INTRODUÇÃO À COMPUTAÇÃO

UD II - ALGORITMOS

ESTRUTURAS DE REPETIÇÃO

Estruturas de repetição

ELEMENTOS DE COMPETÊNCIA

- Empregar recursos para operar em ambientes humanizados, integrando as dimensões física, humana e informacional deste ambiente operacional.
- Tomar decisões e conduzir ações, em situações de crise.

UD II - INTRODUÇÃO À ALGORITMOS

Estruturas de Repetição

🎯 OBJETIVOS

1. Descrever o conceito de iteração. (FACTUAL/CONCEITUAL)
2. Compreender a aplicação da iteração na construção de algoritmos. (CONCEITUAL)
3. Aplicar as estruturas de repetição na elaboração de algoritmos. (PROCEDIMENTAL)

Estruturas de Repetição

ATITUDES

1. Organização: capacidade de desenvolver atividades de forma sistemática e eficiente.
2. Dedicação: agir, realizando espontaneamente, com empenho e entusiasmo, as atividades necessárias ao cumprimento da missão.
3. Responsabilidade: capacidade de cumprir suas atribuições assumindo e enfrentando as consequências de suas atitudes e decisões.

Estruturas de Repetição

- Estruturas de repetição (laços de repetição ou loops) possibilitam a repetição de alguma parte do algoritmo, tantas vezes quantas forem necessárias, dispensando a necessidade de reescrever comandos ou instruções.
- A existência das estruturas de repetição reduz o tamanho do algoritmo seja reduzido, facilitando sua leitura e entendimento.
- A cada execução do bloco de instruções contidas dentro dos laços de repetição é dado o nome de iteração.

Tipos de Estruturas de Repetição

Laço Contado

Estas estruturas são utilizadas quando já se sabe de antemão o número de vezes que o trecho do código deverá ser repetido.

```
Para <variável de controle> de <início*1> até <fim*1> passo <valor*1> faça
        <instrução/bloco de instruções)>
FimPara
```

Observações:

- *(*1) Valores numéricos.*
- *A <variável de controle> também deve ser declarada na área de variáveis, do tipo inteiro.*

Laço Contado

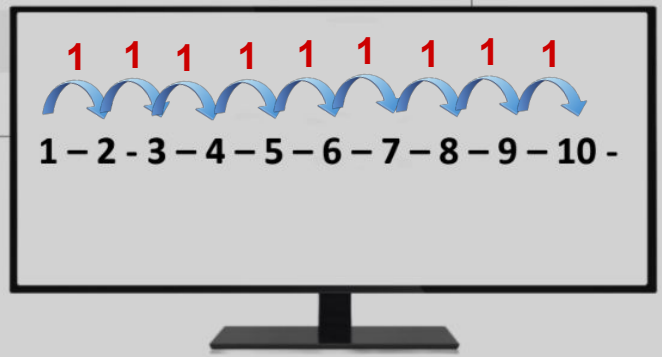

```
Para <variável de controle> de <início> até <fim> passo <valor> faça
        <instrução/bloco de instruções)>
FimPara
```

Parâmetros:

- <início> é o valor inicial da variável de controle
- <valor> incrementado a cada iteração. Se omitido assume o valor 1
- <fim> valor limite de repetições

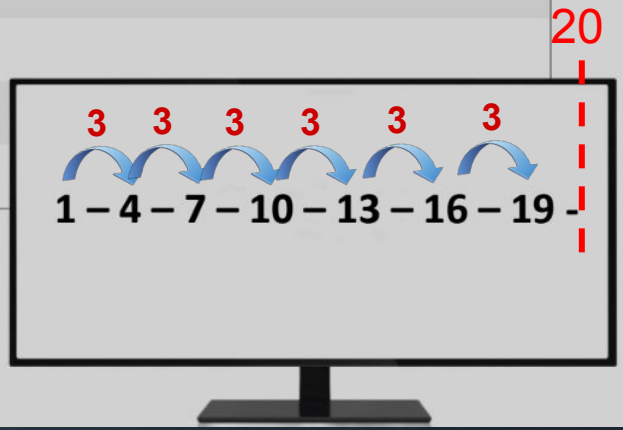
Laço Contado


```
1  Algoritmo "07_01 Uso de Laço Contado"
2  // Algoritmo que demonstra a utilização de um laço de repetição contado ,
3  // evidenciando o uso de uma variável de controle.
4  Var
5      contador: inteiro
6  Inicio
7      Para contador de 1 até 10 passo 1 faça
8          Escreva (contador , " - ")
9      Fimpara
10 FimAlgoritmo
```



