# Introdução à Programação de Computadores para Biologia

## Subrotinas II

Aula 14

https://tttorres.github.io/introprog2021/

# Resultado da enquete

# Resultado da enquete

- Pós-graduação (BIO5793): 15/17 respostas

1 Você concorda com a mudança no programa da disciplina?

| Resposta | Média | Total |
|---|---|---|
| - SIM | 100% | 15 |
| Total de respostas para questão | 100% | 15/15 |

- Graduação (BIO0454): 6/11 respostas

1 Você concorda com a mudança no programa da disciplina?

| Resposta | Média | Total |
|---|---|---|
| - SIM | 100% | 6 |
| Total de respostas para questão | 100% | 6/6 |

# Resultado da enquete

# Resultado da enquete

- Pós-graduação (BIO5793): 15/17 respostas

1 Você concorda com a mudança no programa da disciplina?

| Resposta | Média | Total |
|---|---|---|
| - SIM | 100% | 15 |
| Total de respostas para questão | 100% | 15/15 |

- Graduação (BIO0454): 6/11 respostas

1 Você concorda com a mudança no programa da disciplina?

| Resposta | Média | Total |
|---|---|---|
| - SIM | 100% | 6 |
| Total de respostas para questão | 100% | 6/6 |

3/4 votantes concordam com a alteração

# Proposta

| Aula | Data | Tema |
|---|---|---|
| 1 | 19/08 | Apresentação da disciplina |
| 2 | 26/08 | Introdução à Programação / Bash |
| 3 | 02/09 | Bash (continuação) / Algoritmos |
| 4 | 09/09 | Introdução ao Perl / Variáveis e tipos de dados |
| 5 | 16/09 | Arrays e Hashes |
| 6 | 23/09 | Estruturas de controle: condicionais |
| 7 | 30/09 | Estruturas de controle: loops I |
| 8 | 07/10 | Estruturas de controle: loops II |
| 9 | 14/10 | Manipulação de arquivos I |
| 10 | 21/10 | Manipulação de arquivos II |
| ❌ | 28/10 | Dia do servidor público. Não haverá aula |
| 12 | 04/11 | Manipulação de strings e expressões regulares |
| 13 | 11/11 | Subrotinas I |
| 14 | 18/11 | Subrotinas II |
| 15 | 25/11 | Módulos I: introdução e CPAN |
| 16 | 02/12 | Módulos II: BioPerl |

## Proposta

> Exercícios semanais: a cada novo tema de aula, será disponibilizado um exercício que deverá ser entregue no prazo de sete dias. A entrega dos exercícios será realizada via moodle (https://edisciplinas.usp.br/). O atraso na entrega implica na redução da nota em 10%. Serão 8 exercícios ao longo do semestre.

# Proposta

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

my ($E1, $E2, $E3, $E4, $E5, $E6, $E7, $E8, $nota_final);
($E, $M, $H) = @ARGV;

$nota_final = ($E1+$E2+$E3+$E4+$E5+$E6+$E7+$E8)/8;

if ($nota_final >= 5) {
  print "Aluno aprovado\n";
} else {
  print "Aluno reprovado\n";
}

#conversao de nota para conceitos, apenas para a Pos-graduacao
if ($nota_final < 5) {
  print "R, Reprovado, sem direito a credito\n";
} elsif ($nota_final <= 7.0) {
  print "C, Regular, com direito a credito\n";
} elsif ($nota_final <= 8.5) {
  print "B, Bom, com direito a credito\n";
} else {
  print "A, Excelente, com direito a credito\n";
}
exit;
```

# SUBROTINAS

## Definição pelo usuário

- Em qualquer ponto do script
- Geralmente, todas juntas, no início ou fim

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

# Sintaxe da definicao de subrotinas, separadas do script
# principal

# Antes de cada subrotina, um comentario detalhado com o
# da subrotina, sua funcao, os inputs e outputs

sub subName {
   # Bloco de comandos para fazer alguma coisa
   # Nao esquecer a indentacao!
}
```

# SUBROTINAS

## Definição pelo usuário

- Em qualquer ponto do script
- Geralmente, todas juntas, no início ou fim
- A subrotina pode retornar valores

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

subName();                          # nao sao passados argumentos
                                    # nenhum valor eh retornado

subName($arg, "arg", 1);            # sao passados 3 argumentos
                                    # nenhum valor eh retornado

$res = subName($arg, "arg", 1);     # sao passados 3 argumentos
                                    # retorna uma escalar

