Introdução à Programação de Computadores para Biologia

Subrotinas

Aula 12

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

FUNÇÕES PRÉ-DEFINIDAS

Código para realizar uma tarefa específica

Exemplos de funções que já vimos:

```
$newString = substr ($str,1,4);   #retorna uma substring

@array = split (/\t/,$line);      #quebra uma linha em um a

push (@array, $num);              #encaixa uma variável no
                                  #de um array

chop($string);                    #remove último caracter

$primeiro = shift(@array);        #remove e retorna o prime
                                  #elemento de um array
```

FUNÇÕES PRÉ-DEFINIDAS

Código para realizar uma tarefa específica

Funções tem argumentos e retornam valores:

```
$newString = substr ($str,1,4);
```

Valor retornado:
A função *substr* retorna uma string

Argumentos:
(STRING, INÍCIO, COMPRIMENTO)

SUBROTINAS

Função definida pelo usuário

SUBROTINAS

Função definida pelo usuário

- Organização (manter, atualizar, compartilhar)
- Divisão de trabalho (simplificar)

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

sub SUB_NAME {
        # Fazer alguma coisa
}

sub printHello {
        print "Hello World!\n";
}

sub latir {
        print "Au-Au\n";
}
```

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

Utilização

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como subrotinas.pl
4. Copiar **exemplo01** da página da disciplina.
5. Utilizar a subrotina em um script.
6. Qual a função da subrotina?

SUBROTINAS

Utilização

Script: subrotinas.pl

```
#/usr/bin/perl/

exemplo01();

exit;

sub exemplo01 {
    for ($i = 0; $i < 40; $i++) {
        $base = int(rand(4));
        if ($base == 0) { print "A"; next; }
        if ($base == 1) { print "T"; next; }
        if ($base == 2) { print "C"; next; }
        if ($base == 3) { print "G"; next; }
    }
    print "\n";
}
```

SUBROTINAS

Utilização

Script: subrotinas.pl

```
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
Darwin:~ Tatiana$ perl exemplo01.pl >>out.txt
```

SUBROTINAS

Utilização

Arquivo: out.txt

```
CAAACCAGCCCTGAGCGTCCGTTTACGAAACGACCGCCCA
GTCGTGATCTGTAGATCGTACACGTGCCGCATTTTACAAT
GGCCAGACGTGCGAGGAAAGTAGTAAAAGGCGATCTGTTG
GATGTTTTAGTCATACCACCGATAGTTTCCTTGATGTCTT
TTGGAATAGTATACTGTACGCTATTGCTGAGGTGCCGCCC
TTTCTACAACGACTACTACGCCACCACTTCGGCGGGGTGC
AATAAGGTCAATAGTGTTGATATTGCCTAATTTCGAAGTC
CCCGTTCTGGTGGTCTCTAGCGCGCCCTCGGACGCCGACG
AGTTCTTGGCTTCGGTTTGTTTGTCAGATTATGCGATTCA
GATAACAACTCGATCCAGTACCTGTACCACGGTAGTCTCC
```

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

Argumentos

1. No Geany, alterar a subrotina para passar como argumento, o tamanho desejado da sequência.
2. Use a subrotina passando o argumento de 100 nt.

Variável @_ recebe os argumentos de subrotinas

Muito similar ao @ARGV

SUBROTINAS

Argumentos

1. No Geany, alterar a subrotina para passar como argumento, o tamanho desejado da sequência.
2. Use a subrotina passando o argumento de 100 nt.

