Refresh

EMBOSS package

Unix-Windows operating system (login, ftp),

Basic commands

Programming

List of instructions or program statements composed in such a way as to enable a computer to solve a problem

- **State the problem**
- **Examine the problem and break down into several parts.**
- **Examine the parts and refine into smaller parts.**
- **Sketch a picture/structure plan**
- **Write the main program with references to the subprograms.**
- **Test the program.**

Programming in Perl: Basics

Practical Extraction and Reporting Language

#! /usr/bin/perl (shebang)
A special command required to tell the operating system what is going to run the script (Perl) and it always starts with #!.

An interpreter translates some form of source code into a target representation that it can immediately execute and evaluate.

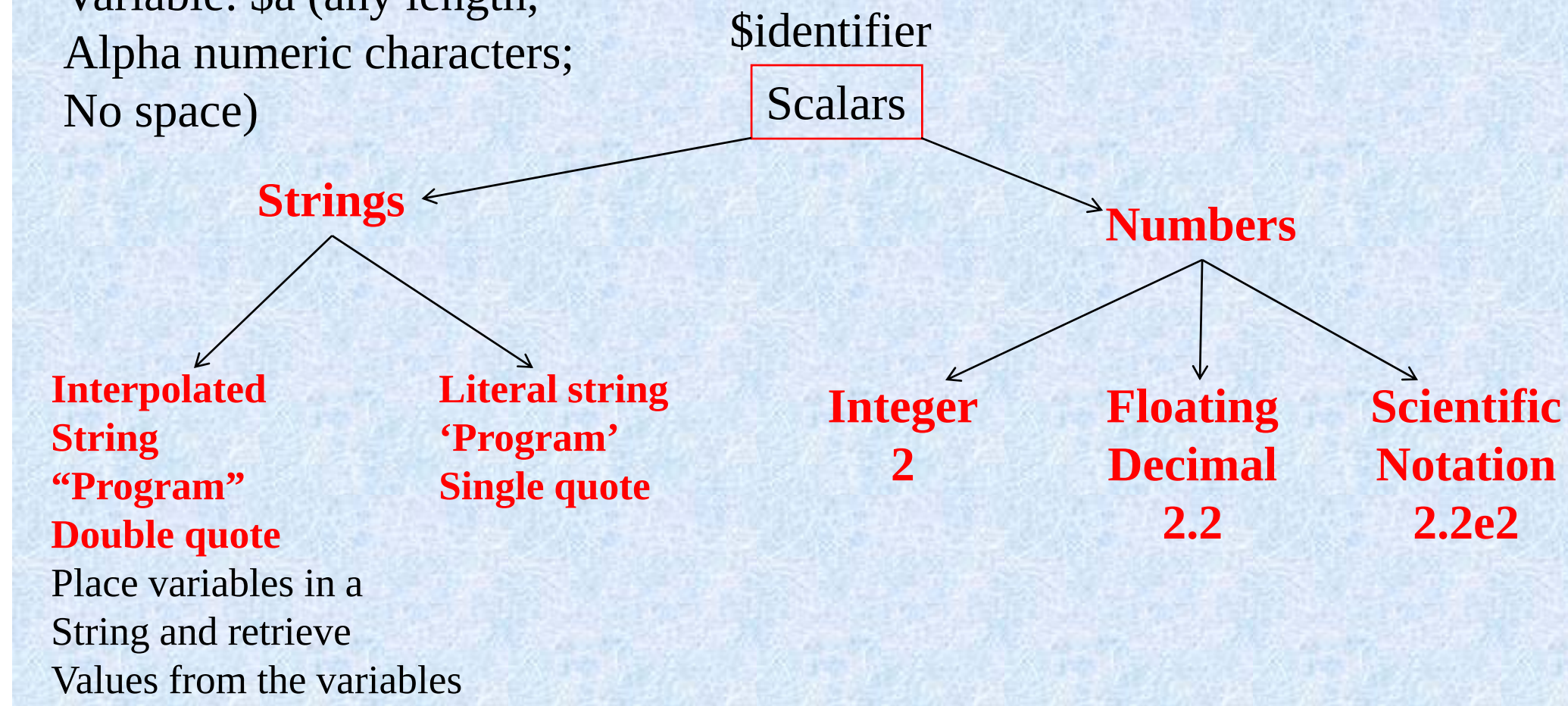
PERL statement ends with semicolon ;