Laço Contado


```
1  Algoritmo "07_02 Uso Laço Contado Passo 3"
2  // Algoritmo que demonstra a utilização de um laço de repetição contado ,
3  // evidenciando o passo diferente de 1, que é o valor padrão
4  Var
5      contador: inteiro
6  Inicio
7      Para contador de 1 até 20 passo 3 faça
8          Escreva (contador , " - ")
9      Fimpara
10 FimAlgoritmo
```



Laço Contado

ALGORITMO NO VISUALG

```
1  Algoritmo "07_03 Média5"
2  // Algoritmo que calcula a média dos alunos a partir das notas dos dois semestres
3  // e apresenta a mensagem de aprovação (nota>=5) ou reprovação(nota<5)
4  Var
5      nota1, nota2, media: real
6      num_alunos: inteiro
7  Inicio
8      Para num_alunos de 1 até 10 passo 1 faça
9          Escreva ("Digite a nota do primeiro semestre: ")
10         Leia (nota1)
11         Escreva ("Digite a nota do segundo semestre: ")
12         Leia (nota2)
13         media ← (nota1 + nota2)/2
14         Escreva ("A média é: ",media)
15         Se media >= 5 então
16                 Escreva ("Aluno aprovado.")
17         Senão
18                 Escreva ("Aluno reprovado.")
19         Fimse
20     Fimpara
21 FimAlgoritmo
```

Todo este bloco de instruções será repetido **10** vezes

Laço Contado

ALGORITMO
NO VISUALG

```
1  Algoritmo "07_04 Média6"
2  // Melhora algoritmo anterior, possibilitando indicar quantidade de alunos da classe
3  Var
4      nota1, nota2, media: real
5      Qtd, num_alunos: inteiro
6  Inicio
7      Escreva ("Digite a quantidade de alunos da classe: ")
8      Leia(qtd)
9      Para num_alunos de 1 até qtd passo 1 faça
10         Escreva ("Digite a nota do primeiro semestre: ")
11         Leia (nota1)
12         Escreva ("Digite a nota do segundo semestre: ")
13         Leia (nota2)
14         media ← (nota1 + nota2)/2
15         Escreva ("A média é: ",media)
16         Se media >= 5 então
17                 Escreva ("Aluno aprovado.")
18         Senão
19                 Escreva ("Aluno reprovado.")
20         Fimse
21     Fimpara
22 FimAlgoritmo
```

O valor final pode ser um valor numérico inteiro ou uma variável que armazene um valor numérico inteiro

Laço Condicional

Estas estruturas são mais vocacionadas para os casos em que o número de repetições a ser executado não é conhecido.

Apesar disso, elas podem ser usadas também nas situações em que o número de vezes que um trecho de código deverá ser repetido já é conhecido.

Estrutura Enquanto

Caracteriza-se pela realização do teste lógico no início do laço de repetição. Se o resultado do teste lógico (condição) for verdadeiro, as instruções que se encontram no interior do laço são executadas; caso contrário, o laço não será executado, ou seja, o programa passa para a próxima instrução após a estrutura.

Sintaxe:

```
Enquanto <condição> faça
        | < instrução/bloco de instruções) >
FimEnquanto
```

EXEMPLO DE ENQUANTO

ALGORITMO NO VISUALG

```
1  Algoritmo "07_05 Media7"
2  // Algoritmo que calcula e exibe a média dos alunos a partir das notas dos
3  // dois semestres usando laço condicional enquanto.
4  Var
5     nota1, nota2, media: real
6     resp: inteiro
7  Inicio
8     resp ← 1
9     Enquanto resp = 1 faça
10       Escreva ("Digite a nota do primeiro semestre: ")
11       Leia (nota1)
12       Escreva ("Digite a nota do segundo semestre: ")
13       Leia (nota2)
14       media ← (nota1 + nota2)/2
15       Escreval ("A média é: ",media)
16       Escreval ("Digite 1 para continuar ou qualquer outro número para sair: ")
17       Leia (resp)
18    Fimenquanto
19 FimAlgoritmo
```

Enquanto a variável **resp** contiver o valor 1, o bloco de instruções dentro do laço de repetição será executado.

