

Comandos de repetição

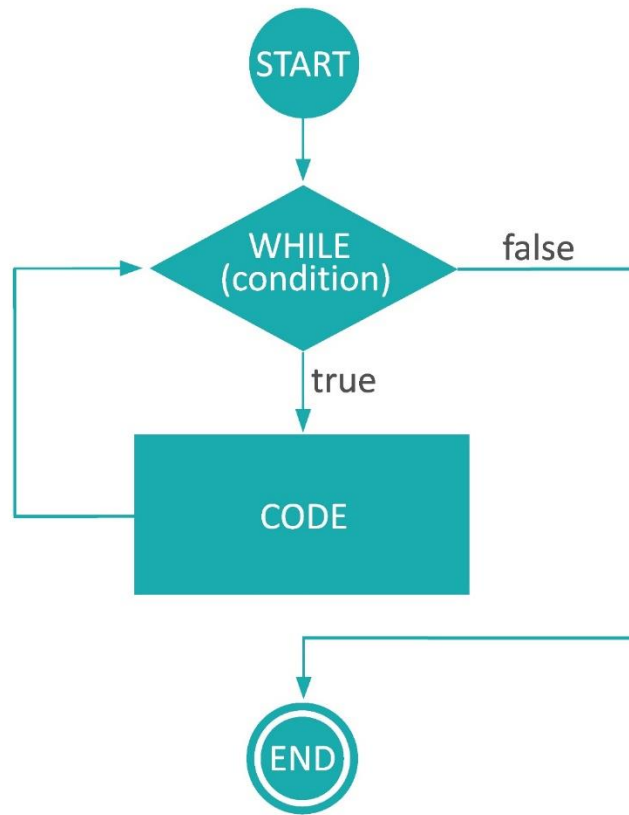
Roberto Rocha

O que é um LOOP?

