

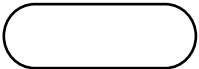


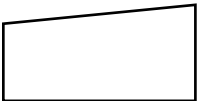

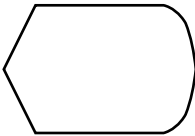
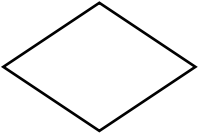
Fluxograma

Analisaremos agora o método de representação de algoritmos denominado fluxograma.

Conceitualmente um fluxograma é um tipo de diagrama, e pode ser entendido como uma representação esquemática de um processo, constitui uma representação gráfica que ilustra de forma descomplicada a seqüência de execução dos elementos que o compõem. Podemos entendê-lo, na prática, como a documentação dos passos necessários para a execução de um processo qualquer.

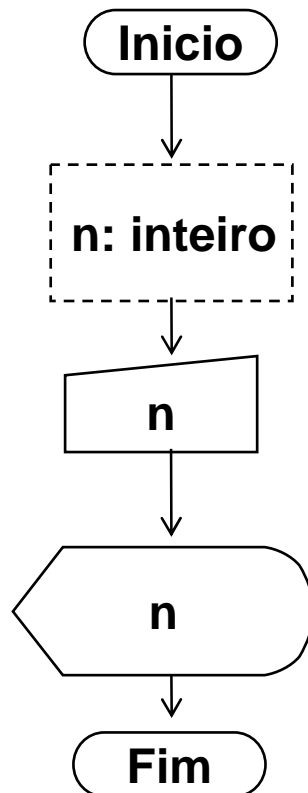
Veremos agora alguns símbolos empregados na construção de fluxogramas.







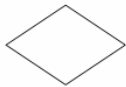
Fluxograma

Símbolo	Nome	Descrição
	Terminador	Indica o início e o fim do fluxo do algoritmo.
	Seta de fluxo	Indica o sentido do fluxo de execução do algoritmo. É através dela que os símbolos do fluxograma são conectados.
	Declaração	Delimita a seção de declaração de variáveis.
	Entrada de dados	Corresponde à instrução de entrada de dados através do teclado.
	Atribuição	Símbolo utilizado para indicar cálculos e atribuição de valores.
	Saída de dados	Corresponde à instrução de saída de dados. Os dados serão exibidos na tela do computador.
	Desvio condicional	Divide o fluxo do programa em dois caminhos, dependendo do teste lógico que fica dentro do losango.

