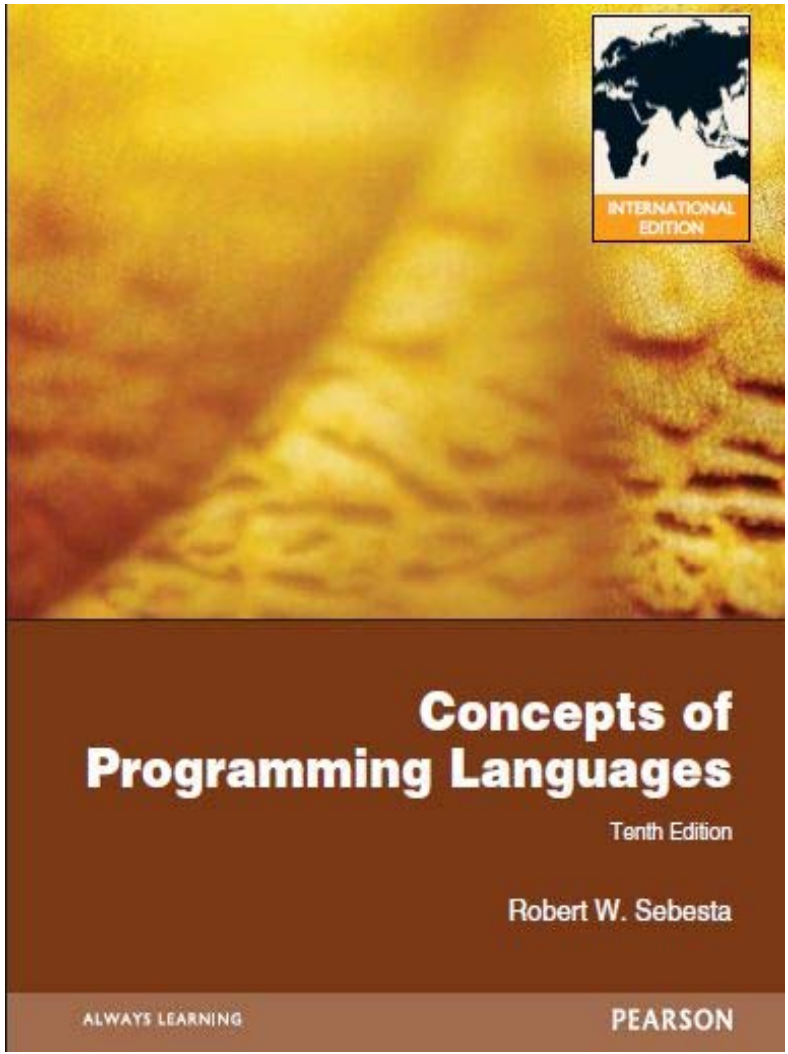
Programming Language

Instructor:
Min-Chun Hu
anita_hu@mail.ncku.edu.tw



Lecture 4
Regular Expression
Perl

Introduction

- Use special symbols to accomplish complex pattern matching tasks such as searching/replacing/deleting strings
- Useful in text processing, especially when dealing with HTML

Introduction

- Examples of regular expression:
  - Use “?” to represent that the previous character can be ignored
    - “color” and “colour” :
  - Use “\b” to represent boundary
    - “port” and “ports”, but not “export” or “important” :
  - Use “[ ]” to represent any element in the bracket
    - airport code such as “TPE”, “BOS”, “LAX” :
  - Use “\d” to represent any element in [0-9]
    - mobile phone number :
  - Use “+” to represent one or multiple occurrence
    - IP address :
  - Use “^”, “$”, “*” and “\s” to represent the start of a line, the end of a line, any number of occurrences, and any blank character [\t\n], respectively
    - Blank line :

Introduction

- Examples of regular expression:
  - Use “?” to represent that the previous character can be ignored
    - “color” and “colour” : colou?r
  - Use “\b” to represent boundary
    - “port” and “ports”, but not “export” or “important” : \bports?\b
  - Use “[ ]” to represent any element in the bracket
    - airport code such as “TPE”, “BOS”, “LAX” : \b[A-Z][A-Z][A-Z]\b
  - Use “\d” to represent any element in [0-9]
    - mobile phone number : 09\d\d-?\d\d\d-?\d\d\d
  - Use “+” to represent one or multiple occurrence
    - IP address : \d+\.\d+\.\d+\.\d+
  - Use “^”, “$”, “*” and “\s” to represent the start of a line, the end of a line, any number of occurrences, and any blank character [\t\n], respectively
    - Blank line : ^\s*$

Perl

- Things Perl can do easily
  - Find out if a string contains some specific pattern.
  - Parse a string
    - Extract components from a string according to some syntactical rules.
  - Replace parts of a string with other strings.
    - Change “abc” to “cde” in an article

