Algoritmos e Lógica de Programação

Lógica de Programação

Conjunto de técnicas para encadear pensamentos visando atingir um determinado objetivo.

- Identificar claramente as componentes do problemas
- Definir os passos para a solução
- Implementar/Executar a solução

Algoritmo

- Conjunto finito e bem definido de passos ou instruções precisas que devem ser executados em um sequência para se executar uma determinada tarefa.

- Não é a solução do problema, mas o conjunto de passos ou ações para que implementam ou conduzem à solução do problema.

Pseudocódigo

Pseudocódigo é uma forma intuitiva e informal de se descrever as ações e a sequência exata na qual o computador deve executá-la.

- Representação bastante útil para se organizar as ideias e estruturar o caminho para se solucionar um problema.

- Compiladores e interpretadores não são capazes de compreender pseudocódigos, logo, para converter um algoritmo em um programa computacional é necessário escrevê-lo, utilizando uma linguagem de programação cuja sintaxe e a semântica são rigidamente definidas.

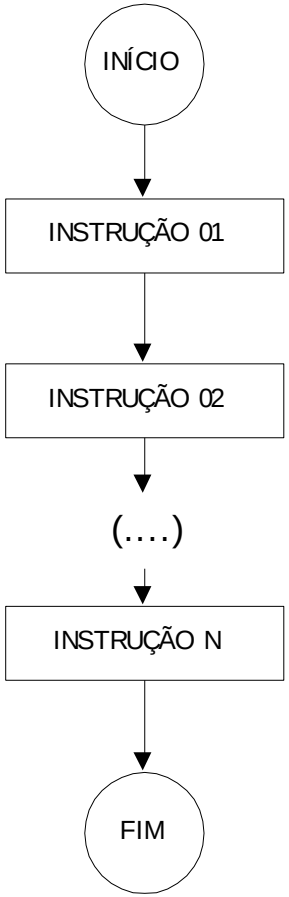
- Podem ser escritos em qualquer linguagem humana na forma de uma lista ordena ou numerada de instruções.

Representação Utilizando Fluxogramas

Pseudocódigo

```
INÍCIO
Instrução 01;
instrução 02;
....
instrução N;
FIM
```

Fluxograma



Expressões

Uma expressão uma é uma parte de uma sentença ou de uma instrução no código na qual um conjunto de variáveis que estão relacionadas por meio de operadores que uma vez avaliada gera um resultado.

Tipos de Expressões
As expressões podem ser classificadas em:

- Aritmética
- Relacional
- Lógica

Os operadores podem variar de acordo com linguagem. Quando se inicia o estudo de uma nova linguagem é conveniente conhecer logo de início os operadores básicos.

Nomes de Variáveis

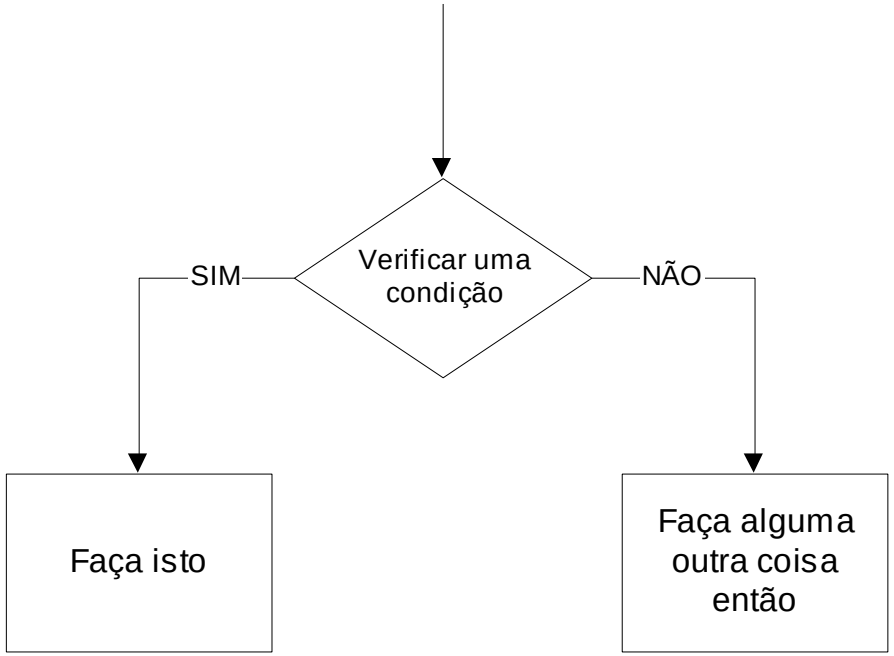
Para dar nomes às variáveis é necessário seguir algumas regras que podem variar de acordo com a linguagem e o estilo do programador, no entanto, algumas regras são básicas:

