Introdução à Programação de Computadores para Biologia

Subrotinas II

Aula 13

https://tttorres.github.io/introprog2024/

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
```

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

```
@notas = (7, 9, 5);
@pesos = (2, 3, 5);
$nota_final = media_pond(@notas, @pesos);
```

SUBROTINAS

Passando argumentos

```
@notas = (7, 9, 5);
@pesos = (2, 3, 5);
$nota_final = media_pond(@notas, @pesos);
```

Argumentos em @_:

$_[0] tem valor 7
$_[1] tem valor 9
$_[2] tem valor 5
$_[3] tem valor 2
$_[4] tem valor 3
$_[5] tem valor 5

SUBROTINAS

Passando argumentos

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como argumentos.pl
4. Copiar **exemplo01** da página da disciplina.
5. Analisar o script.
    - Como deve ser o arquivo de entrada?
    - Como os argumentos são indicados na linha de comando?
    - Como argumentos de duas listas foram passados para a subrotina?
    - Como a subrotina captura as informações na ordem?

SUBROTINAS

Passando argumentos

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

```
#!/usr/bin/perl
use strict;

# Pegando argumentos da linha de comando
my @pesos = ($ARGV[0],$ARGV[1],$ARGV[2]);
my $input  = $ARGV[3];
my $output = $ARGV[4];

open(NOTAS, "<$input")  || die "Nao foi possivel abrir $input\n";
open(FINAL, ">$output") || die "Nao foi possivel abrir $output\n";

# Main
while (<NOTAS>) {
   my @linha = split('\t', $_);
   my $nome = $linha[0];
   my $nota_final = media_ponderada($linha[1], $pesos[0],
                                    $linha[2], $pesos[1],
                                    $linha[3], $pesos[2]);
   print FINAL "$nome\t$nota_final\n";
}
exit;
```

SUBROTINAS

Passando argumentos

Arquivo de entrada:

```
Nome1    7.0    8.0    9.0
Nome2    6.5    7.0    8.0
Nome3    5.5    4.0    7.0
```

Linha de comando:

```
tatiana$ perl argumentos.pl 2 3 5 notas.txt final.txt
```

Arquivo de saída:

```
Nome1    8.3
Nome2    7.4
Nome3    5.8
```

SUBROTINAS

Passando argumentos

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

```perl
sub media_ponderada {
    my (@notas_e_pesos) = @_;

    # Variáveis para somar as notas ponderadas e os pesos
    my $soma_ponderada = 0;
    my $soma_pesos = 0;

    # Itera sobre o array de argumentos, dois a dois (nota e peso)
    for (my $i = 0; $i < @notas_e_pesos; $i += 2) {
        my $nota = $notas_e_pesos[$i];
        my $peso = $notas_e_pesos[$i + 1];

        # Calcula o produto nota * peso e soma aos totais
        $soma_ponderada += $nota * $peso;
        $soma_pesos += $peso;
    }

    # Retorna a média ponderada
    return $soma_ponderada / $soma_pesos;
}
```

SUBROTINAS

Passando argumentos

Testando a subrotina:

```
#!/usr/bin/perl
use strict;

# Pegando argumentos da linha de comando
my @pesos = ($ARGV[0],$ARGV[1],$ARGV[2]);
#my $input  = $ARGV[3];
#my $output = $ARGV[4];

#open(NOTAS, "<$input")  || die "Nao foi possivel abrir $input\n";
#open(FINAL, ">$output") || die "Nao foi possivel abrir $output\n";

# Main
#while (<NOTAS>) {
#   my @linha = split('\t', $_);
   my $nome = $linha[0];
   my $nota_final = media_ponderada($linha[1], $pesos[0],
                                    $linha[2], $pesos[1],
                                    $linha[3], $pesos[2]);
   print FINAL "$nome\t$nota_final\n";
#}
exit;
```

SUBROTINAS

Passando argumentos

Testando a subrotina:

```
#!/usr/bin/perl
use strict;

