Introdução à Programação de Computadores para Biologia

Tipos de dados

Arrays e Hashes

Aula 05

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

TIPOS DE DADOS EM PERL

Variáveis

1. Escalares ($):

```
my $variavel_escalar = 1;
my $cidade = "Sao Paulo";
my $sequencia = "ATCCTACTGTGCGTCAGGCTAAGCTA";
```

2. Arrays, vetores (@):

```

```

3. Hashes, vetores associativos (%):

```

```

TIPOS DE DADOS EM PERL

Variáveis

1. Escalares ($):

```
my $variavel_escalar = 1;
my $cidade = "Sao Paulo";
my $sequencia = "ATCCTACTGTGCGTCAGGCTAAGCTA";
```

2. Arrays, vetores (@):

```
@genes = ("CG7856", "scpr-B", "CG4294", "Sgt","CG42308");
```

3. Hashes, vetores associativos (%):

```
%genes = ("FBgn0033056" => "CG7856",
          "FBgn0037888" => "scpr-B",
          "FBgn0034742" => "CG424",
          "FBgn0032640" => "Sgt")
```

VARIÁVEIS ESCALARES

Variáveis

1. Nomes precedidos de "$":

```
my $cidade = "Sao Paulo";       #correto
my $ cidade = "Sao Paulo";      #incorreto
```

2. Nomes podem conter uma ou mais letras "A-Z" ou "a-z" incluindo "_" e depois dela(s) números:

```
my $v = 1;              #correto
my $var = 1;            #correto
my $var1 = 1;           #correto
my $var2 = 2;           #correto
my $var_1 = 1;          #correto
my $var 1 = 1;          #incorreto
my $1var = 1;           #incorreto
my $2var = 2;           #incorreto
my $variavel_escalar_criada_para_armazenar_um_numero_real = 1;
```

VARIÁVEIS ESCALARES

Valores de Escalares

PERL tem dois tipos básicos de escalares:

1. Números:

```
$y=1;          # inteiro positivo
$z=-5;         # inteiro negativo
$x = 3.14;     # real em ponto flutuante
$w = 2.75E-6; # real em notação científica
$t = 0377;     # octal
$u = 0xffff;   # hexadecimal
```

2. Strings:

```
$string1 = "Oi, eu sou uma string!";      # string
$string2 = 'Oi, eu tb sou uma string!';   # string
$string3 = "ATCGATCGATCGATCGATTGGATC";    # string
```

VARIÁVEIS ESCALARES

Valores de Escalares

PERL é diferente de algumas linguagens de programação: o tipo de variável NÃO precisa ser declarado.

Exemplo em C

```c
#include <stdio.h>

int main(void){
        int inteiro;  //guarda numeros inteiros
        char caract;  //guarda caracteres
        float real;   //um número real com precisão simples
        double reald; //um número real com precisão dupla
        void vazio;   //tipo vazio

        inteiro = 5;
        caract = 'C';
        real = 27.25;
        reald = 22.442e2;

return 1;
}
```

VARIÁVEIS ESCALARES

Valores

Script: escalares.pl; output:

```
$y = 1
$z = -5
$x = 3.14
$w = 2.75e-06
$t = 255
$u = 65535
$s = 12

Oi, eu sou uma string!
Oi, eu tb sou uma string
ATCGATCGATCGATCGATTGGATC
```

$t = atribuir um valor octal = 0377

$u = atribuir um valor hexadecimal = 0xffff

$s = atribuir um valor binário = 0b1100

VARIÁVEIS ESCALARES

Valores

Script: escalares.pl;

```
#! /usr/bin/perl

# atribuindo valores as variaveis
$y = 1;        # inteiro positivo
$z = -5;       # inteiro negativo
$x = 3.14;     # real em ponto flutuante
$w = 2.75e-6; # real em notação científica
$t = 0377;     # octal
$u = 0xffff;   # hexadecimal
$s = 0b1100;   # binario

$string1 = "Oi, eu sou uma string!";    # string
$string2 = 'Oi, eu tb sou uma string';  # string
$string3 = "ATCGATCGATCGATCGATTGGATC";  # string
```

VARIÁVEIS ESCALARES

Valores

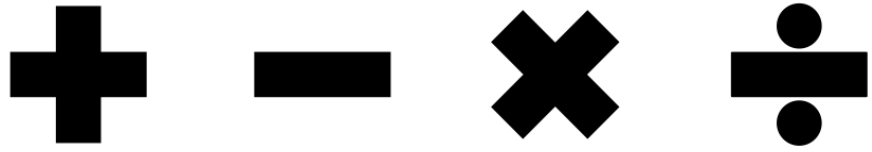
Script: escalares.pl;

```
#continuacao

# imprimindo
print "\$y \= $y\n";
print "\$z \= $z\n";
print "\$x \= $x\n";
print "\$w \= $w\n";
print "\$t \= $t\n";
print "\$u \= $u\n";
print "\$s \= $s\n\n";

print "$string1\n$string2\n$string3\n\n";

exit;
```