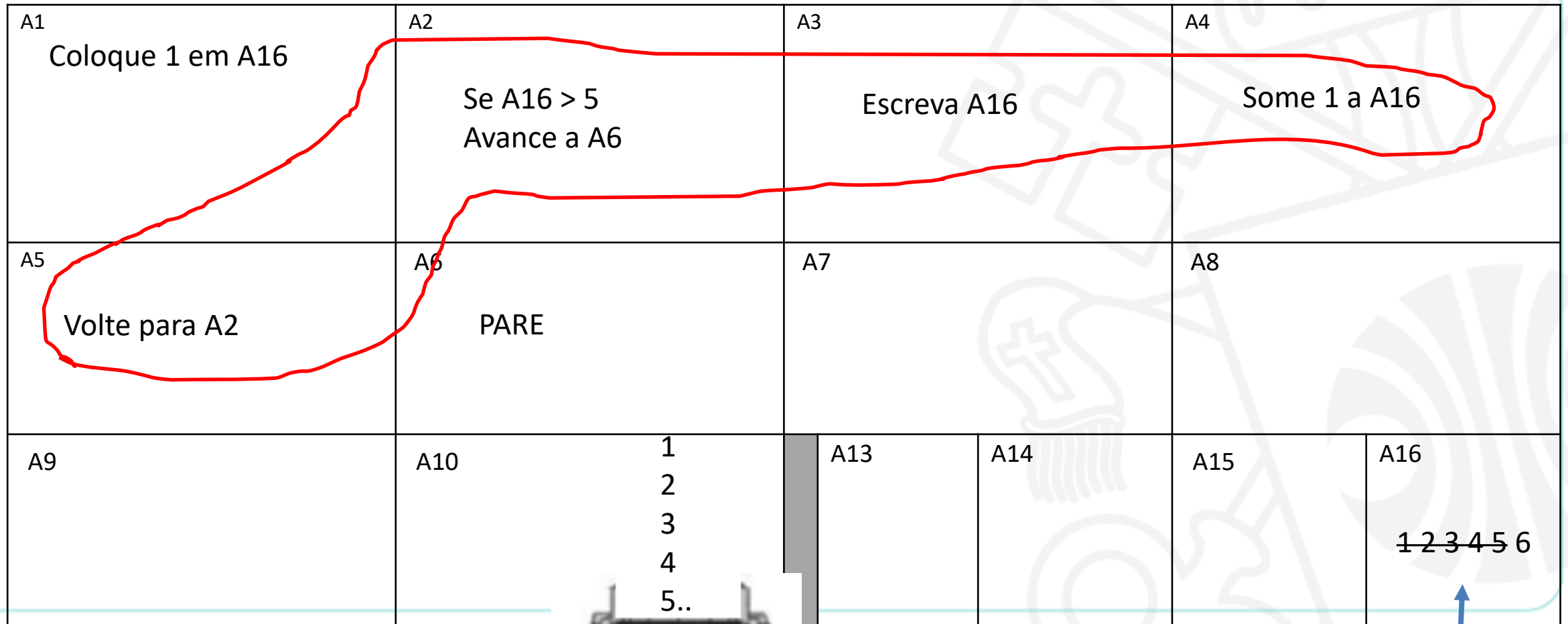
LOOP



Repetição



Exiba os números de 1 a 5



Aqui guardaremos
um valor

Ética e o comportamento de um programador de computador

Professor Manzano*: Um médico-cirurgião desatento pode matar um paciente numa cirurgia; um programador desatento pode "matar" uma empresa.

Na tarefa de programar computadores o programador pode correr três riscos:

Erro de sintaxe: escrita de comandos e/ou instruções de forma incorreta, sem respeitar as regras gramaticais da linguagem de programação em uso. É considerado de baixa gravidade, pois é fácil de ser corrigido.

Erro de requisito: não se atende ao que é solicitado. Esse erro denota que o programador não sabe ou não quer ouvir o que lhe é pedido. É considerado de media gravidade, pois é possível corrigi-lo com certa facilidade.

