Places To Put Things

Exploring Perl's Built-In Variable Containers:
Arrays and Hashes

Beyond Scalars

Arrays: Associating Data With Numbers

```
@list_of_sequences
@totals
@protein_structures
```

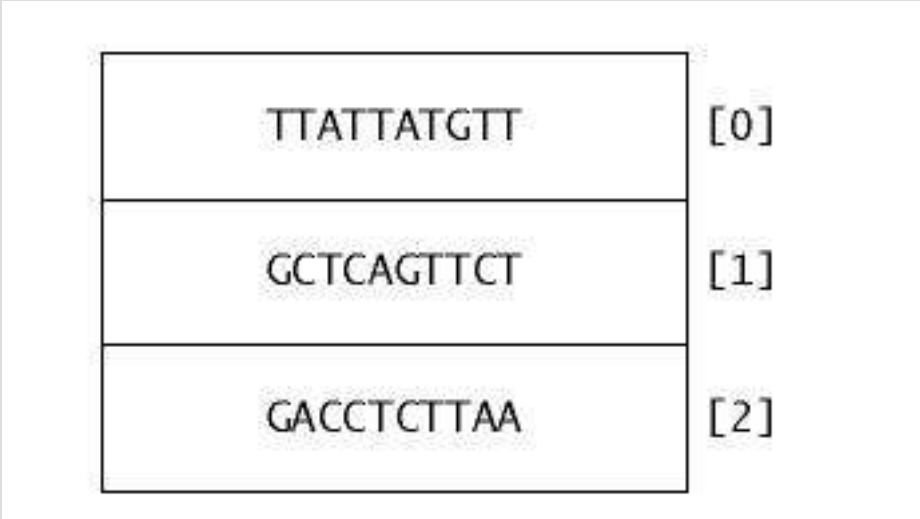
```
( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' )
```

```
@list_of_sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' );
```

Maxim 4.1

Lists in Perl are comma-separated collections of scalars

The @list_of_sequences Array



Maxim 4.2

Perl starts counting from zero, not one

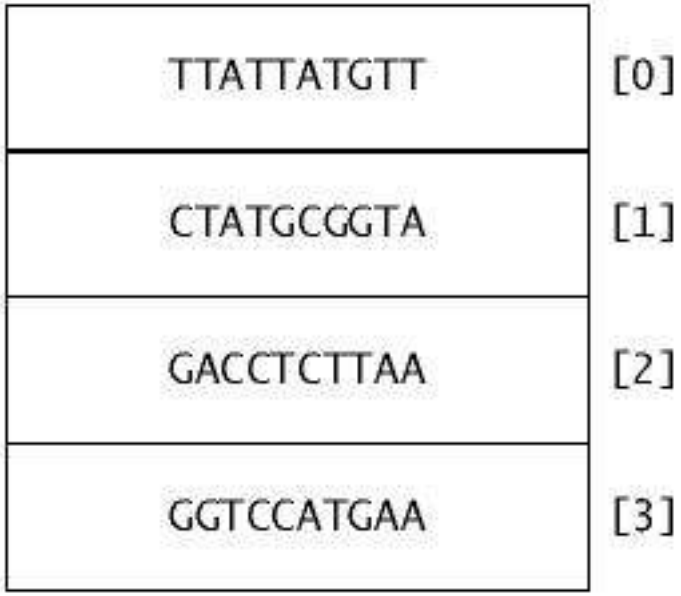
Working with array elements

```
print "$list_of_sequences[1]\n";



$list_of_sequences[1] = 'CTATGCGGTA';
$list_of_sequences[3] = 'GGTCCATGAA';
```

The Grown @list_of_sequences Array



How big is the array?

```
print "The array size is: ", $#list_of_sequences+1, ".\n";
print "The array size is: ", scalar @list_of_sequences,
   ".\n";

The array size is: 4.
```

Maxim 4.3

There are three main contexts in Perl: numeric, list and scalar

Adding elements to an array

```
@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' );
@sequences = ( @sequences, 'CTATGCGGTA' );

print "@sequences\n";

TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA

@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' );
@sequences = ( 'CTATGCGGTA' );
print "@sequences\n";

CTATGCGGTA
```