1. O primeiro caractere deve ser uma letra.
2. Se houver mais de um caractere só podem ser utilizadas letras, números e underline (“_”).
3. Apesar de permitido em algumas linguagens não utilizar acentuação ou caracteres especiais. Exemplo de nome não recomendável: variável_1 = 10;
4. Nomes de variáveis escritos com letras maiúsculas e minúsculas são variáveis distintas, assim, minhaVariavel é diferente de MinhaVariavel.
5. Não utilizar palavras reservadas da linguagem, ou seja, instruções ou códigos de uso exclusivo da linguagem.

Controle de Fluxo de Execução

Condicional: `if, if/else`

Representação gráfica do comando condicional simples:



Condicional: if, if/else

A sintaxe mais simples do comando condicional if:

```
if (<expressão lógica>){
        # Comandos executados se expressão lógica for verdadeira
        .
        .
        .
}
```

A sintaxe do comando condicional if/else:

```
if (<expressão lógica>) {
        # Comandos executados se expressão lógica for verdadeira
        .
        .
} else {
        # Comandos executados se expressão lógica for falsa
        .
        .
}
```

Condicional: Exemplos

Condicional simples `if`:

```
## ATENÇAO: Inicilizar as variáveis 'a` e 'b' na janela de comando

## Verifica se as variáveis são iguais
if (a == b){
  cat(sprintf('a=%f é IGUAL que b=%f\n',a,b));
}

## Verifica se 'a' é menor que 'b'
if (a < b) {
  cat(sprintf('a=%f é MENOR que b=%f\n',a,b));
}

## Verifica se 'a' é maior que 'b'
if (a > b) {
  cat(sprintf('a=%f é MAIOR que b=%f\n',a,b));
}
```

Condicional: Exemplos

Condicional simples `if/else:`

```
## ATENÇAO: Inicilizar as variáveis 'a` e 'b' na janela de comando

## Verifica se as variáveis são iguais
if (a == b) {
    cat(sprintf('a=%f IGUAL que b=%f\n',a,b));
} else {
    if (a < b){
        ## Verifica se 'a' é menor que 'b'
        cat(sprintf('a=%f MENOR que b=%f\n',a,b));
    } else {
        ## Verifica se 'a' é maior que 'b'
        cat(sprintf('a=%f MAIOR que b=%f\n',a,b));
    }
}
```

Estruturas de Repetição: “Enquanto” e “Para”

Considere o problema de calcular a soma dos N primeiros número inteiros.

