Perl Tutorial

We provide you with a comprehensive **Perl tutorial** with practical examples. If you are new to Perl, this tutorial is an excellent start. If you want to review your Perl knowledge, you will find some new features of the Perl language, which has been released in the recent versions. This Perl tutorial is updated to version 5.24.

Getting Started with Perl

Powerful, stable, portable and mature, Perl is one of the most feature-rich programming languages with over two decades of development. Perl is portable and cross-platform. Currently, Perl can run on over 100 platforms. Perl is good for both mission-critical large-scale projects and rapid prototyping.

If you are new to Perl…

* First, get started quickly with Perl by following the [introduction to Perl programming language](http://www.perltutorial.org/introduction-to-perl/).
* Second, [setup a Perl development environment](http://www.perltutorial.org/setting-up-perl-development-environment/) for practicing Perl.
* Third, [develop the first program in Perl](http://www.perltutorial.org/developing-the-first-perl-program/).

Basic Perl Tutorial

In this part, you are going to get familiar with the Perl language. We will show you basic Perl syntax, variables, operators and three most important data types in Perl including scalars, arrays, and hashes.

* [Perl Syntax](http://www.perltutorial.org/perl-syntax/)
* [Perl Variables](http://www.perltutorial.org/perl-variables/)
* [Perl String](http://www.perltutorial.org/perl-string/)
* [Perl Numbers](http://www.perltutorial.org/perl-numbers/)
* [Perl Operators](http://www.perltutorial.org/perl-operators/)
* [Perl List](http://www.perltutorial.org/perl-list/)
* [Perl Array](http://www.perltutorial.org/perl-array/)
* [Perl Hash](http://www.perltutorial.org/perl-hash/)

Perl Control Flow Statements

In this part, you will learn how to use Perl control flow statements including if/elsif/else, for/foreach, while/do while, until/do until, next,last statements. In addition, you will learn how to use a relatively new statement in Perl called given/when statement, which is similar to the switch case statement in C.

* [Perl if](http://www.perltutorial.org/perl-if/)
* [Perl unless](http://www.perltutorial.org/perl-unless/)
* [Perl given/when](http://www.perltutorial.org/perl-given/)
* [Perl for loop](http://www.perltutorial.org/perl-for-loop/)
* [Perl while](http://www.perltutorial.org/perl-while/)
* [Perl do while](http://www.perltutorial.org/perl-do-while/)
* [Perl until](http://www.perltutorial.org/perl-until/)
* [Perl do until](http://www.perltutorial.org/perl-do-until/)
* [Perl next](http://www.perltutorial.org/perl-next/)
* [Perl last](http://www.perltutorial.org/perl-last/)

Perl Regular Expression

Regular expressions allow you to match text based on patterns. Perl regular expressions are powerful and very simple to use. In this part, you will learn about the basic regular expression matching techniques, character classes, alternation, grouping, substitutions, and translation.

* [Basic Matching Techniques](http://www.perltutorial.org/perl-regular-expression/)
* [Character Classes](http://www.perltutorial.org/perl-regular-expression-character-classes/)
* [Alternation and Grouping](http://www.perltutorial.org/alternation-and-grouping/)
* [Extracting Matches](http://www.perltutorial.org/regular-expression-extracting-matches/)
* [Substitution and Translation](http://www.perltutorial.org/substitution-and-translation/)

Perl Reference

Perl reference is simple but very powerful. In this part, you will learn how to create array references, hash references, anonymous references, and other kinds of references. You will also learn how to use Perl references effectively in your programs.

* [Perl Reference](http://www.perltutorial.org/perl-reference/)
* [Perl reference types and autovification](http://www.perltutorial.org/perl-reference-types-and-autovification/)

Perl Subroutine

A large Perl program is divided into reusable units of code which are known as functions or [subroutines](http://www.perltutorial.org/perl-subroutine/). In this part, we will show you how to define your own subroutines, how to pass parameters to the subroutine by references and values, and how to pass an array reference to subroutines.

* [Perl subroutine](http://www.perltutorial.org/perl-subroutine/)
* [Passing parameters to subroutines by reference and value](http://www.perltutorial.org/passing-parameters-to-subroutine/)
* [Passing array reference to subroutines](http://www.perltutorial.org/passing-array-references-to-subroutines/)

Perl packages and modules

This part helps you understand packages and namespaces. You will learn how to define a new module in Perl.

* [Perl module](http://www.perltutorial.org/perl-module/)

Perl File I/O

In this part, we are going to show you how to deal with File I/O such as opening a file, reading from a file and writing to file using file handles. We’ll show you other file handling techniques using a very useful diamond operator (<>).

* [Opening file](http://www.perltutorial.org/perl-open-file/)
* [Reading from file](http://www.perltutorial.org/perl-read-file/)
* [Writing to file](http://www.perltutorial.org/perl-write-to-file/)
* [File test operators](http://www.perltutorial.org/perl-file-test-operators/)

Perl Databases

In this part, you will learn how to interact with databases using Perl DBI module.

