Aula Teórica 03

Comandos Condicionais (Decisão)

Semana 03

Material Didático Proposto

Conteúdos da Aula

- **Programação Estruturada**
- **Comandos Condicionais (Decisão)**
- **Operadores Relacionais**

Programação Estruturada

Programação Estruturada

<u>Conceito</u>:

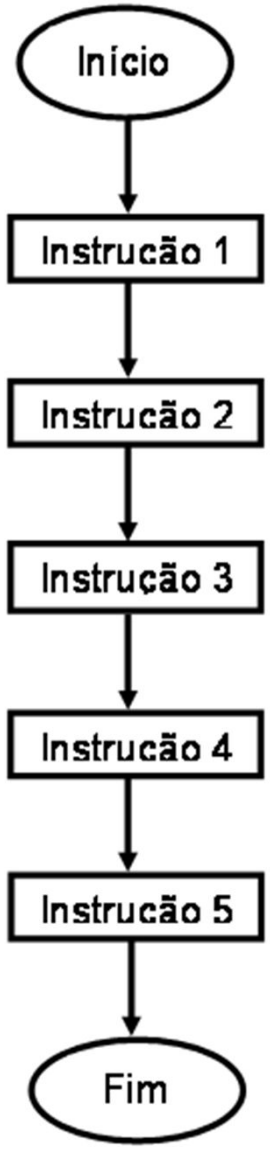
Programação estruturada é uma forma de Programação de computadores que defende que todo programa pode ser escrito com três estruturas:

- **sequência**
- **decisão**
- **iteração**

Programação Estruturada

Sequência

Até a última aula, os programas constituíram-se por uma sequência de instruções (comandos) executados sequencialmente, sem qualquer desvio, conforme o fluxograma ao lado.



Programação Estruturada

Decisão (comandos condicionais)

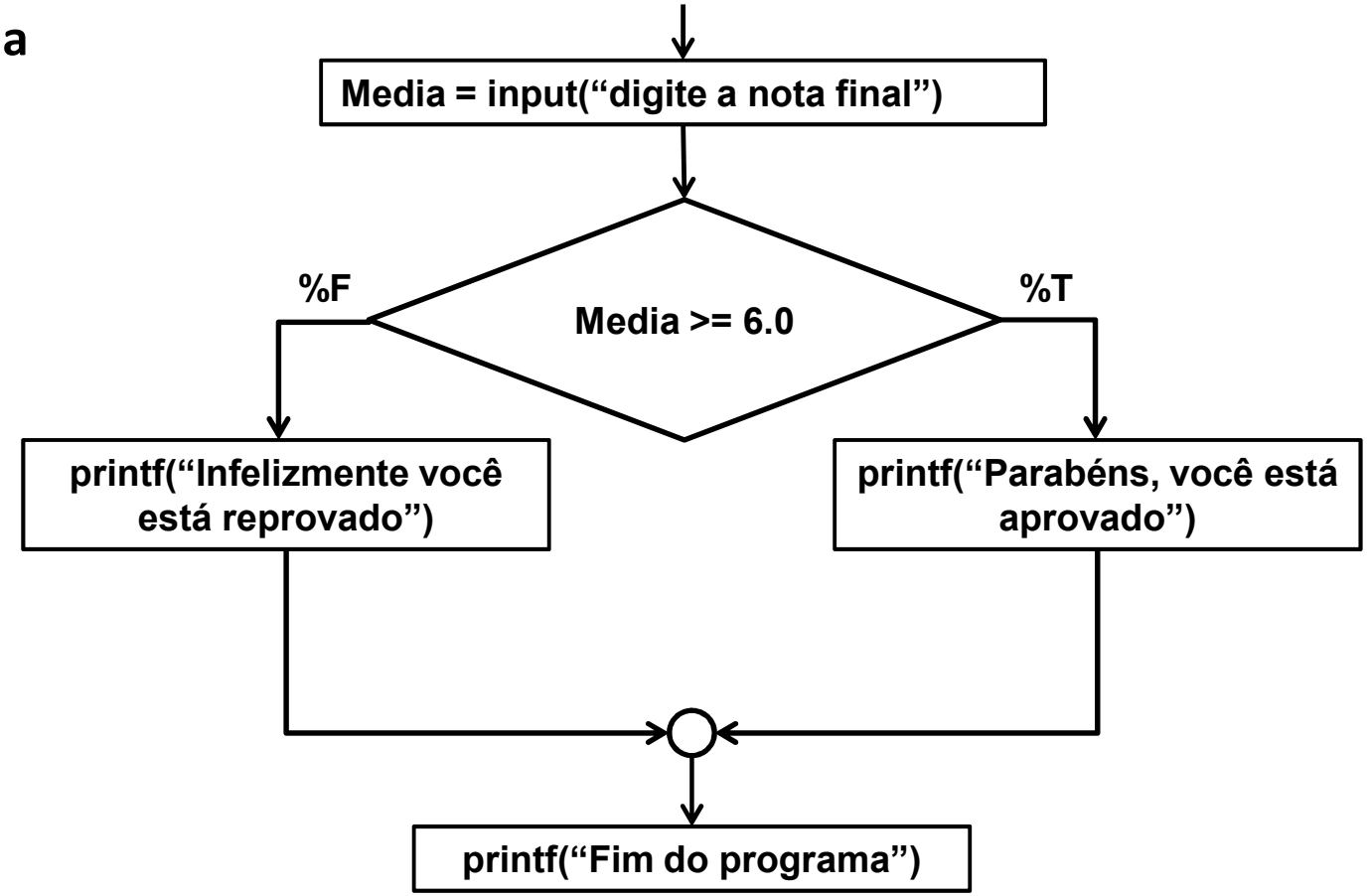
A segunda estrutura é utilizada quando é necessário realizar um desvio de fluxo, com base em uma decisão

```
se <"condição" for verdadeira> então
        Faça a tarefa A;
senão //caso contrário (condição é falsa)
        Faça a tarefa B;
fim
```