Uma solução possível para o problema seria:

```
INÍCIO: N é um número inteiro fixo
S = 1
S = S + 2
S = S + 3
.
.
S = S + i
.
.
S = S + N
FIM
```

Problemas com este código:

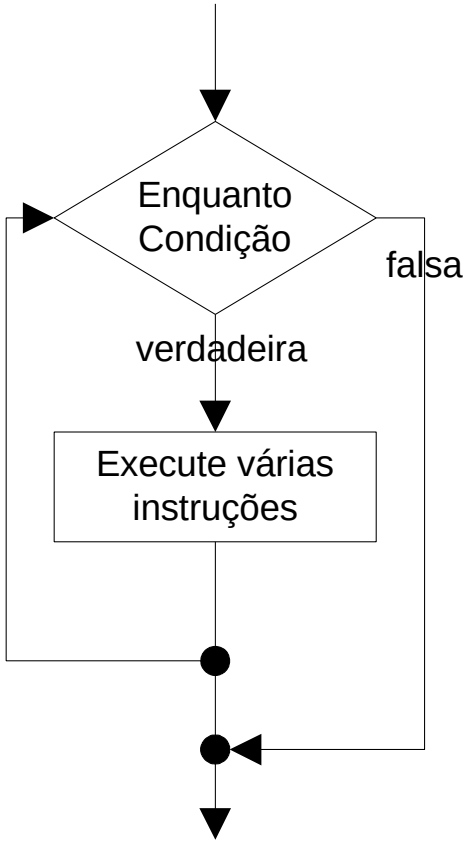
- Ele serve apenas para um único valor de N. Para cada valor de N teríamos que reescrever um código distinto.

- O número de instruções do código depende do valor de N, assim, o trabalho para escrever o código e a probabilidade de se cometer um erro aumenta consideravelmente a medida que N cresce.

Estrutura de Repetição: “Enquanto”

Executa um determinado conjunto de instruções *enquanto* uma condição predefinida for verdadeira.

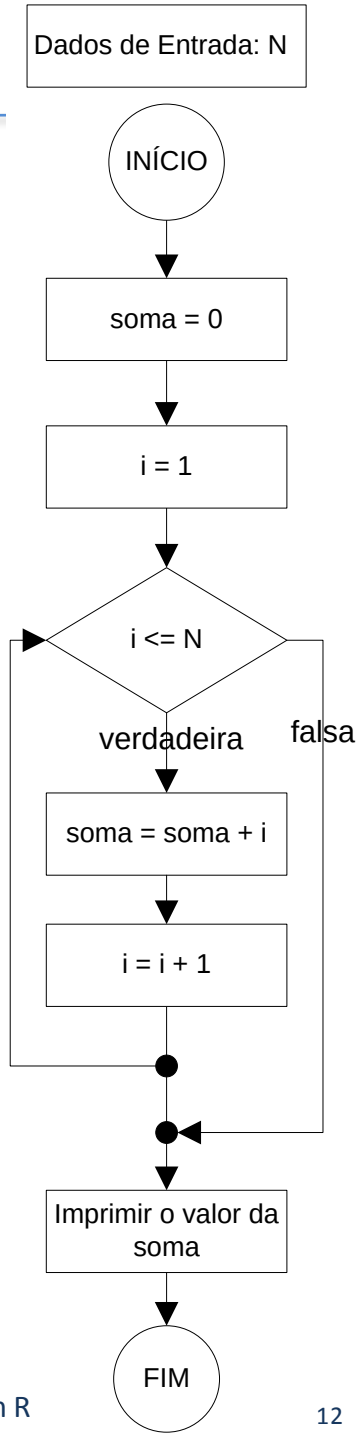
```
ENQUANTO CONDIÇÃO = VERDADEIRA FAÇA
        FAÇA ESTAS INSTRUÇÕES
FIM DO LOOP ENQUANTO
```



Estrutura de Repetição: “Enquanto”

Cálculo da soma dos *N* primeiros números inteiros

```
INÍCIO: N é um número inteiro fixo
i = 1
SOMA = 0
ENQUANTO (i <= N) FAÇA
        SOMA = SOMA + i
        i = i + 1
FIM DO LOOP ENQUANTO
IMPRIMIR SOMA
FIM
```



Observações:

- Código genérico para qualquer valor de *N*
- Variável auxiliar *i* é um contador
- Variáveis *SOMA* e *i* foram inicializadas com valores conhecidos
- O *looping* poderia não ser executado nenhuma vez caso a condição fosse falsa.