# Pegando argumentos da linha de comando
my @pesos = ($ARGV[0],$ARGV[1],$ARGV[2]);
my @linha = ("Nome1",7,8,9);

#my $input  = $ARGV[3];
#my $output = $ARGV[4];
#open(NOTAS, "<$input")  || die "Nao foi possivel abrir $input\n";
#open(FINAL, ">$output") || die "Nao foi possivel abrir $output\n";
# Main
#while (<NOTAS>) {
#   my @linha = split('\t', $_);
   my $nome = $linha[0];
   my $nota_final = media_ponderada($linha[1], $pesos[0],
                                    $linha[2], $pesos[1],
                                    $linha[3], $pesos[2]);
   print "$nome\t$nota_final\n";
#}
exit;
```

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

```
@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

3000 enzimas de restrição

SUBROTINAS

Passando argumentos

1. Na página da disciplina
2. Copiar **exemplo02** da página da disciplina.
3. Colar no arquivo argumentos.pl
4. Analisar o script.

SUBROTINAS

Passando argumentos

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

```
## MAIN ##

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

Passando argumentos

1. Na página da disciplina
2. Copiar **exemplo03** da página da disciplina.
3. Colar no arquivo argumentos.pl
4. Analisar o script.

SUBROTINAS

Referências

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

```
## MAIN ##

restricao(\@nome, \@sitio);
exit;

## SUBROTINAS ##

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

   # bloco de comandos para usar os arrays
   print "Referencia: $ref1; Array 1: @a1\n";
   print "Referencia: $ref2; Array 2: @a2\n";
}
```

SUBROTINAS

Referências

Passando referência de arrays:

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

Usando arrays:

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

SUBROTINAS

Referências

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como referencias.pl
4. Copiar **exemplo04** da página da disciplina.
5. Completar o script com os comandos que faltam.
    - Como as referências são recuperadas na subrotina?
    - Como as referências são passadas como argumentos?

SUBROTINAS

Referências

Subrotina, pegando as referências:

```
# Subrotina para combinar listas de telefones e endereços
sub agenda {
    <COMO RECEBER REFERÊNCIAS?>

    # Copia os arrays referenciados para variáveis locais
    my @telefones = <COMO USAR REFERÊNCIA?>;
    my @enderecos = <COMO USAR REFERÊNCIA?>;

    my $num_pessoas = @telefones;

    # Processa a lista de telefones
    for(my $i=0; $i<$num_pessoas; $i++) {
        my ($nome,  $tel) = split(':', $telefones[$i]);
        my ($nome2, $end) = split(':', $enderecos[$i]);
        print "$nome\t$tel\t$end\n";
    }
}
```

SUBROTINAS

Referências

Subrotina, pegando as referências:

```
# Subrotina para combinar listas de telefones e endereços
sub agenda {
    my ($ref_telefones, $ref_enderecos) = @_;  # Recebe r

    # Copia os arrays referenciados para variáveis locais
    my @telefones = @{$ref_telefones};
    my @enderecos = @{$ref_enderecos};

    my $num_pessoas = @telefones;

    # Processa a lista de telefones
    for(my $i=0; $i<$num_pessoas; $i++) {
        my ($nome,  $tel) = split(':', $telefones[$i]);
        my ($nome2, $end) = split(':', $enderecos[$i]);
        print "$nome\t$tel\t$end\n";
    }
}
```

SUBROTINAS

Referências

Passando referência de arrays para a subrotina:

```
#!/usr/bin/perl
use strict;

# Listas de contatos, nomes, telefones e endereços
my @lista_tel = ('Ana:91234-5678', 'Carlos:95678-9123', 'I
my @lista_end = ('Ana:Rua A, 123', 'Carlos:Rua B, 456', 'I

# Chamando a subrotina e passando referências para os arr
<COMO USAR A SUBROTINA PASSANDO REFERÊNCIAS?>

exit;
```

SUBROTINAS

Referências

Passando referência de arrays para a subrotina:

```
#!/usr/bin/perl
use strict;

# Listas de contatos, nomes, telefones e endereços
my @lista_tel = ('Ana:91234-5678', 'Carlos:95678-9123', 'I
my @lista_end = ('Ana:Rua A, 123', 'Carlos:Rua B, 456', 'I

# Chamando a subrotina e passando referências para os arra
agenda(\@lista_tel, \@lista_end);

exit;
```

SUBROTINAS

Referências

E se quisermos usar retornar um array?