Estrutura Repita

A estrutura Repita se diferencia por realizar o teste lógico apenas no final do laço e, por isso, os comandos internos são executados pelo menos uma vez.

Sua lógica de parada é inversa à da estrutura enquanto, pois o laço continua sendo executado enquanto a condição for falsa e só é interrompido quando a condição se tornar verdadeira.

Sintaxe:

```
Repita
    |   < instrução/bloco de instruções) >
Até < condição >
```

```
1  Algoritmo "07_06 Média8"
2  // Algoritmo que calcula e exibe a média dos alunos a partir das notas dos
3  // dois semestres usando laço condicional repita.
4  Var
5     nota1, nota2, media: real
6     resp: inteiro
7  Inicio
8     Repita
9        Escreva ("Digite a nota do primeiro semestre: ")
10       Leia (nota1)
11       Escreva ("Digite a nota do segundo semestre: ")
12       Leia (nota2)
13       media ← (nota1 + nota2)/2
14       Escreval ("A média é: ",media)
15       Escreval ("Digite 1 para continuar ou outro número para sair: ")
16       Leia (resp)
17    Até resp <> 1
18 FimAlgoritmo
```

Todo este bloco de instruções será repetido até que a condição contida na clausula "até" seja verdadeira.

Avalia se a condição foi satisfeita para encerrar as repetições. Se não for, executa mais uma vez o bloco de instruções dentro do laço de repetição.

Comando: Interrompa

Atua como uma saída controlada dentro das estruturas de repetição (Para, Enquanto ou Repita). O comando interrompa aborta imediatamente a execução do laço de repetição atual, sendo que o fluxo de execução do programa salta instantaneamente para a primeira linha de código após o fim do laço. Será ignorado qualquer código dentro do bloco de repetição que estiver abaixo do comando Interrompa. Será ignorada, também, as verificações de condição que ainda faltavam para terminar o laço de forma normal.

➡ *Otimização de desempenho e a simplificação da lógica*

```
7  Inicio
8      EhPrimo ← Verdadeiro          // todo número é primo até prova em contrário
9      Escreva("Digite um número para checar se é primo:")
```

```
1  Algoritmo "07_07b Demonstra comando Interrompa"
2  // Algoritmo que demonstra o uso do comando Interrompa
3  // Empregado dentro de um algoritmo de verificação de número primo
4  Var
5      numero, contador, resto: inteiro
6      EhPrimo: logico
```

```
16         FimSe
17     FimPara
18     Se EhPrimo = Verdadeiro então
19         Escreva(numero, " é primo")
20     Senão
21         Escreva(número, " não é primo")
22     FimSe
23 FimAlgoritmo
```

ALGORITMOS NO VISUALG

Compare esse algoritmo 07_07b com o 07_07a, que não utiliza o interrompa e veja a diferença de tempo de execução.

