Special Topics in Bioinformatics, 2004

Concept of object-oriented programming

- week4 -

YuFan Liu

10-Mar, 2004
Reading in reference book

- Chapter 8. References
- Chapter 9. Data Structures
The metacharacters - alternatives

- The **barney** is a student.
- The **fred** is a student, too.
- The **betty** is a student, too.

\[
\text{string} =~ /fred|barney|betty/;
\]

Homework 7

- \[
\text{string} =~ /fred \text{ (and|or) } barney/;
\]

a) fred and barney are students. \(\rightarrow\) ok? \ OK
b) fred or barney are students. \(\rightarrow\) ok? \ OK
c) fred and\|or barney are students. \(\rightarrow\) ok? \ not OK
d) fred and\or barney are students. \(\rightarrow\) ok? \ not OK

Copyright 2004 © NYMU Bioinformatics

Email: perl@ym.edu.tw
Homework 8 - To find the common pattern

- `<a href='advantage.htm'>z</a>`
- `<a href='statistic.htm'>z</a>`
- `<a href='accommodation.htm'>z</a>`
- `<a href='study_in_taiwan.htm'>z</a>`

- Regular Expression?

`/<a href=['"]([.\"]+)[^>]*>/`
Memory Parentheses

You remember that parentheses ("( )") may be used for grouping together parts of a pattern.
They also have a second function: they tell the regular expression engine to remember what was in the substring matched by the pattern in the parentheses.

```
/ <a href="advantage.htm"> </a>/
/ <a href="statistic.htm"> </a>/
/ <a href="accommodation.htm"> </a>/
/ <a href="study_in_taiwan.htm"> </a>/
```

Regular Expression
backreferences and memory variables

- \$line = <a href="advantage.htm">
- \$line =~ /<a href=(['"])(.*)['"]>/

```
< a href= " advantage.htm " >

memory variable ("$1") backreferences ("\1")
```

\1 = ""

\1 = "advantage.htm"
The results of homework8

```perl
#!/C:/Perl/bin/perl.exe
#
# Setup the four string into strings list
#
$strings[0] = '<a href="advantage.htm">';
$strings[1] = '<a href="statistic.htm">';
$strings[2] = '<a href="accommodation.htm">';
$strings[3] = '<a href="study_in_taiwan.htm">';
#
# Regular expression for the statements and print results
#
foreach (@strings) {
    /
        <a href="(["\]) (.*)\1>/;
    print $2."\n";
}
```
Homework 9 - Hash table usage

- Write a program to count how many unique words in the “NYMU.txt” file in the other hand counting the frequency of “Yang-Ming”, “Taiwan” and “was”.

Email: perl@ym.edu.tw
The results of homework9

```perl
#!/C:/Perl/bin/perl.exe
#
# Build the filehandle
#
open IN, "NYMU.txt";

while ($line=<IN>) {
  #
  # Search and replacement
  #
  chomp($line);
  $line = uc($line);
  $line =~ s/\////g;
  $line =~ s/\.///g;
  $line =~ s/\s+/\s+/g;
  $line =~ s/\s+/\s+/g;

  # Split the sentence into words
  #
  my @words = split(/\s+/, $line);

  # Build up the hash table for each word
  #
  foreach my $word (@words) {
    if (exists $hash($word)) {
      $hash($word) += 1;
    } else {
      $hash($word) = 1;
    }
  }

  #
  # close the filehandle
  #
  close IN;

  # Count the total unique words using in the documents

  #
  foreach my $word (keys %hash) {
    $all += 1;
  }

  #
  # Print out the results
  #
  print "Total Unique Words using in the document: ".$all."\n";
  print "Yang-Ming: ".$hash("YANG-MING")."\n";
  print "Taiwan: ".$hash("TAIWAN")."\n";
  print "was: ".$hash("WAS")."\n";
```

Copyright 2004 © NYMU Bioinformatics
Homework 10 - Filehandle usage

Write a program to convert the “TW.txt” file 70 nucleotides per-line into a output file “TW1.txt” as 60 nucleotides per-line

