Ambiente MatLab

Processadores e Linguagens de Baixo Nível

- Memória
- Ciclo de Instrução
  - Lê instrução corrente
  - Interpreta
  - Determina próxima instrução
- Esta é a metáfora que um programador de linguagens de baixo nível deve seguir ao construir programas

Inconvenientes da Programação em Baixo Nível

- Extremamente detalhada
- Instruções refletem a arquitetura específica de cada computador
- Programas em baixo nível não são portáteis
- Propensa a erros

Assemblers (Montadores)

- O processo de montagem manual de uma instrução escrita com mnemônicos foi feito pelos primeiros programadores, ainda na década de 40;
- No início dos anos 50 descobriu-se que um programa de imensa utilidade era um *assembler*: um programa que lê programas escritos com mnemônicos, e monta as instruções em binário para serem executadas por um computador.

A Linguagem Fortran

- A idéia de usar programas para facilitar a programação foi adiante
- Em 1954 a linguagem FORTRAN foi proposta por um grupo da IBM
- O primeiro *compilador* – um programa que traduz programas escritos em linguagem de alto nível para instruções de máquina – foi naturalmente escrito em assembler.
- A máquina era um IBM 704 – um computador com 15K de memória

Alguns Marcos das Linguagens de Programação

- Existem literalmente milhares de linguagens de programação, como estas que descendem do Fortran:
  - 1959 – Cobol
  - 1964 – Basic
  - 1970 – Pascal
  - 1971 – C
  - 1983 – C++
  - 1991 – Python
  - 1995 – Java
  - 1995 – PHP
- Existem ainda linguagens como LISP ou Prolog que seguem outros paradigmas

Matlab

- Inventado no fim dos anos 70 por Cleve Moler
- Lançado comercialmente em 1984 pela empresa MathWorks
- Voltado para engenheiros e cientistas
- Grande facilidade para o tratamento de matrizes (MatLab = Matrix Laboratory)
- É um *interpretador*, isto é, um programa que executa programas, por contraste com um *compilador*, que traduz um programa para instruções de máquina

Scilab

- Desenvolvido desde 1990 por pesquisadores do INRIA e da École Nationale des Ponts et Chaussées (França)
- Muito semelhante ao MatLab – e gratuito!
  - http://www.scilab.org
- É também um interpretador
- A linguagem e o sistema têm o mesmo nome, Scilab
- Atualmente na versão 5.3.1.

