# Research Skills 1: Programming Lesson 4

Erik Tjong Kim Sang Herman Stehouwer

27 September 2007

## Week 1: Variables and number processing

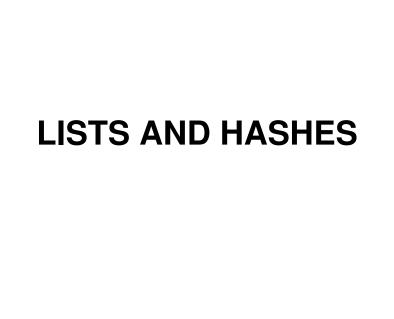
- names of numeric variables start with a dollar sign (\$yearOfBirth)
- several arithmetic operators are available: + − \* / % \*\*
- as well as several functions: abs() int() rand() sqrt()
- input can be read from the keyboard: <STDIN>

#### Week 2: Control structures

- Conditional structures: if (condition) { command }
- Truth expressions: and, or and not
- Iterative structures: while (), for(;;) and foreach ()

## Week 3: String processing

- Basic string operations: concatenation (.); testing (eq and others)
- String substitution: tr///, s/// and //
- Regular expressions: special tokens like \s, \$ and ★



#### List and hashes: overview

Lists and hashes both are sets of objects. But...

- lists are ordered; hashes are unordered
- list names start with @; hash names start with %
- lists use integers (0, 1, 2, ...) as keys; hashes use arbitrary strings
- lists use [] as subscripting operator: \$list[0]; hashes use {}: \$hash{"string"}

### List examples (1)

```
@a = ();  # empty list
@b = (1,2,3);  # three numbers
@c = ("Jan","Piet","M");  # three strings
@d = ("Dirk",1.92,46);  # a mixed list

$a = 1;
@b = ($a,$a+1,$a+2);  # same as: (1,2,3)
@c = ("Jan",("Piet","M"));  # ("Jan","Piet","M")
@d = ("Dirk",1.92,(),46);  # ("Dirk",1.92,46);
@e = (@b,@c);  # (1,2,3,"Jan","Piet","M")
```

## List examples (2)

Shorthand operators for filling lists: .. and qw

```
0x = (1..6); # same as: (1,2,3,4,5,6)

0y = ("a".."e"); # ("a","b","c","d","e")

0z = qw(Jan Piet M); # ("Jan","Piet","M")
```

Finding out how many elements a list contains:

```
@list = qw(a b c d);
$length = @list;  # $length = 4
$lastId = $#list;  # $lastId = 3 (ids start at 0)
```

## List examples (3)

#### Using lists in assignments:

```
(\$a,\$b) = ("one","two"); # \$a = "one" and $b = "two"

(\$one,@many) = (1,2,3); # \$one = 1 and @many = (2,3)

(\$a,\$b) = (\$b,\$a); # \$a = "two" and $b = "one"
```

#### Accessing list elements:

```
@list = qw(a b c d);
$z = $list[0];  # "a"
$y = $list[1];  # "b"
$x = $list[-1];  # "d"
```

#### **List functions**

There are four functions for adding or removing elements from the beginning or end of the list:

```
push (@list1, @list2)
pop(@list1)
shift (@list1)
```

add @list2 to the end of @list1 remove element from the end of @list1 remove element from the start of @list1 unshift (@list1, @list2) add @list2 to the start of @list1

Additionally there is a command for manipulating the middle of a list:

```
splice (@list1, $start, $length, @list2)
```

Remove \$length elements from \$start and replace by @list2

## **List function examples (1)**

```
@list1 = qw(a b);
@list2 = qw(cd);
push(@list1,@list2);  # @list1 = qw(a b cd)
push(@list2,"e");  # @list2 = qw(cd e)
$item = shift(@list2);  # $item = c; @list2 = qw(d e)

@list1 = qw(a b);
@list2 = qw(cd);
unshift(@list1,@list2);  # @list1 is qw(cd a b)
unshift(@list2,"e");  # @list2 is qw(ecd)
$item = pop(@list2);  # $item = d; @list2 = qw(e c)
```

## **List function examples (2)**

```
@list = qw(a b c d);
@cut = splice(@list,1,2);
# @cut = qw(b c) and @list = qw(a d)
@list1 = qw(a b c d);
@list2 = qw(e f);
@cut = splice(@list1,1,2,@list2);
# @cut = qw(b c) and @list1 = qw(a e f d)
```

#### **More list functions**

```
returns alphabetically sorted copy of @list
reverse(@list)
returns reversed copy of list (first ← last)

join ($string, @list)
   converts @list to string with $string as separator
   $string = join("+", (1,2,3)); # "1+2+3"

split ($regexp, $string)
   converts $string to list using $regexp as separator
   @list = split(/::/, "::a::b"); # ("", "a", "b")
```

## Hash examples (1)

Hashes are like lists but use strings as keys:

Remember that hash names start with % and that hashes use {} as subscripting operator rather than the [] that lists use.

#### **Hash functions**

```
keys(%hash)
values(%hash)
reverse(%hash)
exists($hash{"k"})
defined($hash{"k"})
delete($hash{"k"})
```

returns a list of the keys of the hash returns a list of the values of the hash returns a reversed copy of the hash tests if \$hash{"k"} exists tests if \$hash{"k"} contains a valid value deletes \$hash{"k"} from the hash

#### Notes:

- a reversed hash has its values as keys and its keys as values
- defined() can also be used for other variables

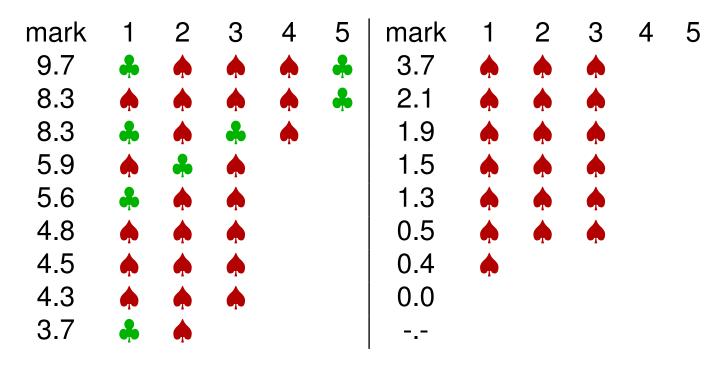
## Hash examples (2)

Printing the contents of a hash is a task which needs to be performed frequently. Here is an example of how this can be done:

```
foreach $key (sort keys %hash) {
    print "The value associated with key $key ";
    print "is $hash{$key}\n";
}
```



#### **Exercise results week 3**



♣ = perfect; ♠ = one or more errors

## Most common problem in the exercises

The regular expression a\* matches strings that contain a's.

However, if we want to make sure that a string *only* contains a's, then we need to test the following:

between the start and the end of the string there should only be a's

This requires the regular expression ∧a \* \$

#### **About the exercises**

see the web site, for example: http://ifarm.nl/erikt/perl2007/31.txt

#### Extra class

We are trying to arrange an extra lab class so that people that need this can work on the exercises with supervision.

The target dates/times are Mondays 14:45-16:30 in room DZ11.

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