Exemplo de Fluxograma

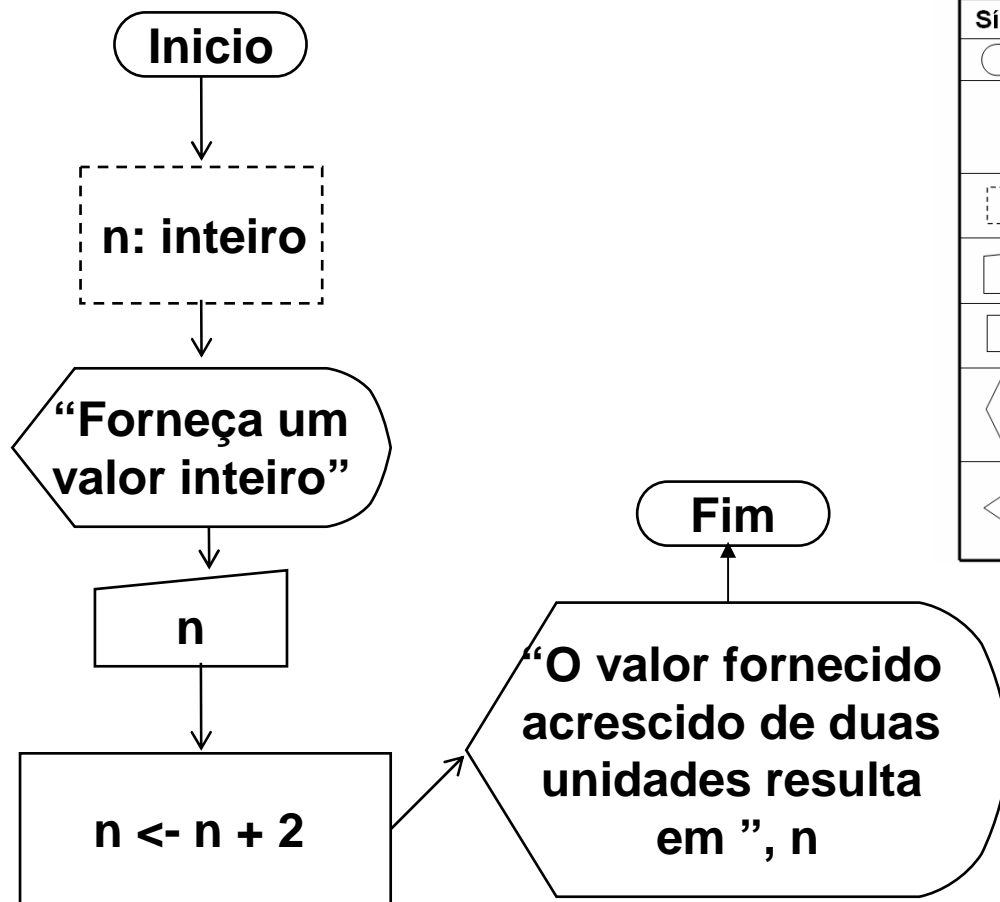
De forma similar à análise feita com pseudocódigo, iniciaremos nossa análise por um fluxograma que efetua a leitura, através do teclado, de um valor inteiro e o retorna no monitor.










Símbolo	Nome	Descrição
	Terminador	Indica o início e o fim do fluxo do algoritmo.
	Seta de fluxo	Indica o sentido do fluxo de execução do algoritmo. É através dela que os símbolos do fluxograma são conectados.
	Declaração	Delimita a seção de declaração de variáveis.
	Entrada de dados	Corresponde à instrução de entrada de dados através do teclado.
	Atribuição	Símbolo utilizado para indicar cálculos e atribuição de valores.
	Saída de dados	Corresponde à instrução de saída de dados. Os dados serão exibidos na tela do computador.
	Desvio condicional	Divide o fluxo do programa em dois caminhos, dependendo do teste lógico que fica dentro do losango.

