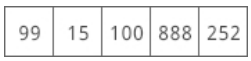
C语言数组指针（指向数组的指针）

<上一节 下一节> 关注我们： 微信公众号 新浪微博 QQ交流群：588321099

数组（Array）是一系列具有相同类型的数据的集合，每一份数据叫做一个数组元素（Element）。数组中的所有元素在内存中是连续排列的，整个数组占用的是一块内存。以 `int arr[] = { 99, 15, 100, 888, 252 };` 为例，该数组在内存中的分布如下图所示：



定义数组时，要给出数组名和数组长度，数组名可以认为是一个指针，它指向数组的第 0 个元素。在C语言中，我们将第 0 个元素的地址称为数组的首地址。以上面的数组为例，下图是 arr 的指向：

> 数组名的本意是表示整个数组，也就是表示多份数据的集合，但在使用过程中经常会转换为指向数组第 0 个元素的指针，所以上面使用了“认为”一词，表示数组名和数组首地址并不总是等价。初学者可以暂时忽略这个细节，把数组名当做指向第 0 个元素的指针使用即可，我们将在VIP教程《数组和指针绝不等价，数组是另外一种类型》和《数组在什么时候会转换为指针》中再深入讨论这一细节。

下面的例子演示了如何以指针的方式遍历数组元素：

```
01.  #include <stdio.h>
02.
03.  int main(){
04.      int arr[] = { 99, 15, 100, 888, 252 };
05.      int len = sizeof(arr) / sizeof(int);  //求数组长度
06.      int i;
07.      for(i=0; i<len; i++){
08.          printf("%d  ", *(arr+i) );  //*(arr+i)等价于arr[i]
09.      }
10.      printf("\n");
11.      return 0;
12.  }
```

运行结果：
99 15 100 888 252

第 4 行代码用来求数组的长度，sizeof(arr) 会获得整个数组所占用的字节数，sizeof(int) 会获得一个数组元素所占用的字节数，它们相除的结果就是数组包含的元素个数，也即数组长度。

第 8 行代码中我们使用了 `*(arr+i)` 这个表达式，arr 是数组名，指向数组的第 0 个元素，表示数组首地址， arr+i 指向数组的第 i 个元素，*(arr+i) 表示取第 i 个元素的数据，它等价于 arr[i]。

> arr 是 `int*` 类型的指针，每次加 1 时它自身的值会增加 sizeof(int)，加 i 时自身的值会增加 sizeof(int) * i，这在《指针变量的运算》中已经进行了详细讲解。

我们也可以定义一个指向数组的指针，例如：

```
01.  int arr[] = { 99, 15, 100, 888, 252 };
02.  int *p = arr;
```

arr 本身就是一个指针，可以直接赋值给指针变量 p。arr 是数组第 0 个元素的地址，所以 `int *p = arr;` 也可以写作 `int *p = &arr[0];`。也就是说，arr、p、&arr[0] 这三种写法都是等价的，它们都指向数组第 0 个元素，或者说指向数组的开头。

> 再强调一遍，“arr 本身就是一个指针”这种表述并不准确，严格来说应该是“arr 被转换成了一个指针”。这里请大家先忽略这个细节，我们将在VIP教程《数组和指针绝不等价，数组是另外一种类型》和《数组在什么时候会转换为指针》中深入讨论。

如果一个指针指向了数组，我们就称它为数组指针（Array Pointer）。

数组指针指向的是数组中的一个具体元素，而不是整个数组，所以数组指针的类型和数组元素的类型有关，上面的例子中，p 指向的数组元素是 int 类型，所以 p 的类型必须也是 `int *`。

反过来想，p 并不知道它指向的是一个数组，p 只知道它指向的是一个整数，究竟如何使用 p 取决于程序员的编码。

更改上面的代码，使用数组指针来遍历数组元素：

```
#include <stdio.h>

int main(){
    int arr[] = { 99, 15, 100, 888, 252 };
    int i, *p = arr, len = sizeof(arr) / sizeof(int);

    for(i=0; i<len; i++){
        printf("%d  ", *(p+i) );
    }
    printf("\n");
    return 0;
}
```