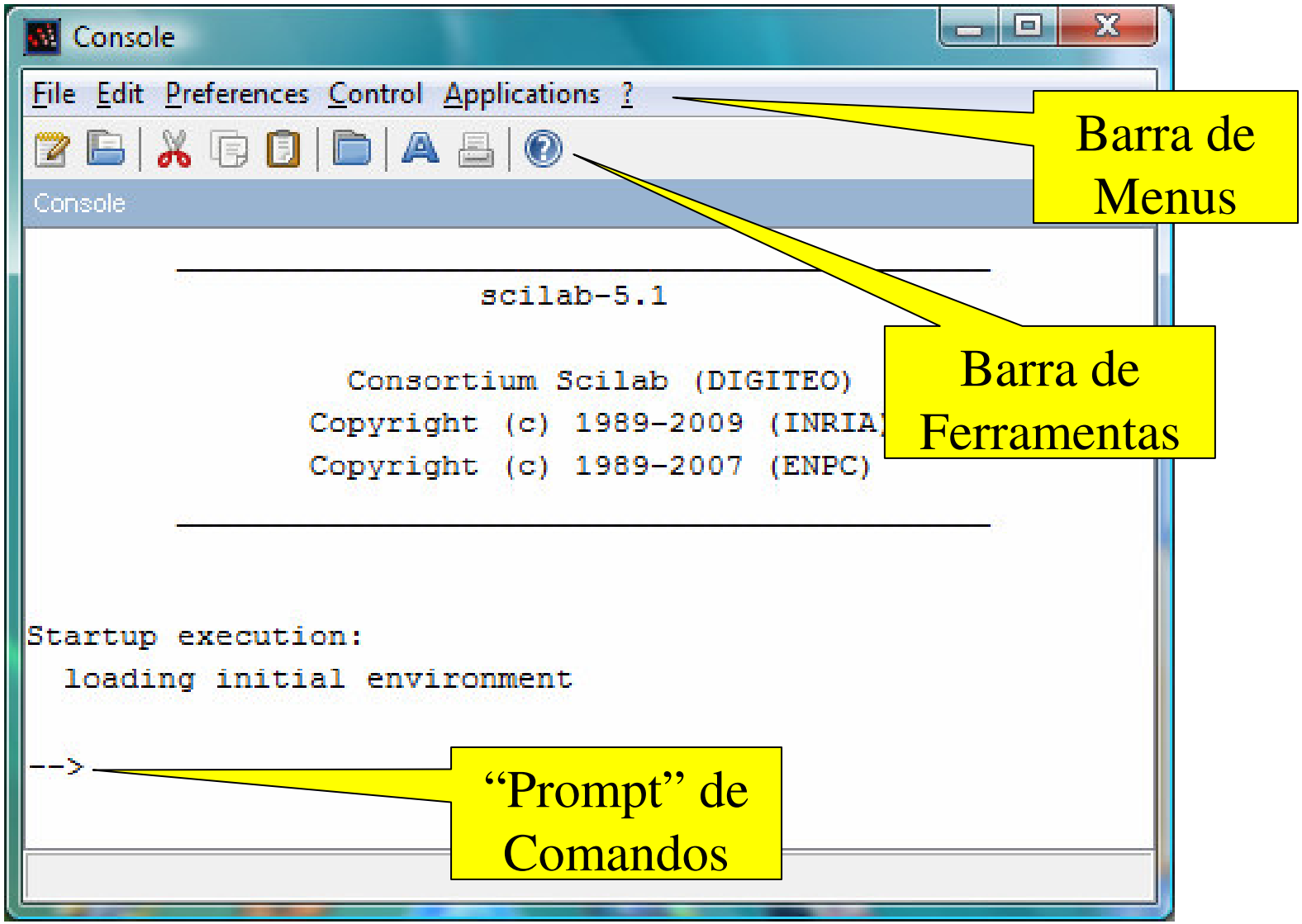
A Linguagem Scilab

- Seu aprendizado exige uma postura paciente, pois envolve no início uma taxa inicial de memorização
- Também como nas linguagens naturais, a fluência vem com o uso