Exemplo de Fluxograma

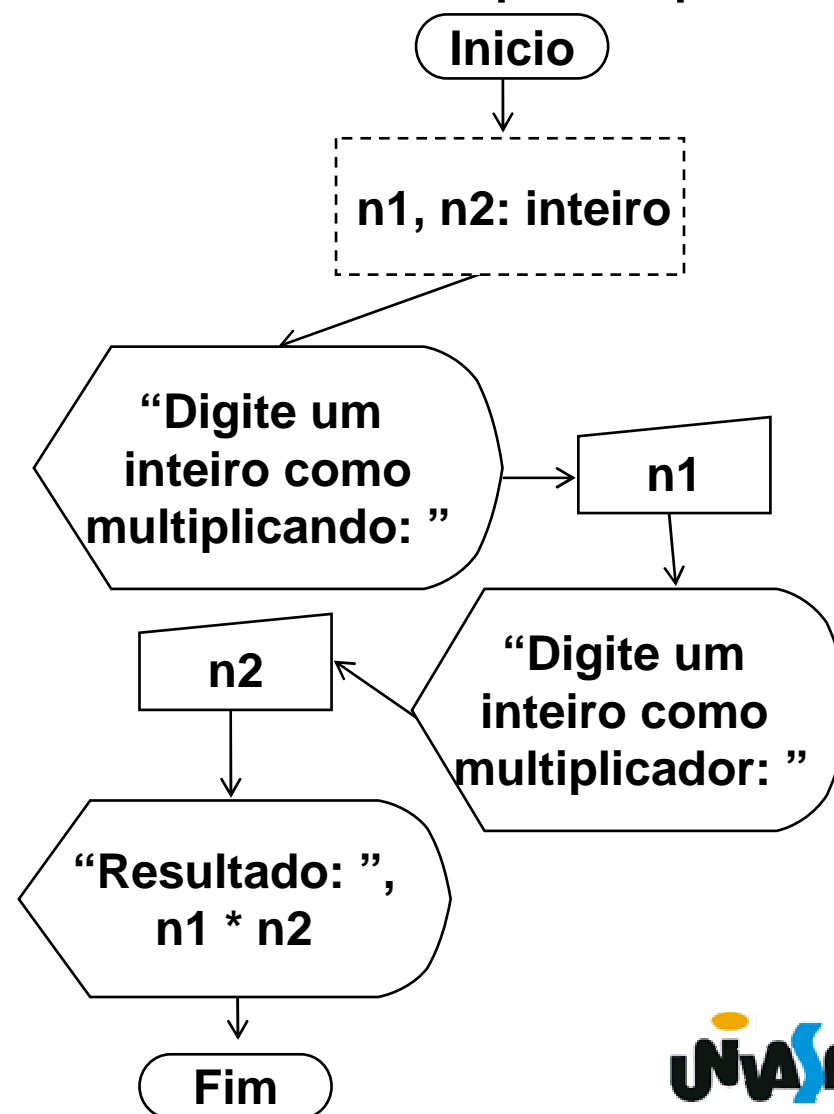
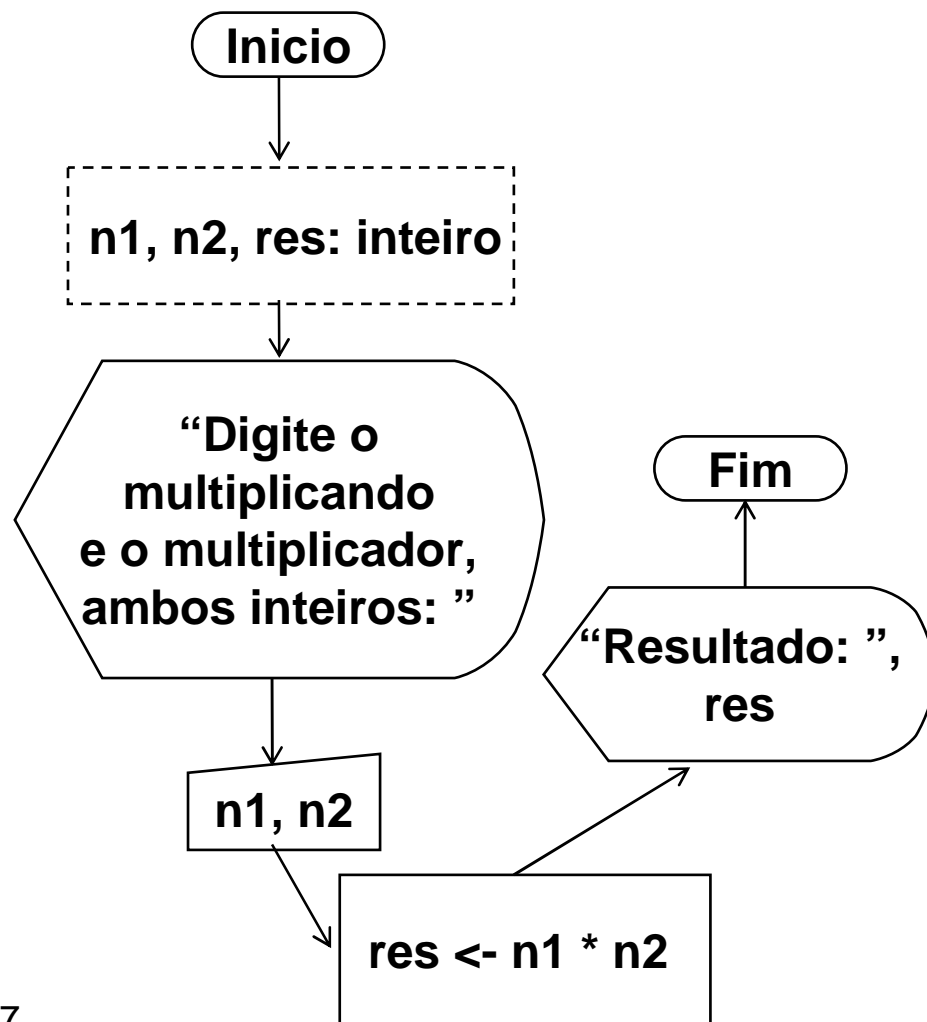
Observaremos agora um fluxograma que recebe um valor inteiro, através da entrada padrão, e acresce duas unidades a este exibindo o resultado na saída padrão.