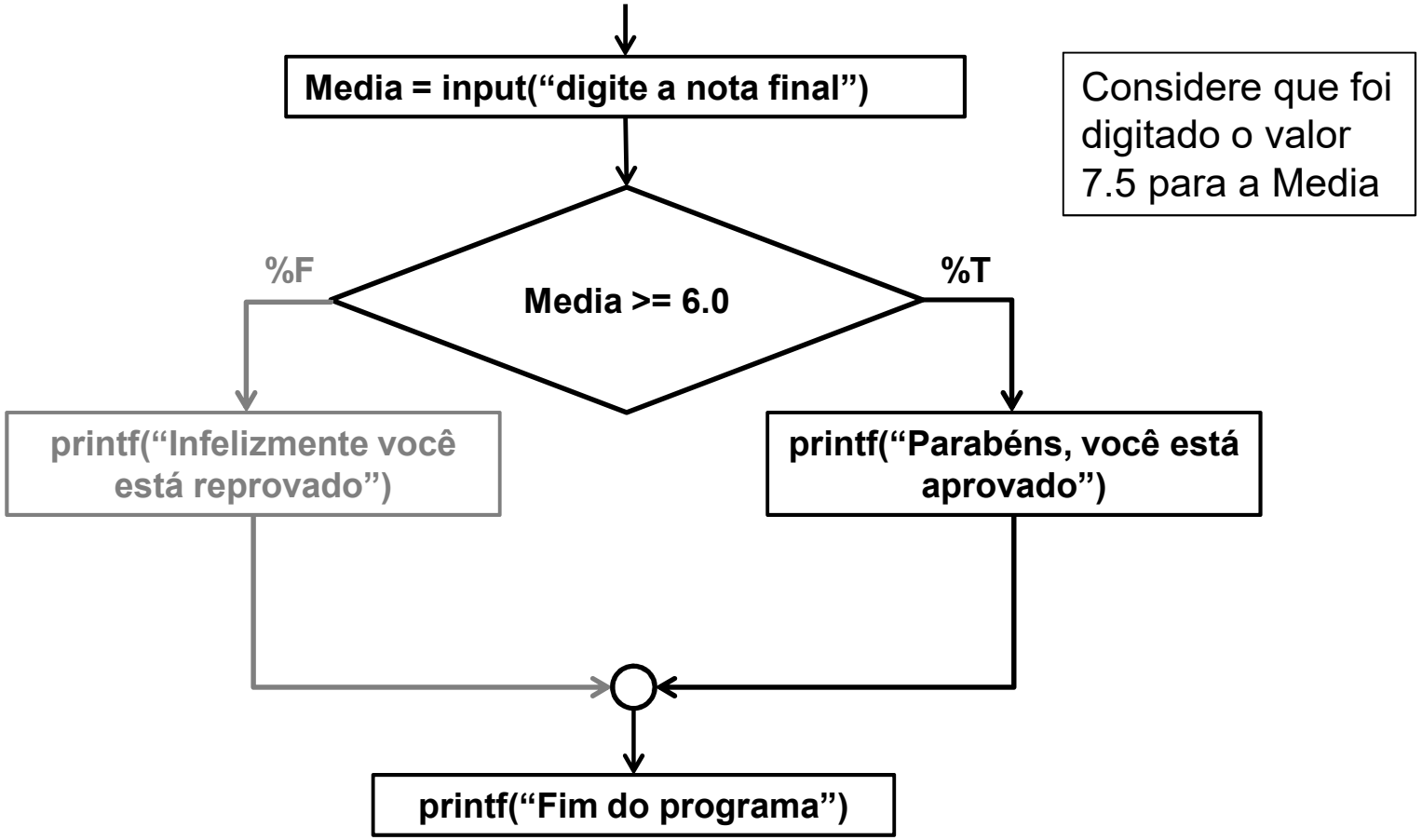
O desvio de fluxo é caracterizado pela "decisão" (condição) entre executar a *tarefa A* ou executar a *tarefa B*, tarefas mutuamente excludentes

Decisão (comandos condicionais)

Fluxograma



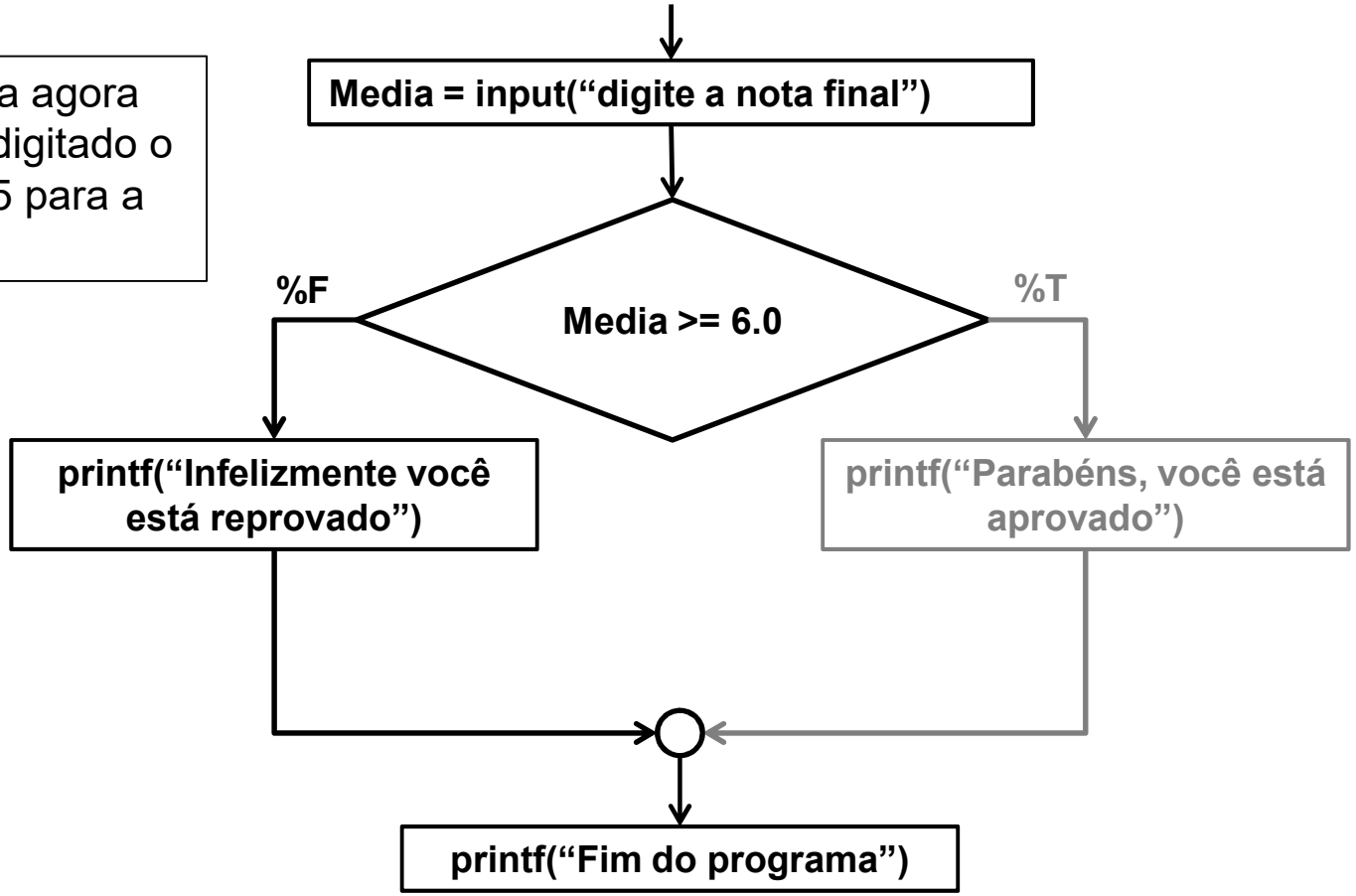
Decisão (comandos condicionais)