#: command line

Character	Function
\n	new line
\t	Tab
\u or \U	force to upper case
\l or \L	force to lower case
\E	end \U or \L

Scalars and strings

Variable: $a (any length,
Alpha numeric characters;
No space)



Place variables in a
String and retrieve
Values from the variables

PERL basics

```
#! /usr/bin/perl
$string = “Good morning!\n”;
Print $string;
```

Output: Good morning!

```
#! /usr/bin/perl
$string = “Good morning!\n”;
Print “The greeting is $string”;
```

Output: The greeting is Good morning!

```
#! /usr/bin/perl
$string = ‘Good morning!\n’;
Print ‘The greeting is $string’;
```

Output: The greeting is $string

Multiplying string output

```
#! /usr/bin/perl
$string = “Good morning!\n” x 3;
Print $string;
```

Output:

```
Good morning!
Good morning!
Good morning!
```

Calculations

Mathematical operations are performed in the standard order of precedence.

Eg. Multiplication has higher precedence than addition.

2+3*4 = 14 and not 20.

(2+3)*4

PERL operators

++	Auto increment
--	Auto decrement
**	Exponentiation
*	Multiply
/	Divide
+	Add
-	Subtract
cos()	Cosine
sqrt()	square root
=	Assign
+=	assign add
-=	assign subtract

Simple programming: Perl/FORTRAN

Print a sentence

$vi strand.pl

```
#! /usr/bin/perl
# A simple perl program
print “Complementary strand\n”;
```

Esc :wq
Command:
$perl strand.pl
Ans:
Complementary strand

$vi strand.f

```
print *, "Complementary strand"
 stop
 end
```

Esc :wq
Command:
$f77 strand.f –o strand
./strand
Ans:
Complementary strand

Simple programming: Perl

Add two numbers

$vi add.pl

```
#! /usr/bin/perl

# Add two numbers

$a = 2;

$b = 3;

$c = $a + $b;

print "$a, $b, $c\n";
```

Command:

$perl add.pl

Ans:

2, 3, 5

$vi add.f

```
 a=2
b=3
c=a+b
write(*,*) a, b, c
stop
end
```

Esc :wq

Command:

$f77 add.f –o add

./add

Ans:

2. 3. 5.

Compute the slope

(x1,y1) and (x2,y2)

(3,4) and (15,8)

m=(y2-y1)/(x2-x1)

```
$x1=3;
$x2=4;
$y1=15;
$y2=8;
$m=($y2-$y1)/($x2-$x1);
Print “The slope is $m\n”;
```

The slope is 0.333333333333

Operators for strings

eq	equal
ne	not equal
lt	less than
gt	greater than
ge	greater than or equal to
le	less than or equal to
++	causes the last letter of the scalar to change so that it increases one place in the alphabet
Comma (,)	Append strings to each other

Operators lt and gt

```
#! /usr/bin/perl
$city1=“Chennai”;
$city2=“Delhi”;
If($city2 gt $city1)
{
    print “$city2 comes later in an alphabetical list than $city1\n”
}
If($city1 gt $city2)
{
    print “$city1 comes later in an alphabetical list than $city2\n”
}
```

Delhi comes later in an alphabetical list than Chennai

PERL

Arrays (@)

Loops

- While
- for
- until
- foreach

Loops

Compute the sum and sum of the squares of the first 100 natural numbers

```
$sum=0;
$sumsq=0;
foreach $i (1 .. 100) {
$sum += $i;
$sumsq += $i*$i;
}
Print “$sum, $sumsq\n”;
```