Símbolo	Nome	Descrição
	Terminador	Indica o início e o fim do fluxo do algoritmo.
	Seta de fluxo	Indica o sentido do fluxo de execução do algoritmo. É através dela que os símbolos do fluxograma são conectados.
	Declaração	Delimita a seção de declaração de variáveis.
	Entrada de dados	Corresponde à instrução de entrada de dados através do teclado.
	Atribuição	Símbolo utilizado para indicar cálculos e atribuição de valores.
	Saída de dados	Corresponde à instrução de saída de dados. Os dados serão exibidos na tela do computador.
	Desvio condicional	Divide o fluxo do programa em dois caminhos, dependendo do teste lógico que fica dentro do losango.

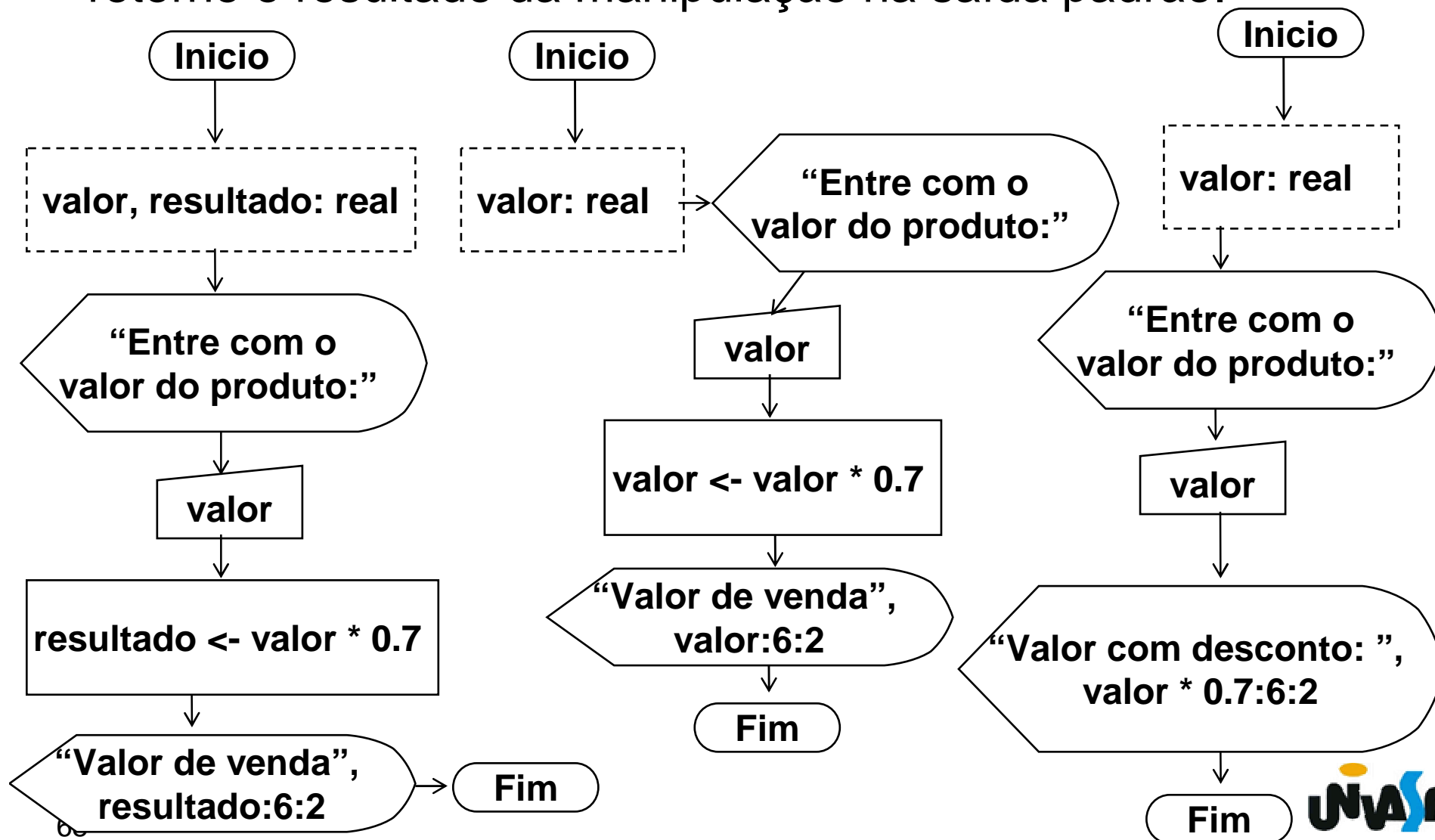
Exercício de Fluxograma

Construa um fluxograma para obter o resultado da multiplicação de dois números inteiros quaisquer fornecidos pelo usuário.



Exercício de Fluxograma

Gere um fluxograma que aplique um desconto de 30% sobre o valor de um produto, recebido como entrada, e retorne o resultado da manipulação na saída padrão.



Estruturas de Controle de Fluxo

Os algoritmos desenvolvidos até o momento constituem uma seqüência de ações que sempre são executadas em sua totalidade indiferente de qual(is) seja(m) o(s) valor(es) da(s) entrada(s).

Contudo para a resolução de determinados problemas ou para a execução de determinadas tarefas é necessária a realização de um conjunto distinto de ações e este conjunto é definido com base em uma análise da(s) entrada(s).