Considere que foi digitado o valor 7.5 para a Media

Decisão (comandos condicionais)

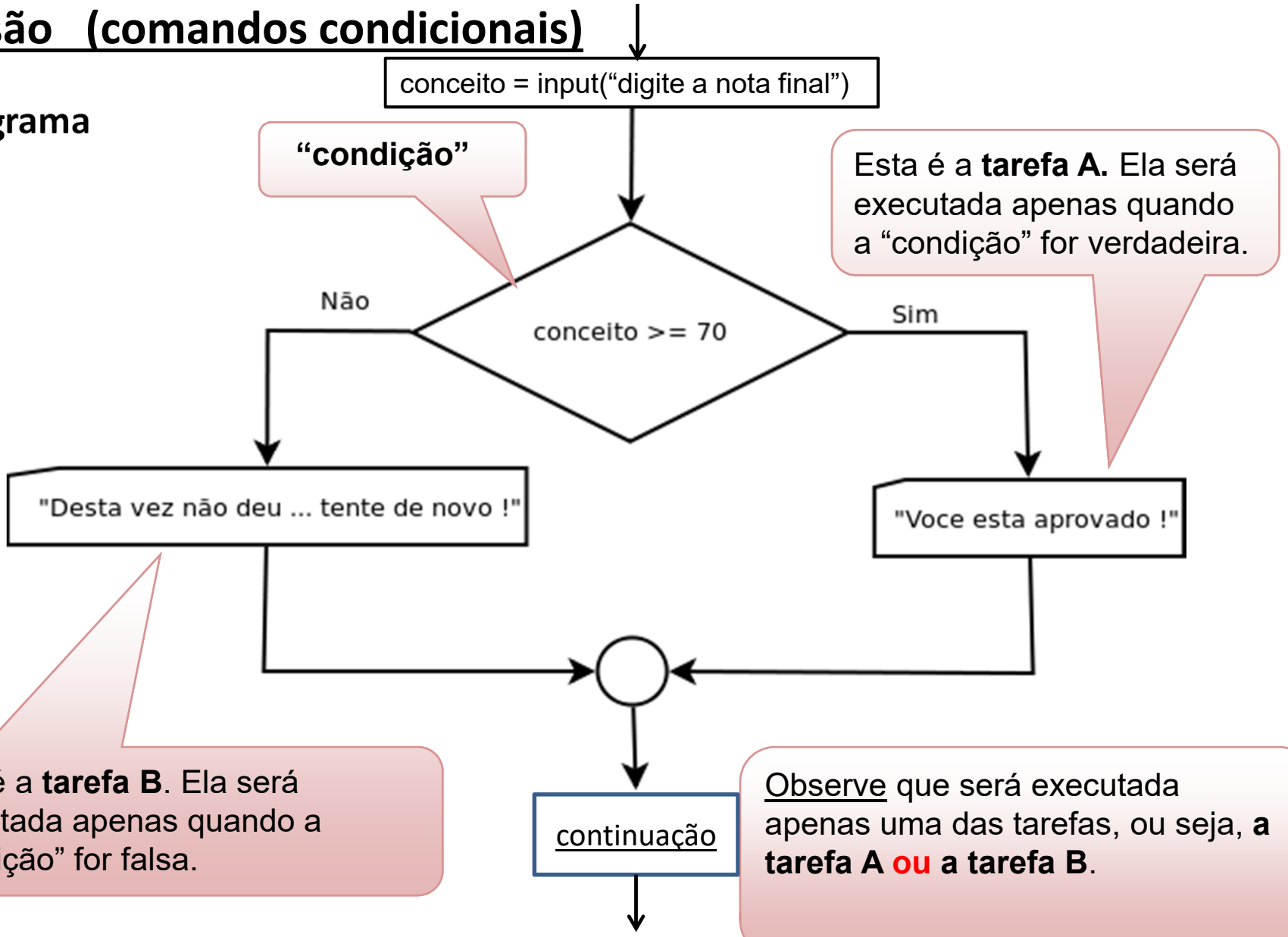
Suponha agora que foi digitado o valor 4.5 para a Media



Programação Estruturada

Decisão (comandos condicionais)

Fluxograma



Esta é a **tarefa A.** Ela será executada apenas quando a “condição” for verdadeira.

Esta é a **tarefa B**. Ela será executada apenas quando a “condição” for falsa.