```
#!/usr/bin/perl
use strict;

# Listas de contatos, nomes, telefones e endereços
my @lista_tel = ('Ana:91234-5678', 'Carlos:95678-9123', 'I
my @lista_end = ('Ana:Rua A, 123', 'Carlos:Rua B, 456', 'I

# Chamando a subrotina e passando referências para os arr
agenda(\@lista_tel, \@lista_end);

exit;
```

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

Exemplo 04: subrotina, retornando referência:

```
sub agenda {
    my ($ref_telefones, $ref_enderecos) = @_;  # Recebe r

    # Copia os arrays referenciados para variáveis locais
    my @telefones = @{$ref_telefones};
    my @enderecos = @{$ref_enderecos};
    my $num_pessoas = @telefones;
    my @minha_agenda;

    # Processa a lista de telefones
    for(my $i=0; $i<$num_pessoas; $i++) {
        my ($nome,  $tel) = split(':', $telefones[$i]);
        my ($nome2, $end) = split(':', $enderecos[$i]);
        $minha_agenda[$i] = $nome.":".$tel.";".$end;
    }
    return \@minha_agenda;
}
```

SUBROTINAS

Referências

Exemplo 04: script, recuperando referência retornada:

```
#!/usr/bin/perl
use strict;

# Listas de contatos (nomes e telefones) e endereços
my @lista_tel = ('Ana:91234-5678', 'Carlos:95678-9123', 'I
my @lista_end = ('Ana:Rua A, 123', 'Carlos:Rua B, 456', 'I

# Chamando a subrotina e passando referências para os arr
my $ref_agenda = agenda(\@lista_tel, \@lista_end);
my @tel_e_end = @{$ref_agenda};

foreach my $pessoa(@tel_e_end) {
   print "$pessoa\n";
}

exit;
```

SUBROTINAS

Referências

Passando referência de hashes:

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

Usando hashes:

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

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};
```

SUBROTINAS

Referências

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como referencias.pl
4. Copiar **exemplo05** da página da disciplina.
5. Completar o script com os comandos que faltam.
    - Como as referências são recuperadas na subrotina?
    - Como as referências são passadas como argumentos?

SUBROTINAS

Referências

Subrotina, pegando as referências:

```
# Subrotina para combinar listas de telefones e endereços
sub agenda {
    <COMO RECEBER REFERÊNCIAS DOS HASHES?>

    # Copia os arrays referenciados para variáveis locais
    my %telefones = <COMO USAR REFERÊNCIA?>;
    my %enderecos = <COMO USAR REFERÊNCIA?>;

    my %minha_agenda;

    # Processa a lista de telefones
    foreach my $pessoa(keys(%telefones)) {
        $minha_agenda{$pessoa} = $telefones{$pessoa}.";".$
    }
    <COMO RETORNAR REFERÊNCIA A HASH?>
}
```

SUBROTINAS

Referências

Subrotina, pegando as referências:

```
# Subrotina para combinar listas de telefones e endereços
sub agenda {
    my ($ref_telefones, $ref_enderecos) = @_;  # Recebe r

    # Copia os arrays referenciados para variáveis locais
    my %telefones = %{$ref_telefones};
    my %enderecos = %{$ref_enderecos};

    my %minha_agenda;

    # Processa a lista de telefones
    foreach my $pessoa(keys(%telefones)) {
        $minha_agenda{$pessoa} = $telefones{$pessoa}.";".$
    }
    return \%minha_agenda;
}
```

