

# Scalar Data Strings

Podstawy programowania

Jolanta Bachan  
2007-12-12

# What is scalar data?

Scalar data is the simplest kind of data. It can be either a **number** (like 2007 or 3.25e20) or a **string** of characters (like “Hello!” or an address).

Scalar variables always begin with the “\$” character.

# Strings

- Strings are sequences of characters. Strings may contain any combination of any characters. Each character is an 8-bit value from the entire 256-character set.
- The shortest possible string has no characters. The longest string fills all the available memory.
  - Single-quoted strings
  - Double-quoted strings

# Single-quoted strings

- A sequence of characters enclosed in single quotes.

'hello' (5 characters)

'don\'t' # don't

' # the null string (no characters)

'silly\\me' # silly\me

'hello\n' # hello\n

'hello

there' # hello, newline, there (11 characters)

- \n within a single quoted string is not interpreted as a newline.

# Double-quoted strings

- A sequence of characters enclosed in double quotes.
  - “hello world\n” # hello world & newline
  - “coke\tsprite” # coke & tab & sprite
  - “c:\\temp” # c: & backslash & temp

# single- vs. double quotes

- Single quotes are literal. They will pass on exactly what is inside of them. Contents are not analysed for variables.

- Example

```
$myname = Jola ;
```

```
print 'My name is $myname.' ;
```

– Result: My name is \$myname. ;

```
print "My name is $myname."
```

– Result: My name is Jola.

# Double-quoted String Representations

Construct	Meaning
<code>\n</code>	Newline
<code>\t</code>	Tab
<code>\\</code>	Backslash
<code>\"</code>	<u>Doublequote</u>
<code>\l</code>	<u>Lowercase</u> next letter
<code>\L</code>	<u>Lowercase</u> all following letters until <code>\E</code>
<code>\u</code>	<u>Uppercase</u> next letter
<code>\U</code>	<u>Uppercase</u> all following letters until <code>\E</code>
<code>\E</code>	Terminate <code>\L</code> or <code>\U</code>

# Scalar Operators for Strings

- “.” - concatenation

"hello" . "world" ; # same as "helloworld"

'hello world' . "\n" ; # same as "hello world\n"

"hello" . " " . "world" ; # same as "hello world"

- “x” - string repetition

"fred" x 3 ; # is "fredfredfred"

"barney" x (3+1) ; # is "barney" x 4, or  
"barneybarneybarneybarney"

(3+2) x 4 ; # is "5555"

# String Comparison Operators

- eq
- ne
- lt
- gt
- le
- ge

# Numeric and String Comparison Operators

Comparison	Numeric	String
Equal	<code>==</code>	<code>eq</code>
Not equal	<code>!=</code>	<code>ne</code>
Less than	<code>&lt;</code>	<code>lt</code>
Greater than	<code>&gt;</code>	<code>gt</code>
Less than or equal to	<code>&lt;=</code>	<code>le</code>
Greater than or equal to	<code>&gt;=</code>	<code>ge</code>

# Numeric and String Comparison Operators

- Example:

```
$b = 20;
```

```
$c = 7 ;
```

```
if ($b > $c) {
```

```
    print "b is greater than c" ;
```

```
} else {
```

```
    print "c is greater than b" ;
```

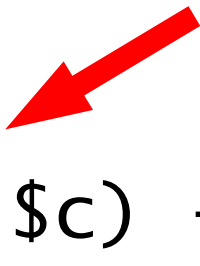
```
}
```

# Numeric and String Comparison Operators

- Example:

```
$b = 20;  
$c = 7 ;  
if ($b > $c) {  
    print "b is greater than c" ;  
} else {  
    print "c is greater than b" ;  
}
```

numeric comparison operator

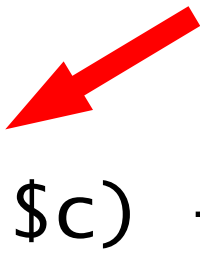


# Numeric and String Comparison Operators

- Example:

```
$b = 20;  
$c = 7 ;  
if ($b > $c) {  
    print "b is greater than c" ;  
} else {  
    print "c is greater than b" ;  
}
```

numeric comparison operator



Result: "b is grater than c"

# Numeric and String Comparison Operators

- Example:

```
$b = 20;
```

```
$c = 7 ;
```

```
if ($b gt $c) {
```

```
    print "b is greater than c" ;
```

```
} else {
```

```
    print "c is greater than b" ;
```

```
}
```

# Numeric and String Comparison Operators

- Example:

```
$b = 20;
```

```
$c = 7 ;
```

```
if ($b gt $c) {
```

```
    print "b is greater than c" ;
```

```
} else {
```

```
    print "c is greater than b" ;
```

```
}
```

string comparison operator

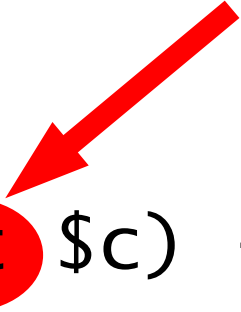


# Numeric and String Comparison Operators

- Example:

```
$b = 20;  
$c = 7 ;  
if ($b gt $c) {  
    print "b is greater than c" ;  
} else {  
    print "c is greater than b" ;  
}
```

string comparison operator



Result: "c is greater than b"

# Conversion between Numbers and Strings

- If you use a string value as an operand for a numeric operator (e.g. +), Perl automatically converts the string to its equivalent numeric value, as if you had entered it as a decimal floating point value.

- Example:

```
$string = "20" ;
```

```
$sum = $string + 10 ; # $sum is 30.
```

# Conversion between Numbers and Strings

- **BUT** something that *is not* a number at all converts to zero without warning.

Example:

```
$string_fred01 = "123.45fred" ;
```

```
$string_fred02 = "20fred" ;
```

```
$sum = $string_fred01 + $string_fred02 ;
```

Result: 143.45

# Conversion between Numbers and Strings

- Likewise, if you give a numeric value when a string is needed (e.g. for the string concatenation operator), the numeric value is converted to a string which that number represents.

Example:

```
$string_fred01 = "123.42fred" ;  
$number = 100 ;  
print $number . $string_fred01 ;
```

Result: 100123.45fred

# The chop() and chomp() Functions

- chop() takes a single argument within its parentheses – the name of the scalar variable – and removes the last character from the string value from that variable.

- Example:

```
$x = "hello world" ;
```

```
chop ($x); # $x is now "hello worl"
```

```
$a = chop($x) # $a is now "l"
```

# The chop() and chomp() Functions

- `chomp()` removes only the newline character.
- Example:

```
$x = "hello world\n" ;  
chomp ($x); # $x is now "hello world"  
chomp ($x); # no change in $x  
chop ($x) ; # oops! $x is now "hello  
world"
```

# length Function

- Returns the length of the string in bytes.
- Example:

```
$length = length "Nauka o  
informacji" ; # $length is 18.
```

# Exercise 1

- Write a program which reads a string and a number, and then prints the string the number of times indicated by the number on separate lines. (Hint: Use the x operator.)

# Exercise 1

- Write a program which reads a string and a number, and then prints the string the number of times indicated by the number on separate lines. (Hint: Use the x operator.)
  - Now modify the script, so that it prints the string on one line.

# Exercise 2

- Write a program which asks for a language. If you enter “Polish”, then it prints “Oh, it's my mother tounge!” If you enter some other language, it prints “I wish I spoke *<language>* fluently.” (Hint: Use the eq and chomp.)

Don't forget to bring gingerbread  
next week! :-)