# PERL INTERVIEW QUESTIONS

 

Dear readers, these **Perl Programming Language Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Perl Programming Language**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer −

What is Perl?

- Perl is a stable, cross platform programming language.
- Though Perl is not officially an acronym but few people used it as **Practical Extraction and Report Language**.
- It is used for mission critical projects in the public and private sectors.
- Perl is an *Open Source* software, licensed under its *Artistic License*, or the *GNU General Public License* *GPL*.
- Perl was created by Larry Wall.
- Perl 1.0 was released to usenet's alt.comp.sources in 1987
- At the time of writing this tutorial, the latest version of perl is 5.16.2
- Perl is listed in the *Oxford English Dictionary*.

What are the features of Perl programming?

- Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.
- Perls database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
- Perl works with HTML, XML, and other mark-up languages.
- Perl supports Unicode.
- Perl is Y2K compliant.
- Perl supports both procedural and object-oriented programming.
- Perl interfaces with external C/C++ libraries through XS or SWIG.
- Perl is extensible. There are over 20,000 third party modules available from the Comprehensive Perl Archive Network (CPAN).
- The Perl interpreter can be embedded into other systems.

What are the benefits of Perl programming in using it in web based applications?

- Perl used to be the most popular web programming language due to its text manipulation capabilities and rapid development cycle.
- Perl is widely known as " the duct-tape of the Internet".
- Perl can handle encrypted Web data, including e-commerce transactions.
- Perl can be embedded into web servers to speed up processing by as much as 2000%.
- Perl's mod_perl allows the Apache web server to embed a Perl interpreter.

- Perl's DBI package makes web-database integration easy.

Is perl a case sensitive language?

Yes! Perl is a case sensitive programming language.

What is a perl identifier?

A Perl identifier is a name used to identify a variable, function, class, module, or other object. A Perl variable name starts with either $, @ or % followed by zero or more letters, underscores, and digits (0 to 9).

What are data types that perl supports?

Perl has three basic data types − scalars, arrays of scalars, and hashes of scalars, also known as associative arrays.

What are scalar data types in perl?

Scalars are simple variables. They are preceded by a dollar sign $. A scalar is either a number, a string, or a reference. A reference is actually an address of a variable, which we will see in the upcoming chapters.

What are Arrays in perl?

Arrays are ordered lists of scalars that you access with a numeric index which starts with 0. They are preceded by an "at" sign @.

What are Hashes in perl?

Hashes are unordered sets of key/value pairs that you access using the keys as subscripts. They are preceded by a percent sign %.

How will you declare a variable in perl?

Perl variables do not have to be explicitly declared to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign = is used to assign values to variables.

What is variable context in perl?

Perl treats same variable differently based on Context, i.e. situation where a variable is being used.

What is scalar context?

Assignment to a scalar variable evaluates the right-hand side in a scalar context.

What is a list context?

Assignment to an array or a hash evaluates the right-hand side in a list context.

What is boolean context?

Boolean context is simply any place where an expression is being evaluated to see whether it's true or false.

What is void context?

This context not only doesn't care what the return value is, it doesn't even want a return value.

What is interpolative context?

This context only happens inside quotes, or things that work like quotes.

What is the difference between single quoted string and double quoted string?

Single quoted string prints the perl variable as a string whereas double quoted string evaluates the variable and used to get the variable's value.

```
#!/usr/bin/perl

$var = "This is string scalar!";
$quote = 'I m inside single quote - $var';
$double = "This is inside double quote - $var";

$escape = "This example of escape -\tHello, World!";

print "var = $var\n";
print "quote = $quote\n";
print "double = $double\n";
print "escape = $escape\n";
```

This will produce the following result –

```
var = This is string scalar!
quote = I m inside single quote - $var
double = This is inside double quote - This is string scalar!
escape = This example of escape - Hello, World!
```

What is V-Strings?

A literal of the form v1.20.300.4000 is parsed as a string composed of characters with the specified ordinals. This form is known as v-strings.

A v-string provides an alternative and more readable way to construct strings, rather than use the somewhat less readable interpolation form "\x{1}\x{14}\x{12c}\x{fa0}".

