Introduction to Perl language

Unit 4

Why use Perl?

- Easy text processing, similar to C
- Cross platform
  - Platform independent
  - Runs on over most platforms
  - Windows, Unix, Linux, Mac
- Interpreted language i.e. there is no explicitly separate compilation step
- The processor reads the whole file, converts it to an internal form and executes it immediately

Running Perl

#!/usr/local/bin/perl (tells the file to run through perl)

Use **.pl** extension

perl *programName (to run the program)*
perl test.pl

perl -d *programName (to run using debugger)*

perl – w *programName (to run with warnings)*

Literals

- Data that do not change with time.
- Also known as **invariants** or **constants**.
- The text “Hello, World!\n” is also a literal as data cannot be changed during the time we are running the program

Numbers – 1/4

- Several classes of numbers:
  - **integers**
  - decimals (known as **floating-point numbers)**

- Integers can be expressed in decimal (base 10), hexadecimal (base 16) or octal (base 8) notation

Numbers – 2/4

- Octal numbers are preceded by a 0 (zero),
- Hexadecimal numbers are preceded by 0x (zero x)
- A – F can be lowercase or uppercase

Numbers – 3/4

- **Integers cannot be delimited by commas or spaces**
- Example is 4_976_297_305. → can read easily by programmers
- **Decimals are those carrying decimal points.**
- If the integral portion is 0, the integral portion is optional, i.e. -0.6 or -.6

Numbers – 4/4

- Exponents (base 10) can also be specified by appending the letter "**e**" and the exponent to the real number portion.

2e3 is equivalent to $2 \times 10^3 = 2000$.

Strings – 1/6

- A **string** is a sequence of characters enclosed (delimited) by either double quotes (“) or single quotes (‘).
- They **differ in variable substitution** and in the way **escape characters** are handled.
- The text “Hello, World!\n” is a string literal, delimited by double quotes

Strings – 2/6

- An escape character consists of a **backslash** \ symbol followed by a letter.
- **Escape characters are usually put inside double-quoted strings**
- **Escape characters are predefined and can be used in double-quoted strings.**

Strings – 3/6

- Backslashes are also used in **character escaping**.
- Suppose we would like to use double quotes in a double-quoted string.
- For example we would like to print

Howdy says, "Give me $500"

Strings – 4/6

Type the code below :

print "Howdy says, "Give me $500".";

- On execution, Perl locates the end of the string by searching forward until the second double quote is found.
- **If the literal contains double quotes itself, Perl will not know where the string literal terminates.**
- Perl will think the string ends after “Howdy says,”

Strings – 5/6

- Hence we place the \ character before the two symbols concerned, and this is what we mean to "**escape**" a character.
- The correct way to print this sentence using double quotes is:

```
print "Howdy says, \"Give me \$500\".";
```

- If we want to **use single quotes** instead, we **don't have to escape** anything:

```
print 'Howdy says, "Give me $500".';
```

Strings – 6/6

- **Single-quoted strings do not support variable substitution**
- There are **only two characters that need to be escaped in single-quoted strings**, namely ' and \
- **Empty strings** are denoted by **“”** or **‘’**, that is, two quotes with nothing in between.

Escape Character	Function
\n	Newline Starts a newline
\r	Carriage Return Returns to the starting point of the line
\t	Tab Analogous to striking the **Tab** key on your keyboard; However, using tab to make formatter output does not always generate the format expected.
\b	Backspace Analogous to the **Backspace** key; erases the last character
\a	Bell Creates a beep sound from the system buzzer (or sound card)
\xnn	ASCII character using hexadecimal notation Outputs the character which corresponds to the specified ASCII index (each n is a hexadecimal digit)

\0nn	ASCII character using octal notation Outputs the character which corresponds to the specified ASCII index (each n is an octal digit)
\cX	Control Character For example, \cC is equivalent to pressing Ctrl-C on your keyboard
\u	Next letter uppercase The letter immediately following \u is converted to uppercase. For example, \uemail is equivalent to Email
\l	Next letter lowercase The letter immediately following \l is converted to lowercase. For example, \lEmail is equivalent to email
\U	All subsequent letters uppercase All the letters immediately following \U are converted to uppercase until \E is reached
\L	All subsequent letters lowercase All the letters immediately following \L are converted to lowercase until \E is reached
\Q	Disables pattern matching until \E
\E	Ends \U, \L, \Q Terminates the effect of \U, \L or \Q.