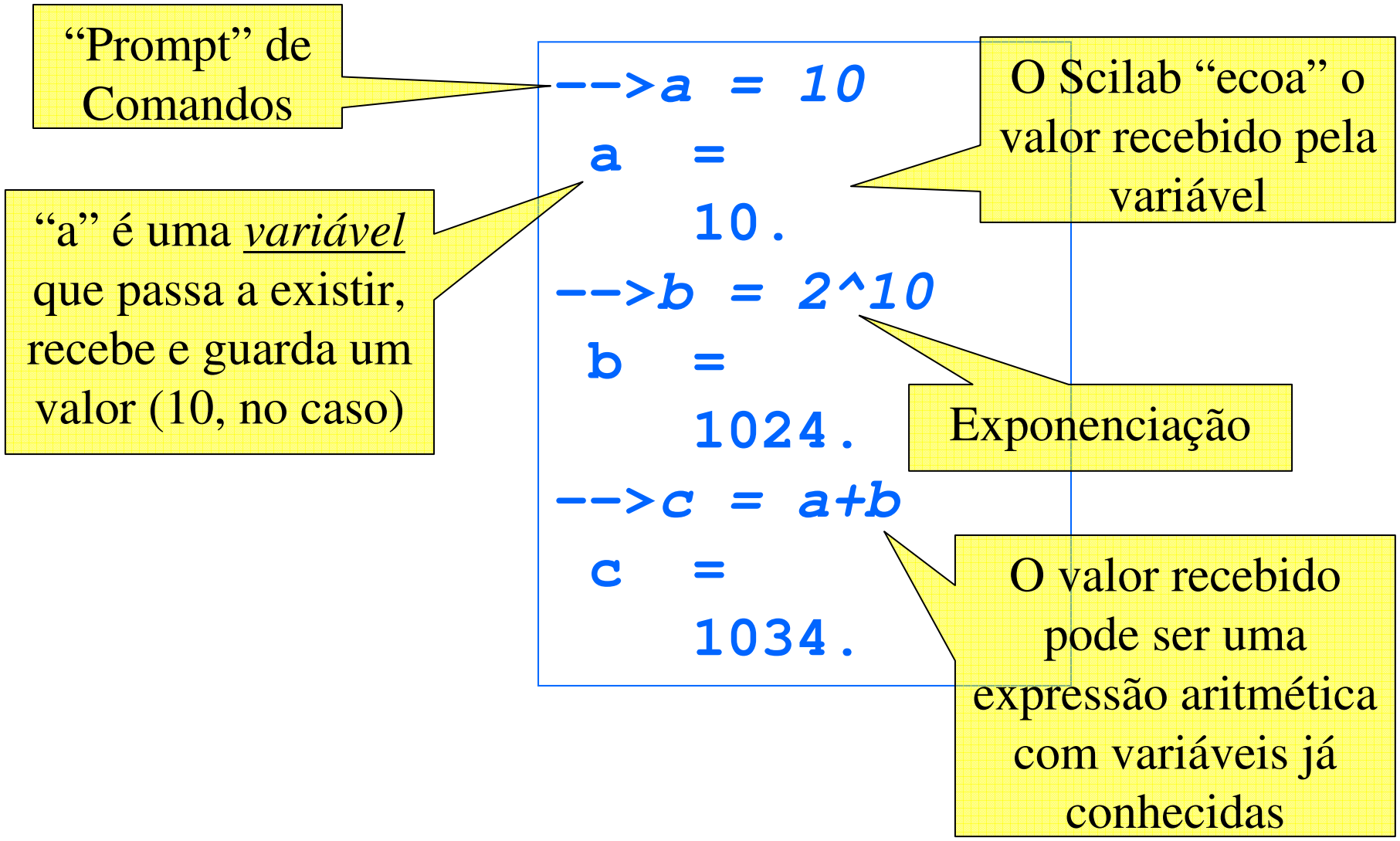
O Ambiente Scilab

- Interpreta comandos e programas
- Oferece um editor para a construção de programas, o SciPad
- Emite mensagens de erros relativos tanto à obediência de comandos e programas às regras da linguagem como a problemas na exeçução, como divisão por zero
- O ambiente também requer familiarização para uso eficiente

Tela Inicial Scilab



Variáveis e Comandos de Atribuição – 1



Variáveis

- São nomes para espaços de memória gerenciados pelo Scilab
- O programador não precisa ter qualquer idéia de como isso é feito
- Variáveis têm seus nomes escolhidos pelo programador segundo algumas regras
  - O primeiro caractere do nome deve ser uma letra, ou qualquer caractere dentre '%', '_', '#', '!', '$‘ e ‘?'.
  - Os outros podem ser letras ou dígitos, ou qualquer caractere dentre '_', '#', '!', '$‘ e '?'

Nomes de Variáveis

- Válidos:
  - `a`, `A`, `jose`, `total_de_alunos`, `#funcionarios`
- Não válidos
  - `1Aluno` (o primeiro caractere é um algarismo)
  - `total de alunos` (tem espaços)
  - `José` (é acentuado)