Data Type

- Scalar
- Scalar Array
- Hash Array
- References

Scalar

- A scalar variable is declared with a start symbol '$'
- A local variable is declared with the syntax "my" or "local"
- Examples:
  - my $x="abc";
  - my $x=123;
  - my $x=4.82;

#註解

String Operators

Operator	Purpose
x	Returns a string consisting of the left operand repeated the number of times specified by the right operand.
.	Concatenates the two strings on both sides of the operator.
.=	The concatenation assignment operator.
eq	Returns True if the two operands are equivalent, False otherwise.
ne	Returns True if the two operands are not equal, False otherwise.
lt	Stringwise less than
le	Stringwise less than or equal
gt	Stringwise greater than
ge	Stringwise greater than or equal
cmp	Returns -1, 0, or 1 if the left operand is stringwise less than, equal to, or greater than the right operand.
++	Increments the string by one alphabetic value.

Value Operators

Operator	Operator	Operator
+	<	++
-	>	--
*	< = >	**
/	&&	+=
%	||	-=
==	&	*=
!=	|	>>
<=	^	<<
>=	~	

Scalar Array

- A scalar array variable is declared with a start symbol ‘@’
- Examples:
  - my @array;
  - my @array=qw(a b c);
  - $array[0]=“a”; $array[1]=“b”; $array[2]=“c”;
  - for($i=0; $i<=$#array; $i++) {
    print "$array[$i]\n";
    }

Hash Array (Associative Array)

- A hash array variable is declared with a start symbol '%'
- The index of a hash array is a string rather than an integer
- Example:
  - my %hash;
  - my %hash=("i1"=>"aaa","i2"=>"bbb","i3"=>"ccc");
  - $hash{'i1'}="aaa"; $hash{'i2'}="bbb"; hash{'i3'}="ccc";
  - foreach $key (keys %hash) {

```
    print "$hash{$key}\n";
}
```

  - foreach $value (values %hash)
  - while(($key,$value)=each %hash)

Reference (Pointer)

- Obtain the address of a variable and use it :
  - $scalarRef=\$scalarVar; print $$scalarRef;
  - $arrayRef=\@arrayVar; print "@$arrayRef";
  - $hashRef=\%hashVar; print $hashRef->{$key};
  - $funcRef=\&funcName; &$funcRef;

Reference (Pointer)

- Anonymous Array References (2D array):
  - $arrayRef=[[1,2,3,4],'a','b',['x','y','z'],'c'];
    print "$arrayRef->[3][2]\t$arrayRef->[2]\n";
- Anonymous Hash References:
  - $hashRef={a=>aa,b=>bb,c=>cc};
    print "$hashRef->{a}\t$hashRef->{b}\t$hashRef->{c}\n";

Control Statement

- Conditional Control Statements:
  - if (Expression) {Code Segment}
  - if (Expression) {Code Segment} else {Code Segment}
  - if (Expression) {Code Segment} elsif (Expression) {Code Segment} else {Code Segment}
  - statement if (Expression);
  - statement unless (Expression);

Control Statement

- Loop Control Statements :
  - for($i=0; $i<=10; $i++) {Code Segment}
  - for $i (0..10) {Code Segment}
  - foreach $i (@array) {Code Segment}
  - while($i<=10) {Code Segment}
  - do {Code Segment} while(Expression);
  - while(chomp($i=<STDIN>)) {

    去除換行符號　　標準輸入：現在key in的line

```
    next if ($i == 5);
    last unless ($i > 10);
}
```

    continue

    break

Subroutines

- Syntax: Function
  - sub NAME {code}
- Call a subroutine:
  - &NAME(para1, para2, …)
- Message passing:
  - @_ Global array　$_ Global variable （global array中的一個element）
- Variable Localization
  - “my” or “local”

必考：
call by value
call by reference
call by address

Example

```
my $s1 = &sum1(11,22,33);
my $s2 = &sum2(22,33,44);
my $s3 = &sum3(11,22,33,44,55);
print "s1=$s1, s2=$s2, s3=$s3\n";
sub sum1 {
   (my $first, my $second, my $third) = @_;
    return $first + $second + $third;
}
sub sum2 {
    my $first = $_[0];
    my $second = $_[1];
    my $third = $_[2];
    return $first + $second + $third;
}
```

Example (Cont.)

```
sub sum3 {
    my $s = shift @_;

    foreach ( @_ ) {
        $s = $s + $_;
    }
    return $s;
}
```

置入的第一個參數

下一個參數

File Open/Close

- Syntax:
  - open(FILEHANDLE,"Expression");
  - close(FILEHANDLE);
- Examples:
  - open(FILE, $filename) || die "Can't open file $filename : $!\n";
  - print while(<FILE>);
  - while($_=<FILE>){print "$_";}