Erro de logica: não se consegue entender e atender ao que de fato é necessário fazer com o programa. O programador não soube pensar a programação de computadores. Para solucionar este problema é necessário mudar o "pensar", o que pode ser tarefa muito trabalhosa. O erro de logica é considerado de alta gravidade, pois é difícil de ser corrigido, exigindo do programador muita disciplina.

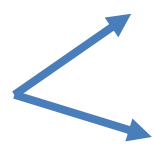
Uma forma de evitar erros, principalmente de requisitos e de logica, é a construção de algoritmos.

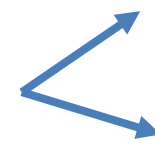
Operadores Relacionais

Operadores Relacionais

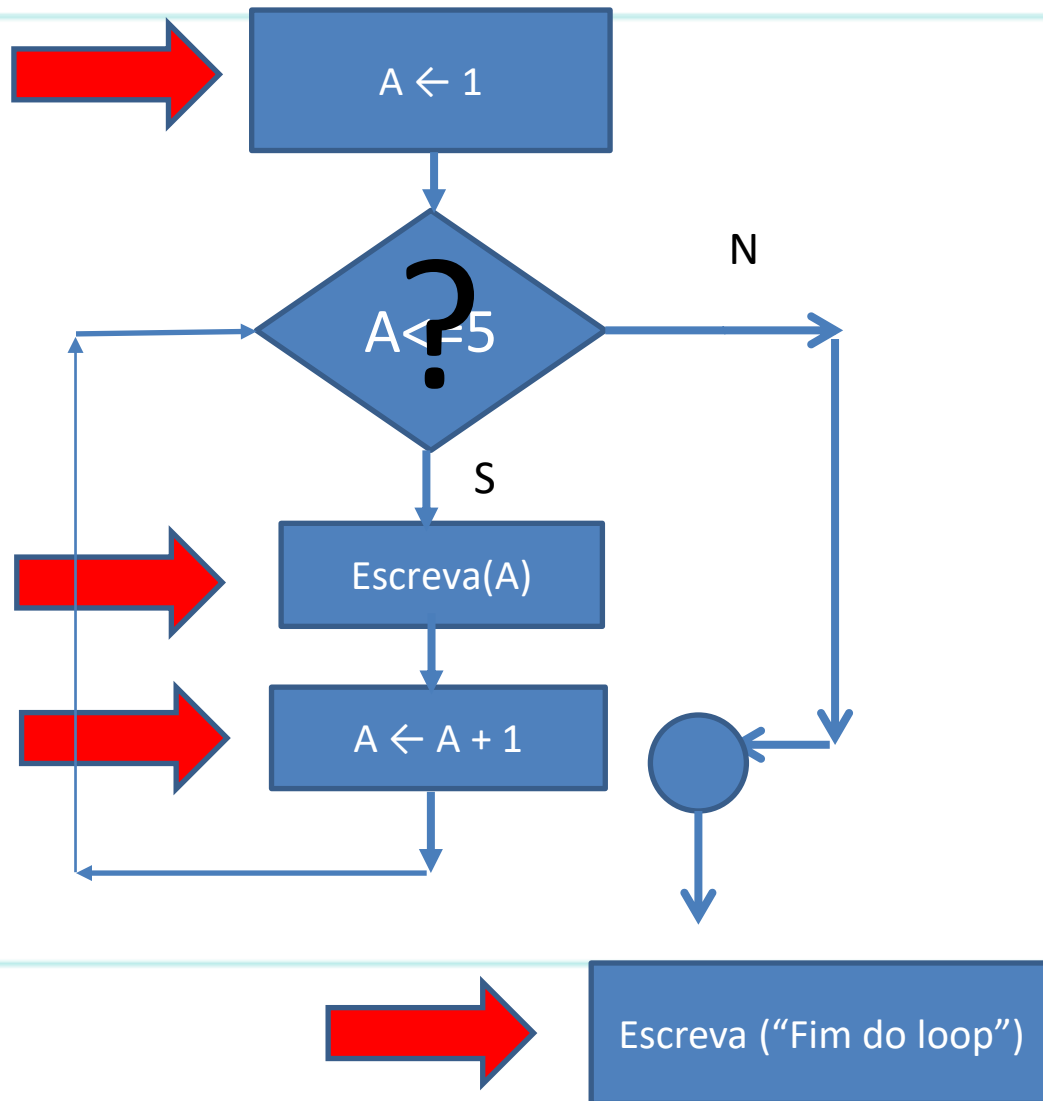
Tabela de operadores relacionais	
Operador	Descrição
=	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
<>	Diferente de