Email: perl@ym.edu.tw
The results of homework10

```perl
#!/C:/Perl\bin\perl.exe

# Open input filehandle
open (IN, "TW.txt");

# Using the filehandle and join the sequence
while ($line = <IN>) {
    chomp($line);
    $seq .= $line;
}

# Split the string into array
@all = split(/\/, $seq);

# Close the filehandle
close IN;

# Open the output filehandle
open (OUT, "> TW1.txt");

# Using the filehandle and glue the sequence
for ($X = 0; $X < $#all+1; $X += 60) {
    $subseq = join("
", @all[$X..$X+59]);
    print OUT $subseq."\n";
}

# close the filehandle
close OUT;
```
Arguments

- Perl pass an argument list to the subroutine, simply place the list expression, in parentheses, after the subroutine invocation

- \$n = &max(10, 15)

- That list is passed to the subroutine, it’s made available for the subroutine to use however it needs to.

- The parameter list is automatically assigned to a special array variable name `@` for the duration of the subroutine
Write your program for sub-routine

Use the default arguments

```perl
#!/C:/Perl/bin/perl.exe

sub max {
    if ($_[0] > $_[1]) {
        return $_[0];
    } else {
        return $_[1];
    }
}

$n = &max(10, 15);

print "The larger number is $n\n";
```

Use the private variables in subroutines

```perl
#!/C:/Perl/bin/perl.exe

sub max {
    my ($a, $b) = @_;  
    if ($a > $b) {
        return $a;
    } else {
        return $b;
    }
}

$n = &max(10, 15);

print "The larger number is $n\n";
```
How to use local in Perl?

- In Perl, variables are global by default. Global variables can be seen and modified anywhere in the code, including subroutines, or any other file that may access the Perl script. To make a variable private to a Perl code block, use `my` or `local`.

- Because variables declared as `local` are still global variables, there is another time where they are handy. Remember that `my` variables are private, and cannot be seen by subroutines.
The different between *local* and *my*

```perl
#!/C:\Perl\bin\perl.exe

$office = "global";

&say();
&fred();
&barney();

sub say { print "$office\n"; }

sub fred { local($office) = "fred"; &say() }

sub barney { my($office) = "barney"; &say(); }
```

Copyright 2004 © NYMU Bioinformatics
Variable-length parameter lists

```perl
#!/C:\Perl\bin\perl.exe

$maximum = @max(3, 5, 10, 4, 6);

print "The maximum number is: ". $maximum. "\n"

sub max {
  my $max_so_far = shift @_;    # Move the first element of @_ to $max_so_far
  foreach (@_) {
    if ($_ > $max_so_far) {
      $max_so_far = $_;
    }
  }
  return $max_so_far;
}
```

<table>
<thead>
<tr>
<th>@</th>
<th>_$</th>
<th>$max_so_far</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 5, 10, 4, 6)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(5, 10, 4, 6)</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>(5, 10, 4, 6)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>(5, 10, 4, 6)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>(5, 10, 4, 6)</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>(5, 10, 4, 6)</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

Copyright 2004 © NYMU Bioinformatics
Perl tends to be a pretty permissive language. But maybe you want Perl to impose a little discipline; that can be arranged with the `use strict` pragma.

A pragma is a hint to a compiler, telling it something about the code. The `use strict` pragma Perl’s internal compiler that it should enforce some good programming rules for the rest of this block or source file.
Example

```
use strict;

$bammbammbam = 3;
$bammbammbam += 1;
```

Copyright © NYMU Bioinformatics 2004
Homework 11

Write a program to convert the “TW.txt” file 70 nucleotides per-line into a output file “TW1.txt” as GenBank format

- Reuse the Homework10 program to transfer ”TW.txt” from 60 nt into 70 nt per line
- Please use the subroutine concept
- Dependent on the user defined the format - “fasta” or “genbank”