What is the purpose of _FILE_ literal?

It is used to get the current file name.

What is the purpose of _LINE_ literal?

It is used to get the current line number.

What is the purpose of _PACKAGE_ literal?

It is used to get the current package name.

How will you access an element of a perl array?

To refer to a single element of an array, you will use the dollar sign $ with the variable name followed by the index of the element in square brackets.

Here is a simple example of using the array variables –

```
#!/usr/bin/perl

@ages = (25, 30, 40);
@names = ("John Paul", "Lisa", "Kumar");

print "\$ages[0] = $ages[0]\n";
print "\$ages[1] = $ages[1]\n";
print "\$ages[2] = $ages[2]\n";
print "\$names[0] = $names[0]\n";
print "\$names[1] = $names[1]\n";
print "\$names[2] = $names[2]\n";
```

When exected, this will produce the following result –

```
$ages[0] = 25
$ages[1] = 30
$ages[2] = 40
$names[0] = John Paul
$names[1] = Lisa
$names[2] = Kumar
```

What is range operator?

range operator .. is used to create sequential arrays.

```
#!/usr/bin/perl

@var_10 = (1..10);
@var_20 = (10..20);
@var_abc = (a..z);

print "@var_10\n";   # Prints number from 1 to 10
print "@var_20\n";   # Prints number from 10 to 20
print "@var_abc\n";  # Prints number from a to z
```

Here double dot .. is called range operator. This will produce the following result –

```
1 2 3 4 5 6 7 8 9 10
10 11 12 13 14 15 16 17 18 19 20
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

How will you get size of an array?

The size of an array can be determined using the scalar context on the array - the returned value will be the number of elements in the array –

```
@array = (1,2,3);
print "Size: ",scalar @array,"\n";
```

The value returned will always be the physical size of the array, not the number of valid elements.

How will you add an element to an end of an array?

push @ARRAY, LIST - Pushes the values of the list onto the end of the array.

```
#!/usr/bin/perl

# create a simple array
@coins = ("Quarter","Dime","Nickel");
print "1. \@coins  = @coins\n";

# add one element at the end of the array
push(@coins, "Penny");
print "2. \@coins  = @coins\n";
```

This will produce the following result –

```
1. @coins = Quarter Dime Nickel
2. @coins = Quarter Dime Nickel Penny
```

How will you add an element to the beginning of an array?

unshift @ARRAY, LIST - Prepends list to the front of the array, and returns the number of elements in the new array.

```
#!/usr/bin/perl

# create a simple array
@coins = ("Quarter","Dime","Nickel");
print "1. \@coins  = @coins\n";

# add one element at the beginning of the array
unshift(@coins, "Dollar");
print "2. \@coins  = @coins\n";
```

This will produce the following result –

```
1. @coins = Quarter Dime Nickel
2. @coins = Dollar Quarter Dime Nickel
```

How will you remove an element from end of an array?

pop @ARRAY − Pops off and returns the last value of the array.

```
#!/usr/bin/perl

# create a simple array
@coins = ("Quarter","Dime","Nickel");
print "1. \@coins  = @coins\n";

# remove one element from the last of the array.
pop(@coins);
print "2. \@coins  = @coins\n";
```

This will produce the following result −

```
1. @coins = Quarter Dime Nickel
2. @coins = Quarter Dime
```

How will you remove an element from begining of an array?

shift @ARRAY − Shifts the first value of the array off and returns it, shortening the array by 1 and moving everything down.

```
#!/usr/bin/perl

# create a simple array
@coins = ("Quarter","Dime","Nickel");
print "1. \@coins  = @coins\n";

# remove one element from the beginning of the array.
shift(@coins);
print "2. \@coins  = @coins\n";
```

This will produce the following result −

```
1. @coins = Quarter Dime Nickel
2. @coins = Dime Nickel
```

How will you get slice from an array?

You can also extract a "slice" from an array − that is, you can select more than one item from an array in order to produce another array.

```
#!/usr/bin/perl

@days = qw/Mon Tue Wed Thu Fri Sat Sun/;

@weekdays = @days[3,4,5];

print "@weekdays\n";
```

This will produce the following result -

```
Thu Fri Sat
```

How will you get replace elements of an array?

splice function will remove the elements of @ARRAY designated by OFFSET and LENGTH, and replaces them with LIST, if specified. Finally, it returns the elements removed from the array.

