CS100: Introduction to Computer Science

Lecture 14: Programming Language: Perl (I)

Review: Images

- Image in HTML
  - Background of your web page
  - Insert images
  - Make a hyperlink of an image
- Image representations & compression
  - GIF and JPEG
  - Pixels, image size, file size, image resolution

Review: Editing images with Photoshop

- Crop the image
- Resize the image
- Save files as GIF or JPEG
- Create images using Photoshop elements
- Change the background of a picture
  - [link]
- Deal with imperfections in images

Programming Languages

- Programming languages are used to facilitate communication about the task of organizing and manipulating information, and to express algorithms precisely.
- Many programming languages exist
  - Basic, Fortran, C, C++, JAVA, Perl
- Computers actually speak only one language, machine language.

Creating and Executing Programs

- Most programs are not written in machine language.
  - General editor for creating a program
  - "pl" for a program in perl
  - "c", for a program in C, "cpp" in C++
- A compiler is used to translate a program from one programming language into machine language.
- The resulting program (the object code) then is ready to run on your computer.
  - "exe" for "executable" in the Windows operating systems

What is an Interpreter?

- An interpreter is a program that implements or simulates a virtual machine using the base set of instructions of a programming language as its machine language.
- You can also think of an interpreter as a program that implements a library containing the implementation of the basic instruction set of a programming language in machine language.
- An interpreter reads the statements of a program, analyzes them and then executes them on the virtual machine or calls the corresponding instructions of the library.
Compiler versus Interpreter

- Using a compiler separates translation and execution of a program. The source code is translated only once.
- The compiled program is machine-dependent, it can only be executed on a machine for which it has been compiled, whereas an interpreted program is not machine-dependent.
- An interpreted program runs much slower than if it had been compiled. But it can take less time to interpret it than the total time required to compile and run it.

Scripting Languages: Perl

- Scripting programming languages or script languages are computer programming languages that are typically interpreted and can be typed directly from a keyboard.
- A Perl program is a text file containing a list of perl commands. (Perl script)
- Then we must start the Perl interpreter and tell it to run the Perl script.
  - On Unix, type `perl myperl.pl`
  - On Windows, double clicking the script file.

A Simple Perl Program

```
#!/usr/local/bin/perl
print "This is My First Perl Program!\n";
```

- Line 1 tells the script where to look for perl interpreter.
- Line 2 is the program. print is a perl command to output text.
  - Note that: The string outputted by print is in double quotes;
  - Every perl command is terminated with a semi-colon;
  - Leading white space is ignored.

Basic Concepts: Variables and Values

- Each variable acts as named storage location where a value is stored.
- Legal names for variables in Perl
  - Begin with $ (for scalar variables that can hold only one value)
  - Combinations of letters, numbers and a few characters including hyphen (-) and underscore (_) characters.
  - Variables are case sensitive.
- The value stored in a variable can be retrieved and changed at any time, and we can perform operations on it.

Data Typing

- A variable can contain more than just numbers
  - An integer number: 7, 100, or 98
  - A real number: 28.96
  - A character string: "Hello world"
- In many languages, you must declare what kinds of values a variable can contain before using it.
  - C, C++,
- In Perl, a variable can contain any data at any time. But decide
  - whether the variable can hold one or more values
  - Scalar variables, arrays, hashes

Examples: a Perl script uses a single variable to store several types of data (assignment operator “=”)

The program
```
$my_variable = 7;
prient ('my variable contains the value: $my_variable \n');
$my_variable = 7.3217;
prient ('my variable contains the value: $my_variable \n');
$my_variable = "apple";
prient ('my variable contains the value: $my_variable \n');
```

The output
```
my variable contains the value: 7
my variable contains the value: 7.3217
my variable contains the value: apple
```
Arithmetic Operators: used to execute general arithmetic procedures

