```
#!/usr/bin/perl -w

# hello.pl by Lassi Paavolainen (lopaavol@cc.jyu.fi)

# Esimerkki perinteisestä hello worldistä Perlillä

use strict; # Sitoo muuttujat lohkoon.

# Kun strict moduuli on otettu käyttöön, täytyy muuttujat sitoa my käskyllä,
# eli käytettäessä kyseisen nimistä muuttujaa ensimmäisen kerran lohkossa,
# täytyy nimen edessä olla my.
my $text = "Hello World!\n"; # Sijoitetaan merkkijono muuttujaan.

print $text; # Tulostaa Hello World!
print "$text"; # Tulostaa saman
print '$text'."\n"; # Tulostaa $text, koska '-merkit eivät lisää muuttujan
                    # sisältöä tulostukseen.
```

```
#!/usr/bin/perl -w

# silmukat.pl by Lassi Paavolainen (lopaavol@cc.jyu.fi)

# Yksinkertainen demonstraatio erilaisista silmukoista Perlissä

use strict; # otetaan strict moduli käyttöön

# Sijoitetaan taulukkoon parametrina saadut alkiot. @ARGV on erityinen taulukko
# jossa ohjelman parametrit tuodaan.
my @arguments = @ARGV;

while (@arguments) { # Niin kauan kun taulukossa on alkioita.
    my $parameter = shift @arguments; # Poistaa ensimmäisen alkion ja sijoittaa
    print $parameter . "\n";
}

@arguments = @ARGV;

until (@arguments == 0) { # Niin kauan kunnes taulukon alkioita on 0 kappaletta
    my $parameter = shift @arguments;
    print $parameter . "\n";
}

for (my $i=0; $i < @ARGV; $i++) { # Perinteinen for-silmukka.
    print $ARGV[$i] . "\n";
}

# Kätevin viimeisenä

foreach (@ARGV) { # Käy taulukon kaikki alkiot läpi. Koska muuttujaa ei ole
    print $_ . "\n"; # nimetty, tallennetaan yksittäinen alkio oletusmuuttujaan
}

# tai

foreach my $alkio (@ARGV) {# Sama kuin edellä, mutta nyt nimetään itse muuttuja
    print $alkio . "\n";
}
```

```
#!/usr/bin/perl -w

# template.pl by Lassi Paavolainen (lopaavol@cc.jyu.fi)

# Esimerkkiskripti HTML::Template-modulin käytöstä.

use strict; # Otetaan strict-moduli käyttöön.
use Template; # Käytetään Template-modulia. Olettaa, että moduli on samassa
              # hakemistossa.

my $file = "template.tmpl"; # Käytettävän template-tiedoston nimi
my $outfile = "template.html"; # Tulostettavan tiedoston nimi

# Luodaan uusi HTML::Template-modulin "ilmentymä". Samalla voidaan määritellä
# käytettävän template-tiedoston nimi.
my $template = HTML::Template->new(filename => "$file");
my @people; # Taulukko pitää sisällään kaikkien henkilöiden tiedot.

# 1. Henkilön tietojen luonti
my $name = "Aku Ankka";
my $address = "Ankkalinnankatu 13";
my @numbers; # Jokaisessa alkiossa yksi puhelinnumero
my %hash_num = (nro => "040-123456");
push(@numbers,\%hash_num); # Viedään viite hajautustaulusta taulukkoon.
my %hash_num1 = (nro => "555-567432");
push(@numbers,\%hash_num1);
# Luodaan hajautustaulu, joka sisältää kaikki henkilön tiedot. HTML::Template-
# modulia varten kaikki tiedot on oltava hajautustauluissa.
my %hash = ( nimi => $name,
             osoite => $address,
             numerot => \@numbers ); # Viite @number-taulukkoon.
push(@people,\%hash); # Sijoitetaan hajautustaulu taulukkoon @people.