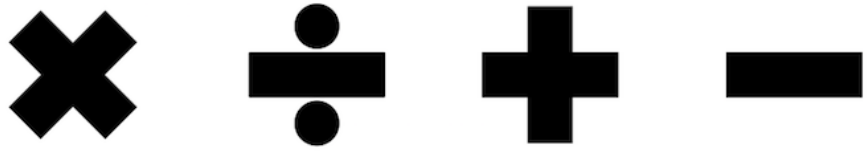
VARIÁVEIS ESCALARES

Operações com números

1. Os "operadores da escola" estão disponíveis:



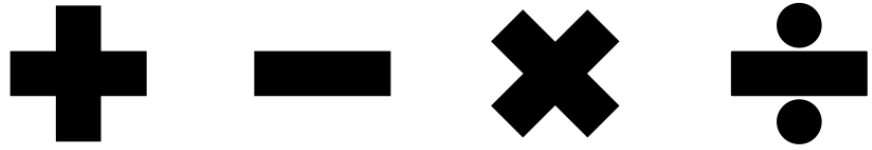
2. Precedência: praticamente igual "da escola"



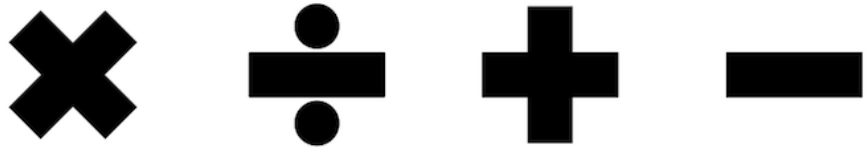
VARIÁVEIS ESCALARES

Operações com números

1. Os "operadores da escola" estão disponíveis:



2. Precedência: praticamente igual "da escola"



Sempre use parênteses!!!

VARIÁVEIS ESCALARES

Operações com números

CARACTER	FUNÇÃO
+	Adição
=	Atribuição
+=	Atribuição após soma
-=	Atribuição após subtração
++	Auto-acréscimo
--	Auto-decréscimo
/	Divisão
%	Módulo (Resto da divisão)
*	Multiplicação
**	Potenciação (Exponenciação)
sqrt()	Raiz quadrada
-	Subtração

VARIÁVEIS ESCALARES

Operações com números

Script: operacoes.pl

```
#! /usr/bin/perl
# script para testar operacoes matematicas

# testando
$a = 1;
print "Atribuicao\:                \$a \= $a\n";

#++$a;
#print "Auto\-acrescimo\:           \$a \= $a\n";

#--$a;
#print "Auto\-decrescimo\:          \$a \= $a\n";

#$b = 3 + 1;
#print "Soma\:                     \$b \= $b\n";

#$c = $a + $b;
#print "Soma\:                     \$c \= $c\n";
```

VARIÁVEIS ESCALARES

Operações com números

Script: operacoes.pl

Output:

```
Atribuicao:                $a = 1
Auto-acrescimo:            $a = 2
Auto-decrescimo:           $a = 1
Soma:                      $b = 4
Soma:                      $c = 5
Multiplicacao:             $d = 20
Divisao:                   $e = 5
Raiz quadrada:             $f = 2
Equacao:                   $g = 25
Modulo:                    $h = 1
Modulo:                    $i = 0
Potenciacao:               $j = 25
Adicao e atribuicao:       $j = 30
Subtracao e atribuicao: $j = 25
```

PROBLEMA

> No arquivo dmel-gene-r5.45.fasta há um conjunto de genes presentes no genoma de *Drosophila melanogaster*. Queremos armazenar cada sequência em uma variável com um número identificador único.

1. Como implementar em Perl?

2. Quantas variáveis seriam geradas?

Revisando comandos Unix

Quantos genes há no arquivo dmel-gene-r5.45.fasta?

1. Gravar arquivo (pasta introprog)
2. Ir para a pasta

```

```

3. O arquivo está íntegro?

```

```

4. Quantas sequências no arquivo (linhas iniciadas em >)?

```

```

Revisando comandos Unix

Quantos genes há no arquivo dmel-gene-r5.45.fasta?

1. Gravar arquivo (pasta introprog)
2. Ir para a pasta

```
cd /mnt/c/Users/Aluno/introprog/
```

3. O arquivo está íntegro?

```

```

4. Quantas sequências no arquivo (linhas iniciadas em >)?

```

```

Revisando comandos Unix

Quantos genes há no arquivo dmel-gene-r5.45.fasta?

1. Gravar arquivo (pasta introprog)
2. Ir para a pasta

```
cd /mnt/c/Users/Aluno/introprog/
```

3. O arquivo está íntegro?

```
md5sum dmel-gene-r5.45.fasta
```

4. Quantas sequências no arquivo (linhas iniciadas em >)?

```

```

Revisando comandos Unix

Quantos genes há no arquivo dmel-gene-r5.45.fasta?

1. Gravar arquivo (pasta introprog)
2. Ir para a pasta

```
cd /mnt/c/Users/Aluno/introprog/
```

3. O arquivo está íntegro?

```
md5sum dmel-gene-r5.45.fasta
```

4. Quantas sequências no arquivo (linhas iniciadas em >)?

```
grep -c "^>" dmel-gene-r5.45.fasta
```

PROBLEMA

> No arquivo dmel-gene-r5.45.fasta há um conjunto de genes presentes no genoma de *Drosophila melanogaster*. Queremos armazenar cada sequência em uma variável com um número identificador único.