@res = subName($arg, "arg", 1);     # sao passados 3 argumentos
                                    # retorna um array
```

# VARIÁVEIS LOCAIS E GLOBAIS

## *my* vs *our*

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como local_global.pl
4. Copiar **exemplo01** da página da disciplina.
5. Executar o script.

# VARIÁVEIS LOCAIS E GLOBAIS

## *my* vs *our*

Script: local_global.pl

O que aconteceu com $valor?

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

# $valor como variavel global

$valor = 0;
print "Antes da subrotina, valor \= $valor.\n";

soma(12, 20);

print "Depois da subrotina, valor \= $valor.\n";
exit;

sub soma {
   ($n1, $n2) = @_;
   $valor = $n1 + $n2;
}
```

# VARIÁVEIS LOCAIS E GLOBAIS

## *my* vs *our*

Script: local_global.pl

O que aconteceu com $valor?

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

# $valor como variavel global

$valor = 0;
print "Antes da subrotina, valor \= $valor.\n";

soma(12, 20);

print "Depois da subrotina, valor \= $valor.\n";
exit;

sub soma {
   ($n1, $n2) = @_;
   my $valor = $n1 + $n2;
}
```

# VARIÁVEIS LOCAIS E GLOBAIS

## *my* vs *our*

Declarando variáveis

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

# Declarando variáveis

my $x;                          # OK
my $x, $y, $z;                  # INCORRETO
my ( $x, $y, $z );              # OK
my $x; my $y; my $z;            # OK

# Inicializando variáveis

my $x = 0;                       # OK
my $x = "";                      # OK
my $x = my $y = my $z = 0;       # OK
my $x = my $y = my $z = "";      # OK
```

# PERL SCRIPT

## Estrutura

Minha estrutura preferida:

```
#!/caminho/para/Perl

# cabecalho do script

# declaracao e inicializacao de variaveis

# captura dos argumentos

# corpo do script

# subrotinas
```

Ver **exemplo02** na página da disciplina.

# PERL SCRIPT

## Estrutura

Minha estrutura preferida:

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

## ------------------------------------------------------------------------ ##
##                                                                          ##
##  SCRIPT: seqAleatoria.pl                                  03.11.2014     ##
##                                                                          ##
##  DESCRIPTION: script para exemplificar subrotinas                        ##
##                                                                          ##
##  USAGE: perl seqAleatoria.pl                                             ##
##                                                                          ##
##  AUTHOR: Tatiana Torres    tttorres at ib.usp.br                         ##
##                                                                          ##
## ------------------------------------------------------------------------ ##

use strict;

## Declaracao de variaveis

my $tamanho;
my $sequencia;
my $revSeq;
```

# PERL SCRIPT

## Estrutura

Minha estrutura preferida:

```
## MAIN ##

# criar a sequencia aleatoria
$tamanho = 200;
$sequencia = seqAleatoria($tamanho);

# inverter a sequencia
$revSeq = reverse($sequencia);

# gerar o complemento reverso
$revSeq =~ tr/ATCG/atcg/;

# imprimir (fasta)
print "\>seq\n";
for (my $i = 0; $i < $tamanho; $i+=70) {
   print substr($revSeq,$i,70), "\n";
}

exit;
```

# PERL SCRIPT

## Estrutura

Minha estrutura preferida:

```perl
## SUBROTINAS ##

## subrotina para gerar sequencias aleatorias de nucleotideos
## argumento: tamanho da sequencia (numero de nucleotideos)
## retorna: string com uma sequencia aleatoria

sub seqAleatoria {
   my $seq = "";
   my $tamanho = shift || 40;   # default: $tamanho = 40
   for (my $i = 0; $i < $tamanho; $i++) {
      my $base = int(rand(4));
      if ($base == 0) { $seq .= "A"; next; }
      if ($base == 1) { $seq .= "T"; next; }
      if ($base == 2) { $seq .= "C"; next; }
      if ($base == 3) { $seq .= "G"; next; }
   }
   return $seq;
}
```

# PERL SCRIPT

## Estrutura

Minha estrutura preferida:

```perl
## SUBROTINAS ##

## subrotina para gerar sequencias aleatorias de nucleotideos
## argumento: tamanho da sequencia (numero de nucleotideos)
## retorna: string com uma sequencia aleatoria

sub seqAleatoria {
   my $seq = "";
   my $tamanho = shift || 40;   # default: $tamanho = 40
   for (my $i = 0; $i < $tamanho; $i++) {
      my $base = int(rand(4));
      if ($base == 0) {
         $seq .= "A";
         next;
      }
      if ($base == 1) { $seq .= "T"; next; }
      if ($base == 2) { $seq .= "C"; next; }
      if ($base == 3) { $seq .= "G"; next; }
   }
   return $seq;
}
```

# SUBROTINAS

## Passando argumentos

# SUBROTINAS

## Passando argumentos