```
splice @ARRAY, OFFSET [ , LENGTH [ , LIST ] ]
```

Following is the example −

```
#!/usr/bin/perl

@nums = (1..20);
print "Before - @nums\n";

splice(@nums, 5, 5, 21..25);
print "After - @nums\n";
```

This will produce the following result −

```
Before - 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
After - 1 2 3 4 5 21 22 23 24 25 11 12 13 14 15 16 17 18 19 20
```

How will you convert a string to an array?

split splits a string into an array of strings, and returns it. If LIMIT is specified, splits into at most that number of fields. If PATTERN is omitted, splits on whitespace.

```
split [ PATTERN [ , EXPR [ , LIMIT ] ] ]
```

Following is the example −

```
#!/usr/bin/perl

# define Strings
$var_string = "Rain-Drops-On-Roses-And-Whiskers-On-Kittens";
$var_names = "Larry,David,Roger,Ken,Michael,Tom";

# transform above strings into arrays.
@string = split('-', $var_string);
@names  = split(',', $var_names);

print "$string[3]\n";  # This will print Roses
print "$names[4]\n";   # This will print Michael
```

This will produce the following result −

```
Roses
Michael
```

How will you convert an array to string?

join function joins the separate strings of LIST into a single string with fields separated by the value of EXPR, and returns the string.

```
join EXPR, LIST
```

Following is the example −

```
#!/usr/bin/perl

# define Strings
$var_string = "Rain-Drops-On-Roses-And-Whiskers-On-Kittens";
$var_names = "Larry,David,Roger,Ken,Michael,Tom";

# transform above strings into arrays.
@string = split('-', $var_string);
@names  = split(',', $var_names);

$string1 = join( '-', @string );
$string2 = join( ',', @names );
```

```
print "$string1\n";
print "$string2\n";
print "$string[3]\n";  # This will print Roses
print "$names[4]\n";   # This will print Michael
```

This will produce the following result –

```
Rain-Drops-On-Roses-And-Whiskers-On-Kittens
Larry,David,Roger,Ken,Michael,Tom
```

How will you sort an array?

The sort function sorts each element of an array according to the ASCII Numeric standards. This function has the following syntax –

```
sort [ SUBROUTINE ] LIST
```

This function sorts the LIST and returns the sorted array value. If SUBROUTINE is specified then specified logic inside the SUBTROUTINE is applied while sorting the elements.

```
#!/usr/bin/perl

# define an array
@foods = qw(pizza steak chicken burgers);
print "Before: @foods\n";

# sort this array
@foods = sort(@foods);
print "After: @foods\n";
```

This will produce the following result –

```
Before: pizza steak chicken burgers
After: burgers chicken pizza steak
```

What is the purpose of $[ variable?

This special variable is a scalar containing the first index of all arrays. Because Perl arrays have zero-based indexing, [ will almost always be 0. But if you set [ to 1 then all your arrays will use on-based indexing. It is recommended not to use any other indexing other than zero. However, let's take one example to show the usage of $[ variable –

```
#!/usr/bin/perl

# define an array
@foods = qw(pizza steak chicken burgers);
print "Foods: @foods\n";

# Let's reset first index of all the arrays.
$[ = 1;

print "Food at \@foods[1]: $foods[1]\n";
print "Food at \@foods[2]: $foods[2]\n";
```

This will produce the following result –

```
Foods: pizza steak chicken burgers
Food at @foods[1]: pizza
Food at @foods[2]: steak
```

How will you merge two arrays?

Because an array is just a comma-separated sequence of values, you can combine them together as shown below.