Um exemplo simples de uma destas situações é um algoritmo capaz de efetuar o cálculo do imposto de renda devido por um determinado contribuinte. Neste caso dependendo da quantidade de dependentes, do valor de sua renda e outras fatores o cálculo será feito de formas distintas.

Estruturas de Controle de Fluxo

Em função do que foi mencionado foram criadas as estruturas de controle de fluxo, as quais são fundamentais para a construção de algoritmos complexos. Estas permitem que o programador especifique a seqüência de instruções que será executada.

1. Instrução condicional

Sintaxe: ...

se (*<expressão-lógica>*) entao

<seqüência-de-comandos>

fimse

...

Estruturas de Controle de Fluxo

Pseudocódigo/Exercício – Construa o pseudocódigo de um algoritmo para obter o resultado da divisão de dois números inteiros quaisquer.

algoritmo “exercício”

var n1, n2: inteiro

res: real

inicio

escreva (“Digite o dividendo inteiro: ”)

leia (n1)

escreva (“Digite o divisor inteiro: ”)

leia (n2)

se (n2<>0) **entao**

res <- n1 / n2

escreva (“Resultado da divisão: ”, res)

fimse

fimalgoritmo

Estruturas de Controle de Fluxo

Pseudocódigo/Exercício – Construa o pseudocódigo de um algoritmo para obter o resultado da divisão de dois números inteiros quaisquer.