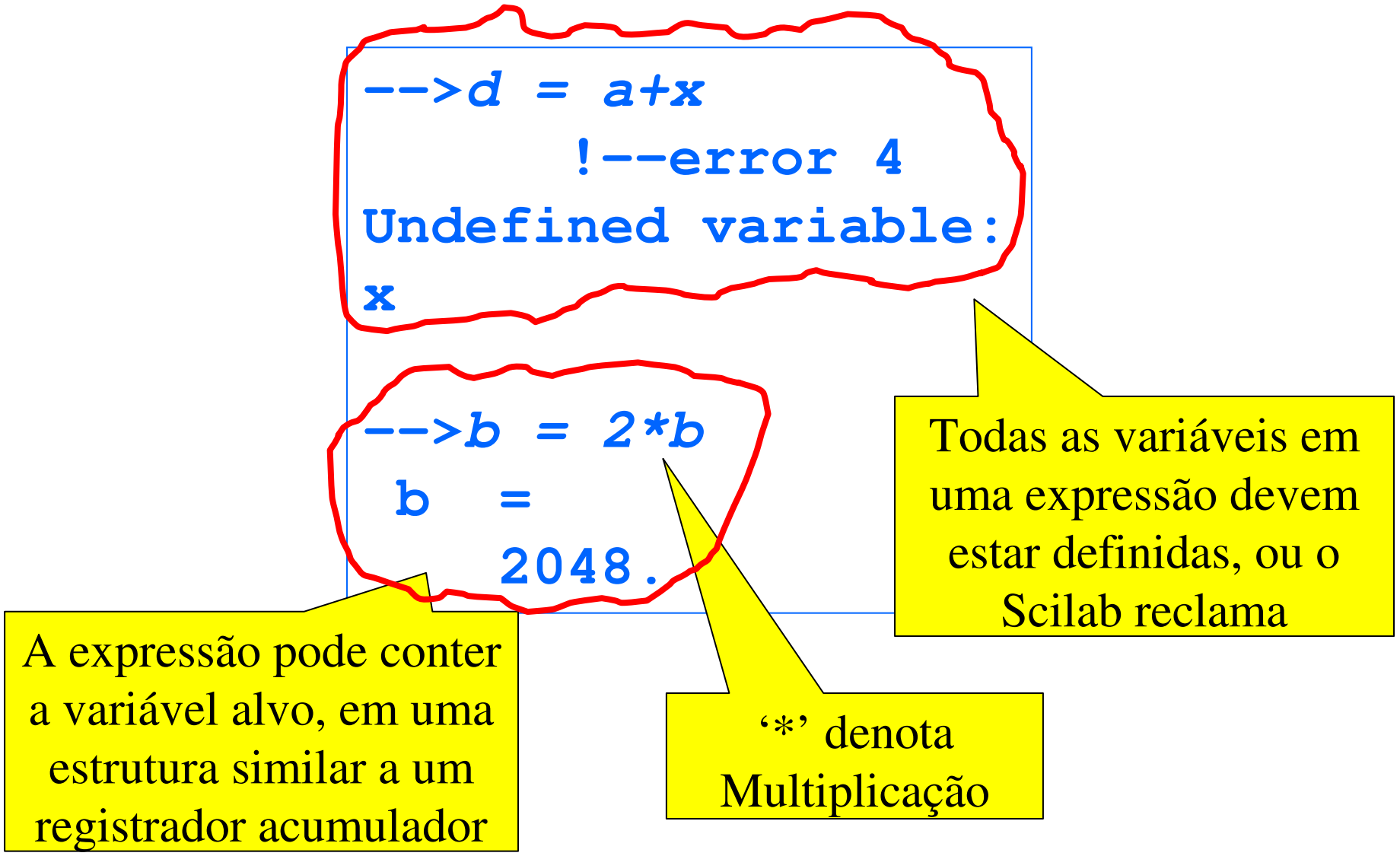
Comando de Atribuição

- Forma:

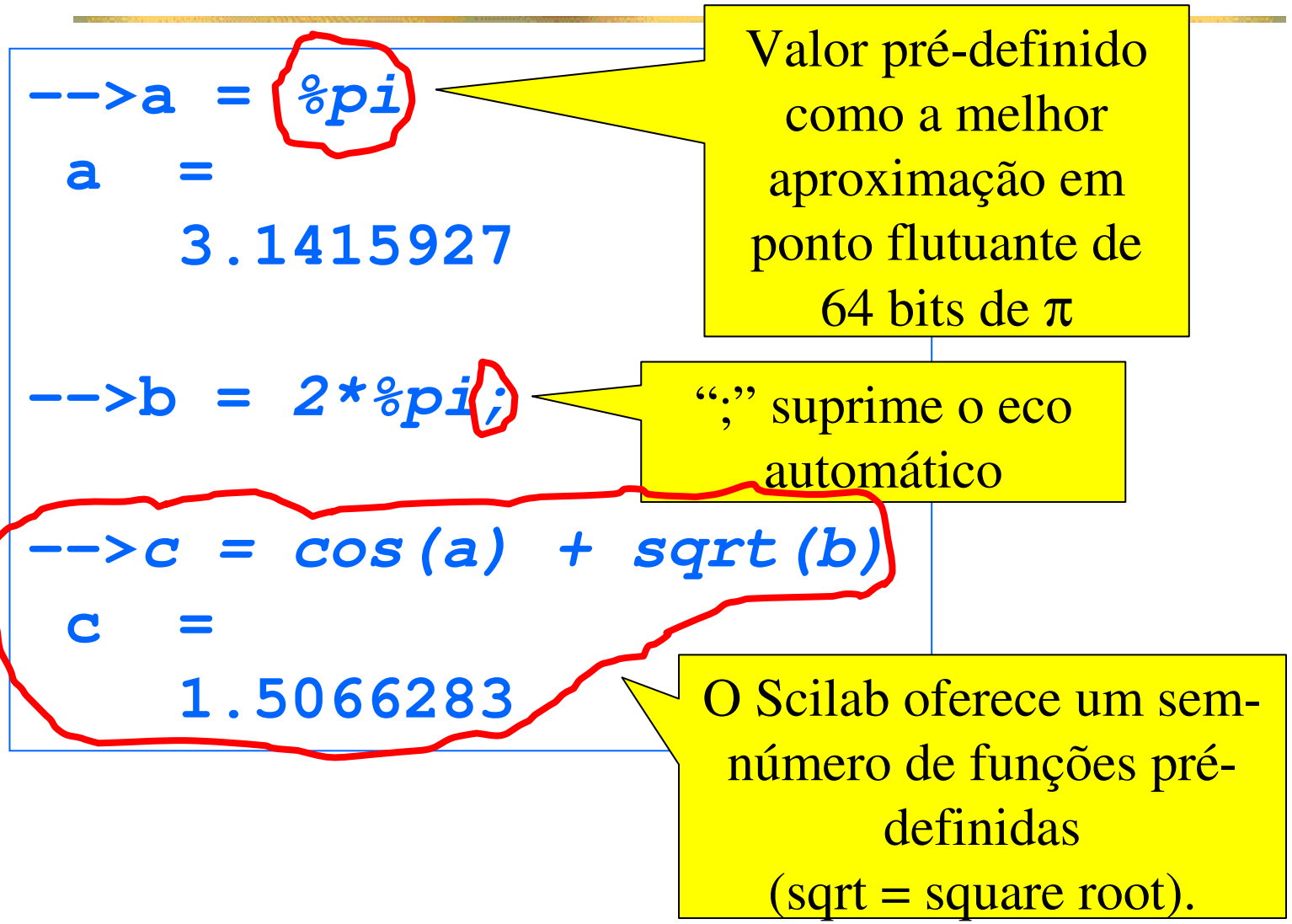
  <variável alvo> = <expressão>

- A <variável alvo>, se não existia, passa a existir
- Se existia, o valor anterior é perdido
- A <expressão> é calculada, e o resultado é atribuído à <variável alvo>
- O comando de atribuição é a construção básica de transformação de informação

