Decode Team

Instituto Politécnico de tomar

Portugol IDE

Equivalências de estruturas

Índice

[Perl 2](#_Toc357553159)

[Introdução 2](#_Toc357553160)

[Variáveis 2](#_Toc357553161)

[Tipos de variáveis 2](#_Toc357553162)

[Definição e atribuição de variáveis 3](#_Toc357553163)

[Estruturas input/output 4](#_Toc357553164)

[Input – Ler 4](#_Toc357553165)

[Output – Escrever 4](#_Toc357553166)

[Estrutura *Início* 5](#_Toc357553167)

[Estrutura *Fim* 5](#_Toc357553168)

[Estruturas de Decisão 5](#_Toc357553169)

[Condição if 5](#_Toc357553170)

[Condição if/else 6](#_Toc357553171)

[Condição else if 6](#_Toc357553172)

[Condição while 7](#_Toc357553173)

[Condição do while 7](#_Toc357553174)

[Funções 8](#_Toc357553175)

[Definir funções 8](#_Toc357553176)

[Chamada de funções 8](#_Toc357553177)

[Operadores 9](#_Toc357553178)

[Aritméticos 9](#_Toc357553179)

[Lógicos 9](#_Toc357553180)

[Relacionais 9](#_Toc357553181)

# Perl

## Introdução

Algumas notas sobre *Perl*:

* Software livre, lançado em 1987 por Larry Wall;
* Tem como lema: “There's More Than One Way To Do It"
* É uma das linguagens mais populares de programação web, devido às suas capacidades de manipulação de textos;
* Influenciada pelas linguagens shell script;
* Suporta vários tipos de base de dados (Oracle, Sybase, [PostgreSQL](http://programacao.livreforum.com/postgresql-f15/), [MySQL](http://programacao.livreforum.com/mysql-f14/), etc);
* Unicode;
* Multiplataforma;
* Usa o ponto e virgula (;) para terminar uma linha de código;
* As funções podem ser definidas antes ou depois do código a executar.

## Variáveis

### Tipos de variáveis

Linguagem *Perl* é considerada como não-tipada.   
Em *Perl*, as variáveis não ficam restringidas a ter um único tipo de dados, o interpretador irá automaticamente definir o seu tipo baseado no contexto em que a variável se associa.  
Não existe necessidade de pré-declarar o tipo da variável.  
Existem três estruturas básicas de dados:

* **Escalares:** assumem um valor único (número – inteiro ou real, string e referência);
* **Listas/Arrays:** conjunto ordenado de escalares;
* **Hashes:** conjunto indexado de escalares, não ordenados, que podem ser acedidos através de chaves.

### Definição e atribuição de variáveis

variável <- valor

Todos os nomes podem conter letras, o caracter *underscore* e números, desde que não se inicie com número.

* **Escalares**
  + Todas as variáveis são declaradas/iniciadas pelo caracter **$**  
    **Ex:**  
    $sum  
    $sumTotal  
    $sum\_1  
      
    Para atribuir dados à variável:
  + $nome\_variavel = valor;
  + O valorpode ser um número inteiro ou real, uma string ou uma referência.
  + **Ex:** $var = 1; $var = 1.5; $var = “Texto”;
* **Arrays**
  + Todas as variáveis são declaradas/iniciadas pelo caracter **@**
  + **Ex:**
  + @animais; @escolaSec; @escola\_1;
  + Para atribuir dados ao array:
  + @variavel =(valor1, valor2);
  + **Ex:**
  + @animais = (“gato”, “cao”, “porco”);
  + @numeros = (1, 2, 3, 4);
* **Hashes**
  + Float variavel =valorF;
  + **Nota**: *valor* é um decimal.
  + O tipo de dados float termina com F ou f

## Estruturas input/output

### Input – Ler

variavel

Para ler do teclado, teremos que atribuir a instrução <STDIN> à variável que vai estar associado o valor introduzido:  
  
$num = **<STDIN>**;

### Output – Escrever

mensagem

Para escrever no ecrã, é utilizada a instrução ***print.***Após escrever a mensagem, permanece na mesma linha, exceto se no final da instrução encontrar **\n**

**Ex:**

print “Nome”;  
print “Apelido;  
Teremos no ecrã:   
NomeApelido

Mas se colocar-mos:  
print “Nome\n”;  
print “Apelido”;  
Teremos:   
Nome  
Apelido

## Estrutura *Início*

Início

Não é necessário nem usada nenhuma instrução específica para indicar o começo do programa.

## Estrutura *Fim*

Fim

Não é necessário nem usada nenhuma instrução específica para indicar o final do programa.

## Estruturas de Decisão

### Condição if

true

condição

If (*condição*) {

Instruções;

}

**Nota**: A condição deve estar dentro de parêntesis

### Condição if/else

condição

false

true

If (*condição*)  
 {  
Instruções;  
}  
else  
 {  
Instruções;  
}

**Nota**: A condição deve estar dentro de parêntesis

### Condição else if

If (*condição1*)  
 {  
Instruções;  
}  
elsif(condição2)  
 {  
Instruções;  
}  
else  
{  
Instruções;  
}

**Nota**: A condição deve estar dentro de parêntesis

### Condição while

condição

false

instruções

true

While (*condição*){

Instruções

}

**Nota**: A condição deve estar dentro de parêntesis

### Condição do while

instruções

true

condição

false

do{

Instruções

}while(*condição*);

**Nota**: A condição deve estar dentro de parêntesis

## Funções

### Definir funções

As funções podem ser definidas antes ou após o código a executar. São habitualmente definidas por:

**sub** nomeFuncao{

}

Sempre que exista necessidade de se utilizar parâmetros, estes são definidos/lidos através da expressão **@\_** que é atribuído a uma variável. Exemplo:  
  
**sub** funcaoExemplo{  
 $numParametros = **@\_** ;

print “O numero de parâmetros é $numParametros \n”;  
}

### Chamada de funções

Aproveitando o exemplo da função acima declarada:  
  
funcaoExemplo(1); #Resultado: **O numero de parâmetros é 1**  
funcaoExemplo(1,2); #Resultado: **O numero de parâmetros é 2**  
funcaoExemplo(1..3); #Resultado: **O numero de parâmetros é 3**  
funcaoExemplo(“A”..”Z”); #Resultado: **O numero de parâmetros é 26**

## Operadores

### Aritméticos

* Adição
  + +
* Subtração
  + -
* Divisão
  + /
* Multiplicação
  + \*
* Resto da divisão inteira
  + %
* Potenciação
  + \*\*
* Concatenação de texto
  + +

### Lógicos

* Disjunção
  + &&
* Conjunção
  + ||
* Negação
  + !

### Relacionais

**Numéricos**  **Strings**

* Igual
  + == eq
* Diferente
  + != ne
* Maior
  + > gt
* Maior ou igual
  + >= ge
* Menor
  + < lt
* Menor ou igual

<= le