algoritmo “exercício b”

var n1, n2: inteiro

res: real

inicio

escreva (“Digite o dividendo inteiro: ”)

leia (n1)

escreva (“Digite o divisor inteiro: ”)

leia (n2)

se (n2<>0) **entao**

res <- n1 / n2

escreva (“Resultado da divisão: ”, res)

fimse

se (n2=0) **entao**

escreva (“Impossível dividir!”)

fimse

fimalgoritmo

Estruturas de Controle de Fluxo

1. Instrução condicional (continuação)

Sintaxe:

```
...  
se (<expressão-lógica>) entao  
    <seqüência-de-comandos-1>  
senao  
    <seqüência-de-comandos-2>  
fimse  
...
```

//**Observação:** esta forma é denominada instrução condicional composta.

Estruturas de Controle de Fluxo

Pseudocódigo/Exercício – Construa o pseudocódigo de um algoritmo para obter o resultado da divisão de dois números inteiros quaisquer.

algoritmo “exercício c”

var n1, n2: inteiro

res: real

inicio

escreva (“Digite o dividendo inteiro: ”)

leia (n1)

escreva (“Digite o divisor inteiro: ”)

leia (n2)

se (n2=0) entao

escreva (“Impossível dividir!”)

senao

res <- n1 / n2

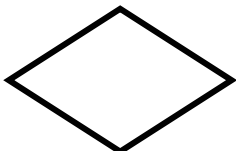
escreva (“Resultado da divisão: ”, res)