1. Como implementar em Perl?

```
$gene00000001 = "ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGC";
$gene00000002 = "AAGAGAGAGGATAGAGAGATCTTCTCTCTCGGGGTAGC";
```

2. Quantas variáveis seriam geradas?

```

```

PROBLEMA

> No arquivo dmel-gene-r5.45.fasta há um conjunto de genes presentes no genoma de *Drosophila melanogaster*. Queremos armazenar cada sequência em uma variável com um número identificador único.

1. Como implementar em Perl?

```
$gene00000001 = "ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGC";
$gene00000002 = "AAGAGAGAGGATAGAGAGATCTTCTCTCTCGGGGTAGC";
```

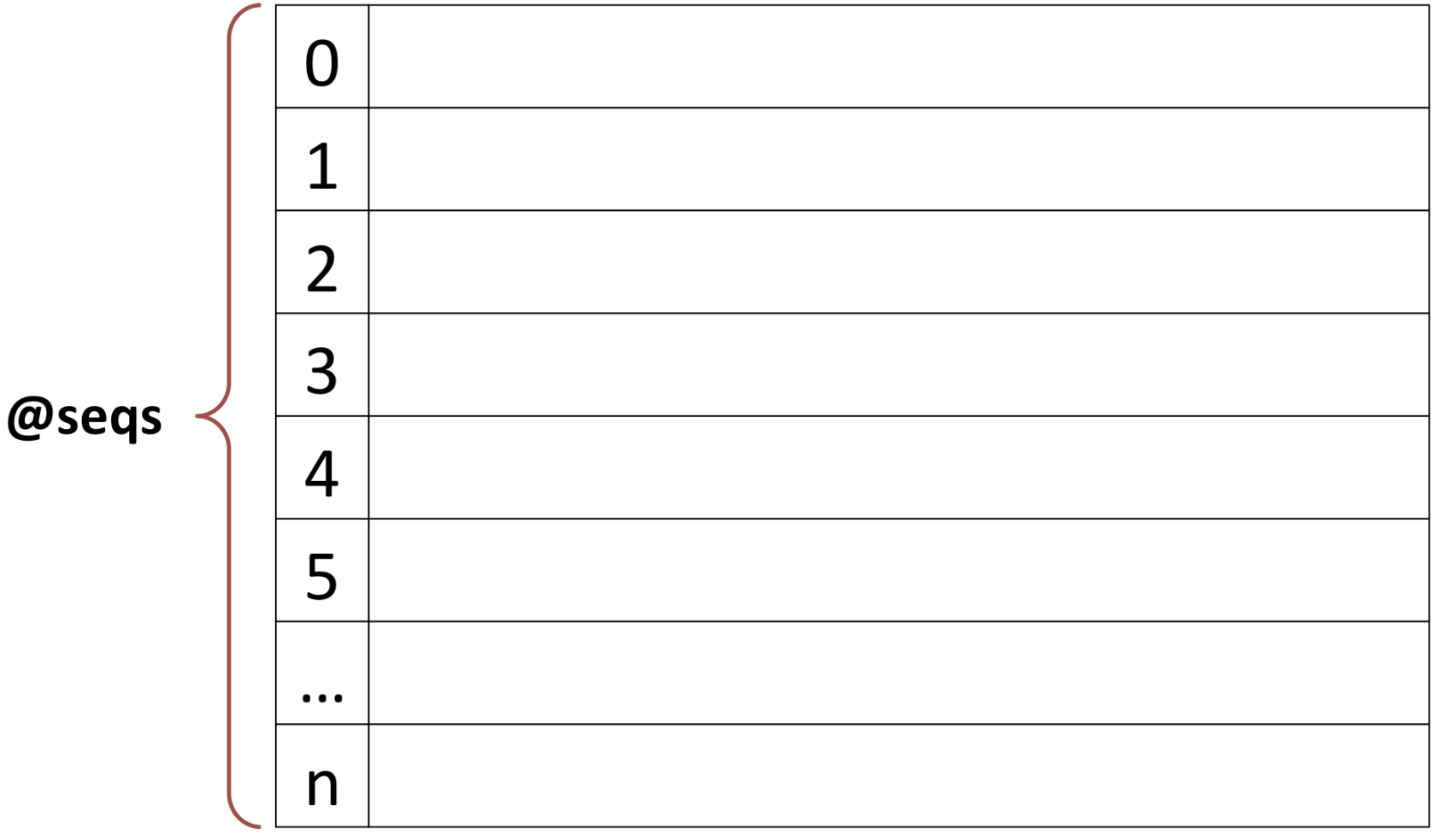
2. Quantas variáveis seriam geradas?

```
$gene00000001 = "ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGC";
$gene00015608 = "AAGAGAGAGGATAGAGAGATCTTCTCTCTCGGGGTAGC";
```

VETORES ("ARRAYS")

0	
1	
2	
3	
4	
5	
...	
n	

VETORES ("ARRAYS")



VETORES ("ARRAYS")