Exemplos:

$A > B$  Verdadeiro
Falso

$A > 5$  Verdadeiro
Falso

Repetição com teste no início

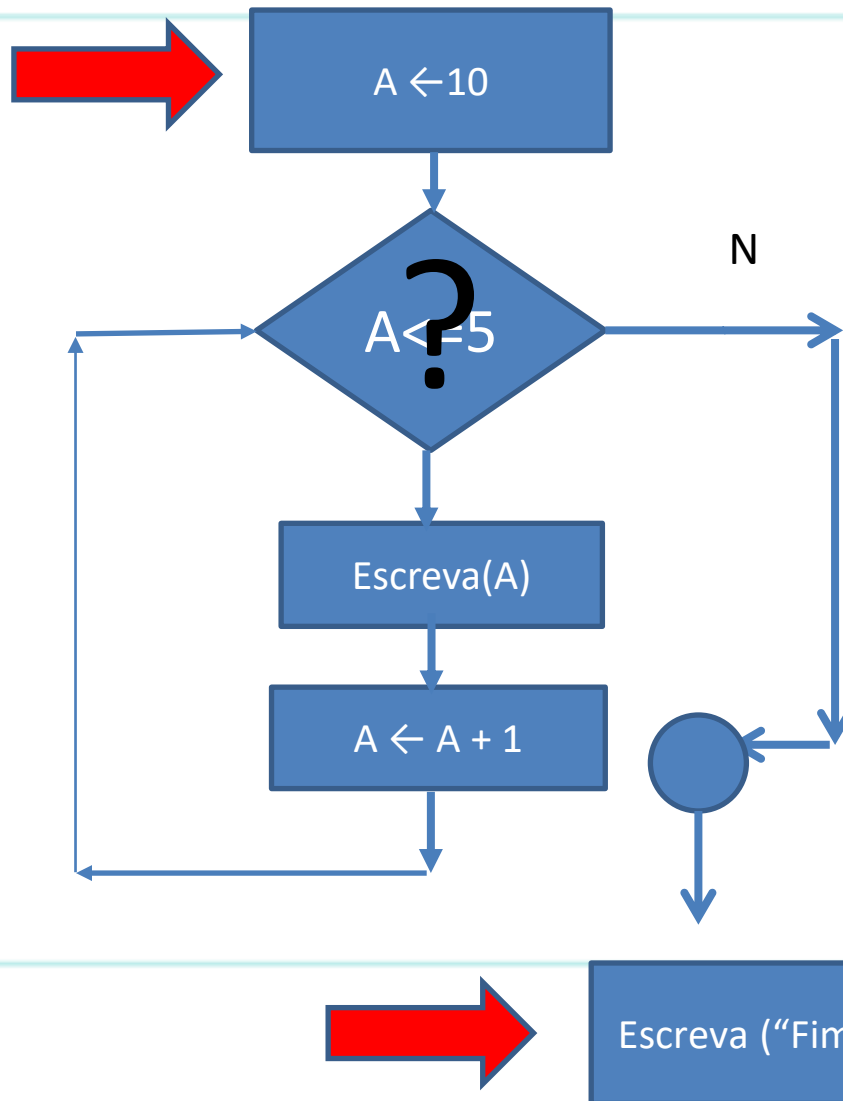


Variável	Valor
A	1 2 3 4 5 6

1 2 3 4 5

Fim do loop

Repetição com teste no início

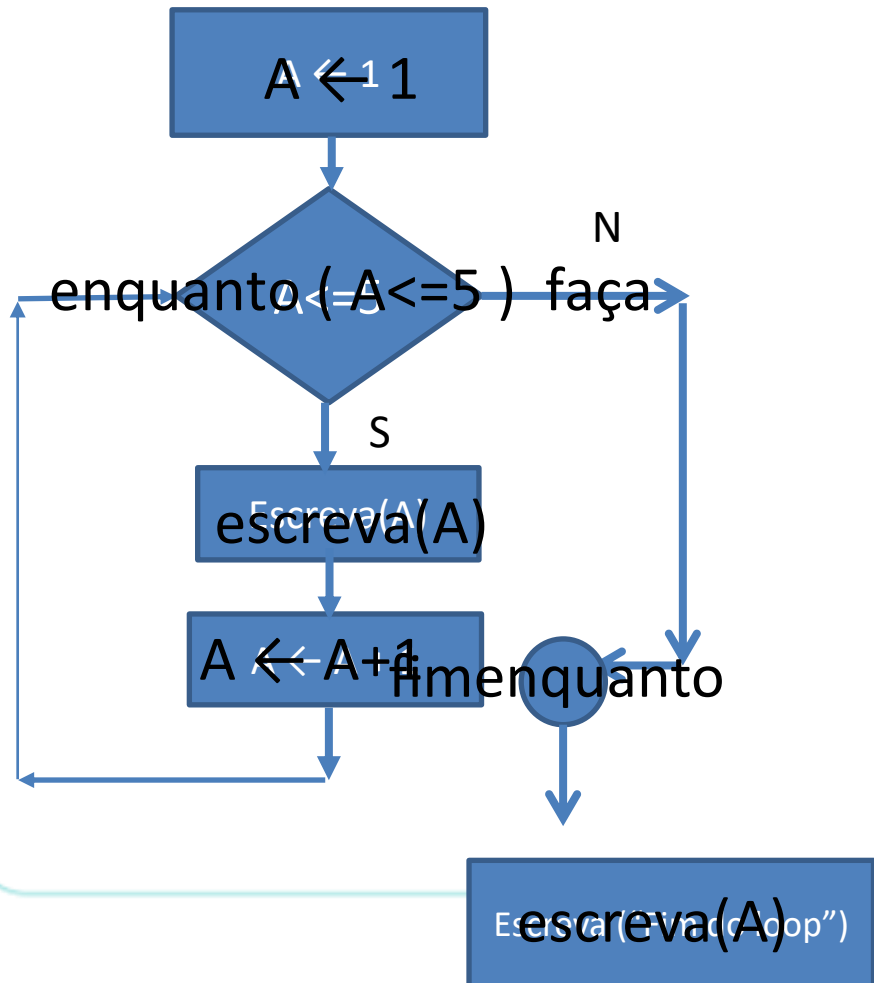


Variável	Valor
A	10

Fim do loop

Repetição com teste no início

Fluxograma



Algoritmo

Algoritmo x Java

Algoritmo	Java
Comando repetição	
enquanto (condição) faça c1 c2 c3 fimenquanto	while (condição) { c1; c2; c3; }
Exemplos	
enquanto (a<=5) faça escreva(a) a←a+1 fimenquanto	while (a<=5) { printf("%d\n",a); a=a+1; }

Tabela de operadores relacionais		
Algoritmo		Java
Operador	Descrição	Operador
=	Igual a	==
>	Maior que	>
<	Menor que	<
>=	Maior ou igual a	>=
<=	Menor ou igual a	<=
<>	Diferente de	!=