```
#!/usr/bin/perl

@numbers = (1,3,(4,5,6));

print "numbers = @numbers\n";
```

This will produce the following result −

```
numbers = 1 3 4 5 6
```

How will you create Hashes in perl?

Hashes are created in one of the two following ways. In the first method, you assign a value to a named key on a one-by-one basis −

```
$data{'John Paul'} = 45;
$data{'Lisa'} = 30;
$data{'Kumar'} = 40;
```

In the second case, you use a list, which is converted by taking individual pairs from the list: the first element of the pair is used as the key, and the second, as the value. For example −

```
%data = ('John Paul', 45, 'Lisa', 30, 'Kumar', 40);
```

How will you get element from Hashes in perl?

When accessing individual elements from a hash, you must prefix the variable with a dollar sign $ and then append the element key within curly brackets after the name of the variable. For example −

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

print "$data{'John Paul'}\n";
print "$data{'Lisa'}\n";
print "$data{'Kumar'}\n";
```

This will produce the following result −

```
45
30
40
```

How will you get all keys from Hashes in perl?

You can get a list of all of the keys from a hash by using keys function, which has the following syntax −

```
keys %HASH
```

This function returns an array of all the keys of the named hash. Following is the example −

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

@names = keys %data;

print "$names[0]\n";
print "$names[1]\n";
print "$names[2]\n";
```

This will produce the following result −

```
Lisa
John Paul
Kumar
```

How will you get all values from Hashes in perl?

You can get a list of all of the values from a hash by using values function, which has the following syntax −

```
values %HASH
```

This function returns an array of all the values of the named hash. Following is the example −

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

@ages = values %data;

print "$ages[0]\n";
print "$ages[1]\n";
print "$ages[2]\n";
```

This will produce the following result −

```
30
45
40
```

How will you check if key exists in a hash or not?

Using the exists function, which returns true if the named key exists, irrespective of what its value might be −

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

if( exists($data{'Lisa'} ) ){
   print "Lisa is $data{'Lisa'} years old\n";
}
else{
   print "I don't know age of Lisa\n";
}
```

Here we have introduced the IF...ELSE statement, which we will study in a separate chapter. For now you just assume that if condition part will be executed only when the given condition is true otherwise else part will be executed. So when we execute the above program, it produces the following result because here the given condition exists($data{'Lisa'} returns true −

```
Lisa is 30 years old
```

How will you get the size of hash?

You can get the size - that is, the number of elements from a hash by using the scalar context on either keys or values. Simply saying first you have to get an array of either the keys or values and then you can get the size of array as follows −

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

@keys = keys %data;
```

```
$size = @keys;
print "1 - Hash size:  is $size\n";

@values = values %data;
$size = @values;
print "2 - Hash size:  is $size\n";
```

This will produce the following result –

```
1 - Hash size: is 3
2 - Hash size: is 3
```

How will you add an element to a hash?

Adding a new key/value pair can be done with one line of code using simple assignment operator.

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);
@keys = keys %data;
$size = @keys;
print "1 - Hash size:  is $size\n";

# adding an element to the hash;
$data{'Ali'} = 55;
@keys = keys %data;
$size = @keys;
print "2 - Hash size:  is $size\n";
```

This will produce the following result –

```
1 - Hash size: is 3
2 - Hash size: is 4
```

How will you remove an element from a hash?

To remove an element from the hash you need to use delete function as shown below in the example–

```
#!/usr/bin/perl

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);
@keys = keys %data;
$size = @keys;
print "1 - Hash size:  is $size\n";

# delete the same element from the hash;
delete $data{'John Paul'};
@keys = keys %data;
$size = @keys;
print "2 - Hash size:  is $size\n";
```

This will produce the following result –