@seqs

0	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
1	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
2	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
3	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
4	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
5	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
...	...
n	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA

VETORES ("ARRAYS")

@genes

0	for
1	Mvl
2	Cyp6g1
3	Cut
4	Ovo
5	Npf
...	...
n	white

VETORES ("ARRAYS")

Na prática

1. No terminal, descobrir quais são os cinco primeiros genes do arquivo **dmel-gene-r5.45.fasta**

VETORES ("ARRAYS")

Na prática

1. No terminal, descobrir quais são os cinco primeiros genes do arquivo **dmel-gene-r5.45.fasta**

```
grep ">" dmel-all-gene-r5.45.fasta | head -5
>FBgn0033056 loc=2R:complement(1944862..1947063) name=CG7856; length=
release=r5.45; species=Dmel;
>FBgn0037888 type=gene; loc=3R:complement(7065763..7066713); name=scp
length=951; release=r5.45; species=Dmel;
>FBgn0034742 type=gene; loc=2R:complement(18487910..18493140); name=C
length=5231; release=r5.45; species=Dmel;
>FBgn0032640 type=gene; loc=2L:17471603..17474773; name=Sgt; length=3
release=r5.45; species=Dmel;
>FBgn0259204 type=gene; loc=X:6905390..6906170; name=CG42308; length=
release=r5.45; species=Dmel;
```

VETORES ("ARRAYS")

Na prática

1. No terminal, descobrir quais são os cinco primeiros genes do arquivo **dmel-gene-r5.45.fasta**

```
grep ">" dmel-gene-r5.45.fasta | head -5
```

2. No Geany, File > New File.
3. File > Save as...
4. Gravar arquivo como arrays.pl
5. Copiar **exemplo01** da página da disciplina e substituir o nome dos genes.

VETORES ("ARRAYS")

Na prática

Script: arrays.pl

```
# exemplo01
#! /usr/bin/perl
# script para entender arrays

# criando o array de nomes
@gene_names = ("gene1", "gene2", "gene3", "gene4", "gene5"

exit;
```

VETORES ("ARRAYS")

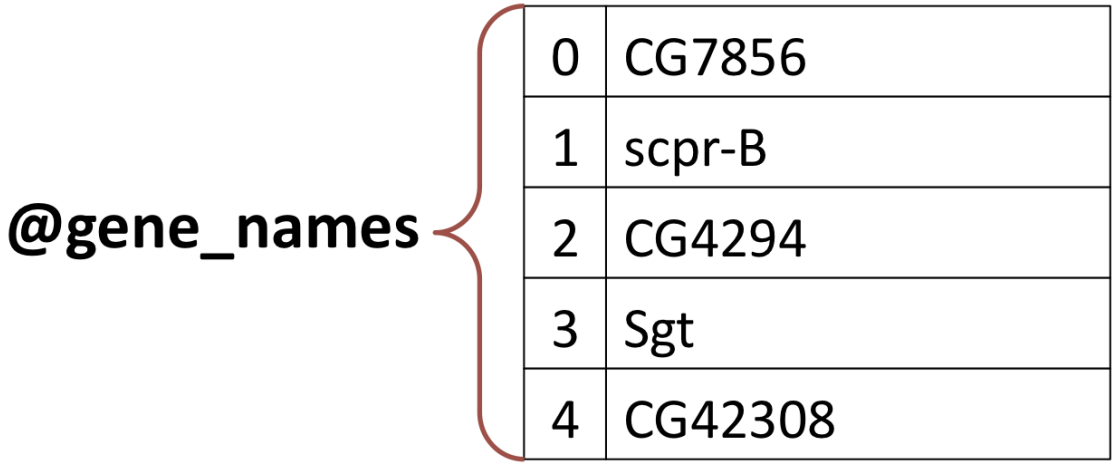
Na prática

Script: arrays.pl

```
# exemplo01
#! /usr/bin/perl
# script para entender arrays

# criando o array de nomes
@gene_names = ("CG7856", "scpr-B", "CG4294", "Sgt", "CG42

exit;
```

VETORES ("ARRAYS")

Na prática



VETORES ("ARRAYS")

Na prática

Script: arrays.pl

Copiar e colar **exemplo02** da página

```
# exemplo01
#! /usr/bin/perl
# script para entender arrays

# criando o array de nomes
@gene_names = ("CG7856", "scpr-B", "CG4294", "Sgt",
               "CG42308");

# criando o array de IDs
@gene_IDs = ("FBgn0033056", "FBgn0037888", "FBgn0034742",
             "FBgn0032640", "FBgn0259204" );

exit;
```

VETORES ("ARRAYS")

Na prática

@gene_names

0	CG7856
1	scpr-B
2	CG4294
3	Sgt
4	CG42308

@gene_IDs

0	FBgn0033056
1	FBgn0037888
2	FBgn0034742
3	FBgn0032640
4	FBgn0259204

VETORES ("ARRAYS")

Acessando elementos individuais de um array

Script: arrays.pl

```
# criando o array de IDs
@gene_IDs = ( "FBgn0033056", "FBgn0037888", "FBgn0034742"
              "FBgn0032640", "FBgn0259204" );

# acessando os elementos individuais do array
print "$gene_IDs[0] \= $gene_names[0]\;\n";




exit;
```

