Perl Programming Language - Detailed Notes

Introduction

Perl (Practical Extraction and Report Language) is a high-level, interpreted, and dynamic programming language.

It is especially known for its strengths in text processing, system administration, web development, and more.

Perl was created by Larry Wall and combines features from other programming languages including C, shell script, AWK, and sed.

Basic Syntax

```
#!/usr/bin/perl
use strict;
use warnings;
print "Hello, World!\n";
```

The first line is the shebang that tells the OS to run the script using Perl. The 'strict' and 'warnings' pragmas enforce good coding practices.

Variables

Perl has three main types of variables:

1. Scalar: $scalar = 10; # A single value (number or string)
2. Array: @array = (1, 2, 3); # Ordered list of scalars
3. Hash: %hash = ('a' => 1, 'b' => 2); # Key-value pairs

Variable names are case-sensitive.

Operators

Arithmetic: +, -, *, /, %

String: . (concatenation), x (repetition)

Comparison:

  Numeric: ==, !=, <, >, <=, >=

  String: eq, ne, lt, gt, le, ge

Control Structures

```
if ($x > 10) {
  print "Greater\n";
} elsif ($x == 10) {
  print "Equal\n";
} else {
  print "Smaller\n";
}
```

Loops:

- for (my $i = 0; $i < 10; $i++)
- foreach my $item (@array)
- while (condition)
- until (condition)

Control keywords: last, next, redo

Subroutines

Subroutines are defined using 'sub':

```
sub greet {
  my ($name) = @_;
  print "Hello, $name\n";
}
greet("Alice");
```

Arguments are passed via @_ array. Use 'my' to declare local variables.

File Handling

```
open(my $fh, '<', 'file.txt') or die "Can't open file: $!";
```

Perl Programming Language - Detailed Notes

```
while (my $line = <$fh>) {
  print $line;
}
close($fh);
```

Modes: '<' read, '>' write, '>>' append.

Regular Expressions

```
$text =~ /pattern/;  # Match
$text =~ s/old/new/;  # Substitute
$text =~ tr/a-z/A-Z/;  # Transliterate
```

Pattern modifiers: i (case-insensitive), g (global), m (multi-line)

Useful Functions

- chomp($str): Removes newline
- length($str): Returns length of string
- split /sep/, $str: Splits string into array
- join("sep", @array): Joins array into string
- push, pop, shift, unshift: Array operations

Modules and Packages

Modules are reusable packages:

```
use Math::Trig;
use POSIX;
```

CPAN is the Comprehensive Perl Archive Network for modules.

Special Variables

- $_: Default variable for many operations

Perl Programming Language - Detailed Notes

- @_: Arguments to subroutine
- $.: Current line number of input file
- @ARGV: Command line arguments
- $$: Process ID
- $!: Last error message