

一、列表二、数组--列表的存贮1、数组的存取2、字符串中的方括号和变量替换3、列表范围4、数组的输出5、列表/数组的长度6、子数组7、有关数组的库函数

## 一、列表

列表是包含在括号里的一序列的值，可以为任何数值，也可为空，如：(1, 5.3, "hello", 2)，空列表：()。

注：只含有一个数值的列表(如：(43.2))与该数值本身(即：43.2)是不同的，但它们可以互相转化或赋值。

列表例：

```
(17, $var, "a string")
(17, 26 << 2)
(17, $var1 + $var2)
($value, "The answer is $value")
```

## 二、数组--列表的存贮

列表存贮于数组变量中，与简单变量不同，数组变量以字符"@"打头，如：

```
@array = (1, 2, 3);
```

注：

(1)数组变量创建时初始值为空列表：()。

(2)因为PERL用@和\$来区分数组变量和简单变量，所以同一个名字可以同时用于数组变量和简单变量，如：

```
$var = 1;
@var = (11, 27.1, "a string");
```

但这样很容易混淆，故不推荐。

## 1、数组的存取

.对数组中的值通过下标存取，第一个元素下标为0。试图访问不存在的数组元素，则结果为NULL，但如果给超出数组大小的元素赋值，则数组自动增长，原来没有的元素值为NULL。如：

```
@array = (1, 2, 3, 4);
$scalar = $array[0];
$array[3] = 5; # now @array is (1,2,3,5)
$scalar = $array[4]; # now $scalar = null;
$array[6] = 17; # now @array is (1,2,3,5,""," ",17)
```

.数组间拷贝

```
@result = @original;
```

.用数组给列表赋值

```
@list1 = (2, 3, 4);
@list2 = (1, @list1, 5); # @list2 = (1, 2, 3, 4, 5)
```

.数组对简单变量的赋值

```
(1) @array = (5, 7, 11);
($var1, $var2) = @array; # $var1 = 5, $var2 = 7, 11被忽略
(2) @array = (5, 7);
($var1, $var2, $var3) = @array; # $var1 = 5, $var2 = 7, $var3 = "" (null)
```

.从标准输入(STDIN)给变量赋值

```
$var = <STDIN>;
```

@array = <STDIN>; # ^D为结束输入的符号

## 2、字符串中的方括号和变量替换

"\$var[0]" 为数组@var的第一个元素。

"\$var\[0]" 将字符\[转义，等价于"\$var". "[0]"，\$var被变量替换，[0]保持不变。

"\${var}[0]" 亦等价于"\$var" . "[0]"。

"\$\{var}"则取消了大括号的变量替换功能，包含文字：\${var}。

## 3、列表范围：

(1..10) = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

(2, 5..7, 11) = (2, 5, 6, 7, 11)

(3..3) = (3)

.用于实数

(2.1..5.3) = (2.1, 3.1, 4.1, 5.1)

(4.5..1.6) = ()

.用于字符串

("aaa".. "aad") = ("aaa", "aab", "aac", "aad")

@day\_of\_month = ("01".. "31")

.可包含变量或表达式

(\$var1..\$var2+5)

.小技巧：

```
$fred = "Fred";
```

```
print ("Hello, " . $fred . "!\n") x 2;
```

其结果为：

```
Hello, Fred!
```

```
Hello, Fred!
```

## 4、数组的输出：

(1) @array = (1, 2, 3);

```
print (@array, "\n");
```

结果为：

```
123
```

(2) @array = (1, 2, 3);

```
print ("@array\n");
```

结果为：

```
1 2 3
```

## 5、列表/数组的长度

当数组变量出现在预期简单变量出现的地方，则PERL解释器取其长度。

```
@array = (1, 2, 3);
```

```
$scalar = @array; # $scalar = 3,即@array的长度
```

```
($scalar) = @array; # $scalar = 1,即@array第一个元素的值
```

注：以数组的长度为循环次数可如下编程：

```
$count = 1;
```

```
while ($count <= @array) {
```

```
print ("element $count: $array[$count-1]\n");
```

```
$count++;
```

```
}
```

## 6、子数组

```
@array = (1, 2, 3, 4, 5);
```

```
@subarray = @array[0,1]; # @subarray = (1, 2)
```

```
@subarray2 = @array[1..3]; # @subarray2 = (2,3,4)
```

```
@array[0,1] = ("string", 46); # @array = ("string",46,3,4,5) now
```

```
@array[0..3] = (11, 22, 33, 44); # @array = (11,22,33,44,5) now
```

```
@array[1,2,3] = @array[3,2,4]; # @array = (11,44,33,5,5) now
```

```
@array[0..2] = @array[3,4]; # @array = (5,5,"",5,5) now
```

可以用子数组形式来交换元素：

```
@array[1,2] = @array[2,1];
```

## 7、有关数组的库函数

### (1)sort--按字符顺序排序

```
@array = ("this", "is", "a","test");  
@array2 = sort(@array); # @array2 = ("a","is", "test", "this")  
@array = (70, 100, 8);  
@array = sort(@array); # @array = (100, 70, 8) now
```

### ( 2)reverse--反转数组

```
@array2 = reverse(@array);  
@array2 = reverse sort (@array);
```

### (3)chop--数组去尾

chop的意义是去掉STDIN（键盘）输入字符串时最后一个字符--换行符。而如果它作用到数组上，则将数组中每一个元素都做如此处理。

```
@list = ("rabbit", "12345","quartz");  
chop (@list); # @list = ("rabbi", "1234","quart") now
```

### ( 4)join/split--连接/拆分

join的第一个参数是连接所用的中间字符，其余则为待连接的字符数组。

```
$string = join(" ", "this", "is","a", "string"); # 结果为"this is a string"  
@list = ("words","and");  
$string = join("::", @list, "colons"); #结果为"words::and::colons"  
@array = split(/::/, $string); # @array = ("words","and", "colons") now
```

[上一章](#) [下一章](#) [目录](#)