```
1 - Hash size: is 3
2 - Hash size: is 2
```

What is the purpose of next statement?

It causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. last statement.

What is the purpose of last statement?

It terminates the loop statement and transfers execution to the statement immediately following the loop. continue statement.

What is the purpose of continue statement?

A continue BLOCK, it is always executed just before the conditional is about to be evaluated again.

What is the purpose of redo statement?

The redo command restarts the loop block without evaluating the conditional again. The continue block, if any, is not executed.

What is the purpose of goto Label statement?

The goto LABEL form jumps to the statement labeled with LABEL and resumes execution from there.

What is the purpose of goto Expr statement?

The goto EXPR form is just a generalization of goto LABEL. It expects the expression to return a label name and then jumps to that labeled statement.

What is the purpose of goto &NAME statement?

It substitutes a call to the named subroutine for the currently running subroutine.

What is the purpose of ** operator?

Exponent − Performs exponential power calculation on operators. Assume variable a holds 10 and variable b holds 20 then a**b will give 10 to the power 20.

What is the purpose of <=> operator?

It checks if the value of two operands are equal or not, and returns -1, 0, or 1 depending on whether the left argument is numerically less than, equal to, or greater than the right argument. Assume variable a holds 10 and variable b holds 20 then $a <=> $b returns -1.

What is the purpose of lt operator?

It returns true if the left argument is stringwise less than the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a lt $b is true.

What is the purpose of gt operator?

It returns true if the left argument is stringwise greater than the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a gt $b is false.

What is the purpose of le operator?

It returns true if the left argument is stringwise less than or equal to the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a le $b is true.

What is the purpose of ge operator?

It returns true if the left argument is stringwise greater than or equal to the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a ge $b is false.

What is the purpose of eq operator?

It returns true if the left argument is stringwise equal to the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a eq $b is false.

What is the purpose of ne operator?

It returns true if the left argument is stringwise not equal to the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a ne $b is true.

What is the purpose of cmp operator?

It returns -1, 0, or 1 depending on whether the left argument is stringwise less than, equal to, or greater than the right argument. Assume variable a holds "abc" and variable b holds "xyz" then $a

cmp $b is -1.

What is the purpose of **= operator?

Exponent AND assignment operator, Performs exponential power calculation on operators and assign value to the left operand. c **= a is equivalent to c = c ** $a

What is the purpose of q{ } operator?

It encloses a string with-in single quotes. q{abcd} gives 'abcd'

What is the purpose of qq{ } operator?

It encloses a string with-in double quotes. qq{abcd} gives "abcd"

What is the purpose of qx{ } operator?

It encloses a string with-in invert quotes. qx{abcd} gives `abcd`

What is the purpose of . operator?

Binary operator dot . concatenates two strings. If a="abc", b="def" then a.b will give "abcdef"

What is the purpose of x operator?

The repetition operator x returns a string consisting of the left operand repeated the number of times specified by the right operand. '-' x 3 will give ---.

What is the purpose of .. operator?

The range operator .. returns a list of values counting up by ones from the left value to the right value. 2..5 will give 2, 3, 4, 5.

What is the purpose of ++ operator?

Auto Increment operator increases integer value by one. $a++ will give 11.

What is the purpose of −− operator?

Auto Decrement operator decreases integer value by one. $a−− will give 9

What is the purpose of −> operator?

The arrow operator is mostly used in dereferencing a method or variable from an object or a class name. obj−>a is an example to access variable a from object obj.

What is the purpose of localtime function?

localtime function, which returns values for the current date and time if given no arguments.

```
#!/usr/local/bin/perl

@months = qw( Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec );
@days = qw(Sun Mon Tue Wed Thu Fri Sat Sun);

($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime();
print "$mday $months[$mon] $days[$wday]\n";
```

When the above code is executed, it produces the following result −

