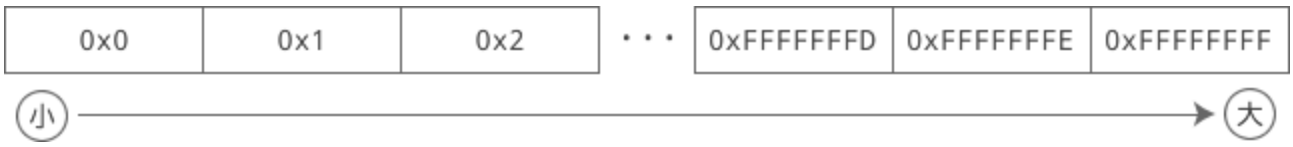
1 分钟彻底理解 C 语言指针的概念

计算机中所有的数据都必须放在内存中，不同类型的数据占用的字节数不一样，例如 int 占用 4 个字节，char 占用 1 个字节。为了正确地访问这些数据，必须为每个字节都编上号码，就像门牌号、身份证号一样，每个字节的编号是唯一的，根据编号可以准确地找到某个字节。

下图是 4G 内存中每个字节的编号（以十六进制表示）：



我们将内存中字节的编号称为地址（Address）或指针（Pointer）。地址从 0 开始依次增加，对于 32 位环境，程序能够使用的内存为 4GB，最小的地址为 0，最大的地址为 0XFFFFFFFF。

下面的代码演示了如何输出一个地址：

```
#include <stdio.h>

int main(){
    int a = 100;
    char str[20] = "c.biancheng.net";
    printf("%#X, %#X\n", &a, str);
    return 0;
}
```

运行结果：

0X28FF3C, 0X28FF10

`%#X` 表示以十六进制形式输出，并附带前缀 `0X`。a 是一个变量，用来存放整数，需要在前面加`&`来获得它的地址；str 本身就表示字符串的首地址，不需要加`&`。

一切都是地址

C 语言用变量来存储数据，用函数来定义一段可以重复使用的代码，它们最终都要放到内存中才能供 CPU 使用。

数据和代码都以二进制的形式存储在内存中，计算机无法从格式上区分某块内存到底存储的是数据还是代码。当程序被加载到内存后，操作系统会给不同的内存块指定不同的权限，拥有读取和执行权限的内存块就是代码，而拥有读取和写入权限（也可能只有读取权限）的内存块就是数据。

CPU 只能通过地址来取得内存中的代码和数据，程序在执行过程中会告知 CPU 要执行的代码以及要读写的数据的地址。如果程序不小心出错，或者开发者有意为之，在 CPU 要写入数据时给它一个代码区域的地址，就会发生内存访问错误。这种内存访问错误会被硬件和操作系统拦截，强制程序崩溃，程序员没有挽救的机会。

CPU 访问内存时需要的是地址，而不是变量名和函数名！变量名和函数名只是地址的一种助记符，当源文件被编译和链接成可执行程序后，它们都会被替换成地址。编译和链接过程的一项重要任务就是找到这些名称所对应的地址。

假设变量 a、b、c 在内存中的地址分别是 0X1000、0X2000、0X3000，那么加法运算 `c = a + b;` 将会被转换成类似下面的形式：

```
0X3000 = (0X1000) + (0X2000);
```

`( )` 表示取值操作，整个表达式的意思是，取出地址 0X1000 和 0X2000 上的值，将它们相加，把相加的结果赋值给地址为 0X3000 的内存

变量名和函数名为我们提供了方便，让我们在编写代码的过程中可以使用易于阅读和理解的英文字符串，不用直接面对二进制地址，那场景简直让人崩溃。

需要注意的是，虽然变量名、函数名、字符串名和数组名在本质上是一样的，它们都是地址的助记符，但在编写代码的过程中，我们认为变量名表示的是数据本身，而函数名、字符串名和数组名表示的是代码块或数据块的首地址。