Estrutura de Repetição: “Enquanto”

Sintaxe do R

```
while (<expressão lógica>) {
        # Comandos executados enquanto expressão lógica for verdadeira
        .
        .
}
```

Exemplo: `WhileSomaN (N)`

```
WhileSomaN = function (N) {
  ## Inicializa as variaveis
  i = 1;
  soma = 0;
  ## Loop While
  while (i <= N) {
    ## Incrementa a soma
    soma = soma + i;
    ## Incrementa o contador
    i = i + 1;
  }
  ## Retorna a soma
  return(soma);
}
```

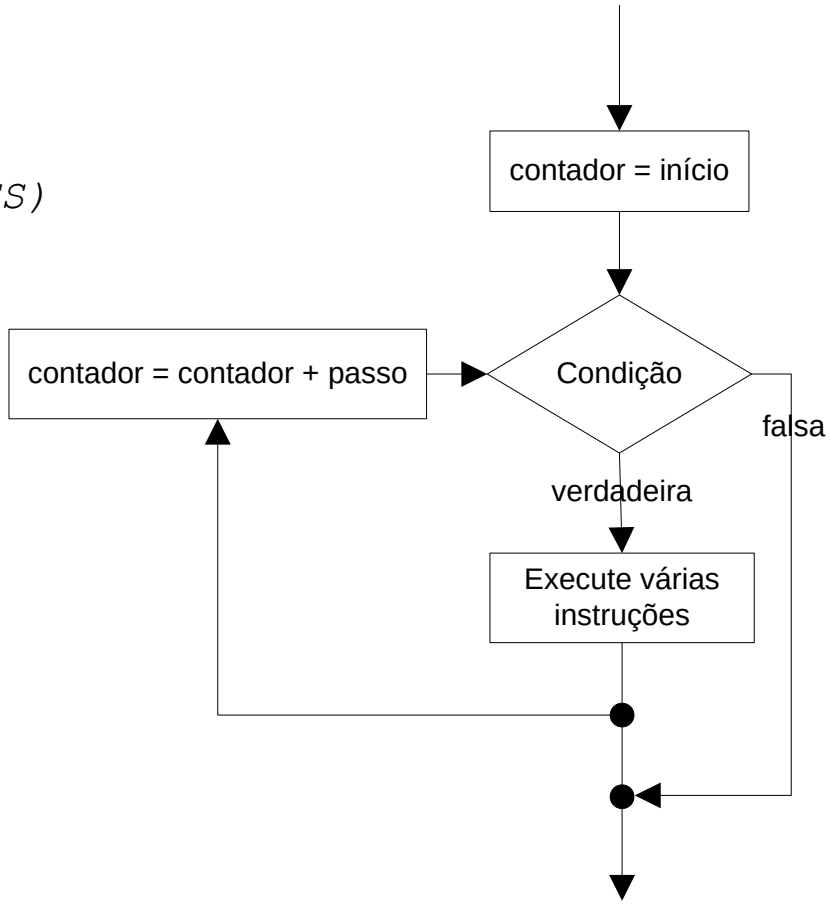
Estrutura de Repetição: “Faça Para”

Executa um determinado conjunto de instruções *para* um determinado número de vezes ou para uma lista de valores

```
FAÇA PARA (CONTADOR = INICIO; CONDIÇÃO; CONTADOR = CONTADOR + PASSO)
        FAÇA ESTAS INSTRUÇÕES
FIM LOOP PARA
```

ou

```
FAÇA PARA (CONTADOR EM LISTA_DE_VALORES)
        FAÇA ESTAS INSTRUÇÕES
FIM LOOP PARA
```



Estrutura de Repetição: “Faça Para”