```
#/usr/bin/perl/

seqAleatoria(100);
exit;

sub seqAleatoria {
   $tamanho = $_[0];
   for ($i = 0; $i < $tamanho; $i++) {
      $base = int(rand(4));
      if ($base == 0) { print "A"; next; }
      if ($base == 1) { print "T"; next; }
      if ($base == 2) { print "C"; next; }
      if ($base == 3) { print "G"; next; }
   }
   print "\n";
}
```

SUBROTINAS

Utilização

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como seqAleatoria.pl
4. Copiar **exemplo02** da página da disciplina.
5. Executar o script.
6. Qual a função do comando "$length = shift || 40;"?
7. Qual a função do bloco "unless ($i%70) {print"\n"};"?

SUBROTINAS

Utilização

Script: seqAleatoria.pl

```
#/usr/bin/perl/

seqAleatoria(200);
exit;

sub seqAleatoria {
    $tamanho = shift || 40;
    for ($i = 0; $i < $tamanho; $i++) {
        unless ($i%70) { print "\n" };
        $base = int(rand(4));
        if ($base == 0) { print "A"; next; }
        if ($base == 1) { print "T"; next; }
        if ($base == 2) { print "C"; next; }
        if ($base == 3) { print "G"; next; }
    }
    print "\n";
}
```

SUBROTINAS

print vs *return*

SUBROTINAS

print vs *return*

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

SUBROTINAS

print vs *return*

Script: soma.pl

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

soma(12, 20);
exit;

sub soma {
    ($n1, $n2) = @_;
    $res = $n1 + $n2;
    print "$res\n";
}
```

SUBROTINAS

print vs *return*

E se quisermos utilizar o valor da soma em outra etapa do script?

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

soma(12, 20);
exit;

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

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

print vs *return*

Alterar a subrotina soma para retornar o valor usando a função *return* em vez de *print*. Usar o valor retornado em outra operação.

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

soma(12, 20);
exit;

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

SUBROTINAS

print vs *return*

Alterar a subrotina soma para retornar o valor usando a função *return* em vez de *print*. Usar o valor retornado em outra operação.

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

$valor = soma(12, 20);
$resultado = $valor*5;
print "$resultado\n";
exit;

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

SUBROTINAS

print vs *return*

Script: seqAleatoria.pl, como retornar a sequência?

```
#/usr/bin/perl/

seqAleatoria(200);
exit;

sub seqAleatoria {
   $tamanho = shift || 40;
   for ($i = 0; $i < $tamanho; $i++) {
      unless ($i%70) { print "\n" };
      $base = int(rand(4));
      if ($base == 0) { print "A"; next; }
      if ($base == 1) { print "T"; next; }
      if ($base == 2) { print "C"; next; }
      if ($base == 3) { print "G"; next; }
   }
   print "\n";
}
```

SUBROTINAS

print vs *return*

Problema:

> Como utilizar o resultado da subrotina seqAleatoria em uma segunda etapa do script?

SUBROTINAS

print vs *return*

1. No Geany, abrir script seqAleatoria.pl
2. Copiar **exemplo04** da página da disciplina.
3. Modificar a subrotina seqAleatoria para retornar uma sequência que será utilizada no corpo do script para gerar seu complemento reverso.

ALGORITMO:

i. Gerar sequência aleatória;
ii. inverter a sequência;
iii. gerar o complemento.

SUBROTINAS

print vs *return*

Script: seqAleatoria.pl, main

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

# gerar sequencia aleatoria com a subrotina seqAleatoria
$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 ($i = 0; $i < $tamanho; $i+=70) {
        print substr($revSeq,$i,70), "\n";
}
exit;
```

SUBROTINAS

print vs *return*

Script: seqAleatoria.pl, subrotina

```
sub seqAleatoria {

   $tamanho = shift || 40;
    for ($i = 0; $i < $tamanho; $i++) {
        unless ($i%70) { print "\n" };
        $base = int(rand(4));
        if ($base == 0) { print "A"; next; }
        if ($base == 1) { print "T"; next; }
        if ($base == 2) { print "C"; next; }
        if ($base == 3) { print "G"; next; }
    }
    print "\n";
}
```

SUBROTINAS

print vs *return*

Script: seqAleatoria.pl, subrotina

```
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;

}
```

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 **exemplo05** 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 usando `my` ?

```
#!/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*

Script: local_global.pl

O que aconteceu com $valor usando `our`?

```
#!/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) = @_;
   our $valor = $n1 + $n2;
}
```

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 **exemplo06** da página da disciplina.
5. Executar o script.