Exercício de fixação:

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Entrada

Número

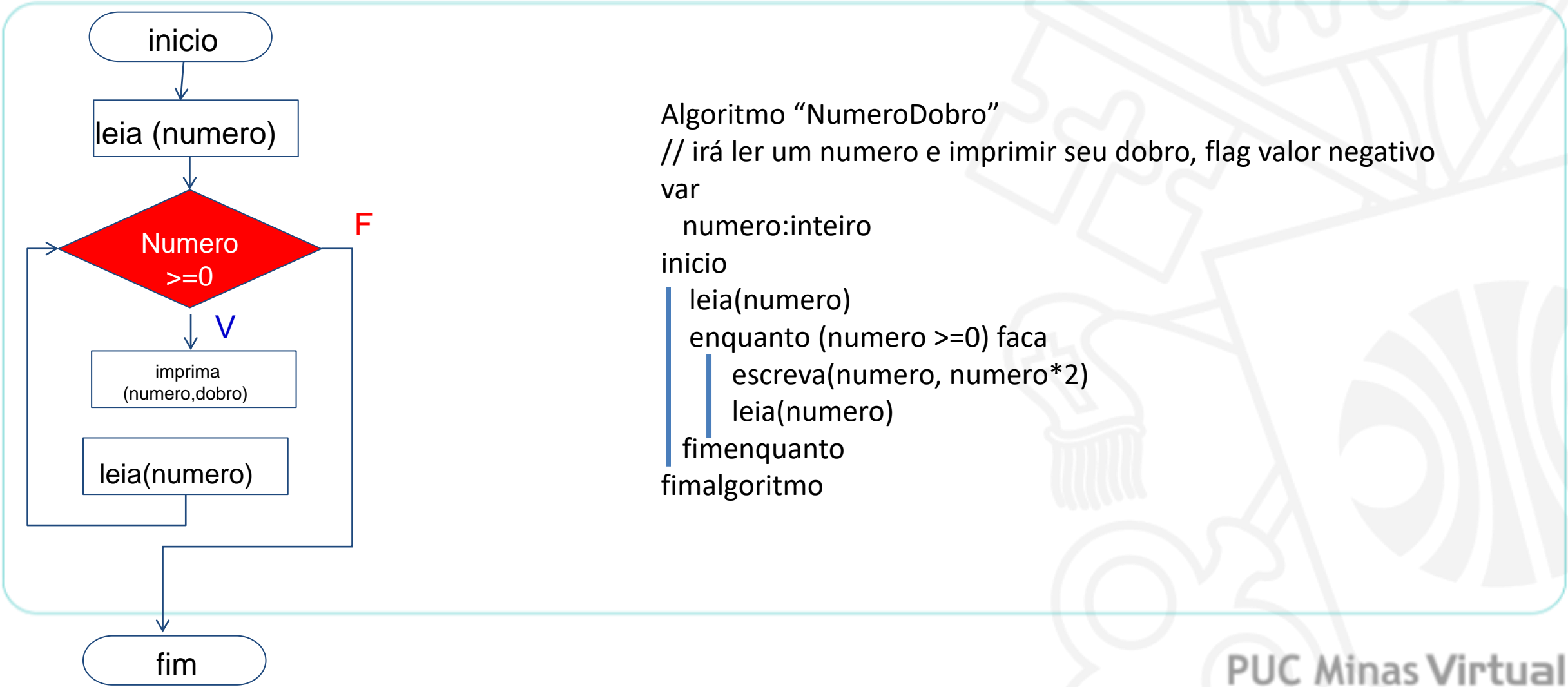
Processamento

Mostrar o número e o seu dobro
Ler novo número até que seja negativo

Saída

Numero, dobro do número

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro
flag – valor negativo



Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Algoritmo “NumeroDobro”

// irá ler um numero e imprimir seu dobro,

// flag valor negativo

var

→ numero: inteiro

início

→ leia(numero)

→ enquanto (numero >=0) faça

→ escreva(numero, numero*2)