```
$valor = soma(1,4,6,7,9);
```

# SUBROTINAS

## Passando argumentos

```
$valor = soma(1,4,6,7,9);
```

**Argumentos em @_:**

$_[0] tem valor 1
$_[1] tem valor 4
$_[2] tem valor 6
$_[3] tem valor 7
$_[4] tem valor 9

# SUBROTINAS

## Passando argumentos

```
$seq = restricao("EcoRI", "HaeIII", "HindIII");
```

# SUBROTINAS

## Passando argumentos

```
$seq = restricao("EcoRI", "HaeIII", "HindIII");
```

**Argumentos em @_:**
$_[0] tem valor EcoRI
$_[1] tem valor HaeIII
$_[2] tem valor HindIII

# SUBROTINAS

## Passando argumentos

```
@nome  = ("EcoRI", "HaeIII", "HindIII");
@sitio = ("GAATTC", "GGCC", "AAGCTT");
$sequencia = restricao(@nome, @sitio);
```

# SUBROTINAS

## Passando argumentos

```
@nome   = ("EcoRI", "HaeIII", "HindIII");
@sitio = ("GAATTC", "GGCC", "AAGCTT");
$sequencia = restricao(@nome, @sitio);
```

**Argumentos em @_:**
$_[0] tem valor EcoRI
$_[1] tem valor HaeIII
$_[2] tem valor HindIII
$_[3] tem valor GAATTC
$_[4] tem valor GGCC
$_[5] tem valor AAGCTT

# SUBROTINAS

## Passando argumentos

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como argumentos.pl
4. Copiar **exemplo03** da página da disciplina.
5. Analisar o script.

# SUBROTINAS

## Passando argumentos

Passando mais de uma lista como argumento (primeira solução):

```
## MAIN ##

$sequencia = restricao($#nome, @nome, $#sitio, @sitio);
exit;

## SUBROTINAS ##

sub restricao {
   my @nome;
   my @sitio;

   my $lastindex = shift;
   for (my $i = 0; $i <= $lastindex; $i++) {
      $nome[$i] = shift;
   }
   $lastindex = shift;
   for (my $i = 0; $i <= $lastindex; $i++) {
      $sitio[$i] = shift;
   }
   # bloco de comandos para usar os arrays
}
```

# SUBROTINAS

## Referências

Exemplo04: Passando mais de uma lista como argumento (segunda solução, passando referências):

```
## MAIN ##

$sequencia = restricao(\@nome, \@sitio);
exit;

## SUBROTINAS ##

sub restricao {
   my ($ref1, $ref2) = @_;
   my @a1 = @{$ref1};
   my @a2 = @{$ref2};

   # bloco de comandos para usar os arrays
   print "@nome\n";
   print "@sitio\n";
}
```

# SUBROTINAS

## Referências

Passando referência de arrays:

```
subRotina(\@arr);
```

Usando arrays:

```
sub subRotina {
        my ($arrRef) = @_;
        my @array = @{$arrRef};
        #...
}
```

# SUBROTINAS

## Referências

Passando referência de hashes:

```
subRotina(\%hash);
```

Usando hashes:

```
sub subRotina {
        my ($hashRef) = @_;
        my %hash = %{$hashRef};
        #...
}
```

# SUBROTINAS

## Referências

Retornando referência de arrays:

```
sub listaTel {
  my @tel;
  $info[0] = <STDIN>;  #nome
  $info[1] = <STDIN>;  #tel
  return \@info;
}
```

Recuperando arrays no corpo do script:

```
my $telRef = listaTel();
my @tel = @{$telRef};
```

# SUBROTINAS

## Referências

Retornando referência de hashes:

```
sub listaTel {
  my %info;
  $info{"nome"} = <STDIN>;
  $info{"tel"}  = <STDIN>;
  return \%info;
}
```

Recuperando hashes no corpo do script:

```
my $telRef = listaTel();
my %tel = %{$telRef};
```

# MÓDULOS

## Conjunto de subrotinas

# MÓDULOS

## comSeqAleatoria.pl

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

##  Script para exemplificar subrotinas

use strict;
use aula14;

## Declaracao de variaveis
my $tamanho; my $sequencia; my $revSeq;

# criar a sequencia aleatoria
$tamanho = 200;
$sequencia = seqAleatoria($tamanho);

# inverter a sequencia
$revSeq = reverse($sequencia);

# gerar o complemento reverso
$revSeq =~ tr/ATCG/atcg/;

# imprimir (fasta)
print "\>seq\n";
for (my $i = 0; $i < $tamanho; $i+=70) {
   print substr($revSeq,$i,70), "\n";
}

exit;
```

# MÓDULOS

## aula14.pm

```
##  Modulo com as minhas subrotinas

##  modulo aula14

sub seqAleatoria {

   $seq = "";
   $tamanho = shift || 40;
   for ($i = 0; $i < $tamanho; $i++) {
      $base = int(rand(4));
      if ($base == 0) { $seq .= "A"; next; }
      if ($base == 1) { $seq .= "T"; next; }
      if ($base == 2) { $seq .= "C"; next; }
      if ($base == 3) { $seq .= "G"; next; }
   }
   return $seq;
}

sub soma {
   ($n1, $n2) = @_;
   $valor = $n1 + $n2;
   return $valor;
}

1;
```

# MÓDULOS

## Conjunto de subrotinas

1. No Geany, abrir script seqAleatoria.pl (exemplo 02).
2. Incluir “use aula14”.
3. Executar o script.

# MÓDULOS

## comSeqAleatoria.pl

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

##  Script para exemplificar subrotinas

use strict;
use aula14;

## Declaracao de variaveis
my $tamanho; my $sequencia; my $revSeq;

# criar a sequencia aleatoria
$tamanho = 200;
$sequencia = seqAleatoria($tamanho);

# inverter a sequencia
$revSeq = reverse($sequencia);

# gerar o complemento reverso
$revSeq =~ tr/ATCG/atcg/;

# imprimir (fasta)
print "\>seq\n";
for (my $i = 0; $i < $tamanho; $i+=70) {
   print substr($revSeq,$i,70), "\n";
}

exit;
```

# MÓDULOS

## Conjunto de subrotinas

comSeqAleatoria.pl

```
perl: warning: Falling back to the standard locale ("C").
Can't locate aula14.pm in @INC (you may need to install
the aula14 module) (@INC contains: /Library/Perl/5.18/dar
win-thread-multi-2level /Library/Perl/5.18 /Network/Libra
ry/Perl/5.18/darwin-thread-multi-2level /Network/Library/
Perl/5.18 /Library/Perl/Updates/5.18.2 /System/Library/Pe
rl/5.18/darwin-thread-multi-2level /System/Library/Perl/5
.18 /System/Library/Perl/Extras/5.18/darwin-thread-multi-
2level /System/Library/Perl/Extras/5.18 .) at saidas.pl l
ine 6.
BEGIN failed--compilation aborted at saidas.pl line 6.
```

# MÓDULOS

## Conjunto de subrotinas

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como aula14.pm
4. Transferir todas as subrotinas para o modulo aula14.pm
5. Executar o script SeqAleatoria.pl

# MÓDULOS

## Conjunto de subrotinas

No terminal:

```
TatianasMacBook:~ tatiana$ perl -V
###varias linhas###
@INC:
    /Library/Perl/5.18/darwin-thread-multi-2level
    /Library/Perl/5.18
    /Network/Library/Perl/5.18/darwin-thread-multi-2level
    /Network/Library/Perl/5.18
    /Library/Perl/Updates/5.18.2
    /System/Library/Perl/5.18/darwin-thread-multi-2level
    /System/Library/Perl/5.18
    /System/Library/Perl/Extras/5.18/darwin-thread-multi-
    /System/Library/Perl/Extras/5.18
    .
```

# MÓDULOS

## SeqAleatoria.pl

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

##  Script para exemplificar subrotinas

use lib '/Users/Tatiana/introprog/modulos/';
use strict;
use aula14;

## Declaracao de variaveis
my $tamanho; my $sequencia; my $revSeq;

# criar a sequencia aleatoria
$tamanho = 200;
$sequencia = seqAleatoria($tamanho);

# inverter a sequencia
$revSeq = reverse($sequencia);

# gerar o complemento reverso
$revSeq =~ tr/ATCG/atcg/;

# imprimir (fasta)
print "\>seq\n";
for (my $i = 0; $i < $tamanho; $i+=70) {
   print substr($revSeq,$i,70), "\n";
}

exit;
```