Tente o número 1.000.005

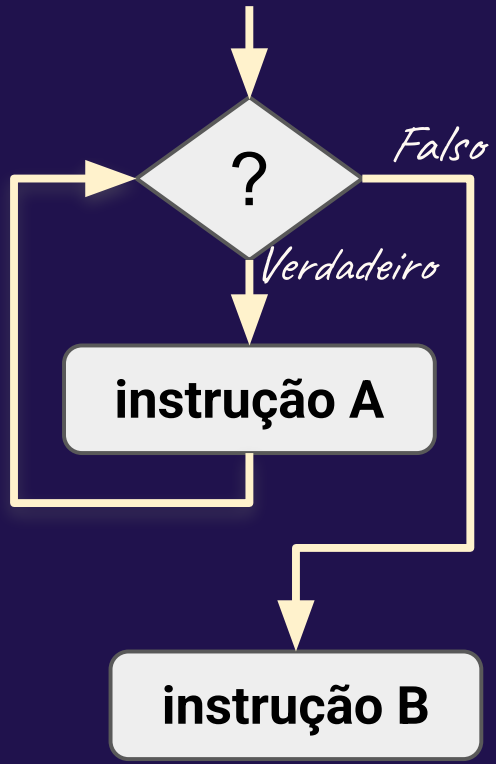
Comando: Interrompa

Atua como uma saída controlada dentro das estruturas de repetição (Para, Enquanto ou Repita). O comando interrompa aborta imediatamente a execução do laço de repetição atual, sendo que o fluxo de execução do programa salta instantaneamente para a primeira linha de código após o fim do laço. Será ignorado qualquer código dentro do bloco de repetição que estiver abaixo do comando Interrompa. Será ignorada, também, as verificações de condição que ainda faltavam para terminar o laço de forma normal.

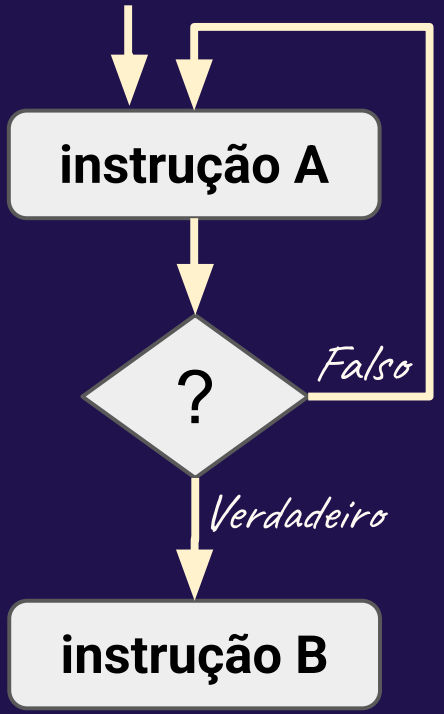
Uso: performance e simplificação do código