VARIÁVEIS LOCAIS E GLOBAIS

my vs *our*

```perl
#!/usr/bin/perl/
my $var_global = "global"; # Variável global

mudar_var_global();  # Modifica a variável global
usar_var_local();    # Declara e usa uma variável local

# Exibe o valor da variável global
print "\nFora das subrotinas: var_global = $var_global e var_local =

exit;

sub mudar_var_global {  # Acessa e modifica a variável global
    $var_global = "VARIAVEL GLOBAL ALTERADA";
    print "Dentro da subrotina: var_global é $var_global\n";
}

sub usar_var_local {
    my $var_local = "VARIAVEL LOCAL";  # Variável local
    print "Dentro da subrotina: val_local é $var_local\n";
}
```

VARIÁVEIS LOCAIS E GLOBAIS

Trocar `my` em `usar_var_local` por `our`

```
#!/usr/bin/perl/
my $var_global = "global"; # Variável global

mudar_var_global();  # Modifica a variável global
usar_var_local();    # Declara e usa uma variável local

# Exibe o valor da variável global
print "\nFora das subrotinas: var_global = $var_global e var_local =

exit;

sub mudar_var_global {  # Acessa e modifica a variável global
    $var_global = "VARIAVEL GLOBAL ALTERADA";
    print "Dentro da subrotina: var_global é $var_global\n";
}

sub usar_var_local {
    our $var_local = "VARIAVEL LOCAL";  # Variável local
    print "Dentro da subrotina: val_local é $var_local\n";
}
```

VARIÁVEIS LOCAIS E GLOBAIS

my vs *our*

`my` : Cria uma variável local ao bloco onde foi declarada, com escopo estritamente limitado.

`our` : Declara uma variável global do pacote, disponível em todo o pacote e em outros pacotes que a referenciem.

VARIÁVEIS LOCAIS E GLOBAIS

use strict

> `use strict` reforça práticas recomendadas ao trabalhar com variáveis, especialmente em relação a escopo.

> Em Perl, `use strict` força o programador a declarar variáveis antes de usá-las, evitando bugs devido a erros de digitação ou uso indevido de variáveis globais e locais.

VARIÁVEIS LOCAIS E GLOBAIS

use strict

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como local_global.pl
4. Copiar **exemplo07** da página da disciplina.
5. Executar o script.

VARIÁVEIS LOCAIS E GLOBAIS

use strict

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

use strict;

our $global_var = "Eu sou global!";

print "$local_var\n"; # erro devido ao escopo

exemplo();

exit;

sub exemplo {
    my $local_var = "Eu sou local!";
    print "$local_var\n";
    print "$global_var\n";
}
```

VARIÁVEIS LOCAIS E GLOBAIS

use strict

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

use strict;

our $global_var = "Eu sou global!";

# print "$local_var\n"; # erro devido ao escopo

exemplo();

exit;

sub exemplo {
    my $local_var = "Eu sou local!";
    print "$local_var\n";
    print "$global_var\n";
}
```

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 **exemplo08** na página da disciplina.

PERL SCRIPT

Estrutura

Minha estrutura preferida:

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

## ------------------------------------------------------------------------ ##
##                                                                          ##
##  SCRIPT: seqAleatoria.pl                                  04.11.2024     ##
##                                                                          ##
##  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 **exemplo07** 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

Exemplo08: 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

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

##  Script para exemplificar subrotinas

use strict;
use aula09;

## 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

aula09.pm

```
##  Modulo com as minhas subrotinas

##  modulo aula09

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.
2. Incluir “use aula09”.
3. Executar o script.

MÓDULOS

comSeqAleatoria.pl

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

##  Script para exemplificar subrotinas

use strict;
use aula09;

## 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 aula09.pm in @INC (you may need to install
the aula09 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

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/scripts/modulos/';
use strict;
use aula09;

## 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

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como aula09.pm
4. Transferir todas as subrotinas sejam transferidas para o modulo aula09.pm
5. Executar o script SeqAleatoria.pl