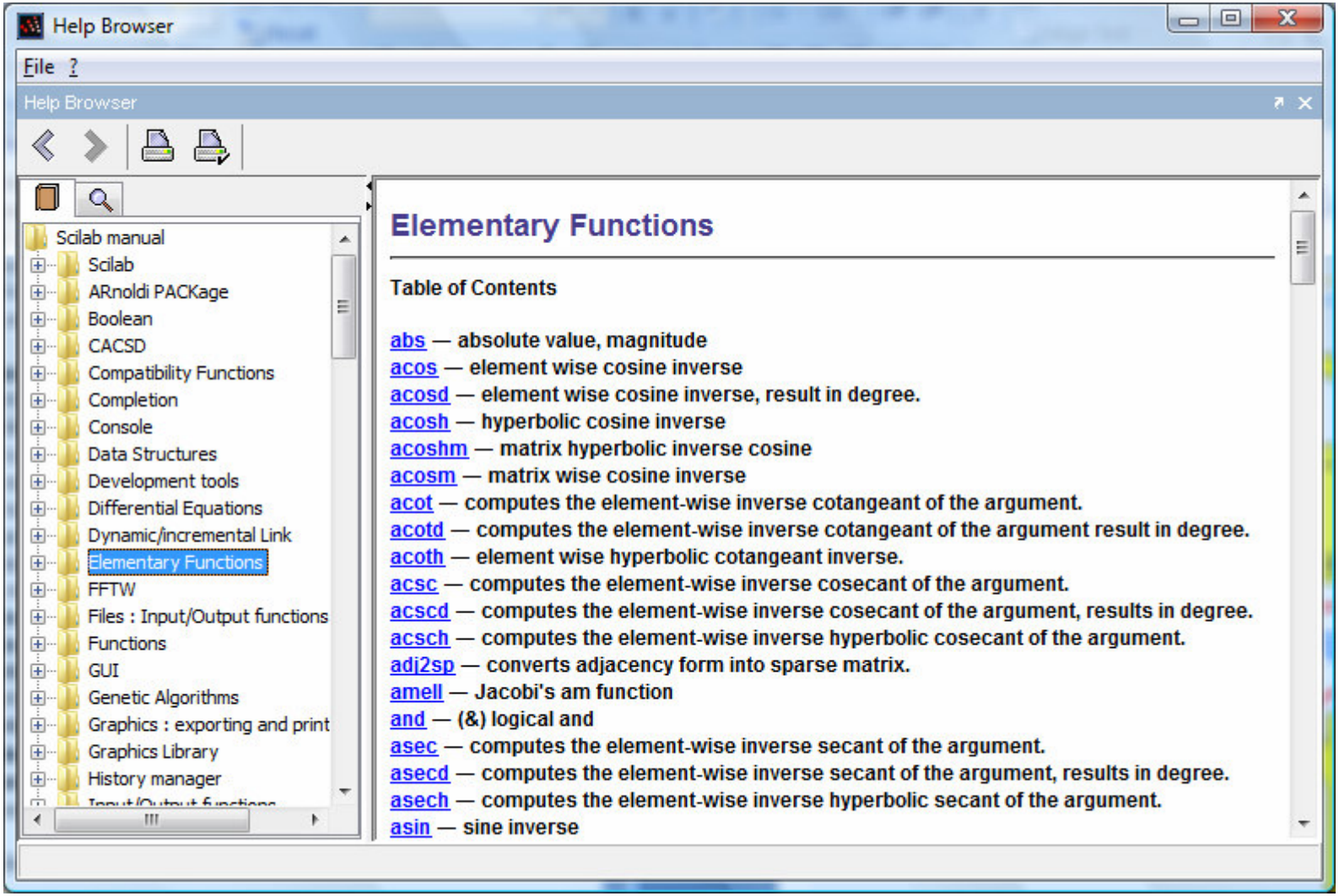
Variáveis e Comandos de Atribuição - 2



Variáveis e Comandos de Atribuição – 3



Help - Funções Elementares do Scilab



Expressões Aritméticas

- Expressões podem ser arbitrariamente complicadas
- A ordem em que operadores são aplicados não é óbvia:

Qual valor o comando `x = 2^3*4` atribui a `x`,

$$2^3.4 = 8.4 = 32$$

ou

$$2^{3.4} = 2^{12} = 4096 \ ?$$

Prioridades entre Operadores

Prioridade	Operação	Associatividade
1ª	Potenciação	Da direita para a esquerda
2ª	Multiplicação, divisão	Da esquerda para a direita
3ª	Adição, subtração	Da esquerda para a direita

- Parênteses podem alterar prioridades

Prioridades e Parênteses

```
-->2^3*4
 ans  =     32.
-->2^(3*4)
 ans  =     4096.
-->2^3^4
 ans  =     2.418D+24
-->2^(3^4)
 ans  =     2.418D+24
-->(2^3)^4
 ans  =     4096.
-->2*3+4
 ans  =     10.
-->2*(3+4)
 ans  =     14.
```