# 2. Henkilön tietojen luonti. Tehdään asiat samalla tavalla kuin äsken.
# Tässä joudutaan keksimään uudet nimet muuttujille, sillä modulia varten
# käytetään viitteitä. Mikäli käytetäisiin edellä olevia muuttujia,
# pyyhkiytyisivät aikaisemmat tiedot niistä pois.
# Normaalisti tämä ei ole ongelma, sillä esimerkiksi silmukassa uusi kierros
# alustaa aina uuden muuttujan my muuttujan_nimi komennolla, eikä tämä siis
# viittaa aikaisempaan muuttujaan.
my @numbers1;
my $name1 = "Roope-setä";
my $address1 = "Rahamäki 1";
my %hash_num2 = (nro => "040-111222");
push(@numbers1,\%hash_num2);
my %hash_num3 = (nro => "335786");
push(@numbers1,\%hash_num3);
my %hash_num4 = (nro => "555-999000");
push(@numbers1,\%hash_num4);
my %hash1 = ( nimi => $name1,
              osoite => $address1,
              numerot => \@numbers1 );
push(@people,\%hash1);
```

```
# 3. henkilön tietojen luonti. Samalla lailla kuin aikaisemmissa.
my @numbers2;
my $name2 = "Hannu Hanhi";
my $address2 = "Onnenkylä 2";
my %hash_num5 = (nro => "555-111111");
push(@numbers2,\%hash_num5);
my %hash2 = ( nimi => $name2,
              osoite => $address2,
              numerot => \@numbers2 );
push(@people,\%hash2);

# Viedään template-modulille uusi parametri. Tässä tapauksessa henkilot
# parametri osoittaa taulukkoon. Yksittäinen muuttuja voitaisiin viedä näin
# $template->param(nimi => $nimi); ja käytettäisiin sitten suoraan template-
# tiedostossa näin <tmpl_var name="nimi">
$template->param(henkilot => \@people);

open(FILE,">$outfile"); # Aukaistaan tiedostokahva.
# $template->output() käsky palauttaa template-tiedoston tekstin korvatuin
# muuttujin.
print FILE $template->output();
close(FILE); # Suljetaan tiedostokahva.
```

```
<!-- template.tmpl by Lassi Paavolainen (lopaavol@cc.jyu.fi) -->

<!-- Esimerkki template-tiedostosta HTML::Template-modulia varten. -->
<!-- Yksittäinen muuttuja korvataan <tmpl_var name="nimi">-tagin tilalle -->
<!-- Taulukko käydään läpi <tmpl_loop name="nimi"> ja -->
<!-- </tmpl_loop>-tagien avulla. -->
<!-- Lisäksi olemassa <tmpl_if name="nimi"> jolla voidaan tutkia -->
<!-- onko taulukko tyhjä tai merkkijono tyhjä. -->
<!-- Lisätietoa http://html-template.sourceforge.net/ -->


<html>
<head>
<title>Esimerkki templaten käytöstä</title>
</head>

<body>
<h1>Tämä on template esimerkki</h1>

<table border="1">
<tr>
<td>Nimi</td>
<td>Osoite</td>
<td>Puh.nro</td>
</tr>
<!-- Seuraavaa osaa </tmpl_loop>-tagiin asti toistetaan niin paljon kuin -->
<!-- taulukossa alkioita. -->
<tmpl_loop name="henkilot">
<tr>
<td><tmpl_var name="nimi"></td> <!-- Tähän korvataan nimi -->
<td><tmpl_var name="osoite"></td> <!-- Tähän osoite -->
<td>
<tmpl_loop name="numerot"> <!-- Taulukon sisällä voi olla taulukko -->
<tmpl_var name="nro"><br /> <!-- Tähän yksittäinen puh.nro -->
</tmpl_loop> <!-- Lopetus tagit pitää muistaa -->
</td>
</tr>
</td>
</tmpl_loop>
</table>

</body>
```