VETORES ("ARRAYS")

Acessando elementos individuais de um array

Script: arrays.pl

Copiar e colar **exemplo03** da página

```
# criando o array de IDs
@gene_IDs = ( "FBgn0033056", "FBgn0037888", "FBgn0034742"
              "FBgn0032640", "FBgn0259204" );

# acessando os elementos individuais do array
print "$gene_IDs[0] \= $gene_names[0]\;\n";
print "$gene_IDs[1] \= $gene_names[1]\;\n";
print "$gene_IDs[2] \= $gene_names[2]\;\n";
print "$gene_IDs[3] \= $gene_names[3]\;\n";
print "$gene_IDs[4] \= $gene_names[4]\;\n";

exit;
```

VETORES ("ARRAYS")

Atribuição de valores individuais

Script: arrays.pl

Copiar e colar **exemplo04** da página

```
print "$gene_IDs[4] \= $gene_names[4]\;\n";

# exemplo04
# continuacao do script para entender arrays

#adicionando elementos no array
$gene_names[5] = "CG15556";
$gene_IDs[5]   = "FBgn0039821";

print "$gene_IDs[5] \= $gene_names[5]\;\n";

exit;
```

VETORES ("ARRAYS")

Atribuição de valores individuais

Script: arrays.pl

Copiar e colar **exemplo05** da página

```
$gene_IDs[5]   = "FBgn0039821";

print "$gene_IDs[5] \= $gene_names[5]\;\n";

# exemplo05
# adicionando mais elementos no array
$gene_names[7] = "CG9773";
$gene_IDs[7]   = "FBgn0037609";

print "$gene_IDs[6] \= $gene_names[6]\;\n";
print "$gene_IDs[7] \= $gene_names[7]\;\n";

exit;
```

VETORES ("ARRAYS")

Atribuição de valores individuais

Script: arrays.pl

Copiar e colar **exemplo06** da página

```
# exemplo05
# adicionando mais elementos no array
$gene_names[7] = "CG9773";
$gene_IDs[7]   = "FBgn0037609";

# exemplo06
# adicionando mais elementos no array
$gene_names[6] = "CG7059";
$gene_IDs[6]   = "FBgn0038957";

print "$gene_IDs[6] \= $gene_names[6]\;\n";
print "$gene_IDs[7] \= $gene_names[7]\;\n";

exit;
```

VETORES ("ARRAYS")

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo07** da página

1. Função push()

```
print "$gene_IDs[7] \= $gene_names[7]\;\n";

# exemplo07
# o que faz a funcao push()?
push(@gene_names, "RabX4");
push(@gene_IDs, "FBgn0051118");

print "\nExemplo07\:\n";
print "$gene_IDs[0] \= $gene_names[0]\;\n";
...
print "$gene_IDs[8] \= $gene_names[8]\;\n";

exit;
```

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo08** da página

2. Função pop()

```
# o que faz a funcao pop()?
pop(@gene_names);
pop(@gene_IDs);

print "\nExemplo10\:\n";
print "$gene_IDs[0] \= $gene_names[0]\;\n";
print "$gene_IDs[1] \= $gene_names[1]\;\n";
print "$gene_IDs[2] \= $gene_names[2]\;\n";
print "$gene_IDs[3] \= $gene_names[3]\;\n";
print "$gene_IDs[4] \= $gene_names[4]\;\n";
print "$gene_IDs[5] \= $gene_names[5]\;\n";
print "$gene_IDs[6] \= $gene_names[6]\;\n";
print "$gene_IDs[7] \= $gene_names[7]\;\n";
print "$gene_IDs[8] \= $gene_names[8]\;\n";

exit;
```

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo09** da página

3. Função shift()

```
# o que faz a funcao shift()?
$gene_name01 = shift(@gene_names);
$gene_id01   = shift(@gene_IDs);

print "\nExemplo09\:\n";
print "$gene_id01 \= $gene_name01\;\n";
print "Novo array\:\n";
print "$gene_IDs[0] \= $gene_names[0]\;\n";
print "$gene_IDs[1] \= $gene_names[1]\;\n";
print "$gene_IDs[2] \= $gene_names[2]\;\n";
print "$gene_IDs[3] \= $gene_names[3]\;\n";
print "$gene_IDs[4] \= $gene_names[4]\;\n";
print "$gene_IDs[5] \= $gene_names[5]\;\n";
print "$gene_IDs[6] \= $gene_names[6]\;\n";
print "$gene_IDs[7] \= $gene_names[7]\;\n";
print "$gene_IDs[8] \= $gene_names[8]\;\n";
```

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo10** da página

4. Função unshift()

```
# o que faz a funcao unshift()?
unshift(@gene_names, $gene_name01);
unshift(@gene_IDs, $gene_id01);

print "\nExemplo10\:\n";
print "$gene_IDs[0] \= $gene_names[0]\;\n";
print "$gene_IDs[1] \= $gene_names[1]\;\n";
print "$gene_IDs[2] \= $gene_names[2]\;\n";
print "$gene_IDs[3] \= $gene_names[3]\;\n";
print "$gene_IDs[4] \= $gene_names[4]\;\n";
print "$gene_IDs[5] \= $gene_names[5]\;\n";
print "$gene_IDs[6] \= $gene_names[6]\;\n";
print "$gene_IDs[7] \= $gene_names[7]\;\n";
print "$gene_IDs[8] \= $gene_names[8]\;\n\n";
```

VETORES ("ARRAYS")

Contexto de lista ou contexto escalar?

