# برنامه نویسی به زبان

# پرل

## مرجع کامل



تالیف
مهندس هادی کیامرثی

**تمام مثال های موجود در این کتاب با کامپیوتر تست شده اند تا از هر گونه خطا مبرا باشند با این حال ممکن است باز هم خطاهایی در آن وجود داشته باشد از کلیه خوانندگان این کتاب ، اساتید و دانشجویان محترم خواهشمندم برای مطلع کردن مولف از این خطا ها لطفا با ایمیل آدرس زیر تماس بگیرند**

**hadikiamarsi@gmail.com**

# فصل دهم

# در این فصل مطالب زیر را خواهید آموخت

تعریف و اجرای یک تابع

ارسال آرگومان به تابع

ارسال لیست به تابع

ارسال هش به تابع

برگرداندن مقدار از یک تابع

متغیرهای خصوصی در تابع

تعریف متغیرهای خصوصی و عمومی هم نام

جلوگیری از نابودی متغیر خصوصی

دریافت مقدار تابع های چند مقداری

# توابع در پرل

در زبان برنامه نویسی پرل یک زیر روال یا تابع شامل یک سری دستورات زبان برنامه نویسی هستند که برای انجام وظیفه خاصی در یک گروه بندی مجزا اجرا می گردند .شما برای خوانایی بیشتر برنامه و نظم بیشتر در کدتویسی می توانید برنامه را به چند وظیفه تقسیم نمایید و برای هر وظیفه یک تابع بنویسید .

## تعریف و اجرای یک تابع

نحوه تعریف یک تابع در زبان برنامه نویسی پرل در زیر نشان داده شده است

```
sub subroutine_name {
   body of the subroutine
}
```

نحوه صدا زدن ( اجرای ) یک تابع در زیر نشان داده شده است

```
subroutine_name( list of arguments );
```

نگارش های مختلف زبان برنامه نویسی پرل تفاوت های اندکی با هم دارند روش زیر در نگارش 5.0 و ما قبل از آن روش صدا زدن ( اجرای ) یک تابع می باشد .

```
&subroutine_name( list of arguments );
```

در زبان برنامه نویسی پرل مهم نیست که یک تابع در کجا تعریف می شود برای آشنایی بیشتر با تابع ها به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Function definition
sub Hello {
   print "Hello, World!\n";
}

# Function call
Hello();
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Hello, World!
```

# ارسال آرگومان به تابع

شما می توانید هر چندتا آرگومان که نیاز داشته باشید به تابع ارسال نمایید و درون تابع از آن ها استفاده نمایید . آرگومان ها درون تابع در یک آرایه به نام _@ ذخیره می شوند و برای دسترسی به عناصر آن از _$ استفاده می شود . به عنوان مثال تابع شما دو عدد آرگومان داشته باشد برای دسترسی به اولین آرگومان از [0]_$ استفاده می نماییم و برای دسترسی به دومین عنصر از [1]_$ استفاده می نماییم

برای روشن شدن بیشتر این درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Function definition
sub Average {
   # get total number of arguments passed.
   $n = scalar(@_);
   $sum = 0;

   foreach $item (@_) {
      $sum += $item;
   }
   $average = $sum / $n;

   print "Average for the given numbers : $average\n";
}

# Function call
Average(10, 20, 30);
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Average for the given numbers : 20
```

# ارسال لیست به تابع

براحتی می توانید یک آرایه را به عنوان آگومان به  یک تابع ارسال کرد در نتیجه یک لیست را بوسیله یک آرایه می توان به یک تابع ارسال کرد .

برای روشن شدن بیشتر این درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Function definition
sub PrintList {
   my @list = @_;
   print "Given list is @list\n";
}
$a = 10;
@b = (1, 2, 3, 4);

# Function call with list parameter
PrintList($a, @b);
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Given list is 10 1 2 3 4
```

## ارسال هش به تابع

هش ها را هم می توانید به راحتی به عنوان آرگومان به توابع ارسال کنید. برای روشن شدن بیشتر این درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Function definition
sub PrintHash {
   my (%hash) = @_;

   foreach my $key ( keys %hash ) {
      my $value = $hash{$key};
      print "$key : $value\n";
   }
}
%hash = ('name' => 'Tom', 'age' => 19);

# Function call with hash parameter
PrintHash(%hash);
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
name : Tom
age : 19
```