Condição de saída nos laços condicionais

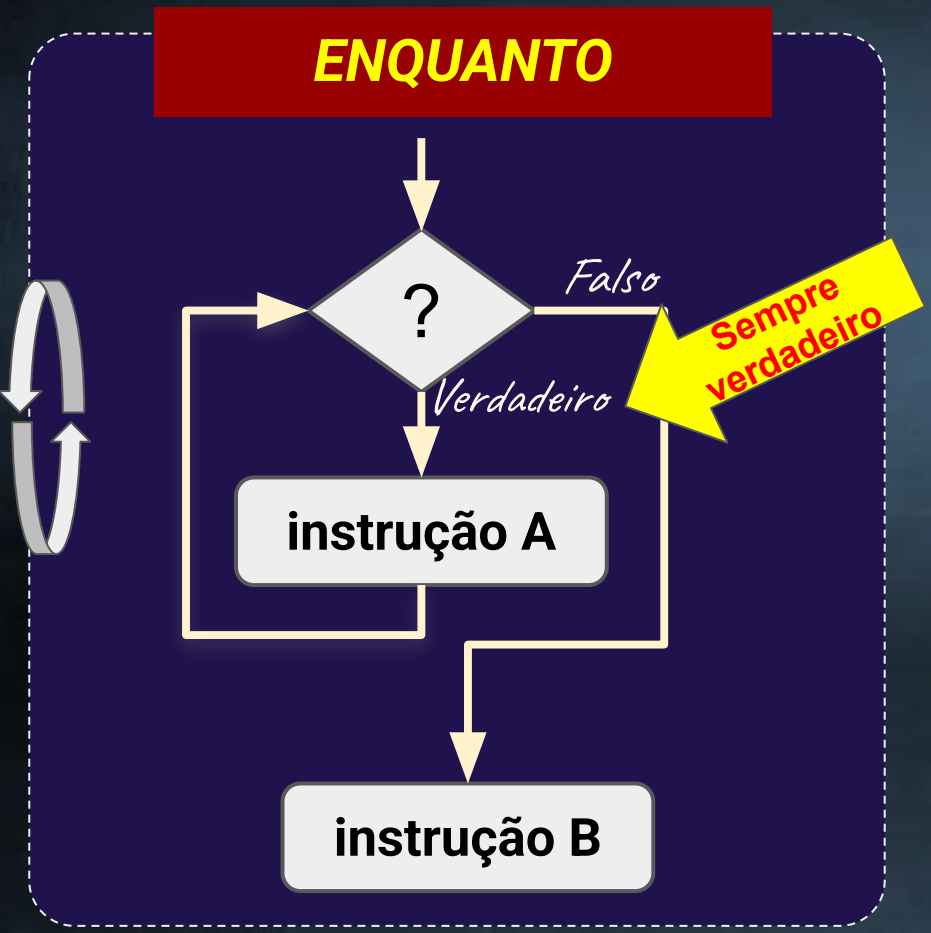
ENQUANTO

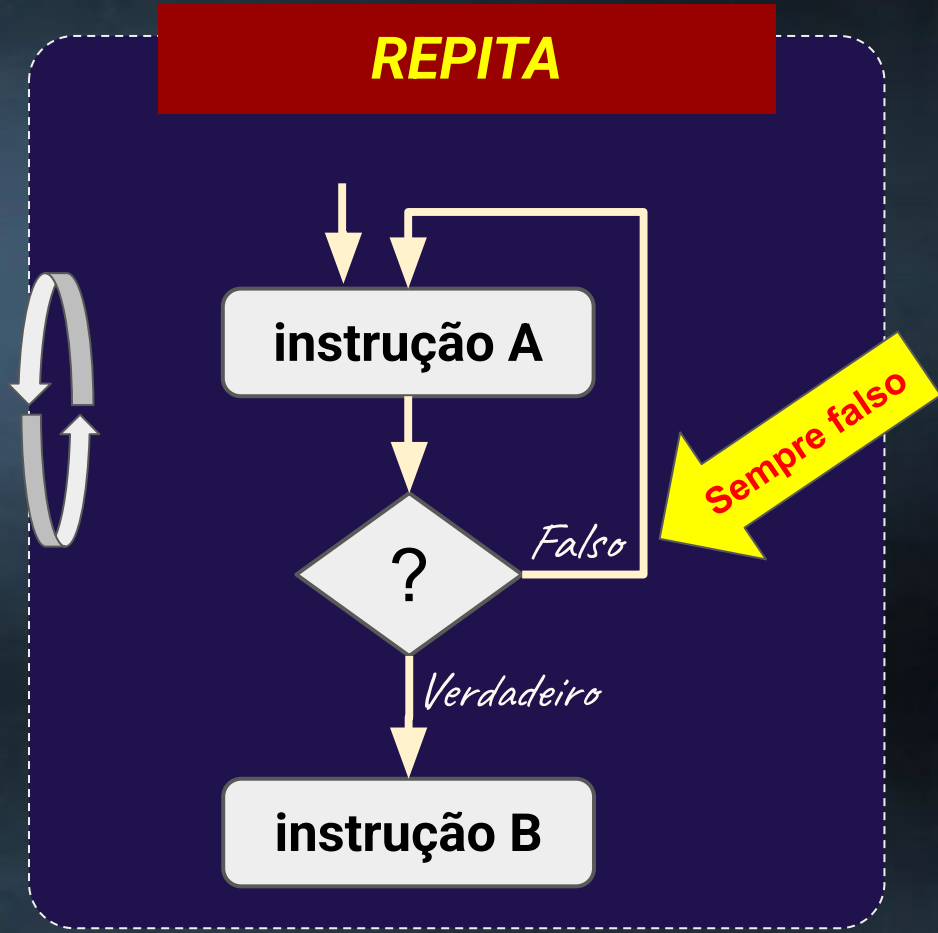


REPITA



Looping Infinito





Validação dos valores de entrada

A validação de entrada de dados é uma técnica essencial para garantir a consistência de um algoritmo, prevenindo que informações incorretas ou fora do escopo (como uma idade negativa ou uma nota acima de 10) causem erros no processamento subsequente.

O princípio é simples: não se deve confiar cegamente no que o usuário digita, mas sim verificar a consistência da informação no momento da entrada do dado.

