

Programação para Bioinformática em Perl

Daniel Moura



Revisão

- Dados -> Inteiros, float, string, boolean
- 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

- Replicação ->
`print (("bioinformática")x3);`
bioinformaticabioinformaticabioinformática
- Comentários múltiplos ->
=begin
*o meu primeiro programa na bioinformática é
fantástico e irá mudar minha forma de trabalhar
com todos os meus dados.
Irei ver o computador de outra forma a partir de
hoje!*
=cut

E hoje?

- Manipulação de arquivos
input / output
- Arrays
- Hashes
- Sub rotina

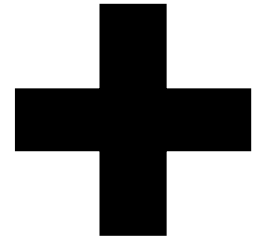
Abrindo arquivos

- `open(arquivo, modo de abertura, nome do arquivo)`

Leitura
<

Escrita
>

Incremental
>>



Porém...!

Deve estar no mesmo diretório

ou

```
open(FILE,  
"/home/user/pasta/subpasta...arquivo.txt");
```

Não esqueça de fechar!

close(FILE);

Em caso de erros use *die*:

open(FILE) or die("falhar ao abrir arquivo);

Let's code!!!

```
#!/usr/local/bin/perl -w
# leitura do arquivo dna humano
use strict;

my $file = "Human_DNA.txt"; #no mesmo diretório
my $linha1;
my $linha2;
open(FILE, $file) or die "não foi possível abrir o arquivo \"$file\"!";
$linha1 = <FILE>;    #ler a linha
print $linha1; #retornar a linha
$linha2 = <FILE>;
print $linha2;

close(FILE);
```

Como verificar se o arquivo abriu?

Opção 1:

```
unless (open(FILE, ">diretório")) {  
    print "erro ao criar arquivo";  
}  
  
else { # o resto do programa}
```

Opção 2:

```
unless(open FILE, ">diretório") {  
    die "erro ao criar arquivo"; }  
#resto do programa
```


Como verificar se o arquivo abriu?

Opção 3:

```
open(FILE, ">diretório") || die "erro ao criar o  
arquivo";
```

Mais comum e mais recomendado!

Lendo um arquivo inteiro

```
my $file2 = "Human_DNA.txt"; #no mesmo diretório  
open(FILE, $file2) or die "não foi possível abrir o arquivo \"$file\"!";
```

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

```
my $sequence = "";  
while(my $linha = <FILE>) {  
    chomp ($linha);  
    $sequence .= $linha;  
}
```

```
close(FILE);  
print "$seq_nome; tamanho: ", length($sequence), "bp\n";  
print "$sequence\n";
```

Arrays

```
my @nucleotideos = ("A","T","G","C");  
print @nucleotideos;    #ATGC
```

```
my @nomes_nucleotideos =  
("Adenina","Timina","Guanina","Citosina",@nucleotideos)  
print @nucleotideos;  
#AdeninaTiminaGuaninaCitosinaATGC
```

Acessando os valores

```
my @nucleotideos = ("A","T","G","C");  
@nucleotideos[$index] = $valor
```

```
print $nucleotideos[0]; #A
```

```
print $nucleotideos[1]; #T
```

```
print $nucleotideos[2]; #G
```

```
print $nucleotideos[3]; #C
```

Foreach

```
foreach $i (@nomes_nucleotideos) {  
    print "$i\n";  
}
```

OU

```
foreach $i(reverse @nomes_nucleotideos) {  
    print "$i\n";  
}
```

Foreach e inteiros

```
my @arr = (1..10);  
foreach $x (@var) {  
    print $x*3; }
```

```
my $total = 0;  
foreach $x (@var) {  
    $total += $x; }  
print $total;
```

Arrays vazias

```
my @array_vazia = ();
```

Se quiser trabalhar com o primeiro elemento basta usar a função SHIFT (remove) e UNSHIFT (adiciona).



```
push(@array_vazia, "string"); # ou vários elementos para o fim.
```

```
pop(@array_vazia); #porém ele retornar o elemento.
```

```
my $ultimo = pop(@array_vazia);
```

```
print $ultimo; #string
```

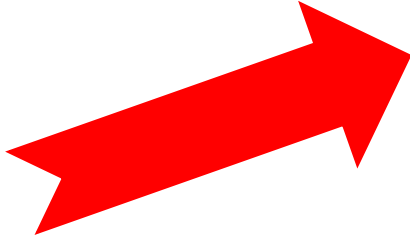
Outras funções

- Length

```
@number =(13, 2, 11, 24, 3, 36, 40, 4);  
$numeros = @number;  
print $numeros; #8
```

- sort

```
@array_organizada = sort @number;  
print "@array_organizada\n"; # 2 3 4 11 13 24 36 40
```



\$#number retorna a posição do último elemento. Lembrando que a contagem começa do zero.

Let's code!!!



Hashes

```
my %nucleotideos = ("A" => "Adenina", "T" => "Timina");  
print $nucleotideos{"A"}; #Adenina
```

"Chave" => "Valor"

Hashes

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

my %nucleotideos = (
    "A" => "Adenina",
    "T" => "Timina",
    "G" => "Guanina",
    "C" => "Citosina",
);

foreach (keys %nucleotideos) {
    print "$_ representa a $nucleotideos{$_}\n";
}
```

Inserindo e removendo valores

```
my %nucleotideos = ("A" => "Adenina", "T" => "Timina");
```

```
$nucleotideos{"G"} = "Guanina";
```

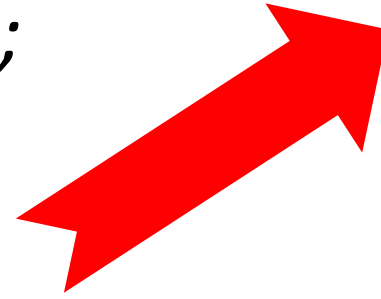
```
delete $nucleotideos{"A"};
```

Tamanho

```
my $lenght = keys %nucleotideos;  
print $lenght;
```

OU

```
print scalar keys %nucleotideos;
```



Se quiser trabalhar com os valores basta trocar a função “keys” por “values”.