Observe que será executada apenas uma das tarefas, ou seja, **a tarefa A ou a tarefa B**.

Condição

<condição> é uma expressão relacional :

<expr 1> <operador Relacional> <expr 2>

Onde:

<expr n> é uma expressão, que pode ser um valor numérico, ou uma expressão matemática que resulta em um valor numérico.

A avaliação de uma expressão relacional pode resultar em:

verdadeiro representado por **%t** ou **%T**

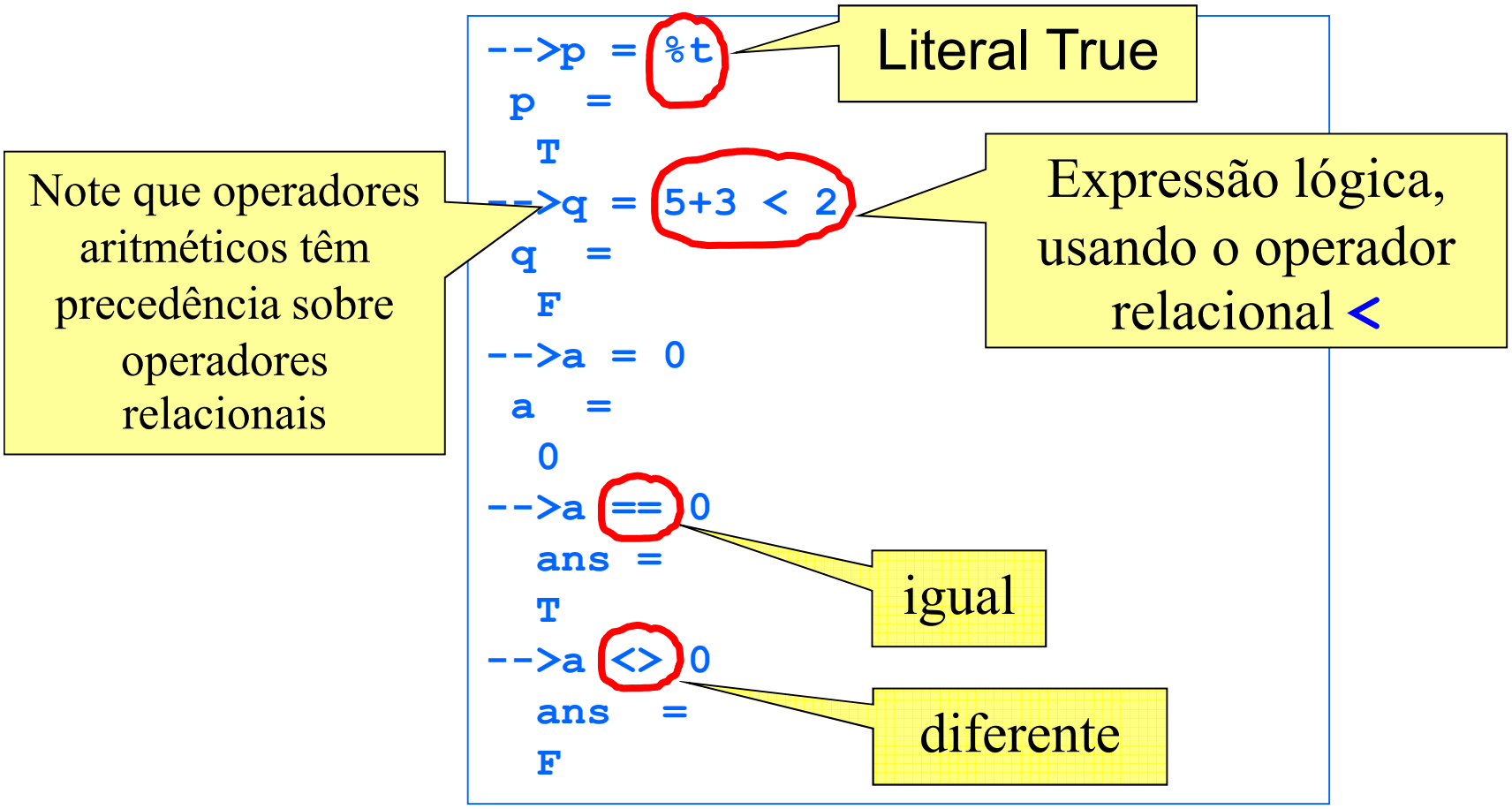
ou

falso representado por **%f** ou **%F**

Operadores Relacionais - Scilab

Operador	Descrição
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
<> ou ~=	Diferente de

Expressões com Tipo Lógico



Operadores Relacionais - Exemplos

Prioridade de Execução

- **Quando temos uma combinação entre expressões matemáticas e expressões lógicas, primeiramente são calculas as expressões matemáticas; a seguir, são calculadas as expressões lógicas.**

- **os operadores matemáticos tem maior prioridade de execução, com relação aos operadores relacionais.**

Na expressão lógica $4 + 5 > 8$ **, é evidente que a soma (+) é realizada antes de ser feita a comparação (>)**

(+) operador matemático

(>) operador relacional

Exemplo de Comando Condicional
Equação de 2º grau com raízes reais

- Ler os valores dos coeficientes a, b e c
- Calcular o valor de delta
- Verificar o valor de delta
  - Se (**delta < 0**) então
    - Imprimir "A equação não tem raízes reais"
  - Senão (**delta >= 0**)
    - Calcular as raízes reais da equação e
    - Imprimir as raízes da equação "X1 = ..., X2 = ..."
  - Final da verificação
- Finalizar o programa imprimindo "Fim do programa"

Condições mutuamente excludente

Equações de Segundo Grau:

Passos do algoritmo:

1. Ler os coeficientes: a, b, c

2. Calcular o valor de $\Delta = b^2 - 4ac$

3. **Se** Δ < 0 **então**

   informar: "não existem raízes reais"

   Senão (Δ será >= 0, não precisa testar!)