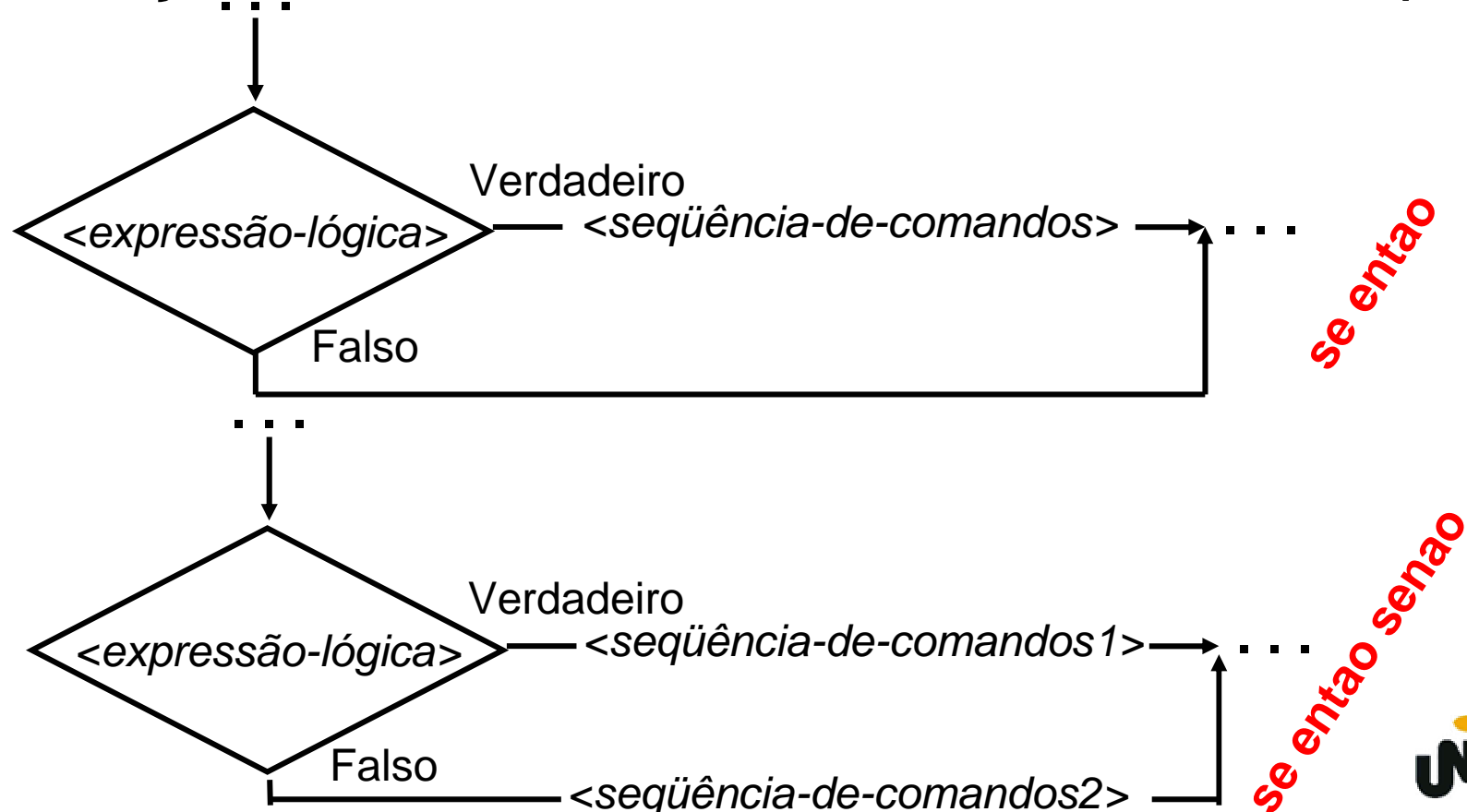
fimse

fimalgoritmo

Estruturas de Controle de Fluxo

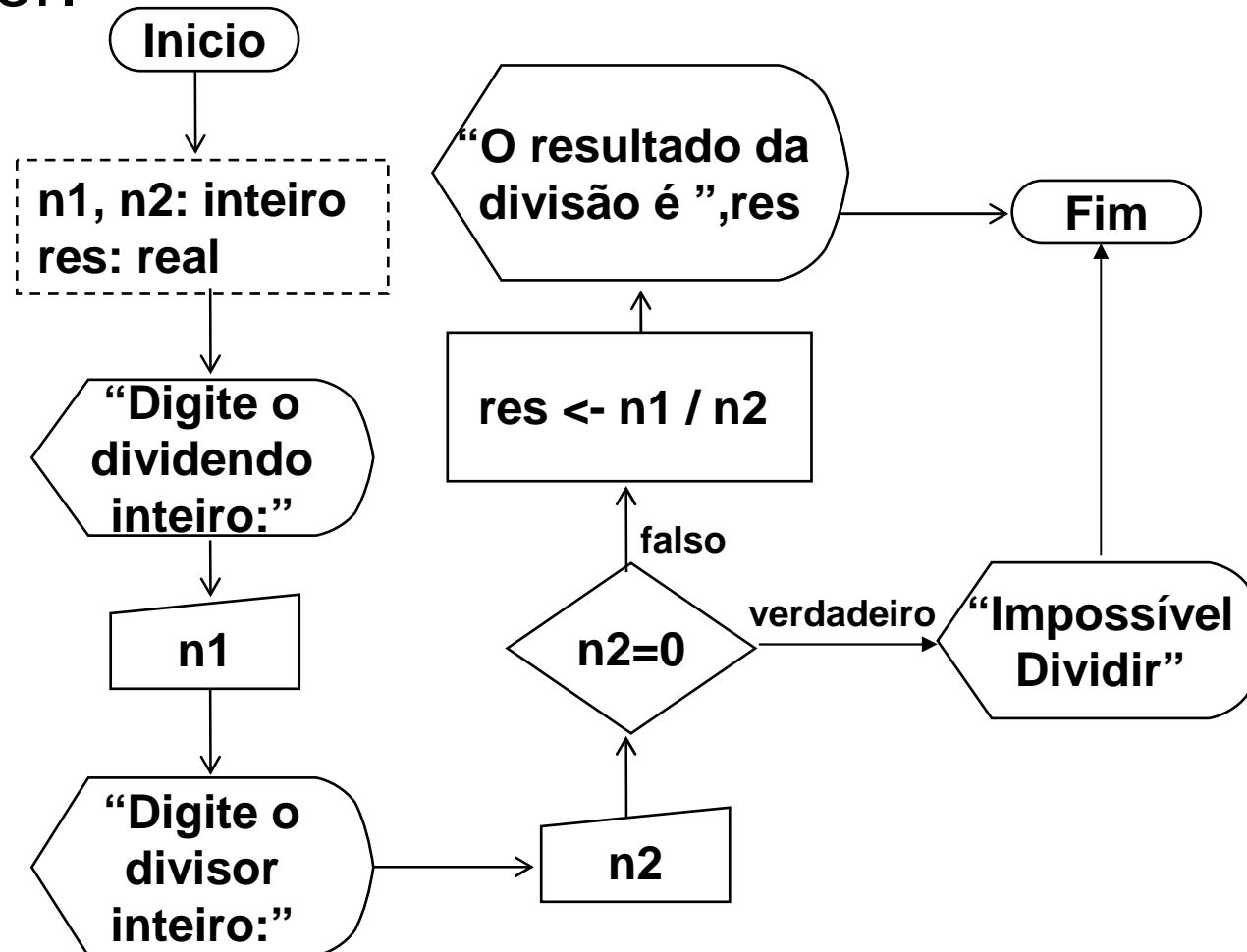
Fluxograma

Vimos o símbolo  quando falamos sobre fluxograma. Este símbolo nos permite implementar a instrução “se então” e “se então senão”. Exemplo:



Estruturas de Controle de Fluxo

Fluxograma/Exercício – Com base no que foi exposto construa um fluxograma para obter o resultado da divisão de dois números inteiros quaisquer.



Teste de Mesa

Ao nos recordarmos dos passos necessários para a construção de um algoritmo veremos que após a elaboração de um algoritmo devemos testá-lo realizando simulações com o propósito de verificarmos se este está ou não correto.

Existem alguns softwares disponíveis que efetuam a interpretação de algoritmos representados em pseudocódigos ou em fluxogramas.

Porém, existe uma técnica denominada “teste de mesa” que permite a simulação do processo de interpretação de um algoritmo utilizando apenas um papel e uma caneta.

Teste de Mesa