* [Perl DBI](http://www.perltutorial.org/perl-dbi/)

Perl Object Oriented Programming

This part helps you understand about the object and the rules of Perl’s object-oriented system. In addition, you will learn how to define a class and use it in other programs.

* [Perl OOP](http://www.perltutorial.org/perl-oop/)

Advanced Perl tutorial

* [Perl sort](http://www.perltutorial.org/perl-sort/)– shows you various sorting techniques in Perl.

## Installing DBD::mysql Module

Before installing the DBD::mysql module, please ensure that you have MySQL server installed on your system.

First, you need to install MySQL driver to connect to MySQL databases. There are several ways to install a module. We are going to use CPAN shell to install the DBD::mysql module.

First, type the following command:

|  |  |
| --- | --- |
| 1  2  3  4 | C:\>perl -MCPAN -e shell    cpan shell -- CPAN exploration and modules installation (v1.9800)  Enter 'h' for help. |

Second, to download and install  DBD::mysql module, you use the following command:

|  |  |
| --- | --- |
| 1 | cpan> install DBD:mysql |

* #!/usr/bin/perl
* use strict;
* use warnings;
* use DBI;
* my $dbh = DBI->connect("DBI:mysql:classicmodels",'root','');
* die "failed to connect to MySQL database:DBI->errstr()" unless($dbh);
* # prepare SQL statement
* my $sth = $dbh->prepare("SELECT lastname, firstname, extension FROM employees")
* or die "prepare statement failed: $dbh->errstr()";
* $sth->execute() or die "execution failed: $dbh->errstr()";
* my($lname,$fname,$ext);
* # loop through each row of the result set, and print it
* while(($lname,$fname,$ext) = $sth->fetchrow()){
* print("$lname, $fname\t$ext\n");
* }
* $sth->finish();
* $dbh->disconnect();

#!/usr/bin/perl

use strict;

use warnings;

use DBI;

my $dbh = DBI->connect("DBI:mysql:classicmodels",'root','');

die "failed to connect to MySQL database:DBI->errstr()" unless($dbh);

# prepare SQL statement

my $sql = <<EOF;

SELECT lastname, firstname, extension

FROM employees

WHERE lastname = ? OR firstname = ?

EOF

my $sth = $dbh->prepare($sql)

                 or die "prepare statement failed: $dbh->errstr()";

my($lname,$fname,$ext);

my($name, $answer);

print("\nPlease enter the employee firstname or lastname:");

while(<STDIN>){

   $name = $\_;

   chomp($name);

   $sth->execute($name,$name) or die "execution failed: $dbh->errstr()";

   # loop through each row of the result set, and print it

   while(($lname,$fname,$ext) = $sth->fetchrow()){

      print("$lname, $fname\t$ext\n");

   }

   print("\nDo you want to continue? (Y/N)");

   $answer = <STDIN>;

   chomp($answer);

   last if $answer eq 'N';

   print("\nPlease enter the employee firstname or lastname:");

}

$sth->finish();

$dbh->disconnect();

Sorting

## Introduction to the Perl sort function

Perl sort() function sorts a [list](http://www.perltutorial.org/perl-list/)and returns a sorted list. The original list remains intact. The sort()function has three forms:

|  |  |
| --- | --- |
| 1  2  3 | sort list;  sort block list;  sort subroutine\_name list |

#!/usr/bin/perl

use warnings;

use strict;

my @list = sort qw(perl sort function alphabetically);

print "@list\n";

#!/usr/bin/perl

use warnings;

use strict;

my @nums = sort qw/1 11 2 22 10 100/;

print "@nums\n";

#!/usr/bin/perl

use warnings;

use strict;

my @nums = sort { $a <=> $b } qw/1 11 2 22 10 100/;

print "@nums\n";

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #!/usr/bin/perl  use warnings;  use strict;    my @nums = sort compare qw/1 11 2 22 10 100/;  print "@nums\n";    # compare two numbers  sub compare{     if($a < $b){        return -1;     }elsif($a == $b){        return 0;     }else{        return 1;     }  } |

### Perl reverse sorting example

Sometimes, you want a list reversed. You may do it quickly as follows:

|  |  |
| --- | --- |
| 1 | @rlist = reverse sort @list; |

#!/usr/bin/perl

use warnings;

use strict;

my @nums = sort { $b <=> $a } qw(1 11 2 22 10 100);

my @chars = sort { $b cmp $a } qw(perl sort function alphabetically);

print "@nums\n";

print "@chars\n";

#!/usr/bin/perl

use warnings;

use strict;

my @products = (

   {

      name => 'iPhone',

      price => 600,

      discount => 0.5,

   },

   {

      name => 'Nexus',

      price => 299,

      discount => 0.1,

   },

   {

      name => 'Samsung Galaxy',

      price => 600,

      discount => 0.8,

   }

);

@products = sort {

  $a->{price} <=> $b->{price} ||

  $b->{discount} <=> $a->{discount}

} @products;

foreach my $p (@products){

   printf "%-15s %2d USD %2.1f\n" => @{$p}{qw(name price discount)};

}