Introdução à Programação de Computadores para Biologia

Subrotinas

Aula 13

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

Programa atual

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
11	04/11	Manipulação de strings e expressões regulares
12	11/11	Subrotinas
13	18/11	Módulos I: introdução e CPAN
14	25/11	Módulos II: BioPerl
15	02/12	Revisão e encerramento da disciplina

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	11/11	Subrotinas II
15	18/11	Módulos I: introdução e CPAN
16	25/11	Módulos II: BioPerl

Forma de avaliação atual

> 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 pela metade. Serão 10 exercícios ao longo do semestre, com pesos diferentes conforme a complexidade do exercício.

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 10 exercícios ao longo do semestre, com pesos diferentes conforme a complexidade do exercício.

Critério atual para aprovação e pesos

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

my ($E, $M, $H, $nota_final); #$E, easy; $M, medium; $H, hard
($E, $M, $H) = @ARGV;

$nota_final = ((2*$E)+(3*$M)+(5*$H))/10;

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;
```

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;
```

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

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

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;
}
```

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

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 {

   $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;

}
```