

## Assignment 1: Perl Hashes and Arrays — Solutions

**Submission:** by 5pm, Friday, 11 October 2002

**Where:** On paper, to the office

**Important:** All programs you write must start with these lines:

```
#!/usr/bin/perl -w  
use strict;
```

and should compile without errors.

**Cheating:** Your work *must* be original. Copying will be *severely* dealt with.

### Assignment Requirements

1. What happens if you assign an array to a hash? Explain what happens in this example:

```
#!/usr/bin/perl -w  
use strict;  
my @array = ( 'word1', 'word2', 'word3', 'word4' );  
my %hash = @array;  
# Now what values are stored in %hash?
```

Examine the documentation for the builtin functions `keys` and `each`, with `perldoc -f keys` and `perldoc -f each`.

Add to this program code to print the keys and values stored in `%hash`, showing which is a key, and which is a value, and write an explanation of the output.

**Solution:** Here is a little extension to the above code to print the contents of the hash:

```
#!/usr/bin/perl -w  
use strict;  
my @array = ( 'word1', 'word2', 'word3', 'word4' );  
my %hash = @array;  
# Now what values are stored in %hash?  
foreach my $key ( keys %hash ) {  
    print "key = '$key', value = '$hash{$key}'\n";  
}
```

If this program file is called `hash-array-ans`, then here is the result of executing it:

```
$ ./hash-array-ans
key = 'word1', value = 'word2'
key = 'word3', value = 'word4'
```

The result is quite simple: the elements of the array with an even index (0, 2, 4, ...) are the keys of the hash; the elements of the array with an odd index (1, 3, 5, ...) are the corresponding values for the hash.

A very important thing you must understand is that a hash is not ordered (well, not in any way that you might imagine by the ordinary meaning of “ordered”.) You cannot have a hash with its contents sorted.

- Write a program to read the lines from any number of input files, and print the words from the file, one word on each line.

Use the builtin `split` function described in the last laboratory exercise. You should also use the angle-bracket operator “<>”.

**Solution:** With Perl, the motto is “There is more than one way to do it” (TIMTOWTDI), so many good solutions will look different from what is here.

A possible solution is:

```
#!/usr/bin/perl -w
use strict;

while ( <> )
{
    next unless /\S/; # Skip blank lines
    my @line = split;
    foreach my $word ( @line )
    {
        print "$word\n";
    }
}
```

- Extend your program from the previous exercise to print a summary of how many times each word occurred in all of the input files. Use a hash as the basic data structure in your program.

**Solution:** See the solution to question 5.

- Modify your program from the previous exercise to also sort the output by the ASCII order of the words.

Study the documentation for the builtin function `sort`; do `perldoc -f sort`.

**Solution:** See the solution to question 5.

- Modify your program from question 3 so that the output is sorted by frequency of the words, so that the most frequent words come at the beginning of the output.

**Solution:** Here is a program that does what is required for question 3, question 4 and for question 5.

```
#!/usr/bin/perl -w
use strict;

our %words;
while ( <> )
{
    next unless /\S/; # Skip blank lines
    my @line = split;
    foreach my $word ( @line )
    {
        ++$words{$word};
    }
}

print "Words unsorted, in the order they come from the hash:\n\n";

foreach my $word ( keys %words )
{
    printf "%4d %s\n", $words{$word}, $word;
}

print "Words sorted in ASCII order:\n\n";

foreach my $word ( sort keys %words )
{
    printf "%4d %s\n", $words{$word}, $word;
}

print "\n\n\nNow sorted by frequency:\n\n";

foreach my $word ( sort { $words{$b} <=> $words{$a} } keys %words )
{
    printf "%4d %s\n", $words{$word}, $word;
}
```