Email: perl@ym.edu.tw
The results of homework 11

“TW.txt”
70 nucleotides

“TW1.txt”
Genbank format

Copyright 2004 © NYMU Bioinformatics
The concept of reference

Symbol reference

Hard reference

Variable

Value

Reference
The symbol table

- Perl maintains mapping between variable names and values in a “symbol table”
- By default, there is one, global symbol table called “main” Generally, any variable is accessible anywhere.
Example

```
Default die handler restored.
Loading DB routines from perl5db.pl version 1.07
Editor support available.

Enter h or 'h h' for help, or 'perldoc perldebug' for more help.

main::<C:\DOCUME~1\yfliu\LOCALS~1\Temp\dir36D.tmp\Homework9.pl:5>: 5:
   open IN, "NYMU.txt";
   DB<1> r
Total Unique Words using in the document: 183
   Yang-Ming: 9
   Taiwan: 1
   was: 4
Debugged program terminated. Use q to quit or R to restart,
   use 0 inhibit_exit to avoid stopping after program termination,
   h q, h R or h 0 to get additional info.
   DB<1> x %hash

新注 半:
```
The data type

- scalar and its reference

```perl
$foo = "Hello";  # string
$foo = 12;       # number
$scalarRef = $foo;  # reference to another scalar variable
```

- array and its reference

```perl
@cities = ("Chicago", "New York", "Los Angles");  # array
$arrayRef = @array;  # reference to array
```

- hash and its reference

```perl
%person = ( "derek2" => "29",  
             "chris61vw" => "22",  
             "mary" => "22" );  # hash
$hashRef = %person;  # reference to hash
```
References

- Hard reference (new)
  - $foo = "bot"
  - $bar = \$foo
  - $$bar = "bot"

- Symbol reference (old)
  - $foo = "bot"
  - $bar = "foo"
  - $$bar = "bot"
Comparison between the HARD and SYMBOL references

```perl
#!/C:/Perl/Bin/perl.exe
#
# Comparison of References
#
$foo = "bot";
$bar = "$foo;

print "Hard reference\n";
print "$foo= '.$foo.'\n";
print "$bar= '.$bar.'\n";
print "$.bar='.$.bar.'\n";

$foo = "bot";
$bar = "foo";

print "Symbol reference\n";
print "$foo= '.$foo.'\n";
print "$bar= '.$bar.'\n";
print "$.bar='.$.bar.'\n";
```

---

Copyright 2004 © NYMU Bioinformatics
Create References

- The backslash operator
  - $scalarref = \$foo
  - $constref = \1.414
  - $arrayref = \@ARGV
  - $hashref = \%ENV
  - $coderef = \&handle

- Anonymous reference
  - $array_ref = [1, 2, 3]
  - $hash_ref = {}
### Comparison of backslash and anonymous references

#### list or array

<table>
<thead>
<tr>
<th>Backslash reference</th>
<th>Anonymous reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>@list = (1..10, 12, 20);</code></td>
<td><code>$list_ref = @list;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Backslash reference</th>
<th>Anonymous reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$list_ref = \@list;</code></td>
<td><code>$list_ref = [1..10, 12, 20];</code></td>
</tr>
</tbody>
</table>

#### hash

<table>
<thead>
<tr>
<th>Backslash reference</th>
<th>Anonymous reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%hash = ( one   =&gt; 1, two   =&gt; 2, three =&gt; 3 );</code></td>
<td><code>$hash_ref = %hash;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Backslash reference</th>
<th>Anonymous reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$hash_ref = \%hash;</code></td>
<td><code>$hash_ref = { one   =&gt; 1, two   =&gt; 2, three =&gt; 3 };</code></td>
</tr>
</tbody>
</table>
Using Hard reference (I)

- Using a **VARIABLE** as a variable name

- $foo = "three humps";
- $scalarref = \$foo;
- print $$scalarref;
- @$ref = qw/May June July/;
- push(@$arrayref, $filename);
- %$hashref = (KEY => "RING", BIRD => "SING");
- $$hashref {KEY} = "VALUE";
Using Hard reference (II)

- Using a **BLOCK** as a variable name

  ```perl
  $foo = "three humps";
  $scalarref = \$foo;
  print \${$scalarref};
  @{$ref} = qw/May June July/;
  push(@{$arrayref}, $filename);
  %{$hashref} = (KEY => "RING", BIRD => "SING");
  \${$hashref}{KEY} = "VALUE";
  ```
Using Hard reference (III)

- Using a **ARROW** as operator

  - $foo = "three humps";
    $scalarref = \$foo;
    print $scalarref->[0];
  
  - @{$ref} = qw/May June July/;
    $ref->[0] = "May";
  