Cálculo dos *N* primeiros números inteiros

```
INÍCIO: N é um número inteiro fixo
SOMA = 0
FAÇA PARA (i = 1; i<=N; i = i + 1)
        SOMA = SOMA + i
FIM LOOP PARA
IMPRIMIR SOMA
FIM
```

ou

```
INÍCIO: N é um número inteiro fixo
SOMA = 0
FAÇA PARA (i EM LISTA_DE_1:N)
        SOMA = SOMA + i
FIM LOOP PARA
IMPRIMIR SOMA
FIM
```

Estrutura de Repetição: “Faça Para”

Sintaxe do R

```
for (<contador> in <lista_de_valore>) {
  ## Comandos utilizando o i-ésimo elemento.
        .
        .
}
```

Obsevações:

- O valor do <contador> não pode ser alterado no interior do looping.

Exemplo: `ForSomaN (N)`

```
ForSomaN = function (N) {
  ## Inicializa as variaveis
  soma = 0;
    ## Loop for
  for (i in 1:N) {
    ## Incrementa a soma
    soma = soma + i;
  }
  ## Retorna a soma
  return(soma);
}
```

Programação Estruturada - Exercícios

1. Considere o problema de calcular as raízes de uma equação de segundo grau :

$$ax^2 + bx + c = 0 \qquad a \neq 0$$

Escrever a função:

`RaizesQuadratica(a, b, c)`

Argumentos:

`a`: vetor numérico com `n` elementos

`b`: vetor numérico com `n` elementos

`c`: vetor numérico com `n` elementos

Retorno:

`x`: Matriz `n x 2` com a solução da equação quadrática.

Duas raízes reais distintas `x[i,1]`= x_1 e `x[i,2]`= x_2 com $x_1 < x_2$

Uma única raiz real `x[i,1]` = x_1 e `x[i,2] = NaN`

Não existe raízes reais: `x[i,1] = x[i,2] = NaN`

(continua...)

Programação Estruturada - Exercícios

Testar para os seguintes exemplo:

$$f(x) = x^2 + 3x + 2$$

$$f(x) = x^2 - 4$$

$$f(x) = 3x^2 - 4x + 5$$

$$f(x) = 4x^2 + 16x$$

$$f(x) = -3x^2 + 27$$

$$f(x) = 9x^2 - 18x + 9$$

INPUT

```
a = c(1,1,3,4,-3,9);

b = c(3,0,-4,16,0,-18);

c = c(2,-4,5,0,27,9);
```

OUTPUT

```
     [,1] [,2]
[1,]   -2   -1
[2,]   -2    2
[3,]  NaN  NaN
[4,]   -4    0
[5,]   -3    3
[6,]    1  NaN
```

Programação Estruturada - Exercícios

Solução:

RaizesQuadratica = function(a, b, c){

Inicializa estrutura de retorno x com NaN

Para cada elemento dos vetores a, b, c

Calcula o delta;

se delta = 0 então

x_1 = -b/2a

senão

se delta > 0 então

se (a>0)

x_1 = (-b - raiz(delta))/2a

x_2 = (-b + raiz(delta))/2a

senão

x_1 = (-b + raiz(delta))/2a

x_2 = (-b - raiz(delta))/2ª

fim se

fim se

fim se

fim Para

Retorna x

fim da função

```
RaizesQuadratica = function(a, b, c){

  ## Estrutura de retorno
  x = array(NaN,c(length(a),2));

  ## Looping principal
  for (i in 1:length(a)) {

    ## Calcula o delta
    delta = b[i]^2 - 4*a[i]*c[i];

    ## Calcula as raízes
    if (delta==0){
      x[i,1] = -b[i]/(2*a[i]);
    } else{
      if (delta > 0) {
        if (a[i]>0){
          x[i,1] = (-b[i] - delta^(1/2))/(2*a[i]);
          x[i,2] = (-b[i] + delta^(1/2))/(2*a[i]);
        } else {
          x[i,1] = (-b[i] + delta^(1/2))/(2*a[i]);
          x[i,2] = (-b[i] - delta^(1/2))/(2*a[i]);
        }
      }
    }
  }
  return(x);
}
```