Script: arrays.pl

Atribuindo o array a uma variável escalar

```
# continuacao do script para entender arrays
# contexto lista ou escalar?

$n = @gene_names;
print "n \= $n!\n";




exit;
```

VETORES ("ARRAYS")

Contexto de lista ou contexto escalar?

Script: arrays.pl

Atribuindo o array a uma variável escalar

```
# continuacao do script para entender arrays
# contexto lista ou escalar?

$n = @gene_names;
print "n \= $n!\n";

print @gene_names;




exit;
```

VETORES ("ARRAYS")

Contexto de lista ou contexto escalar?

Script: arrays.pl

Atribuindo o array a uma variável escalar

```
# continuacao do script para entender arrays
# contexto lista ou escalar?

$n = @gene_names;
print "n \= $n!\n";

print @gene_names;

print "\n@gene_names\n";



exit;
```

VETORES ("ARRAYS")

Contexto de lista ou contexto escalar?

Script: arrays.pl

Atribuindo o array a uma variável escalar

```
# continuacao do script para entender arrays
# contexto lista ou escalar?

$n = @gene_names;
print "n \= $n!\n";

print @gene_names;

print "\n@gene_names\n";

print join(", ", @gene_names);

exit;
```

VETORES ("ARRAYS")

Valores sequenciais

Script: arrays.pl

Atribuindo valores sequenciais a um array

```
# continuacao do script para entender arrays
# valores sequenciais

@var_10 = (1..10);
print "@var_10\n";




exit;
```

VETORES ("ARRAYS")

Valores sequenciais

Script: arrays.pl

Atribuindo valores sequenciais a um array

```
# continuacao do script para entender arrays
# valores sequenciais

@var_10 = (1..10);
print "@var_10\n";

@var_20 = (10..20);
print "@var_20\n";




exit;
```

VETORES ("ARRAYS")

Valores sequenciais

Script: arrays.pl

Atribuindo valores sequenciais a um array

```
# continuacao do script para entender arrays
# valores sequenciais

@var_10 = (1..10);
print "@var_10\n";

@var_20 = (10..20);
print "@var_20\n";

@var_abc = (a..z);
print "@var_abc\n";

exit;
```

VETORES ("ARRAYS")

Subconjuntos

Script: arrays.pl

Selecionando subconjuntos em arrays

```
# continuacao do script para entender arrays
# subconjuntos

@subset_genes = @gene_names[2..4];

print "\n@subset_genes\n";



exit;
```

VETORES ("ARRAYS")

Subconjuntos

Script: arrays.pl

Selecionando subconjuntos em arrays

```
# continuacao do script para entender arrays
# subconjuntos

@subset_genes = @gene_names[2,4];

print "\n@subset_genes\n";



exit;
```

VETORES ("ARRAYS")

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo11** da página

5. Função splice()

```
# exemplo11
# funcao splice() para substituicao
# uso: splice(@array, inicio, tamanho, @lista_substituir)

@nums = (1..20);

print "\nExemplo11\:\n";
print "Antes - @nums\n";

splice(@nums,5 ,5 , 21..25);

print "Depois - @nums\n";

exit;
```

VETORES ("ARRAYS")

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo12** da página

5. Função split()

```
# funcao split()
# transformar string em array

# strings
$var_music="Rain-Drops-On-Roses-And-Whiskers-On-Kittens";
$var_Mando="This is the way";

# transformar strings em arrays
@music = split('-', $var_music);
@Mando  = split(' ', $var_Mando);

print "$music[5]\n";
print "$Mando[3]\n";
```

VETORES ("ARRAYS")

Funções para a manipulação de arrays

Script: arrays.pl

Copiar e colar **exemplo13** da página

5. Função join()

```
# funcao join()
# transformar array em string

$string1 = join( ' ', @music );
$string2 = join( '-', @Mando );

print "\nExemplo13\:\n";
print "$string1\n";
print "$string2\n";

exit;
```

VETORES ("ARRAYS")