   calcular

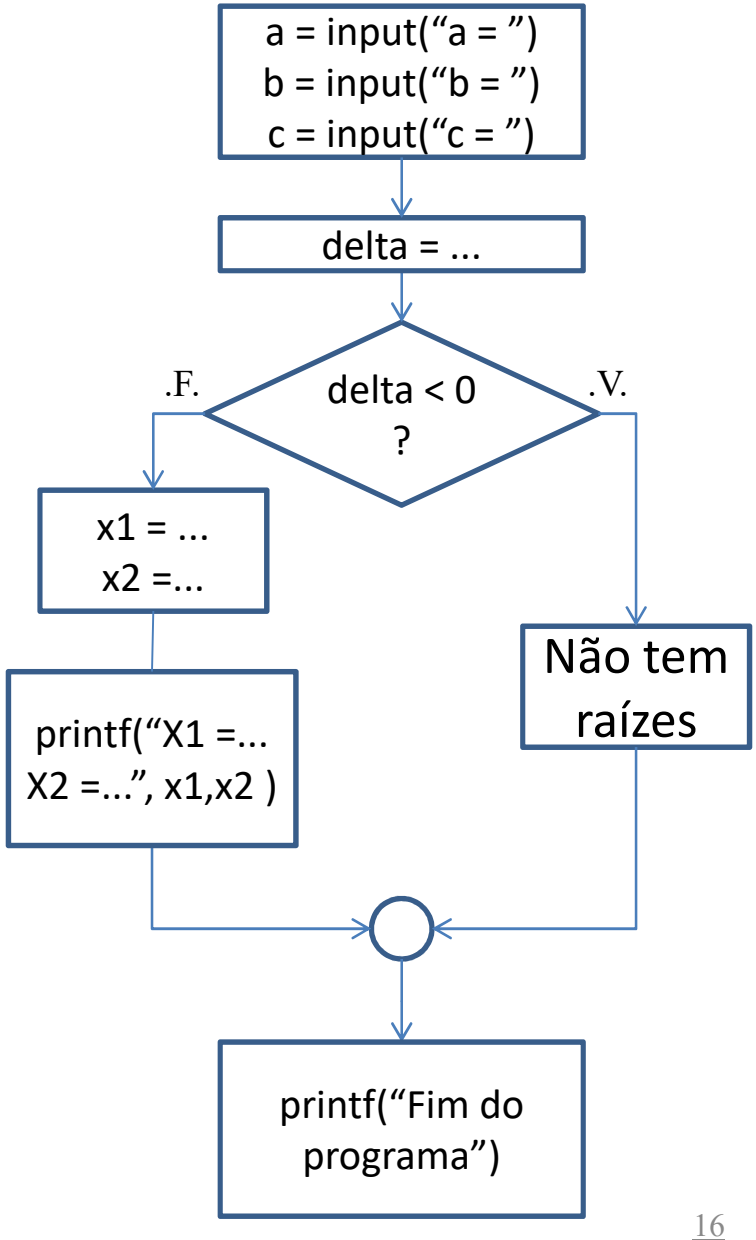
$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

   informar o valor de x_1 e x_2

   Fim

4. Imprimir "Fim do programa"

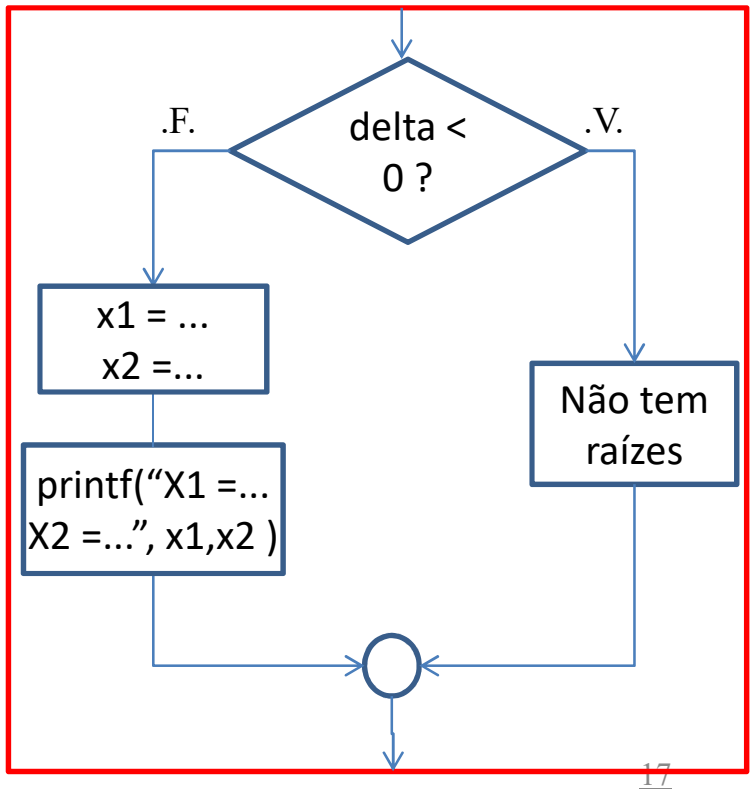


Equação de Segundo Grau:

```
printf("\nRaízes reais de uma equação de segundo grau");
// Entrada dos dados da equação
a = input("entre com o valor de a (a diferente de 0: ");
b = input ("entre com o valor de b: ");
c = input ("entre com o valor de c: ");

// Resolvendo a equação
delta = b^2 - 4 * a * c;
printf("O valor delta é: %g\n", delta)
if delta < 0 then
        printf("\nA equação não tem raízes reais");
else  //(delta será necessariamente >= 0)
        x1 = ( -b + sqrt(delta) ) / (2 * a);
        x2 = ( -b - sqrt(delta) ) / (2 * a);
        printf(" x1 = %g\n", x1);
        printf(" x2 = %g", x2);
end
printf("\n Fim do programa")
```