SUBROTINAS

Referências

Passando referência de hashes para a subrotina:

```
#!/usr/bin/perl
use strict;
# Listas de contatos em hashes
my %lista_tel = ('Ana'     => '91234-5678',
                 'Carlos'  => '95678-9123',
                 'Beatriz' => '99123-5678');
my %lista_end = ('Ana'     => 'Rua A, 123',
                 'Carlos'  => 'Rua B, 456',
                 'Beatriz' => 'Rua C, 789');
# Chamando a subrotina e passando referências para os arr
my $ref_agenda = <COMO USAR A SUBROTINA PASSANDO REF?>;
my %tel_e_end  = <COMO RECUPERAR O HASH PASSADO COMO REF?

foreach my $pessoa(keys(%tel_e_end)) {
        print "$pessoa: $tel_e_end{$pessoa}\n";
}
exit;
```

SUBROTINAS

Referências

Passando referência de hashes para a subrotina:

```
#!/usr/bin/perl
use strict;
# Listas de contatos em hashes
my %lista_tel = ('Ana'     => '91234-5678',
                 'Carlos'  => '95678-9123',
                 'Beatriz' => '99123-5678');
my %lista_end = ('Ana'     => 'Rua A, 123',
                 'Carlos'  => 'Rua B, 456',
                 'Beatriz' => 'Rua C, 789');
# Chamando a subrotina e passando referências para os arr
my $ref_agenda = agenda(\%lista_tel, \%lista_end);
my %tel_e_end  = %{$ref_agenda};

foreach my $pessoa(keys(%tel_e_end)) {
        print "$pessoa: $tel_e_end{$pessoa}\n";
}
exit;
```

MÓDULOS

Conjunto de subrotinas

MÓDULOS

Organizando e modularizando

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como SeqAleatoria.pl
4. Copiar **exemplo06** da página da disciplina.

MÓDULOS

SeqAleatoria.pl

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

use strict;

## Declaracao de variaveis

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

## 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;
```

MÓDULOS

SeqAleatoria.pl

```
## 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;
}
```

MÓDULOS

aula13.pm

```perl
##  Modulo com as minhas subrotinas

##  modulo aula13

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;
}

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

1;
```

MÓDULOS

Conjunto de subrotinas

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como aula13.pm
4. Transferir todas a subrotina para o modulo aula13.pm
5. Ao final do arquivo, incluir a linha `1;`
6. Executar o script SeqAleatoria.pl

MÓDULOS

Conjunto de subrotinas

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como aula13.pm
4. Transferir todas a subrotina para o modulo aula13.pm
5. Ao final do arquivo, incluir a linha `1;`
6. Executar o script SeqAleatoria.pl

```
Undefined subroutine &main::seqAleatoria called at
aula13.pl line 27.
```

MÓDULOS

Conjunto de subrotinas

1. No Geany, no script seqAleatoria.pl.
2. Incluir `use aula13;`
3. Executar o script.

MÓDULOS

SeqAleatoria.pl

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

use strict;
use aula13;

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

## 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),

exit;
```

MÓDULOS

Conjunto de subrotinas

perl SeqAleatoria.pl

```
Can't locate aula13.pm in @INC (you may need to install
the aula13 module) (@INC contains: /Library/Perl/5.30/
darwin-thread-multi-2level /Library/Perl/5.30/Network/
Library/Perl/5.30/darwin-thread-multi-2level /Network/
Library/Perl/5.30 /Library/Perl/Updates/5.30.3 /System/
Library/Perl/5.30/darwin-thread-multi-2level /System/
Library/Perl/5.30 /System/Library/Perl/Extras/5.30/darwin
-thread-multi-2level /System/Library/Perl/Extras/5.30) at
aula13.pl line 16.
BEGIN failed--compilation aborted at aula13.pl line 16.
```

MÓDULOS

Conjunto de subrotinas

`@INC` é uma variável especial em Perl que contém uma lista de diretórios onde o interpretador Perl procura por módulos e bibliotecas quando o comando `use` é executado.

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/';
use strict;
use aula13;

## 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

SeqAleatoria.pl

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

##  Script para exemplificar subrotinas

use lib '/mnt/c/Usuarios/Aluno/introprog/';
use strict;
use aula13;

## 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;
```