```
16 Feb Sat
```

What is the purpose of gmtime function?

The function gmtime works just like localtime function but the returned values are localized for the standard Greenwich time zone. When called in list context, $isdst, the last value returned by gmtime, is always 0 . There is no Daylight Saving Time in GMT.

What is the difference between localtime and gmtime functions?

localtime will return the current local time on the machine that runs the script and gmtime will return the universal Greenwich Mean Time, or GMT or UTC.

What is the purpose of time function?

You can use the time function to get epoch time, i.e. the numbers of seconds that have elapsed since a given date, in Unix is January 1, 1970.

What is the purpose of strftime function?

You can use the POSIX function strftime to format date and time.

How will you define a subroutine in perl?

The general form of a subroutine definition in Perl programming language is as follows −

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

How will you call a subroutine in perl?

The typical way of calling that Perl subroutine is as follows −

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

How will you access the parameters passed to a perl subroutine?

they can be acessed inside the function using the special array @_. Thus the first argument to the function is in _[0], the second is in _[1], and so on.

How will you get the count of parameters passed to a perl subroutine?

using scalar@_, we can get the total number of arguments passed.

What is the purpose of my operator?

The my operator confines a variable to a particular region of code in which it can be used and accessed. Outside that region, this variable cannot be used or accessed.

What is the default scope of perl variables?

By default, all variables in Perl are global variables, which means they can be accessed from anywhere in the program.

What are lexical variables in perl?

Lexical variables are private variables created using my operator.

What is purpose of local operator in perl?

The local is used when the current value of a variable must be visible to called subroutines.

What is dynamic scoping?

A local just gives temporary values to global meaning package variables. This is known as dynamic scoping.

What is lexical scoping?

Lexical scoping is done with my operator. A lexical scope is usually a block of code with a set of braces around it, such as those defining the body of the subroutine or those marking the code blocks of if, while, for, foreach, and eval statements. The my operator confines a variable to a particular region of code in which it can be used and accessed. Outside that region, this variable cannot be used or accessed.

What are state variables in perl?

There are another type of lexical variables, which are similar to private variables but they maintain their state and they do not get reinitialized upon multiple calls of the subroutines. These variables are defined using the state operator and available starting from Perl 5.9.4.

What is Subroutine Call Context?

The context of a subroutine or statement is defined as the type of return value that is expected. This allows you to use a single function that returns different values based on what the user is expecting to receive. For example, the following localtime returns a string when it is called in scalar context, but it returns a list when it is called in list context.

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

In this example, the value of $timestr is now a string made up of the current date and time, for example, Thu Nov 30 15:21:33 2000. Conversely −

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

Now the individual variables contain the corresponding values returned by localtime subroutine.

What is a Perl references?

A Perl reference is a scalar data type that holds the location of another value which could be scalar, arrays, or hashes. Because of its scalar nature, a reference can be used anywhere, a scalar can be used.

How will you create a reference for a variable?

You can create a reference for any variable by prefixing it with a backslash as follows −

```
$scalarref = \$foo;
```

How will you create a reference for a array?

You can create a reference for any array by prefixing it with a backslash as follows −

```
$arrayref  = \@ARGV;
```

How will you create a reference for a hash?

You can create a reference for any hash by prefixing it with a backslash as follows −

```
$hashref   = \%ENV;
```

How will you create a reference for a subrouting?

You can create a reference for any subrouting by prefixing it with a backslash as follows −

```
$cref = \&PrintHash;
```

What is dereferencing?

Dereferencing returns the value from a reference point to the location.

How will you dereference a reference?

To dereference a reference simply use $, @ or % as prefix of the reference variable depending on whether the reference is pointing to a scalar, array, or hash.

What is circular reference?

A circular reference occurs when two references contain a reference to each other. You have to be careful while creating references otherwise a circular reference can lead to memory leaks.

Following is an example −

```
#!/usr/bin/perl
my $foo = 100;
$foo = \$foo;

print "Value of foo is : ", $$foo, "\n";
```

When above program is executed, it produces the following result −

```
Value of foo is : REF(0x9aae38)
```

How will you open a file in read-only mode?

Following is the syntax to open file.txt in read-only mode. Here less than < sign indicates that file has to be opend in read-only mode.

```
open(DATA, "<file.txt");
```

Here DATA is the file handle which will be used to read the file.

How will you open a file in writing mode?

Following is the syntax to open file.txt in writing mode. Here less than > sign indicates that file has to be opend in the writing mode.

```
open(DATA, ">file.txt") or die "Couldn't open file file.txt, $!";
```

How will you open a file in writing mode without truncating it?

Following is the syntax to open file.txt in writing mode without truncating it. Here less than +< sign indicates that file has to be opend in the writing mode without truncating it.

```
open(DATA, "+<file.txt") or die "Couldn't open file file.txt, $!";
```

What is the purpose of close function?

To close a filehandle, and therefore disassociate the filehandle from the corresponding file, you use the close function. This flushes the filehandle's buffers and closes the system's file descriptor.

What is the purpose of getc function?

The getc function returns a single character from the specified FILEHANDLE, or STDIN if none is specified.

What is the purpose of read function?

The read function reads a block of information from the buffered filehandle: This function is used to read binary data from the file.

## What is Next ?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)