Problema: criar uma agenda de endereços

@agenda_nomes

0	Ahsoka Tano
1	Sabine Wren
2	Hera Syndulla
3	Asajj Ventress
4	Rey
5	Leia Organa
6	Bo-Katan Kryze
7	Jyn Erso

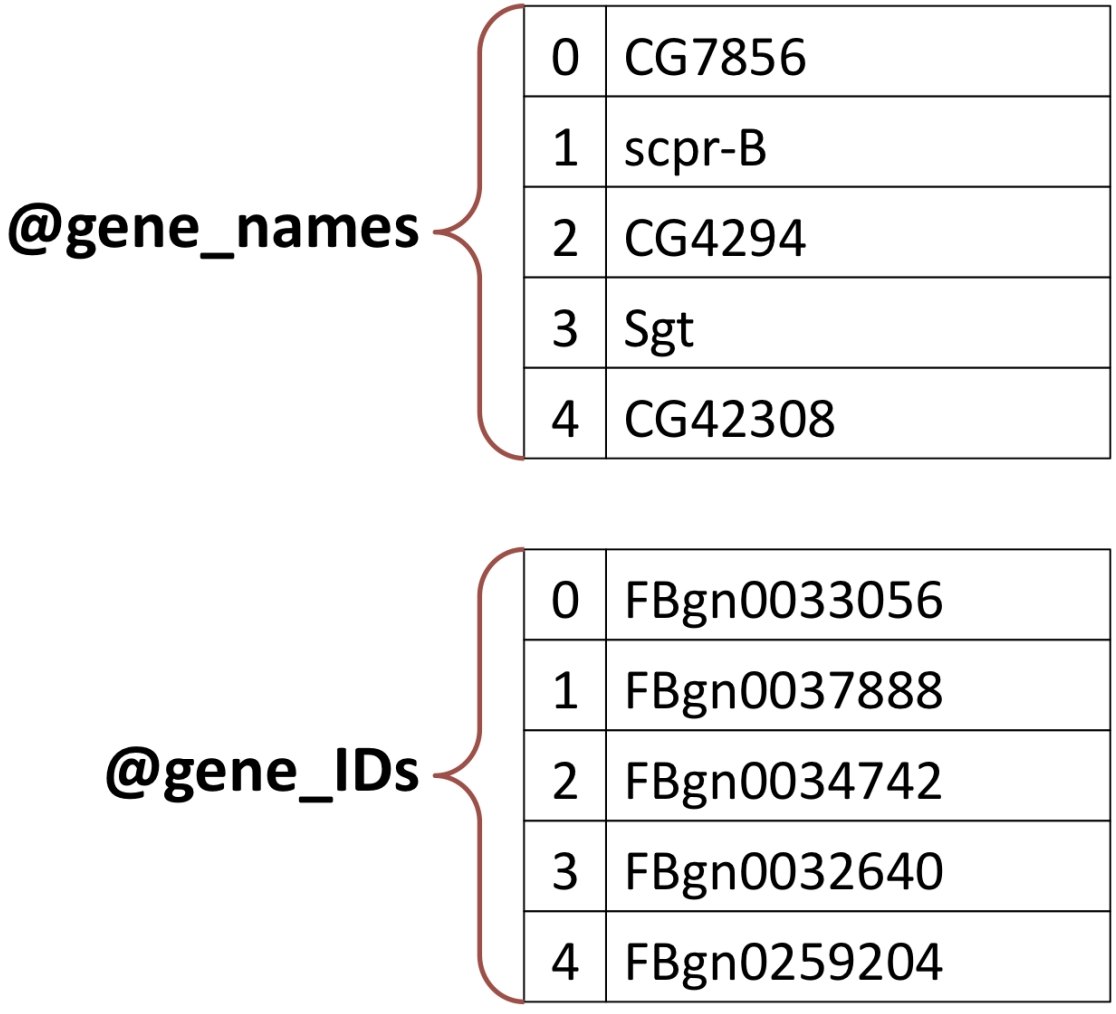
VETORES ("ARRAYS")

Problema: criar uma agenda de endereços

@agenda_enderecos

0	Shili
1	Mandalore
2	Ryloth
3	Dathomir
4	Jakku
5	Alderaan
6	Mandalore
7	Vallt

VETORES ("ARRAYS")



VETORES ASSOCIATIVOS (HASHES)

genes

CG7856	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCT
scpr-B	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCT
CG4294	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCA
Sgt	ATAGCGCTAGCTGAGCTAGCTGGTAGCTGAGCTGAGTATA
CG42308	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG15556	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG7059	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG9773	ATAGCGCTAGCTGAGCTAGCTGAGCTAGCTGAGCTGAGT

VETORES ASSOCIATIVOS (HASHES)

%genes

CG7856	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCT
scpr-B	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCT
CG4294	ATAGCGCTAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCA
Sgt	ATAGCGCTAGCTGAGCTAGCTGGTAGCTGAGCTGAGTATA
CG42308	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG15556	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG7059	ATAGCTGAGCTAGCTGAGCTGCGTAGCTGAGCTGAGTATA
CG9773	ATAGCGCTAGCTGAGCTAGCTGAGCTAGCTGAGCTGAGT

keys **value**

VETORES ASSOCIATIVOS (HASHES)

Na prática

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como hashes.pl
4. Copiar **exemplo14** da página da disciplina.