→ leia(numero)

fimenquanto

fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

Como é um teste condicional temos que realizar vários testes

Primeiro teste : números 100, 5 e -1

numero 100

100, 200

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Algoritmo “NumeroDobro”

```
// irá ler um numero e imprimir seu dobro,  
// flag valor negativo  
var  
    numero: inteiro  
inicio  
    leia(numero)  
    enquanto (numero >=0) faca  
        escreva(numero, numero*2)  
        leia(numero)  
    fimenquanto  
finalgoritmo
```

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

Como é um teste condicional temos que realizar vários testes
Primeiro teste : números 100, 5 e -1

numero 5

100, 200

5, 10

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Algoritmo “NumeroDobro”

// irá ler um numero e imprimir seu dobro,

// flag valor negativo

var

numero: inteiro

inicio

leia(numero)

→ enquanto (numero >=0) faça
 escreva(numero, numero*2)

leia(numero)

→ fimenquanto

→ fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

Como é um teste condicional temos que realizar vários testes

Primeiro teste : números 100, 5 e -1

numero -1

100, 200

5, 10

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Algoritmo “NumeroDobro”

// irá ler um numero e imprimir seu dobro,
// flag valor negativo

var

numero: inteiro

inicio

leia(numero)

enquanto (numero >=0) faça

escreva(numero, numero*2)

leia(numero)

fimenquanto

fimalgoritmo

```
import java.util.Scanner;
public class NumeroDobro
{
    // irá ler um numero e imprimir seu dobro,
    // flag valor negativo
    public static void main (String[] args)
    {
        Scanner leia = new Scanner (System.in);
        int numero;
        System.out.println("Informe um valor:");
        numero= leia.nextInt();
        while (numero >=0 )
        {
            System.out.println(numero+" "+(numero*2));
            System.out.println("Informe um valor:");
            numero= leia.nextInt();
        }
    }
}
```

Testes:

```
----jGRASP exec: java NumeroDobro
Informe um valor:
100
100 200
Informe um valor:
5
5 10
Informe um valor:
-1
----jGRASP: operation complete.
```

Ler um conjunto indeterminado de números a cada número lido mostre o número e seu dobro flag – valor negativo

Caso o primeiro valor já for negativo não executa os comando do loop

Algoritmo “NumeroDobro”

// irá ler um numero e imprimir seu dobro,

// flag valor negativo

var

numero: inteiro

inicio

leia(numero)

enquanto (numero >=0) faca

escreva(numero, numero*2)

leia(numero)

fimenquanto

fimalgoritmo

```
import java.util.Scanner;
public class NumeroDobro
{
    // irá ler um numero e imprimir seu dobro,
    // flag valor negativo
    public static void main (String[] args)
    {
        Scanner leia = new Scanner (System.in);
        int numero;
        System.out.println("Informe um valor:");
        numero= leia.nextInt();
        while (numero >=0 )
        {
            System.out.println(numero+" "+(numero*2));
            System.out.println("Informe um valor:");
            numero= leia.nextInt();
        }
    }
}
```

Teste:

```
----jGRASP exec: java NumeroDobro
Informe um valor:
-1
----jGRASP: operation complete.
```

Ler um conjunto indeterminado de números flag – valor negativo

Esquema geral

Algoritmo “....”

// ler um conjunto indeterminado de numeros,

// flag valor negativo

var

 numero: inteiro

 :

inicio

 leia(numero)

 :

 enquanto (numero ≥ 0) faça

 :

 leia(numero)

 fimenquanto

 :

fimalgoritmo

Exercício de fixação:

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Entrada

Número
Número
Número
:
Valor <0

Processamento

O primeiro número lido é considerado o menor. A partir daí todos os outros números são testados com esse número e se for menor será trocado e esse será o menor número.

Saída

Menor número lido.
Obs.: flag não entra nos cálculos

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Terceiro passo:

Isolar ações consideradas primitivas

início

// Ler um conjunto de números e mostrar qual é o menor. Flag valor negativo

“ definir um local para armazenar o número a ser lido.”

“ definir outro local para armazenar o menor número.”

“ ler o primeiro numero e armazena-lo”

“ colocar o primeiro numero em menor valor”

“ enquanto numero lido ≥ 0 ”

“ se numero lido $<$ menor então coloque numero lido em menor”

“ leia próximo numero”

“ quando acabar os números mostre o menor valor”

fim.