# برگرداندن مقدار از یک تابع

برای برگرداندن مقدار در توابع در زبان برنامه نویسی پرل از دستور return استفاده می گردد برای آشنایی بیشتر با این بخش از درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Function definition
sub Average {
   # get total number of arguments passed.
   $n = scalar(@_);
   $sum = 0;

   foreach $item (@_) {
      $sum += $item;
   }
   $average = $sum / $n;

   return $average;
}

# Function call
$num = Average(10, 20, 30);
print "Average for the given numbers : $num\n";
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Average for the given numbers : 20
```

# متغیرهای خصوصی در تابع

متغیر ها بر اساس حوزه دسترسی به دو دسته تقسیم می شود متغیر های خصوصی و متغیرهای عمومی . متغیر های خصوصی فقط در همان تابعی که تعریف می شوند قابل دسترسی هستند و پس از خروج از تابع تمام متغیرهای خصوصی تعریف شده در تابع به همراه مقادیر آن ها ازبین می روند و حافظه اشغالی آن ها آزاد می گردد . برای تعریف متغیرهای خصوصی در زبان برنامه نویسی پرل از کلمه کلیدی **my** استفاده می گردد برای روشن شدن بیشتر این درس به مثال زیر توجه نمایید

```
sub somefunc {
   my $variable; # $variable is invisible outside somefunc()
   my ($another, @an_array, %a_hash); # declaring many variables at once
}
```

برای مشخص شدن تفاوت بین متغیرهای عمومی و خصوصی به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Global variable
$string = "Hello, World!";

# Function definition
sub PrintHello {
   # Private variable for PrintHello function
   my $string;
   $string = "Hello, Perl!";
   print "Inside the function $string\n";
}
# Function call
PrintHello();
print "Outside the function $string\n";
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Inside the function Hello, Perl!
Outside the function Hello, World!
```

# تعریف متغیرهای خصوصی و عمومی هم نام

برای تعریف یک متغیر عمومی و خصوصی هم نام از کلمه کلیدی **local** استفاده می گردد برای مشخص شدن این مفهوم به مثال زیر توجه نمایید

```
#!/usr/bin/perl

# Global variable
$string = "Hello, World!";

sub PrintHello {
   # Private variable for PrintHello function
   local $string;
   $string = "Hello, Perl!";
   PrintMe();
   print "Inside the function PrintHello $string\n";
}
sub PrintMe {
   print "Inside the function PrintMe $string\n";
}

# Function call
PrintHello();
print "Outside the function $string\n";
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Inside the function PrintMe Hello, Perl!
Inside the function PrintHello Hello, Perl!
Outside the function Hello, World!
```

# جلوگیری از نابودی متغیر خصوصی

برای جلوگیری از نابودی متغیر خصوصی در هنگام خروج از تابع از کلمه کلیدی **state** استفاده می گردد . با استفاده از این دستور مقدار متغیر در هر بار خروج از تابع نابود نمی گردد در نتیجه می توانید روی آن عملیات بیشتری انجام دهید اما بیاد داشته باشید این به معنای عمومی شدن متغیر نیست برای روشن شدن بیشتر این درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

use feature 'state';

sub PrintCount {
   state $count = 0; # initial value

   print "Value of counter is $count\n";
   $count++;
}

for (1..5) {
   PrintCount();
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of counter is 0
Value of counter is 1
Value of counter is 2
Value of counter is 3
Value of counter is 4
```

برای نگارش های قبل از پرل 5.10 می توانید به صورت زیر نیز عمل نمایید

```
#!/usr/bin/perl

{
   my $count = 0; # initial value

   sub PrintCount {
      print "Value of counter is $count\n";
      $count++;
   }
}

for (1..5) {
   PrintCount();
}
```

# دریافت مقدار تابع های چند مقداری

بعضی توابع به جای یک مقدار چندین مقدار بر می گردانند . به عنوان مثال تابع localtime() که برای دریافت اطلاعات زمانی بکار می رود . شما می توانید کل مقدارهای دریافتی را به صورت یک رشته دریافت نمایید مانند مثال زیر

```
my $datestring = localtime( time );
```

یا هر یک را به صورت تک تک مانند مثال زیر دریافت نمایید

```
($sec,$min,$hour,$mday,$mon, $year,$wday,$yday,$isdst) = localtime(time);
```