Programação Estruturada - Exercícios

2. Escrever o pseudocódigo e implementar um script que calcula a média e a variância populacional para o seguinte vetor de dados:

x = c(4.15, 3.05, 8.74, 0.15, 7.67, 9.70, 9.90, 7.88, 4.38, 4.98);

3. Escrever o pseudocódigo e implementar um script que calcula a média e a variância populacional para cada uma das colunas da matriz:

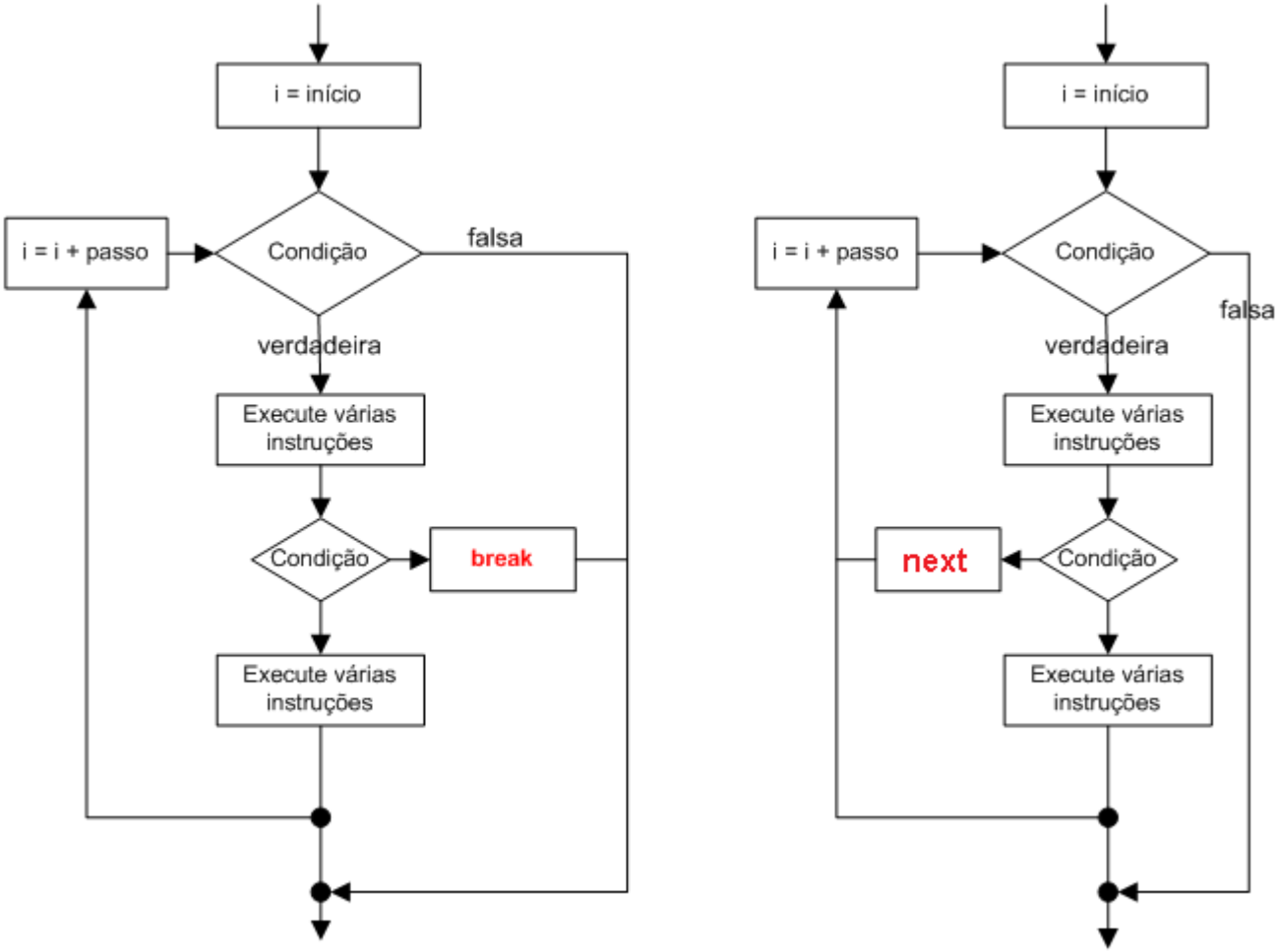
$$m = \begin{bmatrix} 3.8 & 6.0 & 0.57 \\ 6.8 & 0.1 & 3.67 \\ 0.9 & 0.1 & 6.31 \\ 0.3 & 1.9 & 7.17 \\ 6.1 & 5.8 & 6.92 \end{bmatrix}$$