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Terceiro passo:

Isolar ações consideradas primitivas

início

```
// Ler um conjunto de números e mostrar qual é o menor. Flag valor negativo
var
  "definir um local para armazenar o número a ser lido."
  numero, menor: inteiro
  "definir outro local para armazenar o menor número."
  "ler o primeiro numero e armazena-lo"
  leia(numero)
  "colocar o primeiro numero em menor valor"
  menor ← numero
  "enquanto numero lido >= 0"
  enquanto (numero >= 0) faça
    "se (numero < menor) então coloque numero lido em menor"
    se (numero < menor) então
      menor ← numero
    leia(numero)
  fimse
  "quando acabar os números mostre o menor valor"
  escreva(menor)
fim
  fimenquanto
```

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

início

fimalgoritmo

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Testando o algoritmo

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

var

➡ numero, menor: inteiro

inicio

➡ leia(numero)

➡ menor ← numero

➡ enquanto (numero >= 0) faça **Verdadeiro**

➡ se (numero < menor) então **Falso**
 menor ← numero

fimse

➡ leia(numero)

fimenquanto

escreva(menor)

fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

teste : números 5,4,7,-1

numero	5
menor	5

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Testando o algoritmo

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

var
numero, menor: inteiro

inicio

leia(numero)

menor \leftarrow numero

enquanto (numero \geq 0) faça **Verdadeiro**

se (numero < menor) então **Verdadeiro**

menor \leftarrow numero

fimse

leia(numero)

fimenquanto

escreva(menor)

fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

teste : números 5,4,7,-1

numero	4
menor	4

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Testando o algoritmo

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

var
numero, menor: inteiro

inicio

leia(numero)

menor ← numero

enquanto (numero ≥ 0) faça **Verdadeiro**

se (numero < menor) então **Falso**

menor ← numero

fimse

leia(numero)

fimenquanto

escreva(menor)

fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

teste : números 5,4,7,-1

numero	7
menor	4

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Testando o algoritmo

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

var
numero,menor:inteiro

inicio

leia(numero)

menor \leftarrow numero

enquanto (numero \geq 0) faca **Falso**

se (numero<menor) entao

menor \leftarrow numero

fimse

leia(numero)

fimenquanto

escreva(menor)

fimalgoritmo

Vamos agora conferir nosso algoritmo para ver se ele dará a resposta desejada

teste : números 5,4,7,-1

numero	-1
menor	4

4

Ler um conjunto indeterminado de números e mostrar qual o menor número. Flag valor negativo.

Definindo os nomes e as instruções
algoritmo "Menor valor"

// Ler um conjunto de números mostrar qual é o menor

var

numero, menor: inteiro

inicio

leia(numero)

menor ← numero

enquanto (numero ≥ 0) faça

se (numero < menor) então

menor ← numero

fimse

leia(numero)

fimenquanto

escreva(menor)

fimalgoritmo

```
import java.util.Scanner;
public class MenorValor
{
    // Ler um conjunto de números mostrar qual é o menor
    public static void main (String[] args)
    {
        Scanner leia = new Scanner (System.in);
        int numero, menor;
        System.out.println("Informe um valor:");
        numero= leia.nextInt();
        menor = numero;
        while (numero >=0 )
        {
            if (numero < menor)
            {
                menor = numero;
            }
            System.out.println("Informe um valor:");
            numero= leia.nextInt();
        }
        System.out.println("Menor valor do conjunto:"+menor);
    }
}
```

Teste:

```
----jGRASP exec: java MenorValor
Informe um valor:
>> 5
Informe um valor:
>> 7
Informe um valor:
>> 4
Informe um valor:
>> -1
Menor valor do conjunto:4
----jGRASP: operation complete.
```

Exercícios de fixação:

1. Escrever a sequência de 1 até 10
2. Escrever a sequência de 10 até 1
3. Leia um número e imprima os números ímpares de 1 até esse número.
4. Escrever um algoritmo, para calcular e imprimir o fatorial de um número lido do teclado.
Ex. Fatorial de $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.
5. A série de Fibonacci é formada pela sequência:
1, 1, 2, 3, 5, 8, 13, 21, 34, ...
Escreva um algoritmo que peça um número N maior que 2. Gere e imprima a série de 1 até este enésimo termo.
6. O número 3025 possui a seguinte característica:
 $30 + 25 = 55$
 $55^2 = 3025$
Quantos e quais são os números de 4 dígitos possuem essa característica?
7. Faça um menu para o usuário escolher qual exercício deverá ser executado.



PUC Minas
Virtual