VETORES ASSOCIATIVOS (HASHES)

Na prática

Script: hashes.pl

```
# exemplo14
#! /usr/bin/perl
# script para entender hashes

# criando o hash de genes
%genes = ("FBgn0033056", "CG7856",
          "FBgn0037888", "scpr-B",
          "FBgn0034742", "CG424",
          "FBgn0032640", "Sgt",
          "FBgn0259204", "CG42308",
          "FBgn0039821", "CG15556",
          "FBgn0038957", "CG7059",
          "FBgn0037609", "CG9773");

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Na prática

Script: hashes.pl

```
# exemplo14
#! /usr/bin/perl
# script para entender hashes

# criando o hash de genes
%genes = ("FBgn0033056","CG7856","FBgn0037888","scpr-B",
"FBgn0034742","CG424","FBgn0032640","Sgt","FBgn0259204",
"CG42308","FBgn0039821","CG15556","FBgn0038957","CG7059",
"FBgn0037609","CG9773");




exit;
```

VETORES ASSOCIATIVOS (HASHES)

Na prática

Script: hashes.pl

```
# exemplo14
#! /usr/bin/perl
# script para entender hashes

# criando o hash de genes
%genes = ("FBgn0033056" => "CG7856",
          "FBgn0037888" => "scpr-B",
          "FBgn0034742" => "CG424",
          "FBgn0032640" => "Sgt",
          "FBgn0259204" => "CG42308",
          "FBgn0039821" => "CG15556",
          "FBgn0038957" => "CG7059",
          "FBgn0037609" => "CG9773");

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Acessando elementos individuais de um hash

Script: hashes.pl

```
# exemplo14
#! /usr/bin/perl
# script para entender hashes

# criando o hash de genes
%genes = ("FBgn0033056" => "CG7856",
          "FBgn0037888" => "scpr-B",
          "FBgn0034742" => "CG424",
          "FBgn0032640" => "Sgt",
          "FBgn0259204" => "CG42308",
          "FBgn0039821" => "CG15556",
          "FBgn0038957" => "CG7059",
          "FBgn0037609" => "CG9773");



exit;
```

VETORES ASSOCIATIVOS (HASHES)

Acessando elementos individuais de um hash

Script: hashes.pl

```
# criando o hash de genes
%genes = ("FBgn0033056" => "CG7856",
          "FBgn0037888" => "scpr-B",
          "FBgn0034742" => "CG424",
          "FBgn0032640" => "Sgt",
          "FBgn0259204" => "CG42308",
          "FBgn0039821" => "CG15556",
          "FBgn0038957" => "CG7059",
          "FBgn0037609" => "CG9773");

# acessando elementos individuais do hash
$gene01 = $genes{FBgn0033056};
$gene02 = $genes{FBgn0037888};

print "$gene01 e $gene02\n";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Criando uma nova entrada em um hash

Script: hashes.pl

```
# acessando elementos individuais do hash
$gene01 = $genes{FBgn0033056};
$gene02 = $genes{FBgn0037888};

print "$gene01 e $gene02\n";




exit;
```

VETORES ASSOCIATIVOS (HASHES)

Criando uma nova entrada em um hash

Script: hashes.pl

```
# acessando elementos individuais do hash
$gene01 = $genes{FBgn0033056};
$gene02 = $genes{FBgn0037888};

print "$gene01 e $gene02\n";

# criar uma nova entrada
$genes{"FBgn0051118"} = "RabX4";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de hashes

Script: hashes.pl

Recuperando as chaves dos hashes

1. Função keys()

```
print "$gene01 e $gene02\n";

# criar uma nova entrada
$genes{"FBgn0051118"} = "RabX4";

# recuperar os IDs
@gene_IDs = keys(%genes);



exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de hashes

Script: hashes.pl

Recuperando as chaves dos hashes

1. Função keys()

```
print "$gene01 e $gene02\n";

# criar uma nova entrada
$genes{"FBgn0051118"} = "RabX4";

# recuperar os IDs
@gene_IDs = keys(%genes);

print "@gene_IDs\n";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de hashes

Script: hashes.pl

Recuperando os valores dos hashes

2. Função values()

```
# recuperar os IDs
@gene_IDs = keys(%genes);

print "@gene_IDs\n";

# recuperar os valores
@gene_values = values(%genes);



exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de hashes

Script: hashes.pl

Recuperando os valores dos hashes

2. Função values()

```
# recuperar os IDs
@gene_IDs = keys(%genes);

print "@gene_IDs\n";

# recuperar os valores
@gene_values = values(%genes);

print "@gene_values\n";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de arrays/hashes

Script: hashes.pl

Ordenando arrays

3. Função sort()

```
# recuperar os valores
@gene_values = values(%genes);

print "@gene_values\n";

# recuperar os IDs em ordem alfabética



exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de arrays/hashes

Script: hashes.pl

Ordenando arrays

3. Função sort()

```
# recuperar os valores
@gene_values = values(%genes);

print "@gene_values\n";

# recuperar os IDs em ordem alfabética
@gene_IDs = sort(keys(%genes));

print "@gene_IDs\n";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

Funções para a manipulação de arrays/hashes

Script: hashes.pl

Ordenando arrays

4. Função reverse()

```
# recuperar os IDs em ordem alfabética
@gene_IDs = sort(keys(%genes));

print "@gene_IDs\n";

# o que faz a funcao reverse()
@gene_values_rev = reverse(@gene_values);

print "@gene_values_rev\n";

exit;
```

VETORES ASSOCIATIVOS (HASHES)

PROBLEMA

> Como encontrar valores específicos em um hash? Como imprimir elementos de um hash?