Programação para Bioinformática em Perl

Daniel Moura

UNIVERSITEIT GENT

Perícia Universitária

VIRTUS IMPAVIDA

UFPE

Revisão

- Dados -> Inteiros, float, string, boollean
- Operadores aritméticos -> +, -, *, /, %, **, ++, --
- Operadores relacionais -> >, >=, <, <=, !=, ==
- String -> . , uc, lc, substr, length, index, chomp
- Operadores relacionais-> eq, ne, gt, lt, ge, le, cmp
- Operadores lógicos -> &&, ||, !
- Diamante -> <STDIN>

- Condicional -> if, elsif, else

- Loop -> While, do...while, for, foreach*, last, next

- @array -> pop, push, shift, unshift, sort, index, lenght
- %hashe -> inserir, delete, scalar, Keys, values

- Slices arrays:
  (my $primeiro, my $segundo) = @array;
  (my $primeiro, my @outros) = @array;
  @array1 = @array2[1,2,3,4];
- Função map:
  map($_*3, @array);
- Função grep:
  grep($_>100, @array);
- @ARGV
  array automático
- Função split:
  $seq = “ATGC”;
  @array = spit //, $seq;
  @array: A T G C

E hoje?

- Subrotinas
- Módulo
- Expressões regulares

Subrotina

- sub nome {
  #comado1;
  #comado2;
  etc...
}

- Nome();

Subrotina

```
sub hello{
        print “Hello Word!\n”;
}
hello();    #Hello World!

sub latir{
        print “Au au!\n”;
}
latir();          #Au au!
```

Let’s code!!!

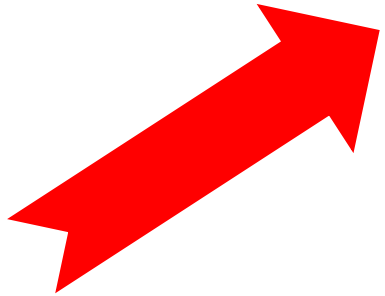
```
#! /usr/local/bin/perl –w
use strict;


print “Digite um numero: \n”;
my $numero = <STDIN>;
chomp($numero);


sub par_ou_impar{
        if( $numero%2 == 0){ print “O numero $numero e par!\n”;}
        else { print “O numero $numero e impar!\n”;}
}


par_ou_impar();
```

Let’s code!!!

```
#! /usr/local/bin/perl –w
use strict;


print “Digite um numero: \n”;
my $numero = <STDIN>;
chomp($numero);
my $resposta;


sub par_ou_impar{
    if( $numero%2 == 0){ return  $resposta = "par";}
    else{ return $resposta = "impar";}
}



par_ou_impar($numero);
print "o numero $numero e $resposta\n";
```

Variável global

```
sub latir{
        my $latido = “Au au!\n”;
        print $latido;
}

my $latido = “Miau!\n”;

latir();          #Au au!
print $latido          #Miau!
```



Quando quiser usar uma variável que seja aplicada em todo o programa use a declaração *our $variável;*

Módulo

```
package Calculadora;

sub soma {
  my $x = $_[0];
  my $y = $_[1];
  return $x + $y;
}
sub divisao {
  my $x = $_[0];
  my $y = $_[1];
  return $x / $y;
}
sub multiplicacao {
  my $x = $_[0];
  my $y = $_[1];
  return $x * $y;
}
sub subtracao {
  my $x = $_[0];
  my $y = $_[1];
  return $x - $y;
}

1;
```

Módulo

```
use strict;
use Calculadora;


# SOMA
print "2 + 2 = ";
print Calculadora::soma(2,2);

# SUBTRACAO
print "\n10 - 7 = ";
print Calculadora::subtracao(10,7);

# DIVISAO
print "\n10 / 5 = ";
print Calculadora::divisao(10,5);

# MULTIPLICACAO
print "\n7 * 8 = ";
print Calculadora::multiplicacao(7,8);

print "\n";
```

Expressões regulares

- Chamadas de *regex* ou ER
- Básicas ou estendidas
- Busca padrões e executa funções
- =~ ou =!

Exemplo

my $dna = “human_dna.txt”

if ($ frase =~ /ggagggg/){ print "Padrão 'ggagggg’ foi encontrado.\ n"; }

elsif ($ frase !~ /ggagggg/){ print "Padrão 'ggagggg’ não foi encontrado.\ n"; }

Operações com Regex

- Procurar padrão (match): m/padrão/ ou /padrão/
  - $sequence =~ m/ATG/;
- Substituir: s/padrão/padrão2/
  - $sequence =~ s/T/U/;
- Modificar (translate): tr/padrão/padrão/
  - $sequence =~ tr/ACGT/TGCA/;
- g = procura global
- i = procura case-insensitive

Let’s code

```
use strict;

my $file = "Human_DNA.txt"; #no mesmo diretório
open(FILE, $file) or die "não foi possível abrir o arquivo \"$file\"!";

my $seq_nome = <FILE>;
chomp ($seq_nome);

while(my $linha = <FILE>) {
        chomp ($linha);
        if($linha =~ /aaaaaaaaa=/){
                print $linha;
        }

}


close(FILE);
```

Sua vez

Crie um programa usando REGEX que converta sequencia de DNA em cDNA e RNA!

Não esqueça o ; !

Metacaracteres

- Símbolos especiais
- Podem ser usados juntos ou separados
- Representantes, âncora, quantificadores e outros

Metacaracteres representantes

.	Ponto busca qualquer caractere, excerto os \n,\b,\t etc...
[]	A lista pode buscar um conjunto de caracteres.
[^]	A lista negada pode buscar um conjunto de caracteres.

Metacaracteres âncoras

^	Circunflexo busca qualquer caractere no inicio de uma linha ou string.
$	Cifrão busca qualquer caractere no fim de uma linha ou string.

Metacaracteres qualificadores

*	O asterisco busca caracteres que aparecem nenhuma ou mais de uma vez.
?	O opcional busca caracteres que aparecem nenhuma ou uma única vez.
+	O mais busca caracteres que aparece uma ou mais vezes.
{x,y}	A chave busca por caractere que apareça no mínimo x vezes e no máximo y vezes, de acordo com o indicado.

Outros metacaracteres

|	O ou busca pelo caractere à direita ou pelo caractere à esquerda.
()	O agrupamento busca por grupos de caracteres.
\	O escape converte metacaractere em um caractere normal.

Let’s code

João descobriu que um de seus parentes desenvolveu câncer. Preocupado com essa situação, ele decidiu analisar a sua própria sequencia de DNA para descobrir se possui algum gene de algum tipo de tumor maligno.

Sua função é desenvolver um programa que percorra todo o DNA de João para encontrar algum desses genes em sua sequência.

Os genes não -> ATGCATGATGCAT para câncer de intestino,
GATTGCTCACTCATGCAGT para câncer de pâncreas,
CATTAGCTTTATTGCATTA para câncer de pele.