  - $anonymous_ref = {};
    push($anonymous_ref, $filename);
    $anonymous_ref->[0];
  
  - %{$hashref} = ( KEY => "RING", BIRD => "SING" );
    $hashref->{KEY} = "RING";
  
  - $hashref = {};
    $hashref->{KEY} = "VALUE";
Reference to Arrays and Hashes

```perl
#!C:/Perl/bin/perl.exe
#
# References to Array and Hashes
#
@derek_friends = qw( John Sandy Laura );  # A list of derek's friends
@sandy_friends = qw( John Mark Ralph );  # A list of sandy's friends
@bob_friends = qw( Freida Jim Jenny );  # A list of bob's friends

$arrayRef = \@friend_derek;  # Note the \ operator in front of the @, tells
# perl you want a REFERENCE to the array

$first_friend = $arrayRef->[0];  # John

$friend{derek} = "john";  # Make a key called "derek" with value "john" in
# a hash called %friend

$hashRef = \%friend;
$dereks_friend = $hashRef->{derek};
```
#!/C:/Perl/bin/perl.exe
#
# Anonymous Array and Hash References
#

$hash = ('key1' => 'value', 'key2' => 'value');
$a_reference = \%hash;

# anonymous array
#
$derekArrayRef = [ 'John', 'Sandy', 'Laura' ];
$sandyArrayRef = [ 'John', 'Mark', 'Ralph' ];
$bobArrayRef = [ 'Freida', 'Jim', 'Jenny' ];

# Print derek's first friend
print "$derekArrayRef->[0]\n";  # prints John

# anonymous hash
#
$friendsHashRef = { 'derek' => 'John', 'Sandy' => 'John', 'Bob' => 'Freida' };

# Print Sandy's friend
print "$friendsHashRef->{Sandy}\n";
Data structures

- Briefly, references are rather like pointers that know what they point to.
- This means that when you have something which looks to you like an access to a two-or-more-dimensional array and/or hash, what's really going on is that the base type is merely a one-dimensional entity that contains references to the next level.

```plaintext
$array[7][12]  # array of arrays
$hash{string}[7]  # hash of arrays
$array[7]{string}  # array of hashes
$hash{string}{'another string'}  # hash of hashes
```
Data structure (I) - Arrays of Arrays

$AOA_ref = [
    [
        "fred", "narney", "pebbles", "bamm bamm", "dinp",
    ],
    [
        "homer", "bart", "marge", "maggie",
    ],
    [
        "george", "jane", "el roy", "judy",
    ],
];

print $AOA_ref->[2]->[3];  (arrow operator)
print $AOA_ref->[1][3];   (bracket operator)
Example 1 - Array of Arrays

Please create a 9x9 table using the Array of Array method.

```perl
$array_ref = []; for my $x (1..9) { for my $y (1..9) { $array_ref->[$x][$y] = $x * $y; } } ....
```
Debugging (I)

```perl
#!/C:\Perl\bin\perl.exe
#
# Fetch the Array of Array
#
$AOA_ref = [
    [
        "fred", "narney", "pebbles", "bamm bamm", "dinp", ,
        "homer", "bart", "marge", "maggie", ,
        "george", "jane", "elroy", "judy",,
    ];

for my $x ( 0 .. $#{$AOA_ref} ) {
    for my $y ( 0 .. $#{$AOA_ref->[x]} ) {
        print "[".$x."][".$y."] => ", $AOA_ref->[x][$y],"\n";
    }
}
```