数组在内存中只是数组元素的简单排列，没有开始和结束标志，在求数组的长度时不能使用 `sizeof(p) / sizeof(int)`，因为 p 只是一个指向 int 类型的指针，编译器并不知道它指向的到底是一个整数还是一系列整数（数组），所以 sizeof(p) 求得的是 p 这个指针变量本身所占用的字节数，而不是整个数组占用的字节数。

也就是说，根据数组指针不能逆推出整个数组元素的个数，以及数组从哪里开始、到哪里结束等信息。不像字符串，数组本身也没有特定的结束标志，如果不知道数组的长度，那么就无法遍历整个数组。

上节我们讲到，对指针变量进行加法和减法运算时，是根据数据类型的长度来计算的。如果一个指针变量 p 指向了数组的开头，那么 p+i 就指向数组的第 i 个元素；如果 p 指向了数组的第 n 个元素，那么 p+i 就是指向第 n+i 个元素；而不管 p 指向了数组的第几个元素，p+1 总是指向下一个元素，p-1 也总是指向上一个元素。

更改上面的代码，让 p 指向数组中的第二个元素：

```
#include <stdio.h>

int main(){
    int arr[] = { 99, 15, 100, 888, 252 };
    int *p = &arr[2];  //也可以写作 int *p = arr + 2;

    printf("%d, %d, %d, %d, %d\n", *(p-2), *(p-1), *p, *(p+1), *(p+2) );
    return 0;
}
```

运行结果：
99, 15, 100, 888, 252

引入数组指针后，我们就有两种方案来访问数组元素了，一种是使用下标，另外一种是使用指针。

1) 使用下标

也就是采用 arr[i] 的形式访问数组元素。如果 p 是指向数组 arr 的指针，那么也可以使用 p[i] 来访问数组元素，它等价于 arr[i]。

2) 使用指针

也就是使用 *(p+i) 的形式访问数组元素。另外数组名本身也是指针，也可以使用 *(arr+i) 来访问数组元素，它等价于 *(p+i)。

不管是数组名还是数组指针，都可以使用上面的两种方式来访问数组元素。不同的是，数组名是常量，它的值不能改变，而数组指针是变量（除非特别指明它是常量），它的值可以任意改变。也就是说，数组名只能指向数组的开头，而数组指针可以先指向数组开头，再指向其他元素。

更改上面的代码，借助自增运算符来遍历数组元素：

```
01.  #include <stdio.h>
02.
03.  int main(){
04.      int arr[] = { 99, 15, 100, 888, 252 };
05.      int i, *p = arr, len = sizeof(arr) / sizeof(int);
06.
07.      for(i=0; i<len; i++){
08.          printf("%d  ", *p++ );
09.      }
10.      printf("\n");
11.      return 0;
12.  }
```

运行结果：
99 15 100 888 252

第 8 行代码中，*p++ 应该理解为 *(p++)，每次循环都会改变 p 的值（p++ 使得 p 自身的值增加），以使 p 指向下一个数组元素。该语句不能写为 *arr++，因为 arr 是常量，而 arr++ 会改变它的值，这显然是错误的。

关于数组指针的谜题

假设 p 是指向数组 arr 中第 n 个元素的指针，那么 *p++、*++p、(*p)++ 分别是什么意思呢？

*p++ 等价于 *(p++)，表示先取得第 n 个元素的值，再将 p 指向下一个元素，上面已经进行了详细讲解。

*++p 等价于 *(++p)，会先进行 ++p 运算，使得 p 的值增加，指向下一个元素，整体上相当于 *(p+1)，所以会获得第 n+1 个数组元素的值。

(*p)++ 就非常简单了，会先取得第 n 个元素的值，再对该元素的值加 1。假设 p 指向第 0 个元素，并且第 0 个元素的值为 99，执行完该语句后，第 0 个元素的值就会变为 100。

<上一节 下一节> 关注我们： 微信公众号 新浪微博 QQ交流群：588321099

关于C语言中文网 | 关于站长 | 如何才能完成一部教程 | 联系我们 | 网站地图 | 手机版网站