Adding more elements to an array

```
@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' );
@sequences = ( @sequences, ( 'CTATGCGGTA', 'CTATTATGTC' ) );
print "@sequences\n";

TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA CTATTATGTC

@sequence_1 = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA' );
@sequence_2 = ( 'GCTCAGTTCT', 'GACCTCTTAA' );
@combined_sequences = ( @sequence_1, @sequence_2 );
print "@combined_sequences\n";

TTATTATGTT GCTCAGTTCT GACCTCTTAA GCTCAGTTCT GACCTCTTAA
```

Removing elements from an array

```
@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                    'TTATTATGTT' );
@removed_elements = splice @sequences, 1, 2;
print "@removed_elements\n";
print "@sequences\n";

GCTCAGTTCT GACCTCTTAA
TTATTATGTT TTATTATGTT



@sequences = ();
```

Slicing arrays

```
@dnas[ 1, 4, 9 ]

@dnas[ 1 .. 9 ]
```

The slices program

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

# The 'slices' program - slicing arrays.

@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                      'CTATGCGGTA', 'ATCTGACCTC' );
print "@sequences\n";
@seq_slice = @sequences[ 1 .. 3 ];
print "@seq_slice\n";
print "@sequences\n";
@removed = splice @sequences, 1, 3;
print "@sequences\n";
print "@removed\n";
```

Results from slices ...

```
TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA ATCTGACCTC
GCTCAGTTCT GACCTCTTAA CTATGCGGTA
TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA ATCTGACCTC
TTATTATGTT ATCTGACCTC
GCTCAGTTCT GACCTCTTAA CTATGCGGTA
```

Maxim 4.4

To access a list of values from an array, use a slice

Maxim 4.5

To remove a list of values from an array, use splice

Pushing, popping, shifting and unshifting

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

# The 'pushpop' program - pushing, popping, shifting
# and unshifting.

@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                    'CTATGCGGTA', 'ATCTGACCTC' );


print "@sequences\n";
$last = pop @sequences;
print "@sequences\n";
$first = shift @sequences;
print "@sequences\n";
unshift @sequences, $last;
print "@sequences\n";
push @sequences, ( $first, $last );
print "@sequences\n";
```

Results from pushpop ...

```
TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA ATCTGACCTC
TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA
GCTCAGTTCT GACCTCTTAA CTATGCGGTA
ATCTGACCTC GCTCAGTTCT GACCTCTTAA CTATGCGGTA
ATCTGACCTC GCTCAGTTCT GACCTCTTAA CTATGCGGTA TTATTATGTT ATCTGACCTC
```

Processing every element in an array

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

# The 'iterateW' program - iterate over an entire array
# with 'while'.

@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                    'CTATGCGGTA', 'ATCTGACCTC' );

$index = 0;
$last_index = $#sequences;

while ( $index <= $last_index )
{
    print "$sequences[ $index ]\n";
    ++$index;
}
```

Results from iterateW ...

TTATTATGTT
GCTCAGTTCT
GACCTCTTAA
CTATGCGGTA
ATCTGACCTC

The iterateF program

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

# The 'iterateF' program - iterate over an entire array
# with 'foreach'.

@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                    'CTATGCGGTA', 'ATCTGACCTC' );

foreach $value ( @sequences )
{
    print "$value\n";
}
```

Maxim 4.6

Use foreach to process every element in an array

Making lists easier to work with

```
@sequences = ( 'TTATTATGTT', 'GCTCAGTTCT', 'GACCTCTTAA',
                    'CTATGCGGTA', 'ATCTGACCTC' );

@sequences = ( TTATTATGTT, GCTCAGTTCT, GACCTCTTAA,
                   CTATGCGGTA, ATCTGACCTC );

@sequences = qw( TTATTATGTT GCTCAGTTCT GACCTCTTAA
                    CTATGCGGTA ATCTGACCTC );
```