Run in command mode

```
Run in Command Prompt
Switches:
  -d
Arguments:
```

Copyright 2004 © NYMU Bioinformatics
Debugging (II)

```
C:\WINDOWS\System32\cmd.exe

Default die handler restored.

Loading DB routines from perl5db.pl version 1.07
Editor support available.

Enter h or `h h' for help, or `perldoc perldebug' for more help.

main:<C:\DOCUME~1\c00liu00\LOCALS~1\Temp\dir99.tmp\AOA.pl:6>:
  6:   $AOA_ref = [
  7:     ["fred", "narney", "pebbles", "bamm bamm", "dinp", ],
  8:     ["homer", "bart", "marge", "maggie", ],
  9:     ["george", "jane", "elroy", "judy",],
 10:   ];
  DB<1>
```
Debugging (III)
Debugging (IV)
How to fetch data from array of arrays?

```perl
#!/C:Perl\bin\perl.exe
#
# Fetch the Array of Array
#

$AOA_ref = [
    [
        "fred", "narney", "pebbles", "bamm bamm", "dinp", ],
    [
        "homer", "bart", "marge", "maggie", ],
    [
        "george", "jane", "elroy", "judy",],
];

for my $x ( 0 .. $#{$AOA_ref} ) {
    for my $y ( 0 .. $#{$AOA_ref->[$x]} ) {
        print "[" . $x . "][" . $y . "] => \"" . $AOA_ref->[$x][$y] . \\
    \n";
    }
}
```

http://www.perldoc.com/perl5.8.0/pod/perllol.html

Copyright 2004 © NYMU Bioinformatics
Data structure (II) - Hashes of Arrays

```perl
$HOA_ref = {
    flintstones => ["fred", "barney"],,
    jetsons    => ["homer", "bart", "marge", "maggie"],,
    simpsons   => ["george", "jane", "el roy", "judy"],,
};

print $HOA_ref->{"simpsons"}[3];  (arrow operator)
print $HOA_ref->{"jetsons"}[2];   (bracket operator)
```
Example II - Hash of Arrays

```perl
#!/C:/Perl/bin/perl.exe
#
# Example : Hash of Array
#
use warnings;
use strict;

my %friendsHash = ('derek' => [ qw/John Sandy Laura/ ],
                    'sandy' => [ qw/John Mark Ralph/ ],
                    'bob'   => [ qw/Freida Jim Jenny/ ]);

for $person (keys %friendsHash) {
    print "$person Friends: ";

    # Print out the array of friends for this key.
    # We have a real hash this time, but still have to dereference
    # the array
    for my $friend (@{$friendsHash{$person}}) {
        print "$friend ";
    }
    print "\n";
}
```
Question 2

Please printout the Hashes of Arrays.

```perl
$HOA_ref = {
    flintstones => ["fred","barney",],
    jetsons => ["homer","bart","marge","maggie",],
    simpsons => ["george","jane","el roy","judy",],
};

foreach my $key (keys %{$HOA_ref}) {
    foreach my $item (@{$HOA_ref->{$key}}) {
        print $key."\t".$item."\n";
    }
}
```
Data structure (III) - Array of Hashes

push @{$AOH_ref}, { husband    => "fred",
                   wife       => "wilma",
                   daughter   => "pebbles",};
push @{$AOH_ref}, { husband    => "homer",
                   wife       => "marge",
                   daughter   => "bart",};

print $AOH_ref->[0]->"wife";
print $AOH_ref->[1]->"daughter";
Question 3 - Array of Hashes

Please parse following data into Array of Hashes

AAA=Lys
AAG=Lys
AAC=Asn
AAU=Asn

$ref = {}; 
while ( <> ) { 
    my @words = split (/=/, $line) 
    push @{$ref->{$words[1]}}, $words[0]; 
} 
....
How to fetch data from array of hashes?

```perl
#!/C:\Perl\bin\perl.exe

# Fetch the Array of Hash
#

push @{$AOH_ref}, { husband => "fred",
                   wife   => "wilma",
                   daughter => "pebbles"};
push @{$AOH_ref}, { husband => "homer",
                   wife   => "marge",
                   daughter => "bart"};

for my $x ( 0 .. $#{$AOH_ref} ) {
    foreach my $key ( keys @{$AOH_ref}[{$x}] ) {
        print "[".$key."] => ".".$AOH_ref[{$x}]->{$key}.
    }
}
```
Please build up a complexity hashes of hashes data structure for store the translation table information