Validação dos valores de entrada

Para implementar essa checagem, podem ser utilizados laços de repetição (tipicamente o enquanto ou repita). A lógica funciona como uma "barreira de contenção": se o valor digitado for inválido, o laço "prende" o fluxo de execução e solicita o dado novamente, repetindo o processo quantas vezes forem necessárias até que o usuário forneça uma entrada que satisfaça as regras estabelecidas.

```
1  Algoritmo "07_08 Validação de Entrada"
2  // Algoritmo que demonstra o uso de laços de repetição para validar, de forma
3  // persistente, a entrada de valores em um programa.
4  Var
5    nota : real
6  Inicio
7    escreva("Digite a nota do aluno (0 a 10): ")
8    leia(nota)
9    // A condição do laço verifica valores inconsistentes:
10   // Enquanto a nota for menor que 0 ou maior que 10, repete a pergunta.
11   enquanto (nota < 0) ou (nota > 10) faca
12       escreval("Erro: Valor inválido! A nota deve ser entre 0 e 10.")
13       escreva("Tente novamente: ")
14       leia(nota)
15   fimenquanto
16   escreval("Nota válida registrada: ", nota)
17 FimAlgoritmo
```

Exercícios em sala 1

Faça um algoritmo que mostre os quadrados perfeitos que tenham como base números entre 1 e 10.

```
--- Quadrados Perfeitos (Bases de 1 a 10) ---
Base  1 -> Quadrado:  1
Base  2 -> Quadrado:  4
Base  3 -> Quadrado:  9
Base  4 -> Quadrado:  16
Base  5 -> Quadrado:  25
Base  6 -> Quadrado:  36
Base  7 -> Quadrado:  49
Base  8 -> Quadrado:  64
Base  9 -> Quadrado:  81
Base  10 -> Quadrado:  100

>>> Fim da execução do programa !
```

Exercícios em sala 1 - Solução

```
1  algoritmo "QuadradosPerfeitos"
2  var
3     i, quadrado: inteiro
4  inicio
5     escreval("--- Quadrados Perfeitos (Bases de 1 a 10) ---")
6     // O laço 'para' percorre os números de 1 até 10
7     para i de 1 ate 10 faca
8        // Calcula o quadrado multiplicando a base por ela mesma
9        quadrado <- i * i
10       escreval("Base ", i, " -> Quadrado: ", quadrado)
11    fimpara
12 fimalgoritmo
```

Exercícios em sala 2

Faça um algoritmo que mostre os quadrados perfeitos que tenham como base um intervalo de números informado pelo usuário. Dica: modifique o programa anterior para atender a estes novos requisitos.

```
Digite o valor inicial do intervalo: 3
Digite o valor final do intervalo: 7
Quadrados Perfeitos (Bases de  3 a  7)
Base  3 -> Quadrado:  9
Base  4 -> Quadrado:  16
Base  5 -> Quadrado:  25
Base  6 -> Quadrado:  36
Base  7 -> Quadrado:  49

>>> Fim da execução do programa !
```

Ex 2

```
1  algoritmo "QuadradosPerfeitosIntervalo"
2  Var
3    valorInicial, valorFinal, i, quadrado: inteiro
4  Inicio
5    // Coleta dos dados que definirão os limites do laço
6    escreva("Digite o valor inicial do intervalo: ")
7    leia(valorInicial)
8    escreva("Digite o valor final do intervalo: ")
9    leia(valorFinal)
10   escreval("Quadrados Perfeitos (Bases de ", valorInicial, " a ", valorFinal, ")")
11   // O laço 'para' agora utiliza as variáveis informadas pelo usuário
12   para i de valorInicial ate valorFinal faca
13     quadrado <- i * i
14     escreval("Base ", i, " -> Quadrado: ", quadrado)
15   Fimpara
16 Fimalgoritmo
```

Exercícios em sala 3

Elabore um algoritmo que permita que um usuário informe vários números inteiros. Para cada número informar se é par ou ímpar. Se informar o número -1 o algoritmo é encerrado.

```
Digite outro número (ou -1 para sair): 3
Resultado: O número  3 é ÍMPAR.

Digite outro número (ou -1 para sair): 0
Resultado: O número  0 é PAR.

Digite outro número (ou -1 para sair): -1
Programa encerrado pelo usuário.

>>> Fim da execução do programa !
```