Diálogo com o usuário

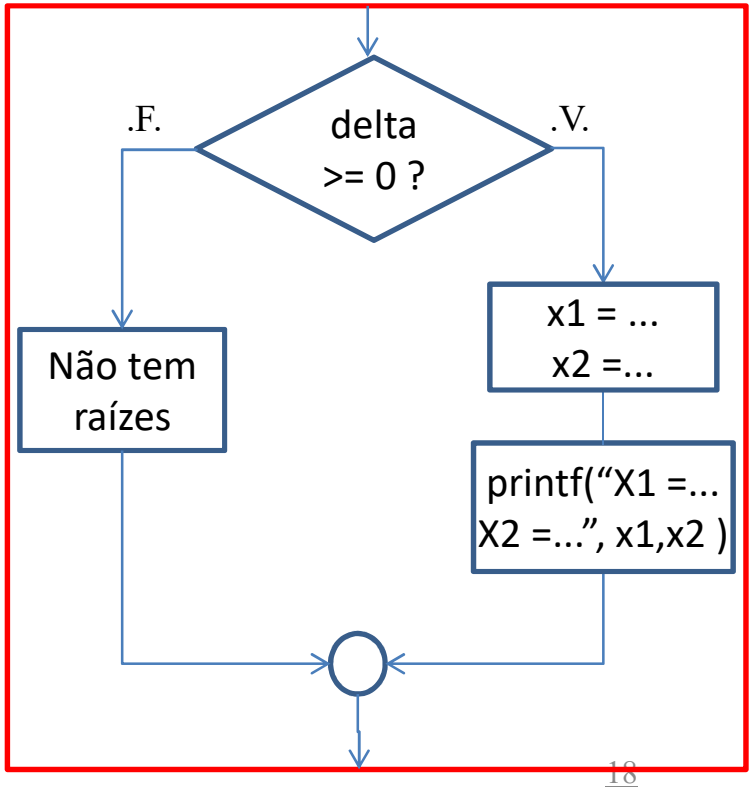


Equação de Segundo Grau:

```
printf("\nRaízes reais de uma equação de segundo grau");
// Entrada dos dados da equação
a = input("entre com o valor de a (a diferente de 0: ");
b = input ("entre com o valor de b: ");
c = input ("entre com o valor de c: ");

// Resolvendo a equação
delta = b^2 - 4 * a * c;
printf("O valor delta é: %g\n", delta)
if delta >= 0 then
        x1 = ( -b + sqrt(delta) ) / (2 * a);
        x2 = ( -b - sqrt(delta) ) / (2 * a);
        printf(" x1 = %g\n", x1);
        printf(" x2 = %g", x2);
else  //(delta será necessariamente < 0)
        printf("\nA equação não tem raízes reais");
end
printf("\n Fim do programa")
```

Diálogo com o usuário



Erros Comuns

- Escrever `delta = b^2 - 4ac`, omitindo os operadores de multiplicação `*`
  - Um *erro de sintaxe*, que é apontado pelo Scilab

- Escrever `r1 = (-b+sqrt(delta))/2*a` , o que na verdade calcula

$$r_1 = \left(\frac{-b+\sqrt{\Delta}}{2} \right) .a$$

  - Um *erro de semântica*, que só pode ser descoberto por meio de testes, que o programador deve fazer

O comando `if` (versão simples)

A condição deve ser uma expressão lógica, ou seja, ao ser avaliada gera "verdade" ou "falso"

```
if <condição> then
    <bloco do então>
end
```

OBSERVAÇÕES:

um bloco é um conjunto de quaisquer comandos Scilab sintaticamente corretos (inclusive outro `if`).

if, then, else e end: são palavras reservadas do Scilab e não podem ser usadas para nomear variáveis.

O comando `if` (versão simples)

Considere que será dado um ponto a mais para quem frequentou pelo menos 75% das aulas de tutoria. Abaixo temos o código que acrescenta este ponto devidamente.

```
nota_alu = input("nota do aluno: ")
freq_tut = input("frequência na tutoria: ")
if (freq_tut >= 75) then
      nota_alu = nota_alu + 1
end
.
.
.
```

O que acontecerá se freq_tut < 75?

Utilizando o if (versão simples)

```
faltas = input("Entre com o número de faltas do aluno.")
n1 = input("Entre com a nota1")
n2 = input("Entre com a nota2")
media = (n1+n2)/2
printf("Média = %g", media)
if faltas > 18 then
  printf("Reprovado por faltas");
end
```

mas ainda não sabemos se o aluno está aprovado ou não, vai depender se a média é >= 6.0 ou não!

O comando `if` (completo)

```
if <condição> then
    <bloco do então>
else
    <bloco do senão>
end
```

`<condição>` é uma expressão relacional. Assim , resulta em um valor verdade (`%t`) ou falso (`%f`).

`<bloco do então>` será executado somente quando a condição resultar em verdadeiro (`%t`).

`<bloco do senão>` será executado somente quando a condição resultar em falso (`%f`).

- Faça um programa que:
  - Leia o nome do usuário
  - Leia o total de pontos feitos pelo usuário
  - Imprima, conforme o caso
  - Se pontos >= 60
    - <usuário>, com <pontos> você passou!
  - Caso contrário (pontos < 60)
    - <usuário>, com <pontos> você não passou!
  - Exemplo de execução:

```
Digite seu nome: José
Digite sua pontuação: 75
José, com 75 pontos passou!
```

Programa PassouNaoPassou.sce

```
// Leitura do nome do usuário
Nome = input("Digite seu nome: “, “s”)

// Leitura da pontuação
Pontos = input(“Digite a sua pontuação: ")

// Impressão do resultado
if Pontos >= 60 then
  printf("%s, com %g pontos você passou!", Nome, Pontos);
else
  printf("%s, com %g pontos você não passou :(",..
  Nome,Pontos);
end
```

Comando continua na próxima linha

Utilizando o if (versão simples)

```
faltas = input("Entre com o número de faltas do aluno.")
n1 = input("Entre com a nota1")
n2 = input("Entre com a nota2")
media = (n1+n2)/2
printf("Média = %g", media)
if faltas > 18 then
  printf("Reprovado por faltas");
end
```

mas ainda não sabemos se o aluno está aprovado ou não, vai depender se a média é >= 6.0 ou não!