Para acompanhar o desenvolvimento de um algoritmo é importante verificar o estado dos dados a cada instrução, verificando o conteúdo de todas as variáveis contidas no algoritmo.

Sendo assim deve-se enumerar as linhas do algoritmo e em seguida criar uma tabela onde, a cada linha, são mostrados os conteúdos das variáveis do algoritmo e o número da linha executada.

Para uma melhor visualização do processo adotaremos a seguinte convenção: nas linhas em que uma variável é lida (entrada), o valor da variável ficará entre colchetes [] e quando o conteúdo de uma variável for escrito (saída), ficará entre chaves {}.

Aplicaremos a técnica do teste de mesa sobre o algoritmo abaixo o qual visa determinar o número de vértices de uma figura geométrica.

algoritmo “vértices”

var vertices, faces, arestas: inteiro

inicio

escreva (“Entre com o número de faces da figura geométrica: ”)

leia (faces)

escreva (“Entre com um número da arestas da figura geométrica:”)

leia (arestas)

 vertices <- arestas + 2 – faces

escreva (“O número de vértices do objeto especificado é: ”, vertices)

fimalgoritmo

Linha	Vertices	Faces	Arestas
1	?	?	?
2	?	[6]	?
3	?	6	?
4	?	6	[12]
5	8	6	12
6	{8}	6	12