PERL SWITCH STATEMENT

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each **switch case**.

A switch case implementation is dependent on **Switch** module and **Switch** module has been implemented using *Filter::Util::Call* and *Text::Balanced* and requires both these modules to be installed.

Syntax

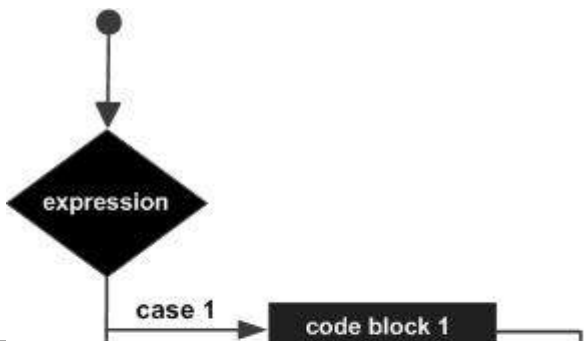
The synopsis for a **switch** statement in Perl programming language is as follows −

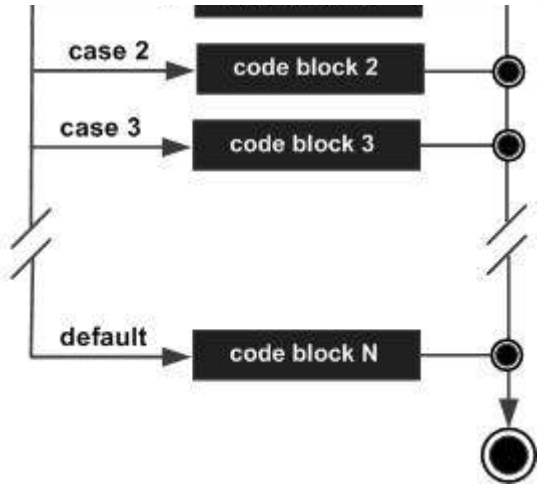
```
use Switch;

switch(argument){
   case 1             { print "number 1" }
   case "a"           { print "string a" }
   case [1..10,42]    { print "number in list" }
   case (\@array)     { print "number in list" }
   case /\w+/         { print "pattern" }
   case qr/\w+/       { print "pattern" }
   case (\%hash)      { print "entry in hash" }
   case (\&)       { print "arg to subroutine" }
   else               { print "previous case not true" }
}
```

The following rules apply to a **switch** statement −

- The **switch** statement takes a single scalar argument of any type, specified in parentheses.
- The value is followed by a block, which may contain one or more case statement followed by a block of Perl statements.
- A case statement takes a single scalar argument and selects the appropriate type of matching between the case argument and the current switch value.
- If the match is successful, the mandatory block associated with the case statement is executed.
- A **switch** statement can have an optional **else** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is matched.
- If a case block executes an untargeted **next**, control is immediately transferred to the statement after the case statement *i. e.* , *usuallyanothercase*, rather than out of the surrounding switch block.
- Not every case needs to contain a **next**. If no **next** appears, the flow of control will *not fall through* subsequent cases.

Flow Diagram





Example

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

use Switch;

$var = 10;
@array = (10, 20, 30);
%hash = ('key1' => 10, 'key2' => 20);

switch($var){
   case 10            { print "number 100\n" }
   case "a"           { print "string a" }
   case [1..10,42]    { print "number in list" }
   case (\@array)     { print "number in list" }
   case (\%hash)      { print "entry in hash" }
   else               { print "previous case not true" }
}
```

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

```
number 100
```

Fall-though is usually a bad idea in a switch statement. However, now consider a fall-through case, we will use the **next** to transfer the control to the next matching case, which is a list in this case –

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

use Switch;

$var = 10;
@array = (10, 20, 30);
%hash = ('key1' => 10, 'key2' => 20);

switch($var){
   case 10            { print "number 100\n"; next; }
   case "a"           { print "string a" }
   case [1..10,42]    { print "number in list" }
   case (\@array)     { print "number in list" }
   case (\%hash)      { print "entry in hash" }
   else               { print "previous case not true" }
}
```

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

```
number 100
number in list
```