Utilizando o if (completo)

```
faltas = input("Entre com as faltas do aluno")
n1 = input("Entre com a nota1")
n2 = input("Entre com a nota2")
media = (n1+n2)/2
printf("Média = %g", media)
if faltas > 18 then
    printf("Reprovado por faltas");
else (faltas <= 18, aluno não está reprovado por faltas, vamos
        verificar se obteve média >= 6.0)
    if media >= 6 then
        printf("Aprovado")
    else (media < 6, reprovado por notas)
        printf("Reprovado por nota")
    end
end
```

Acontece dentro do **else**, ou seja, somente se faltas <= 18

Um jovem programador

Certa vez a mãe disse ao filho estudante de computação:

“Filho, por favor vá ao mercado e compre 1 caixa de leite. Se eles tiverem ovos, traga 6.

Ele retornou com 6 caixas de leite.

A mãe disse: "Porque diabos você comprou 6 caixas de leite?".

Ele disse: "PORQUE ELES TINHAM OVOS!".

O raciocínio na ambiguidade

```
se tiverem ovos então
    traga 6 caixas de leite;
senão
    traga 1 caixa de leite;
```

http://www.hardware.com.br/artigos/computadores-receitas-bolo-aprendendo-comunicar-com-maquina/aprendendo-pensar-como-maquina.html

EXERCÍCIOS

Exercício 1

Codifique um programa que calcule o volume de uma pirâmide, em cm^3, através da fórmula:

```
Volume = 1/3 * ÁreaBase * altura
```

onde

```
ÁreaBase = comprimento * largura
```

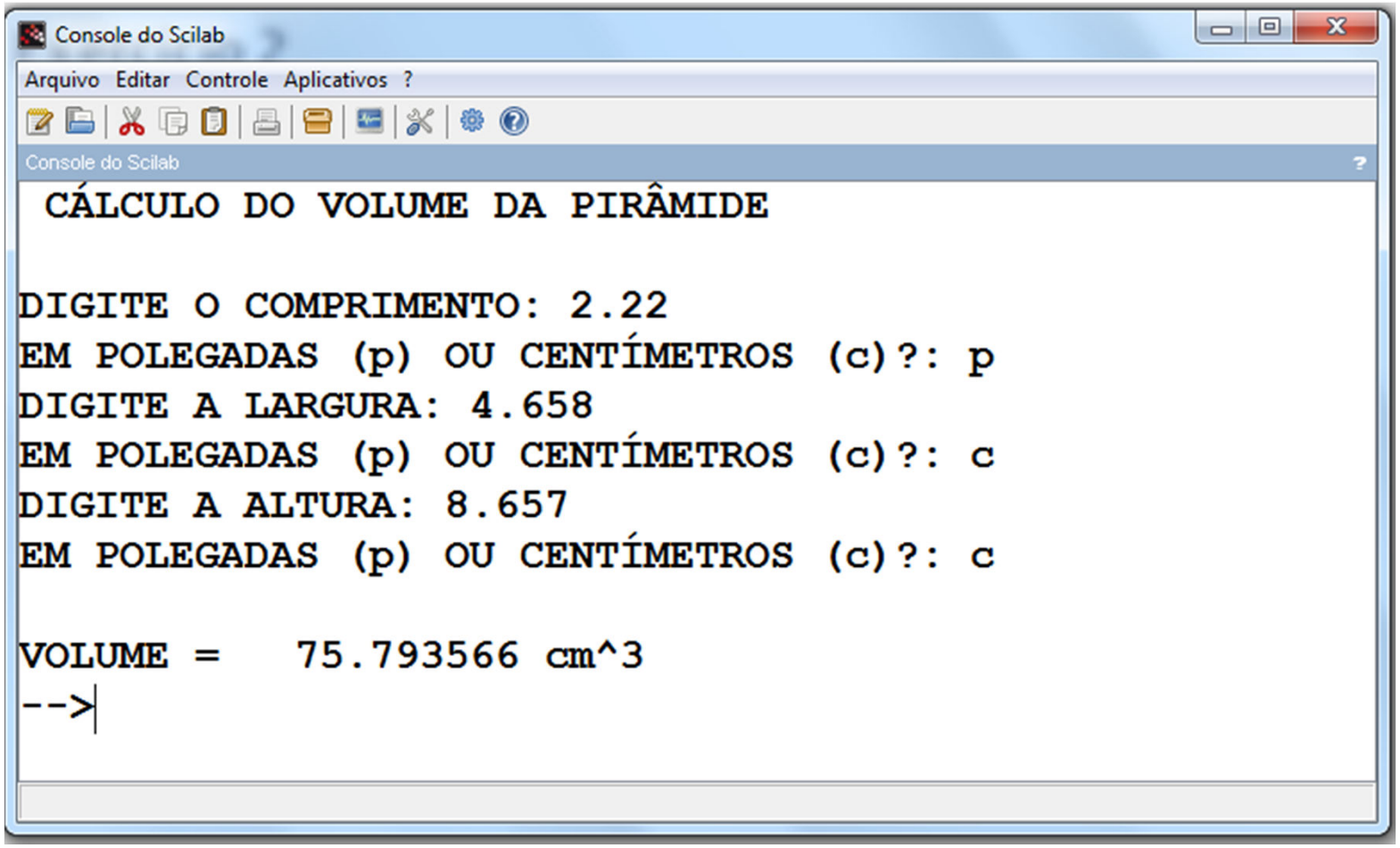
O usuário deve fornecer os valores do comprimento, da largura e da altura. Ao entrar um valor, ele também será solicitado a indicar se o valor digitado foi em polegadas ('p') ou em centímetros ('c').

