用0补齐，对应的输出字符为“=
4. (1)基类的私有成员无论何种继承方式在派生类中均不可直接访问。  
   (2)在公有继承方式下，基类的保护和公有成员在派生类中均保持原访问属性。  
   (3)在保护继承方式下，基类的保护和公有成员在派生类中的访问属性均为保护属性。  
   (4)在私有继承方式下，基类的保护和公有成员在派生类中的访问属性均为私有属性。
5. 标准ASCII编码是7位编码
6. char\*pa[7] pa是一个具有7个元素的指针数组，每个元素是一个char型指针
7. sizeof是运算符而strlen()是函数

char str[20]="0123456789";   
int a=strlen(str); //a=10; >>>> strlen 计算字符串的长度，以结束符 0x00 为字符串结束。   
int b=sizeof(str); //而b=20; >>>> sizeof 计算的则是分配的数组 str[20] 所占的内存空间的大小，不受里面存储的内容改变。

输出结果是4,5

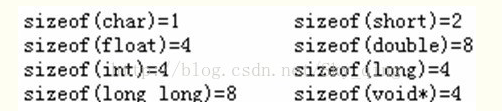
Sizeof是运算符是求类型的大小， 传给他的类型是指针啊他不会遍历的 只是求指针大小 ，指针的作用是寻址啊。32位系统 是32位的地址线。要对所有地址都能访问到，需要4字节 就是 32位 。



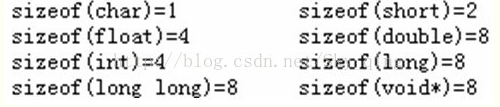
1. 堆栈溢出一般是循环的递归调用导致的,如果使用的大数据结构的局部变量,也可能导致

2015.8.7

1. 32位系统下



1. 64位系统下



1. 如果只是论一个汉字占用的字节数，那么 UTF-8 占用3个字节， UTF-16 占用2个字节。但是如果存储文本的话，需要在文本使用 EF BB BF 三个字节表示使用 UTF-8 编码，使用 FE FF 表示使用 UTF-16 编码。
2. 关于源码反码和补码<http://www.cnblogs.com/zhangziqiu/archive/2011/03/30/computercode.html>

机器数：十进制数+3，计算机字长是8位，转成二进制是00000011，如果是-3则是：10000011，就是机器数。

原码：

反码：正数的反码是其本身，负数的反码是符号位不变，其余各位取反

补码：正数的补码是其本身，负数的补码是在其原来的基础上符号位不变，其余各位取反然后再加上1

1. ASCII码是7位编码，编码范围是0x00-0x7F。ASCII字符集包括英文字母、阿拉伯数字和标点符号等字符。其中0x00-0x20和0x7F共33个控制字符。
2. A extern 外部变量，可供所以源文件使用

B，register寄存器变量，放在寄存器而非内存中，效率更高，一般是临时变量

C，auto自动变量，所有未加 static 关键字的都默认是 auto 变量，也就是我们普通的变量

D，static静态变量，在内存中只存在一个，可供当前源文件的所有函数使用

2015年8月16日

（1）关于++i 和i++

#include "stdio.h"

int main()

{

int a, b, c, d;

a = 10;

b = a++;

c = ++a;

d = 10 \* a++;

printf("b,c,d:%d,%d,%d", b, c, d);

return 0;

}

**b,c,d:10,12,120**

**a++是当前语句中不执行++,知道遇到下一关于a的操作，就先执行++**

**（1）float、double、long double等类型不允许直接进行位与操作符啊，可用间接的方法变通，如float取地址（也是&符号）转换为unsigned int类型，再用取值操作符（\*），这样编译器会以为是unsigned int类型。**

**（2）使用int、short、long移位时最好加上unsigned，这样就是汇编中逻辑移位（即全部移位），如果不加unsigned情况就较为复杂，正数全是逻辑移位，负数左移时保持符号位为1、右边补0，负数右移时保持符号位为1，左边补1，所以-1不管怎么右移都是-1。**

**（3）位与操作符就是将两个数进行与操作，&0x01就相当于掩码取出最后一位，其他位置成0**

**（4）位与操作符&的操作优先级小于移位操作符，但移位操作符小于取地址操作符&（取值操作符\*），所以上面代码不会出错**

**全局变量和static修饰的变量，可以自动进行初始化**

概念解释：广义表L=(A,B,C),表头是A，表尾是（B,C)，这是定义。

tail()表示取字符串的尾部，head()表示取字符串的头，concat()表示字符串拼接。

取两次尾部一次头部，然后合并，答案是C。