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>7 + 7</td>
<td>= 14</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>7 - 7</td>
<td>= 0</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>7 * 7</td>
<td>= 49</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>7 / 7</td>
<td>= 1</td>
<td>Division</td>
</tr>
<tr>
<td>**</td>
<td>7 ** 7</td>
<td>= 823543</td>
<td>Exponents</td>
</tr>
<tr>
<td>%</td>
<td>7 % 7</td>
<td>= 0</td>
<td>Modulus</td>
</tr>
</tbody>
</table>

Assignment Operators: perform an arithmetic operation and then assign the value to the existing variable.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>Addition</td>
<td>($x += 10)</td>
</tr>
<tr>
<td>-=</td>
<td>Subtraction</td>
<td>($x -= 10)</td>
</tr>
<tr>
<td>*=</td>
<td>Multiplication</td>
<td>($x *= 10)</td>
</tr>
<tr>
<td>/=</td>
<td>Division</td>
<td>($x /= 10)</td>
</tr>
<tr>
<td>%=</td>
<td>Modulus</td>
<td>($x %= 10)</td>
</tr>
<tr>
<td>**=</td>
<td>Exponent</td>
<td>($x **= 10)</td>
</tr>
</tbody>
</table>

**Example:**

```bash
$a = 2;
$b = 6;
$c = $a + $b;
print "The value of c is $c \n";
$d = $b % $a;
print "The value of d is $d \n";
$e = $a % $b;
print "The value of e is $e \n";
```

**Working with strings**

- Create a string using double quotes
- Any variables inside the string are replaced with their contents

```bash
$first_name = "John";
$middle_initial = "Q."
$last_name = "Programmer"
$full_name = "$first_name $middle_initial $last_name";
```

**The String Operators (. and x)**

- the concatenation (.) operator
  ```bash
  $first_name = "David"
  $last_name = "Marshall"
  $full_name = $first_name . " " . $last_name;
  ```

- the repetition (x) operator
  ```bash
  $first_name = "David"
  $david_cubed = $first_name x 3;
  ```

**The chop() function**

- Removes the last character of a given string and return the last character.
- For example

```bash
$string = 'frog';
$chr = chop($string);
print "String: $string\n";
print "Char: $chr\n";
```
Problems with strings

- Working with variables and strings: append the letter y to a string variable

```perl
$name = "Mike";
Nickname = "$name y";
```

```perl
$name = "Mike";
Nickname = "$namey";
```

```perl
$name = "Mike";
Nickname = "$(name)y";
```

Problems with strings

- Print the following string to the screen, exactly as it appears below

  His name is Mike, but people call him "Mikey".

- How to include double quotes in the string?

  Escape sequences: special character sequence initiated with a backslash.

  \n, \t, ", '

```perl
print ('His name is Mike, but people call him \"Mikey\":');
```

Conversion between numbers and strings

```perl
$a = 2;
$b = 6;
$c = $a + $b;
print "The value of c is $c\n";
$c = $a - $b;
print "The value of c is $c\n";
$d = $c / 2;
print "The value of d is $d\n";
```

Note: If a string contains any trailing non-number characters they will be ignored.
I.e. "123-abc" would get converted to 123.45 for numeric work.
If no number is present in a string it is converted to 0.

Input from keyboard

```perl
#!/usr/local/bin/perl
print "Please enter your name\n";
#accept keyboard input, read it into $name
$name=<STDIN>;
print "Hello $name\n";
```

- Line 3 is a comment
- Line 4 accept keyboard input, read it into $name. STDIN is a buffer that contains the characters inputted by the keyboard
- Line 5 prints a greeting and the name inputted by the user (contained in $name).

Exercises 1:

- Write a program that display the following shape on the screen.

```
#
##
###
####
######
```

Exercise 2:

- Write a program that will take three numbers and output the sum of them.
Exercise 3: what is the output of the following program?

```perl
$x = "40";
$y = "11";
$z = $x + $y;
print "the value of z is $z \n";
$w = $x . $y;
print "the value of w is $w \n";
```

Announcements:

- Next lecture: perl (II)
- Lab 3 this week. Write codes in perl.