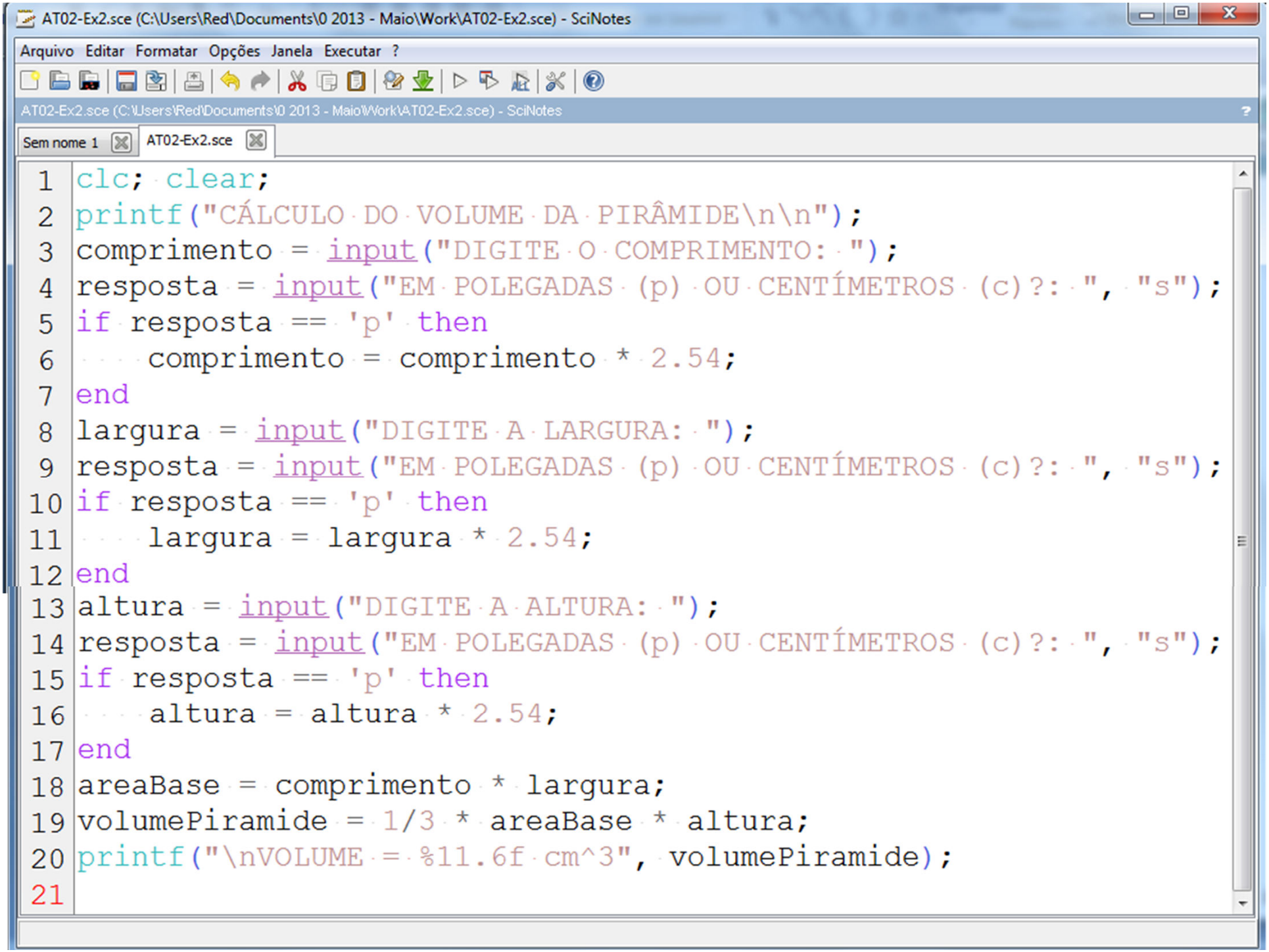
Se a entrada for em polegadas, então o programa a converte automaticamente para centímetros.

Ao final, o programa imprime o volume calculado em cm^3.

Observação: 2.54 cm = 1 polegada.

Exercício 1 - Solução




AT02-Ex2.sce (C:\Users\Red\Documents\0 2013 - Maio\Work\AT02-Ex2.sce) - SciNotes
Arquivo Editar Formatar Opções Janela Executar ?
AT02-Ex2.sce (C:\Users\Red\Documents\0 2013 - Maio\Work\AT02-Ex2.sce) - SciNotes
Sem nome 1
AT02-Ex2.sce
1 clc; clear;
2 printf("CÁLCULO DO VOLUME DA PIRÂMIDE\n\n");
3 comprimento = input("DIGITE O COMPRIMENTO: ");
4 resposta = input("EM POLEGADAS (p) OU CENTÍMETROS (c)?: ", "s");
5 if resposta == 'p' then
6 comprimento = comprimento * 2.54;
7 end
8 largura = input("DIGITE A LARGURA: ");
9 resposta = input("EM POLEGADAS (p) OU CENTÍMETROS (c)?: ", "s");
10 if resposta == 'p' then
11 largura = largura * 2.54;
12 end
13 altura = input("DIGITE A ALTURA: ");
14 resposta = input("EM POLEGADAS (p) OU CENTÍMETROS (c)?: ", "s");
15 if resposta == 'p' then
16 altura = altura * 2.54;
17 end
18 areaBase = comprimento * largura;
19 volumePiramide = 1/3 * areaBase * altura;
20 printf("\nVOLUME = %11.6f cm^3", volumePiramide);
21

Exercício 2

Codifique um programa que converta uma temperatura em graus celsius para graus kelvin, ou para graus fahrenheit.

Após o usuário fornecer a temperatura em celsius, ele de informar se deseja a resposta em Fahrenheit digitando 'f' ou em Kelvin digitando 'k'.

Fórmulas:

$$F = \frac{9}{5}C + 32$$

$$K = C + 273.15$$

Exercício 3

Codifique um programa que leia um valor digitado pelo usuário e depois ele responde se o número é par ou ímpar. Ao executar temos:

`"O NÚMERO DIGITADO É PAR"`

caso contrário imprime a mensagem:

`"O NÚMERO DIGITADO É ÍMPAR"`

Obs. Você pode usar a função modulo(valor, 2) para saber se o valor digitado é um número par ou ímpar.

Exercício 4

Na química, o pH de uma solução aquosa é medido por sua acidez.

A escala do pH varia entre 0 e 14, inclusive. Uma solução como pH igual a 7 é dita neutra; uma solução com o pH maior que 7 é dita básica; e uma solução com o pH menor que 7 é dita ácida.

Codifique um programa que tenha como entrada o pH de uma solução. O programa imprime se o ph é neutro, básico ou ácido.