Research Skills 1: Programming

Erik Tjong Kim Sang
Herman Stehouwer

06 September 2007

Course Goal

To achieve a level of programming in some programming language (we chose Perl) which is sufficient for writing small programs for processing texts and other data sources (for example for supporting Master thesis work).

The course takes six weeks (6 September – 16 October).

Evaluation will take place via weekly programming assignments.

This course requires no prior programming experience.

Contents

The six classes in the course cover the following topics:

1. basic data structures and operators
2. control statements: conditions and loops
3. string processing and regular expressions
4. complex data structures: lists and hashes
5. subroutines and functions
6. file management

Reading material

The theoretical material presented in this course can be found online at **http://ifarm.nl/erikt/perl2007/**

In case you are looking for more information about Perl then we recommend the reference guide: **Programming Perl** by Larry Wall, Tom Christiansen and Jon Orwant.

If you are planning to use Perl for big programming projects then we advise you to take a look at **Perl Testing: A Developer's Notebook** by Ian Langworth and chromatic.

Class organization

Each class will start with a theoretical introduction of the topic (15 – 30 minutes).

After this you will start working on the assignments of the week which can be found on **http://ifarm.nl/erikt/perl2007/**

You are expected to send the programs written for the assignments as well as the associated answers and test results to erikt@science.uva.nl **before the next Wednesday**.

Your assignments will be checked and graded, and you will receive the results the next day.

Evaluation scheme

There will be five exercises each week. Three are obligatory. The other two, which are more difficult than the first three, are optional.

If you hand in results for three exercises you can receive a mark of up to eight points. If you hand in results for all five exercises you can receive a mark of up to ten points.

The final mark of the course will be the average of the six week marks.

Handing in too late: –2 points. Handing in more than one day too late: –4 points. Handing in after the start of the next class: 0 points.

About working together

Everyone should hand in his or her own assignment results.

Feel free to consult your colleagues about difficult parts of the exercises but make sure that the programs, question answers and test results you hand in, are written by you.

Handing in identical exercise results may result in the assessment score for that exercise being divided by the number of copies.

LET’S START!

Example program 1

```
$yearOfBirth = 1983;
$currentYear = 2007;
$age = $currentYear-$yearOfBirth;
print $age;
```

This program computes the age that a person will reach this year.

It contains three variables (`$yearOfBirth`, `$currentYear` and `$age`), an assignment and a printing instruction.

The result of the printing instruction will be displayed on the screen.

Variables

A variable is a storage location with a name.

The previous program contained three variables which consisted of a name preceded by a special token ($). In this course, we will deal with four of these tokens:

- $: variable containing numbers or strings (this lesson)
- @: variable containing a list with numeric keys (lesson 4)
- %: variable containing a list with strings as keys (lesson 4)
- &: subroutine (lesson 5)

Arithmetic operators

Example program 1 contained an assignment with a subtraction operator. Here is an overview of some arithmetic Perl operators:

- +: sum: 1+1=2
- –: subtraction: 3–2=1
- *: product: 2*3=6
- /: division: 6/3=2
- %: modulo division: 8%5=3
- **: exponent: 2**3=8

Arithmetic functions

Perl also contains some built-in arithmetic functions. Here are some useful ones:

- abs($x): absolute value of $x; abs(-1) = abs(1) = 1
- int($x): integer part of $x; int(3.2) = 3
- rand(): random number between 0 and 1
- sqrt($x): square root of $x; sqrt(4) = 2

The operators and functions can be combined in arbitrary ways. Example: $x = abs(rand()+1)+int(sqrt(4)/2);

Example program 2

```
# age calculator
print "Please enter your birth year ";
$yearOfBirth = <STDIN>;
chomp($yearOfBirth);
print "Your age is ",2007-$yearOfBirth,".\n";
```

- lines starting with # are comment lines
- the function `<STDIN>` reads a line from the keyboard
- the function `chomp($x)` removes a newline character from $x
- \n is the code for a newline character

Example program 3

```
# example.03:  compute the day of the week
# usage:  perl example.03
# note:  the program will only work correctly for
# the year 2007 and other non-leap years which start
# on a Monday
# 20000203 erikt@uia.ua.ac.be
# 20070821 erikt@science.uva.nl

$dateNbr = 6; # December 31, 2006 is a Sunday
```

```
# determine month
print "Answer the questions with 1 (yes) or 0 (no).\n";
print ”Is the day after January 31? ”;
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
print "Is the day after February 28?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*28;
print "Is the day after March 31?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
print "Is the day after April 30?  ";
```

```
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*30;
print "Is the day after May 31?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
print "Is the day after June 30?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*30;
print "Is the day after July 31?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
```

```
print "Is the day after August 31?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
print "Is the day after September 30?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*30;
print "Is the day after October 31?  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer*31;
print "Is the day after November 30?  ";
$answer = <STDIN>;
chomp($answer);
```

```
$dateNbr = $dateNbr+$answer*30;
print "\n";

# read day of month
print "Please enter the day of the month:  ";
$answer = <STDIN>;
chomp($answer);
$dateNbr = $dateNbr+$answer;

# show answer
print "In the answer the following table has been used:\n"
print "0 = Monday\n";
print "1 = Tuesday\n";
print "2 = Wednesday\n";
print "3 = Thursday\n";
```

```
print "4 = Friday\n";
print "5 = Saturday\n";
print "6 = Sunday\n";
print "\n";
print "The week day number of your date is ";
print $dateNbr%7;
print "\n";

# exit
exit(0);
```

Programming tips

Programming time can be divided in 10% of writing code and 90% of testing and debugging. So make sure your programs are as easy to understand as possible:

- write clean code
- use comments to explain the code
- use newlines to separate unrelated blocks of code
- use meaningful variable names
- use indentation (after next lesson)

START WITH EXERCISES / LINUX HELP AT
http://ifarm.nl/erikt/perl2007/