AAA=Lys=K
AAG=Lys=K
AAC=Asn=N
AAU=Asn=N

$hash_ref = {}; while ( <> ) {
    my @words = split(/=/, $line);
    $hash_ref->{$words[0]}->{three} = $words[1];
    $hash_ref->{$words[0]}->{one}   = $words[2];
}
Example of Hash of Hashes

```perl
#!/usr/bin/perl

# Example : Hash of Hash
#
# Nice indentation is everything...

%person = ( 'derek' => { 'age' => 29, 'pet' => 'cat' }, 'john' => { 'age' => 41, 'pet' => 'dog' } );

my %person;
$person = 'derek';
$age = '29';
$pet = 'cat';

if (!exists %person{$person}) { # Need to create this key
    %person{$person} = { 'age' => $age, 'pet' => $pet }
} else {
    # key already exists
    %person{$person}{age} = $age;
    %person{$person}{pet} = $pet;
}
```
Question IV - Hash of Hashes

Please build up a complexity hashes of hashes data structure for store the translation table information

AAA=Lys=K
AAG=Lys=K
AAC=Asn=N
AAU=Asn=N

$hash_ref = {}; 
while (<>) {
    my @words = split(/=/, $line);
    $hash_ref @{$words[0]}->{three} = $words[1];
    $hash_ref @{$words[0]}->{one} = $words[2];
}
....
How to fetch data from hash of hashes?

```perl
#!/C:/Perl/bin/perl.exe
#
# Fetch the Hash of Hashes
#
$lines[0] = "AAA=Lys=K";
$lines[1] = "AAG=Lys=K";
$lines[2] = "AAC=Asn=N";
$lines[3] = "AAU=Asn=N";

# Build the Hash of Hashes data structure
#
foreach my $line (@lines) {
    my @words = split( /=/,$line);
    $HOH_ref->{$words[0]}->{three} = $words[1];
    $HOH_ref->{$words[0]}->{one} = $words[2];
}

# Fetch the data
#
# Anonymous reference

foreach my $key1 ( keys %($HOH_ref) ) {
    foreach my $key2 ( keys %($HOH_ref->{$key1}) ) {
        print "$key1.".$key2."\t => \".$HOH_ref->{$key1}->{$key2}."\n";
    }
}
```

Copyright 2004 © NYMU Bioinformatics
Debugging

Enter h or ‘h h’ for help, or ‘perldoc perldebug’ for more help.

main::(C:\DOCUME~1\yfliu\LOCALS~1\Temp\dir2CE.tmp\HOH.pl:6):
6:  $lines[0] = "AAA=Lys=K";
   DB<1> r

Debugged program terminated. Use q to quit or R to restart,
use 0 inhibit_exit to avoid stopping after program termination,
h q, h R or h 0 to get additional info.
   DB<1> x $HOH_ref
   HASH<0x1b2678c>
     'AAA' => HASH<0x1af587c>
       'one'  => 'K'
       'three'  => 'Lys'
     'AAC' => HASH<0x1afb3d4>
       'one'  => 'N'
       'three'  => 'Asn'
     'AAG' => HASH<0x1ae8e88>
       'one'  => 'K'
       'three'  => 'Lys'
     'AAU' => HASH<0x1af2b0c>
       'one'  => 'N'
       'three'  => 'Asn'

   DB<2> =
Homework 12

- Please write a program using Hashes table for translation process
- The input file is FASTA format
  - The output format must support six reading frames translation
  - The output format must support 3-letter and 1-letter for amino acids presentation
  - You can get the translation table from

http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/C/Codons.html

Email: perl@ym.edu.tw
Hints

- Build the translation table
- Get the sequence from FASTA format
- Sequence reverse and complementary
- Translation
- Printout the results
### Build up the translation table

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F Phe</td>
<td>S Ser</td>
</tr>
<tr>
<td></td>
<td>F Phe</td>
<td>S Ser</td>
</tr>
<tr>
<td></td>
<td>L Leu</td>
<td>S Ser</td>
</tr>
<tr>
<td></td>
<td>L Leu</td>
<td>S Ser</td>
</tr>
<tr>
<td>C</td>
<td>L Leu</td>
<td>P Pro</td>
</tr>
<tr>
<td></td>
<td>L Leu</td>
<td>P Pro</td>
</tr>
<tr>
<td></td>
<td>L Leu</td>
<td>P Pro</td>
</tr>
<tr>
<td></td>
<td>L Leu</td>
<td>P Pro</td>
</tr>
<tr>
<td>A</td>
<td>Ile</td>
<td>T Thr</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>T Thr</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>T Thr</td>
</tr>
<tr>
<td></td>
<td>M Met</td>
<td>T Thr</td>
</tr>
<tr>
<td>G</td>
<td>Val</td>
<td>A Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>A Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>A Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>A Ala</td>
</tr>
</tbody>
</table>

http://www2.ebi.ac.uk/cgi-bin/mutations/trtables.cgi

Copyright 2004 © NYMU Bioinformatics
Build the hash of hash data structure

$\text{table_ref}\rightarrow\{\text{codon}\} \rightarrow \{\text{three}\} : \text{aa\_three}$
- $\rightarrow\{\text{one}\} : \text{aa\_one}$

