Perl 时间日期 Perl 引用

Perl 子程序(函数)

Perl 子程序也就是用户定义的函数。

Perl 子程序即执行一个特殊任务的一段分离的代码，它可以使减少重复代码且使程序易读。

Perl 子程序可以出现在程序的任何地方，语法格式如下：

```
sub subroutine{
   statements;
}
```

调用子程序语法格式：

```
subroutine( 参数列表 );
```

在 Perl 5.0 以下版本调用子程序方法如下：

```
&subroutine( 参数列表 );
```

在新版本上，虽然也支持该调用方法，但不推荐使用。

接下来我们来看一个简单是实例：

实例

```
#!/usr/bin/perl
# 函数定义
sub Hello{
print "Hello, World!\n";
}
# 函数调用
Hello();
```

执行以上程序，输出结果为：

```
Hello, World!
```

向子程序传递参数

Perl 子程序可以和其他编程一样接受多个参数，子程序参数使用特殊数组 @_ 标明。

因此子程序第一个参数为 $_[0], 第二个参数为 $_[1], 以此类推。

不论参数是标量型还是数组型的，用户把参数传给子程序时，perl默认按引用的方式调用它们。

实例

```
#!/usr/bin/perl
# 定义求平均值函数
sub Average{
# 获取所有传入的参数
$n = scalar(@_);
$sum = 0;
foreach $item (@_){
$sum += $item;
}
$average = $sum / $n;
print '传入的参数为 : ',"@_\n"; # 打印整个数组
print "第一个参数值为 : $_[0]\n"; # 打印第一个参数
print "传入参数的平均值为 : $average\n"; # 打印平均值
}
# 调用函数
Average(10, 20, 30);
```

执行以上程序，输出结果为：

```
传入的参数为 : 10 20 30
第一个参数值为 : 10
传入参数的平均值为 : 20
```

用户可以通过改变 @_ 数组中的值来改变相应实际参数的值。

向子程序传递列表

由于 @_ 变量是一个数组，所以它可以向子程序中传递列表。

但如果我们需要传入标量和数组参数时，需要把列表放在最后一个参数上，如下所示：

实例

```
#!/usr/bin/perl
# 定义函数
sub PrintList{
my @list = @_;
print "列表为 : @list\n";
}
$a = 10;
@b = (1, 2, 3, 4);
# 列表参数
PrintList($a, @b);
```

以上程序将标量和数组合并了，输出结果为：

```
列表为 : 10 1 2 3 4
```

我们可以向子程序传入多个数组和哈希，但是在传入多个数组和哈希时，会导致丢失独立的标识。所以我们需要使用引用（下一章节会介绍）来传递。

向子程序传递哈希

当向子程序传递哈希表时，它将复制到 @_ 中，哈希表将被展开为键/值组合的列表。

实例

```
#!/usr/bin/perl
# 方法定义
sub PrintHash{
my (%hash) = @_;
foreach my $key ( keys %hash ){
my $value = $hash{$key};
print "$key : $value\n";
}
}
%hash = ('name' => 'runoob', 'age' => 3);
# 传递哈希
PrintHash(%hash);
```

以上程序执行输出结果为：

```
age : 3
name : runoob
```

子程序返回值

子程序可以向其他编程语言一样使用 return 语句来返回函数值。

如果没有使用 return 语句，则子程序的最后一行语句将作为返回值。

实例

```
#!/usr/bin/perl
# 方法定义
sub add_a_b{
# 不使用 return
$_[0]+$_[1];
# 使用 return
# return $_[0]+$_[1];
}
print add_a_b(1, 2)
```