File Handler:若後面無敘述，則為唯讀檔

Expression in Open Function

Expression	Effect
open(FH, "<filename")	Open filename for reading.
open(FH, ">filename")	Open filename for writing.
open(FH, "+<filename")	Open filename for both reading and writing without truncating it.
open(FH, "+>filename")	Open filename for both reading and writing with truncating it first.
open(FH, ">>filename")	Open a file in append mode. The writing point will be set to the end of the file. (cannot read)
open(FH, "+>>filename")	Open a file in append mode. The writing point will be set to the end of the file. (can read)

Input/Output

- Input 截取輸入存成input
  - $input=<STDIN>; chomp $input;
  - chomp($input=<STDIN>);
- Output
  - print "Scalar value is $x\n";
  - print "Scalar value is " . $x . "\n";
  - print FILE "print $x to a file.";
  - print<<XXX;
  - my $output = "標準輸出";
    print "$output\n";
    print STDOUT "$output\n";

- Examples:
  - $x="ls -l";

```
print "$x";          # Output ls -l
print "\$x";         # Output $x
print '$x';          # Output $x
print `$x`;          # Output files in this directory
```

當成system code執行

= 印出string

Special Characters

\t	tab	\x1b	hex char
\n	newline	\c[	control char
\r	return	\l	lowercase next char
\f	form feed	\u	uppercase next char
\b	backspace	\L	lowercase till \E
\a	alarm(bell)	\U	uppercase till \E
\e	escape	\E	end case modification
\033	octalchar	\Q	quoteregexp metacharacters till \E

Regular Expression

- Syntax:
  - $string =~ /regular expression/expression modifier
  - $string != /regular expression/expression modifier
- Examples:
  - $sentence =~ /Hello/

modifier	Effect
g	Match globally, i.e. find all occurrences.
i	Makes the search case-insensitive.
m	If the string has new-line characters embedded within it, the metacharacters ^ and $ will not work correctly. This modifier tells Perl to treat this line as a multiple line.
o	Only compile pattern once.
s	Allows . to match a new-line character.
x	Allows white space in the expression.

Metacharacter

Metacharacter	Effect
\	Accept the following characters as a regular character; this removes special meanings from any metacharacter.
^	Match the ***beginning*** of the string, unless /m is used.
.	Match any character except a new line character, unless /s is used.
$	Match the ***end*** of the string, unless /m is used.
|	Express alternation. This means the expressions will search for multiple patterns in the same string.
( )	Group expressions to assist in alternation and back referencing.
[ ]	Look for a set of characters.

Pattern Quantifier 出現幾次

Quantifier	Effect
*	Matchs 0 or more times.
+	Matchs 1 or more times.
?	Matchs 0 or 1 times.
{n}	Matches exactly n times.
{n,}	Matches at least n times.
{n,m}	Matches at least n times but no more than m times.

```
/b{3}/     #matches three b's
/(ha){3}/  #matches hahaha
```

Examples

/abc/
/^abc/
/abc$/
/a|b/
/ab{2,4}c/
/ab*c/
/ab+c/
/a.c/

/[abc]/ 含有abc任何一個子串
/ \d/ 含有任何數字
/ \w/ 找到字母和數字
/ \s/ 空白、換成、return
/[^abc]/ 沒有abc任一字元
/ */ 找特殊字元
/abc/I 找不論大小寫的abc
/(\d+)\.(\d+)\.(\d+)\.(\d+)/
找ip

Character Patterns

Character Patten	Usage
\r	Carriage return(CR)
\n	New line
\t	Tab
\w	Matches an ***alphanumeric*** character.
\W	Matches a nonalphanumeric character. 即 [^A-Za-z0-9_].
\s	Matches a ***white space*** character. This includes space, tab, FormFeed and CR/LF. 即 [\ \t\f\r\n].
\S	Matches a non-whote space character. 即 [^\ \t\f\r\n].
\d	Matches a ***digit***. 即 [0-9].
\D	Matches a nondigit character. 即 [^0-9].
\b	Matches a word boundary.
\B	Matches a nonword boundary.
\033	octal char
\x1B	hex char

Match Operator Example

/foo/

- Return true if "foo" is a substring of $_
- Return false is there is no "foo" in $_
- If $_ is "food is great" , return true
- If $_ is “blah foo” , return true

Foofinde.pl

```
# read one line at a time into $_
while (<>) {
   # see if the string in $_ contains "foo"
   if (/foo/) {
   # the string matches "foo", so print it
   print “Matched: $_”;
   }
}
```

Special Variables

Special Variable	Effect
$_	The default input and pattern-searching space.
$digit	Contains the subpattern from a successful parentheses pattern match.
$.	The current input line number of last filehandle read.
$!	Contains the current value of errno.
$0	The name of the file of the Perl script.
@ARGV	The command line arguments issued when the script was started.
@_	The parameter array for subroutines.
%ENV	This associative array contains your current environment.