

## C语言指针变量的运算

<上一节

下一节>

关注我们：

微信公众号

新浪微博

QQ交流群：588321099

C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！

### 教程目录

- 1 编程基础
- 2 C语言初探
- 3 变量和数据类型
- 4 输入输出
- 5 分支结构和循环结构
- 6 C语言数组
- 7 C语言函数
- 8 预处理命令
- 9 C语言指针
  - 9.1 1分钟彻底理解指针的概念
  - 9.2 大话C语言指针变量
  - 9.3 C语言指针变量的运算
  - 9.4 数组指针（指向数组的指针）
  - 9.5 字符串指针（指向字符串的指针）
  - 9.6 C语言数组灵活多变的访问形式
  - 9.7 指针变量作为函数参数
  - 9.8 用C语言指针作为函数返回值
  - 9.9 二级指针（指向指针的指针）
  - 9.10 空指针NULL以及void指针
  - 9.11 注意，数组和指针绝不等价
  - 9.12 数组在什么时候会转换为指针
  - 9.13 指针数组（每个元素都是指针）
  - 9.14 一道题目教你玩转指针数组
  - 9.15 指针与二维数组
  - 9.16 函数指针（指向函数的指针）
  - 9.17 只需一招，彻底攻克C语言指针
  - 9.18 用main()函数接收控制台数据
  - 9.19 对C语言指针的总结
- 10 结构体、位运算以及其他
- 11 文件操作
- 12 C语言调试

指针变量保存的是地址，本质上是一个整数，可以进行部分运算，例如加法、减法、比较等，请看下面的代码：

```
01. #include <stdio.h>
02.
03. int main(){
04.     int    a = 10,    *pa = &a, *paa = &a;
05.     double b = 99.9, *pb = &b;
06.     char   c = 'a',  *pc = &c;
07.     //最初的值
08.     printf("&a=%#X, &b=%#X, &c=%#X\n", &a, &b, &c);
09.     printf("pa=%#X, pb=%#X, pc=%#X\n", pa, pb, pc);
10.     //加法运算
11.     pa++; pb++; pc++;
12.     printf("pa=%#X, pb=%#X, pc=%#X\n", pa, pb, pc);
13.     //减法运算
14.     pa -= 2; pb -= 2; pc -= 2;
15.     printf("pa=%#X, pb=%#X, pc=%#X\n", pa, pb, pc);
16.     //比较运算
17.     if(pa == paa){
18.         printf("%d\n", *paa);
19.     }else{
20.         printf("%d\n", *pa);
21.     }
22.     return 0;
23. }
```

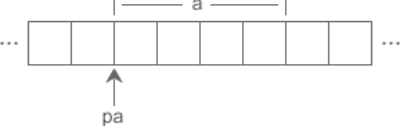
运行结果：

```
&a=0X28FF44, &b=0X28FF30, &c=0X28FF2B
pa=0X28FF44, pb=0X28FF30, pc=0X28FF2B
pa=0X28FF48, pb=0X28FF38, pc=0X28FF2C
pa=0X28FF40, pb=0X28FF28, pc=0X28FF2A
2686784
```

从运算结果可以看出：pa、pb、pc 每次加 1，它们的地址分别增加 4、8、1，正好是 int、double、char 类型的长度；减 2 时，地址分别减少 8、16、2，正好是 int、double、char 类型长度的 2 倍。

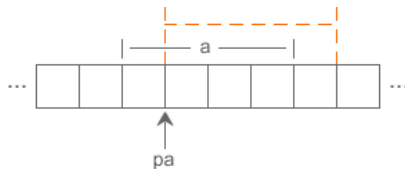
这很奇怪，指针变量加减运算的结果跟数据类型的长度有关，而不是简单地加 1 或减 1，这是为什么呢？

以 a 和 pa 为例，a 的类型为 int，占用 4 个字节，pa 是指向 a 的指针，如下图所示：



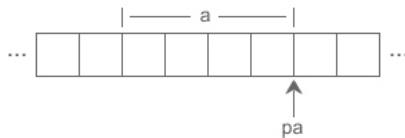
刚开始的时候，pa 指向 a 的开头，通过 \*pa 读取数据时，从 pa 指向的位置向后移动 4 个字节，把这 4 个字节的内容作为要获取的数据，这 4 个字节也正好是变量 a 占用的内存。

如果 pa++，使得地址加 1 的话，就会变成如下图所示的指向关系：



这个时候 `pa` 指向整数 `a` 的中间，`*pa` 使用的是红色虚线画出的 4 个字节，其中前 3 个是变量 `a` 的，后面 1 个是其它数据的，把它们“搅和”在一起显然没有实际的意义，取得的数据也会非常怪异。

如果 `pa++` 使得地址加 4 的话，正好能够完全跳过整数 `a`，指向它后面的内存，如下图所示：



我们知道，数组中的所有元素在内存中是连续排列的，如果一个指针指向了数组中的某个元素，那么加 1 就表示指向下一个元素，减 1 就表示指向上一个元素，这样指针的加减运算就具有了现实的意义，我们将在《C语言和数组》一节中深入探讨。

不过C语言并没有规定变量的存储方式，如果连续定义多个变量，它们有可能是挨着的，也有可能是分散的，这取决于变量的类型、编译器的实现以及具体的编译模式，所以对于指向普通变量的指针，我们往往不进行加减运算，虽然编译器并不会报错，但这样做没有意义，因为不知道它后面指向的是什么数据。

下面的例子是一个反面教材，警告读者不要尝试通过指针获取下一个变量的地址：

```
01. #include <stdio.h>
02.
03. int main(){
04.     int a = 1, b = 2, c = 3;
05.     int *p = &c;
06.     int i;
07.     for(i=0; i<8; i++){
08.         printf("%d, ", *(p+i) );
09.     }
10.     return 0;
11. }
```

在 VS2010 Debug 模式下的运行结果为：

```
3, -858993460, -858993460, 2, -858993460, -858993460, 1, -858993460,
```

可以发现，变量 `a`、`b`、`c` 并不挨着，它们中间还参杂了别的辅助数据。

指针变量除了可以参与加减运算，还可以参与比较运算。当对指针变量进行比较运算时，比较的是指针变量本身的值，也就是数据的地址。如果地址相等，那么两个指针就指向同一份数据，否则就指向不同的数据。

上面的代码（第一个例子）在比较 `pa` 和 `paa` 的值时，`pa` 已经指向了 `a` 的上一份数据，所以它们不相等。而 `a` 的上一份数据又不知道是什么，所以会导致 `printf()` 输出一个没有意义的数，这正好印证了上面的观点，不要对指向普通变量的指针进行加减运算。

另外需要说明的是，不能对指针变量进行乘法、除法、取余等其他运算，除了会发生语法错误，也没有实际的含义。

C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！

编程帮，一个分享编程知识的公众号。跟着站长一起学习，每天都有进步。

通俗易懂，深入浅出，一篇文章只讲一个知识点。

文章不深奥，不需要钻研，在公交、在地铁、在厕所都可以阅读，随时随地涨姿势。

文章不涉及代码，不烧脑细胞，人人都可以学习。

当你决定关注「编程帮」，你已然超越了90%的程序员！



微信扫描二维码关注

<上一节

下一节>

关注我们： 微信公众号

新浪微博

QQ交流群：588321099