以上程序执行输出结果为：

```
3
```

子程序中我们可以返回标量，数组和哈希，但是在返回多个数组和哈希时，会导致丢失独立的标识。所以我们需要使用引用（下一章节会介绍）来返回多个数组和函数。

子程序的私有变量

默认情况下，Perl 中所有的变量都是全局变量，这就是说变量在程序的任何地方都可以调用。

如果我们需要设置私有变量，可以使用 **my** 操作符来设置。

my 操作符用于创建词法作用域变量，通过 **my** 创建的变量，存活于声明开始的地方，直到闭合作用域的结尾。

闭合作用域指的可以是一对花括号中的区域，可以是一个文件，也可以是一个 if, while, for, foreach, eval字符串。

以下实例演示了如何声明一个或多个私有变量：

```
sub somefunc {
   my $variable; # $variable 在方法 somefunc() 外不可见
   my ($another, @an_array, %a_hash); #  同时声明多个变量
}
```

实例

```
#!/usr/bin/perl
# 全局变量
$string = "Hello, World!";
# 函数定义
sub PrintHello{
# PrintHello 函数的私有变量
my $string;
$string = "Hello, Runoob!";
print "函数内字符串: $string\n";
}
# 调用函数
PrintHello();
print "函数外字符串: $string\n";
```

以上程序执行输出结果为：

```
函数内字符串: Hello, Runoob!
函数外字符串: Hello, World!
```

变量的临时赋值

我们可以使用 local 为全局变量提供临时的值，在退出作用域后将原来的值还回去。

local 定义的变量不存在于主程序中，但存在于该子程序和该子程序调用的子程序中。定义时可以给其赋值，如：

实例

```
#!/usr/bin/perl
# 全局变量
$string = "Hello, World!";
sub PrintRunoob{
# PrintHello 函数私有变量
local $string;
$string = "Hello, Runoob!";
# 子程序调用的子程序
PrintMe();
```

```
print "PrintRunoob 函数内字符串值: $string\n";
}
sub PrintMe{
print "PrintMe 函数内字符串值: $string\n";
}
sub PrintHello{
print "PrintHello 函数内字符串值: $string\n";
}
# 函数调用
PrintRunoob();
PrintHello();
print "函数外部字符串值: $string\n";
```

以上程序执行输出结果为：

```
PrintMe 函数内字符串值: Hello, Runoob!
PrintRunoob 函数内字符串值: Hello, Runoob!
PrintHello 函数内字符串值: Hello, World!
函数外部字符串值: Hello, World!
```

静态变量

state操作符功能类似于C里面的static修饰符，state关键字将局部变量变得持久。

state也是词法变量，所以只在定义该变量的词法作用域中有效，举个例子：

实例

```
#!/usr/bin/perl
use feature 'state';
sub PrintCount{
state $count = 0; # 初始化变量
print "counter 值为: $count\n";
$count++;
}
for (1..5){
PrintCount();
}
```

以上程序执行输出结果为：

```
counter 值为: 0
counter 值为: 1
counter 值为: 2
counter 值为: 3
counter 值为: 4
```

注1：state仅能创建闭合作用域为子程序内部的变量。

注2：state是从Perl 5.9.4开始引入的，所以使用前必须加上 use。

注3：state可以声明标量、数组、哈希。但在声明数组和哈希时，不能对其初始化（至少Perl 5.14不支持）。

子程序调用上下文

子程序调用过程中，会根据上下文来返回不同类型的值，比如以下 localtime() 子程序，在标量上下文返回字符串，在列表上下文返回列表:

实例

```
#!/usr/bin/perl
# 标量上下文
my $datestring = localtime( time );
print $datestring;
print "\n";
# 列表上下文
($sec,$min,$hour,$mday,$mon, $year,$wday,$yday,$isdst) = localtime(time);
printf("%d-%d-%d %d:%d:%d",$year+1990,$mon+1,$mday,$hour,$min,$sec);
print "\n";
```

以上程序执行输出结果为：

```
Sun Jun 12 15:58:09 2016
2106-6-12 15:58:9
```

← Perl 时间日期 Perl 引用 →

心绪留香 2个月前 (01-04)