**Example**

$\text{table_ref}\rightarrow\{\text{"TTT"}\} \rightarrow \{\text{three}\} = \text{"Phe"}$
- $\rightarrow\{\text{"TTT"}\} \rightarrow \{\text{one}\} = \text{"F"}$

$\text{table_ref}\rightarrow\{\text{"TTA"}\} \rightarrow \{\text{three}\} = \text{"Leu"}$
- $\rightarrow\{\text{"TTA"}\} \rightarrow \{\text{one}\} = \text{"L"}$
Use the NCBI Entrez database

Display the sequence in FASTA format in NCBI Entrez database

<table>
<thead>
<tr>
<th>Entrez</th>
<th>PubMed</th>
<th>Nucleotide</th>
<th>Protein</th>
<th>Genome</th>
<th>Structure</th>
<th>PMC</th>
<th>Taxonomy</th>
<th>Books</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Search Nucleotide for sca7

Limit | Preview/Index | History | Clipboard | Details |
|------|---------------|---------|------------|---------|

Display FASTA

Send to Text

Items 1-20 of 44

1: AC012557
   Homo sapiens 3 BAC RP11-24519 (Roswell Park Cancer Institute Human BAC Library) complete sequence
gi:12000461gb|AC012557.23[35409720]

2: NM_000333
   Homo sapiens spinocerebellar ataxia 7 (olivopontocerebellar atrophy with retinal degeneration) (SCA7), mRNA
gi:4506796|ref|NM_000333.1[35263003]

3: NT_022517
   Homo sapiens chromosome 3 genomic contig
gi:3750163|ref|NT_022517.16|Hs3_22673[35094984]

4: NM_139227
   Mus musculus spinocerebellar ataxia 7 homolog (human) (Sca7), mRNA
gi:31559971|ref|NM_139227.2[34101941]
Read the sequence file in FASTA format

```fasta
>gi|4506796|ref|NM_000333.1| Homo sapiens spinocerebellar ataxia 7 (olivopontocerebellar atrophy with retinal degeneration) (SCA7), mRNA
CGTTGCTGTCGAAAGGGTGAAAGAGAAACTTGGCGACCTCCGGAGGAGTTCGCGAAGCGACCAGGAGCGTGTTGCCATTGTCCTCACCCGGCACCCAATTCCACCACAGAGTCGGGATTTCGTCGGTGATCGTGATGGGGTGCTTTTATTTTTCTCTTTTGATTTTCAAAAAATGTCTATGTGACTGTCCCTATCTTAAGGGGAAGTTGAAAGTGGGGGCGGGGGTGCTCAATGAGAAACGTTGCCTTGTGTGTAGTTGTTTGGAGCACACTGCAAATTATATTGGCATCTCTTTCCAAAAGTCACTTTGATTCAACTCGGATAGCTTTCTCGTAAATGGCACGTTTAGGTGGTGAGAGGTGGATGAGGAAACAGGCACCAGTGCAGCTGATTTGACC
```

Copyright 2004 © NYMU Bioinformatics
Other parts of translation program

```perl
#!C:/Perl/bin/perl.exe
#
# DNA Translation Program
#
#
# == Methods ==
#
# build_table(filename) : $table_ref
# read_seq(seqname) : $seq_ref
# reverse_complementary(seq_ref) : $seq_ref
# translation(seq_ref) : $pro_ref
# printout(pro_ref)

$table_ref = &build_table("Table.txt");
$seq_ref = &read_seq("sca7.fasta");
$seq_ref = &reverse_complementary($seq_ref);
$pro_ref = &translation($seq_ref);
&printout($pro_ref);
```