m = array(c(3.8, 6.0, 0.5, 6.8, 0.1, 3.6, 0.9, 0.1, 6.3, 0.3, 1.9, 7.1, 6.1, 5.8, 6.9), c(5,3));

Estrutura de Repetição: *break* e *next*

`break`: interrompe as iterações do looping, caso ainda restem iterações, as mesmas não serão executadas.

`next`: fluxo de execução retorne para o início do looping e ignore os comandos abaixo.



Estrutura de Repetição: *break* e *next*

```
## Script: Exemplo de utilização dos comando break e next

for (i in 1:20){

  if (i%%5==0) {
    cat(sprintf("next\n"));
    next;
  }

  cat(sprintf("i=%.0f\n",i));

  if (i==17) {
    cat(sprintf("break\n"));
    break;
  }

}

cat("done!\n");
```

Estrutura de Repetição: “Repetir-Parar”

Repete um determinado conjunto de instruções até que uma condição predefinida for verdadeira.

```
REPETIR
   FAÇA ESTAS INSTRUÇÕES
   SE(CONDIÇÃO = VERDADEIRA) PARAR
FIM DO LOOP
```

Cálculo da soma dos *N* primeiros números inteiros

```
INÍCIO: N é um número inteiro fixo
i = 1
SOMA = 0
REPETIR
        SOMA = SOMA + i
        i = i + 1
        SE (i > N) PARAR
FIM DO LOOP
IMPRIMIR SOMA
FIM
```

Estrutura de Repetição: “Repetir-Para”

Sintaxe do R

```
repeat {
        # Comandos executados enquanto expressão lógica for verdadeira
        .
        .
        if (<expressão lógica>) break;
}
```

Exemplo

```
RepeatSomaN = function (N) {
  ## Inicializa as variaveis
  i = 1;
  soma = 0;
  ## Loop repeat
  repeat {
    ## Incrementa a soma
    soma = soma + i;
    ## Incrementa o contador
    i = i + 1;
    if (i>N) break;
  }
  ## Retorna a soma
  return(soma);
}
```

Estrutura de Repetição - Exercícios

1. Escrever o pseudocódigo e implementar um script que calcula a média e a variância populacional para o seguinte vetor de dados:

x = c(4.15, 3.05, 8.74, 0.15, 7.67, 9.70, 9.90, 7.88, 4.38, 4.98);

2. Escrever o pseudocódigo e implementar um script que calcula a média e a variância populacional para cada uma das colunas da matriz:

$$m = \begin{bmatrix} 3.8 & 6.0 & 0.57 \\ 6.8 & 0.1 & 3.67 \\ 0.9 & 0.1 & 6.31 \\ 0.3 & 1.9 & 7.17 \\ 6.1 & 5.8 & 6.92 \end{bmatrix}$$

m = array(c(3.8, 6.0, 0.5, 6.8, 0.1, 3.6, 0.9, 0.1, 6.3, 0.3, 1.9, 7.1, 6.1, 5.8, 6.9), c(5,3));