编译器为每个变量分配地址（左值），这个地址在编译时可知且一直存在，而它的右值在运行时才能知道。

通俗说，每个变量都有一个地址，这个地址在编译时可以知道，而地址里存储的内容（就是变量的右值）只有在运行时才能知道。

1. 从内存访问形式上，数组与指针的区别
   1. 数组访问方式：每个符号的地址在编译时可知，如果编译器需要一个地址（比如加上偏移量）来执行某种操作，数组可以直接操作，而对于指针，必须在运行时获取它的地址，然后才能对他进行操作。

|  |
| --- |
| IMG_256 |

* 1. 指针访问形式：若声明一个指针，如extern char \*p，它表示p指向一个字符，为了取得这个字符，必须知道地址p的内容，把它作为字符的地址并从这个地址中取得这个字符。

|  |
| --- |
| IMG_256 |

* 1. 定义为指针，但是以数组方式引用

|  |
| --- |
| IMG_256 |

所有作为函数参数的数组名在编译时会自动转换为指针，即退化为指针意思。

什么时候数组和指针是相同的，在C语言标准中，对此有如下说明

1)表达式中的数组名（与声明不同）被编译器当作一个指向该数组第一个元素的指针；在表达式中，指针和数组是可以互换的，因为他们在编译器中最终形式都是指针。如何理解这句话，请看下面的这个例子。

例子：

1: int a[10],\*p,i = 2;

可以通过以下任一种方式访问a[i]

|  |  |  |
| --- | --- | --- |
| p=a;  p[i]; | p = a;  \*(p+i); | p = a+i;  \*p; |

原因是对数组的引用如a[i]在编译时总是被编译器改写成\*(a+i)的形式。

2)下标总是与指针的偏移量相同；

             C语言中，把数组下标作为指针的偏移量。根本原因是指针和偏移量是底层硬件所使用的基本模型。

1. 在函数参数的声明中，数组名被编译器当作指向该数组第一个元素的指针；

      理解这句话首先要明白形参和实参的区别，

|  |  |  |
| --- | --- | --- |
| 术语 | 定义 | 例子 |
| 形参(parameter) | 它是一个变量，在函数定义或者函数声明的原型中定义。又称“形式参数（formal parameter）” | int power(int base,int n); base 和 n 都是形参 |
| 实参(argument) | 在实际调用一个函数时所传递给函数的值。又称“实际参数（actual parameter）” | i = power(10,j); 10和j都是实参。 |

在函数参数这种特殊情况下，编译器必须把数组名当作指向该数组第一个元素的指针形式。编译器只向函数传递数组的地址，而不是整个数组的拷贝。