Hashes: Associating Data With Words

```
%bases
%genomes
%nucleotide_bases

%nucleotide_bases = ( A, Adenine, T, Thymine );
```

Maxim 4.7

A hash is a collection of name/value pairings

The %nucleotide_bases Hash

A	Adenine
T	Thymine

Maxim 4.8

Hash name-parts must be unique

Working with hash entries

```
print "The expanded name for 'A' is $nucleotide_bases{ 'A' }\n";
```

How big is the hash?

```
%nucleotide_bases = ( A, Adenine, T, Thymine );

@hash_names = keys %nucleotide_bases;

print "The names in the %nucleotide_bases hash are: @hash_names\n";

The names in the %nucleotide_bases hash are: A T


%nucleotide_bases = ( A, Adenine, T, Thymine );

$hash_size = keys %nucleotide_bases;

print "The size of the %nucleotide_bases hash is: $hash_size\n";

The size of the %nucleotide_bases hash is: 2
```

Adding entries to a hash

```
$nucleotide_bases{ 'G' } = 'Guanine';
$nucleotide_bases{ 'C' } = 'Cytosine';

%nucleotide_bases = ( A => Adenine, T => Thymine,
                      G => Guanine, C => Cytosine );
```

The Grown %nucleotide_bases Hash

A	Adenine
T	Thymine
C	Cytosine
G	Guanine

Removing entries from a hash

```
delete $nucleotide_bases{ 'G' };

$nucleotide_bases{ 'C' } = undef;
```

Slicing hashes

```
%gene_counts = ( Human => 31000,
                 'Thale cress' => 26000,
                 'Nematode worm' => 18000,
                 'Fruit fly' => 13000,
                 Yeast => 6000,
                 'Tuberculosis microbe' => 4000 );


@counts = @gene_counts{ Human, 'Fruit fly', 'Tuberculosis
   microbe' };

print "@counts\n";
```

Working with hash entries: a complete example

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

# The 'bases' program - a hash of the nucleotide bases.

%nucleotide_bases = ( A => Adenine, T => Thymine,
                      G => Guanine, C => Cytosine );

$sequence = 'CTATGCGGTA';

print "\nThe sequence is $sequence, which expands to:\n\n";

while ( $sequence =~ /(.)/g )
{
    print "\t$nucleotide_bases{ $1 }\n";
}
```

Results from bases ...

```
The sequence is CTATGCGGTA, which expands to:

Cytosine
Thymine
Adenine
Thymine
Guanine
Cytosine
Guanine
Guanine
Thymine
Adenine
```

Processing every entry in a hash

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

# The 'genes' program - a hash of gene counts.

use constant LINE_LENGTH => 60;

%gene_counts = ( Human => 31000,
                 'Thale cress' => 26000,
                 'Nematode worm' => 18000,
                 'Fruit fly' => 13000,
                 Yeast => 6000,
                 'Tuberculosis microbe' => 4000 );
```

The genes program, cont.

```
print '-' x LINE_LENGTH, "\n";

while ( ( $genome, $count ) = each %gene_counts )
{
    print "`$genome' has a gene count of $count\n";
}

print '-' x LINE_LENGTH, "\n";

foreach $genome ( sort keys %gene_counts )
{
    print "`$genome' has a gene count of $gene_counts{ $genome }\n";
}

print '-' x LINE_LENGTH, "\n";
```

Results from genes ...

```
-----------------------------------------------------
`Human' has a gene count of 31000
`Tuberculosis microbe' has a gene count of 4000
`Fruit fly' has a gene count of 13000
`Nematode worm' has a gene count of 18000
`Yeast' has a gene count of 6000
`Thale cress' has a gene count of 26000
-----------------------------------------------------
`Fruit fly' has a gene count of 13000
`Human' has a gene count of 31000
`Nematode worm' has a gene count of 18000
`Thale cress' has a gene count of 26000
`Tuberculosis microbe' has a gene count of 4000
`Yeast' has a gene count of 6000
-----------------------------------------------------
```

Where To From Here