5050, 338350

String substitutions in Perl

Substitution operator, s

Usage: $string =~ s/string_to_replace/replacement_string/modifiers

Eg. $dna = "ACTGACC"; # assign DNA sequence to a string

$dna =~ s/A/T/ig; # swap all A's to T's; i: case insensitive; g: global

Program

```
#! /usr/bin/perl
$dna = "ACTGACC";        # assign DNA sequence to a string
$dna =~ s/A/T/ig;        # swap all A's to T's; i: case insensitive; g: global
print  "$dna\n";
```

Answer

TCTGTCC

Find the complementary strand by replacements of A to T, T to A, C to G, G to C

CCACACA

Complimentary strand using Perl

New function to change all letters together. "tr"

```
#! /usr/bin/perl
    $dna = "ACTGACC";
    $dna =~ tr/ACTGactg/TGACtgac/;
    $dna = reverse($dna);
    print  "$dna\n";
```

GGTCAGT

Convert DNA sequence into protein sequence

```
# This script will convert your DNA sequence to PROTEIN Sequence

print "ENTER THE FILENAME OF THE DNA SEQUENCE:= ";

$DNAfilename = <STDIN>;

chomp $DNAfilename;

unless ( open(DNAFILE, $DNAfilename) ) {

    print "Cannot open file \"$DNAfilename\"\n\n";

}

@DNA = <DNAFILE>;

close DNAFILE;

$DNA = join( '', @DNA);

print " \nThe original DNA file is:\n$DNA \n";

$DNA =~ s/\s//g;

my $protein='';

my $codon;

for(my $i=0;$i<(length($DNA)-2);$i+=3)
{
$codon=substr($DNA,$i,3);
$protein.=&codon2aa($codon);
}
print "The translated protein is :\n$protein\n";
<STDIN>;

sub codon2aa{
my($codon)=@_;
$codon=uc $codon;
my(%g)=('TCA'=>'S','TCC'=>'S','TCG'=>'S','TCT'=>'S','TTC'=>'F','TTT'=>'F
','TTA'=>'L','TTG'=>'L','TAC'=>'Y','TAT'=>'Y','TAA'=>'_','TAG'=>'_','TGC'=
>'C','TGT'=>'C','TGA'=>'_','TGG'=>'W','CTA'=>'L','CTC'=>'L','CTG'=>'L','C
TT'=>'L','CCA'=>'P','CCC'=>'P','CCG'=>'P','CCT'=>'P','CAC'=>'H','CAT'=>'
H','CAA'=>'Q','CAG'=>'Q','CGA'=>'R','CGC'=>'R','CGG'=>'R','CGT'=>'R','A
TA'=>'I','ATC'=>'I','ATT'=>'I','ATG'=>'M','ACA'=>'T','ACC'=>'T','ACG'=>'T'
,'ACT'=>'T','AAC'=>'N','AAT'=>'N','AAA'=>'K','AAG'=>'K','AGC'=>'S','AG
T'=>'S','AGA'=>'R','AGG'=>'R','GTA'=>'V','GTC'=>'V','GTG'=>'V','GTT'=>'
V','GCA'=>'A','GCC'=>'A','GCG'=>'A','GCT'=>'A','GAC'=>'D','GAT'=>'D','G
AA'=>'E','GAG'=>'E','GGA'=>'G','GGC'=>'G','GGG'=>'G','GGT'=>'G');
if(exists $g{$codon})
{
return $g{$codon};
}
else
{
print STDERR "Bad codon \"$codon\"!!\n";
exit;
}
}
```

Running the program

$perl dna2protein.pl

ENTER THE FILENAME OF THE DNA SEQUENCE:= dnafile

The original DNA file is:

ACGTGCGCATGCAACCGAATGA

The translated protein is :

TCACNRM

Books for PERL programming

Beginning Perl For Bioinformatics, James Tisdall

Mastering Perl For Bioinformatics, James Tisdall

PERL PROGRAMMING FOR BIOINFORMATICS & BIOLOGISTS, D. Curtis Jamison