Ex 3

```
1  algoritmo "VerificadorParImpar"
2  var
3      numero: inteiro
4  inicio
5      // Primeira leitura antes do laço
6      escreva("Digite um número inteiro (-1 encerra): ")
7      leia(numero)
8      // O laço testa se a condição de parada foi ativada
9      enquanto numero <> -1 faca
10         // Se o resto da divisão por 2 for zero, é par.
11         se numero % 2 = 0 entao
12             escreval("Resultado: O número ", numero, " é PAR.")
13         senao
14             escreval("Resultado: O número ", numero, " é ÍMPAR.")
15         fimse
16         // Nova leitura no final do laço para a próxima repetição
17         escreva("Digite outro número (ou -1 para sair): ")
18         leia(numero)
19     fimenquanto
20   escreval("Programa encerrado pelo usuário.")
21 fimalgoritmo
```

Exercícios em sala 4

Elabore um algoritmo que solicite uma senha até que seja digitada uma senha alfanumérica correta (você vai definir essa senha no código). Utilize a estrutura repita na solução do problema.

```
--- Sistema de Segurança ---
Digite a senha de acesso: 4567
Erro: Senha incorreta! Tente novamente.

Digite a senha de acesso: 1234
Acesso liberado! Bem-vindo ao sistema.

>>> Fim da execução do programa !
```

Ex 4

```
1  algoritmo "ValidacaoDeSenha"
2  var
3    senhaDigitada: caractere
4  inicio
5    escreval("--- Sistema de Segurança ---")
6    // O laço inicia sem testar nenhuma condição
7    repita
8        escreva("Digite a senha de acesso: ")
9        leia(senhaDigitada)
10       se senhaDigitada <> "1234" entao
11           escreval("Erro: Senha incorreta! Tente novamente.")
12           escreval("")
13       fimse
14       // O laço repete ATÉ que a condição seja Verdadeira.
15   ate senhaDigitada = "1234"
16   escreval("Acesso liberado! Bem-vindo ao sistema.")
17 fimalgoritmo
```

Exercícios em sala 5 (Lista 7.7)

Construa um algoritmo que leia números inteiros até que seja inserido um número negativo. Ao final, o algoritmo deverá informar a média dos números digitados, o maior e o menor valor inserido.

```
Digite o 1° número (-1 encerra): 3
Digite o próximo número (-1 encerra): 4
Digite o próximo número (-1 encerra): 6
Digite o próximo número (-1 encerra): 8
Digite o próximo número (-1 encerra): -1
-------------------------
Quantidade de números digitados:  4
Média dos números:  5.25
Maior valor inserido:  8
Menor valor inserido:  3

>>> Fim da execução do programa !
```

Ex 5

```
1  algoritmo "EstatisticasDeNumeros"
2  var
3     numero, soma, contador, maior, menor: inteiro
4     media: real
5  inicio
6     soma <- 0
7     contador <- 0
8
9     escreva("Digite o 1° número (-1 encerra): ")
10    leia(numero)
11
12    // Inicializa o maior e o menor com o primeiro número lido,
13    // garantindo que a base de comparação seja um dado real do usuário.
14    se numero >= 0 entao
15       maior <- numero
16       menor <- numero
17    fimse
```

Ex 5

```
19    // O laço processa apenas números positivos ou zero
20    enquanto numero >= 0 faca
21       soma <- soma + numero
22       contador <- contador + 1
23       // Verifica se o número atual é o novo maior
24       se numero > maior entao
25          maior <- numero
26       fimse
27       // Verifica se o número atual é o novo menor
28       se numero < menor entao
29          menor <- numero
30       fimse
31       escreva("Digite o próximo número (-1 encerra): ")
32       leia(numero)
33    fimenquanto
```

Ex 5

```
35    escreval("--------------------------")
36
37    // Estrutura de decisão para evitar erro de divisão por zero
38    se contador > 0 entao
39       media <- soma / contador
40       escreval("Quantidade de números digitados: ", contador)
41       escreval("Média dos números: ", media)
42       escreval("Maior valor inserido: ", maior)
43       escreval("Menor valor inserido: ", menor)
44    senao
45       escreval("Nenhum número válido foi processado.")
46    fimse
47 fimalgoritmo
```