Notação Scilab (e Fortran, e C, e Java, e ...) para 2.418×10^{24}

Recomendação: use parênteses; é mais seguro

Equações de Segundo Grau: O Scilab como Calculadora - 1

- Equação $$ax^2 + bx + c = 0$$
- Raízes (reais se Δ>0)

$$r_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$r_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

$$\Delta = b^2 - 4ac$$

- Calcular as raízes para a = 534.2765, b = 9987.3431 e c = 225.7690

Equações de Segundo Grau: O Scilab como Calculadora – 2

- Inicialização

```
-->a = 534.2765
 a   =
    534.2765
-->b = 9987.3431
 b   =
    9987.3431
-->c = 225.7690
 c   =
    225.769
```

Equações de Segundo Grau: O Scilab como Calculadora – 3

- Cálculo das Raízes

```
-->delta = b^2 - 4*a*c
 delta  =
    99264530.
-->r1 = (-b+sqrt(delta))/(2*a)
 r1  =
  - 0.0226329
-->r2 = (-b-sqrt(delta))/(2*a)
 r2  =
  - 18.670578
```

Erros Comuns

- Escrever `delta = b^2 - 4ac`, omitindo os operadores de multiplicação
- Escrever `r1 = (-b+sqrt(delta))/2*a` o que na verdade calcula

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

Verificando os Resultados

```
-->a*r1^2 + b*r1 + c
 ans  =
    3.865D-12
-->a*r2^2 + b*r2 + c
 ans  =
  - 2.274D-13
```

Equações de Segundo Grau: O Scilab como Calculadora – 4

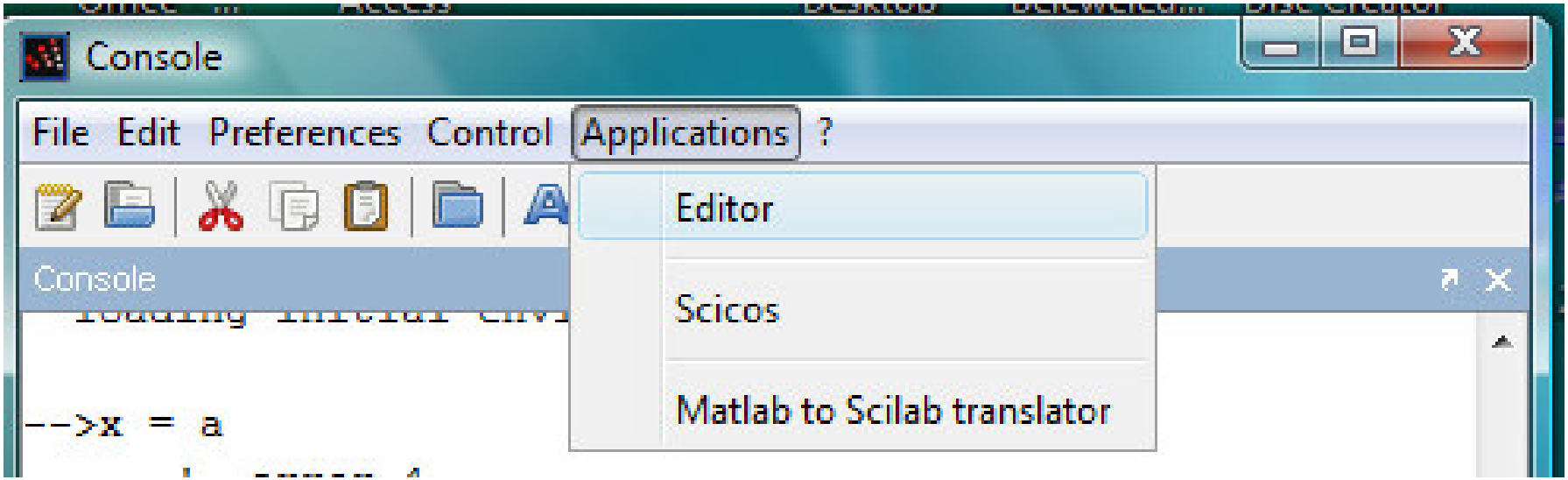
- Ganhos com relação a uma calculadora de mão:
  - Variáveis evitam re-digitações
  - Resultados intermediários são memorizados
  - Fórmulas facilitam a composição de expressões
- Problemas
  - Nova equação, redigitação das fórmulas
- Solução
  - Usar o Scilab como interpretador de *programas*

Programas Scilab

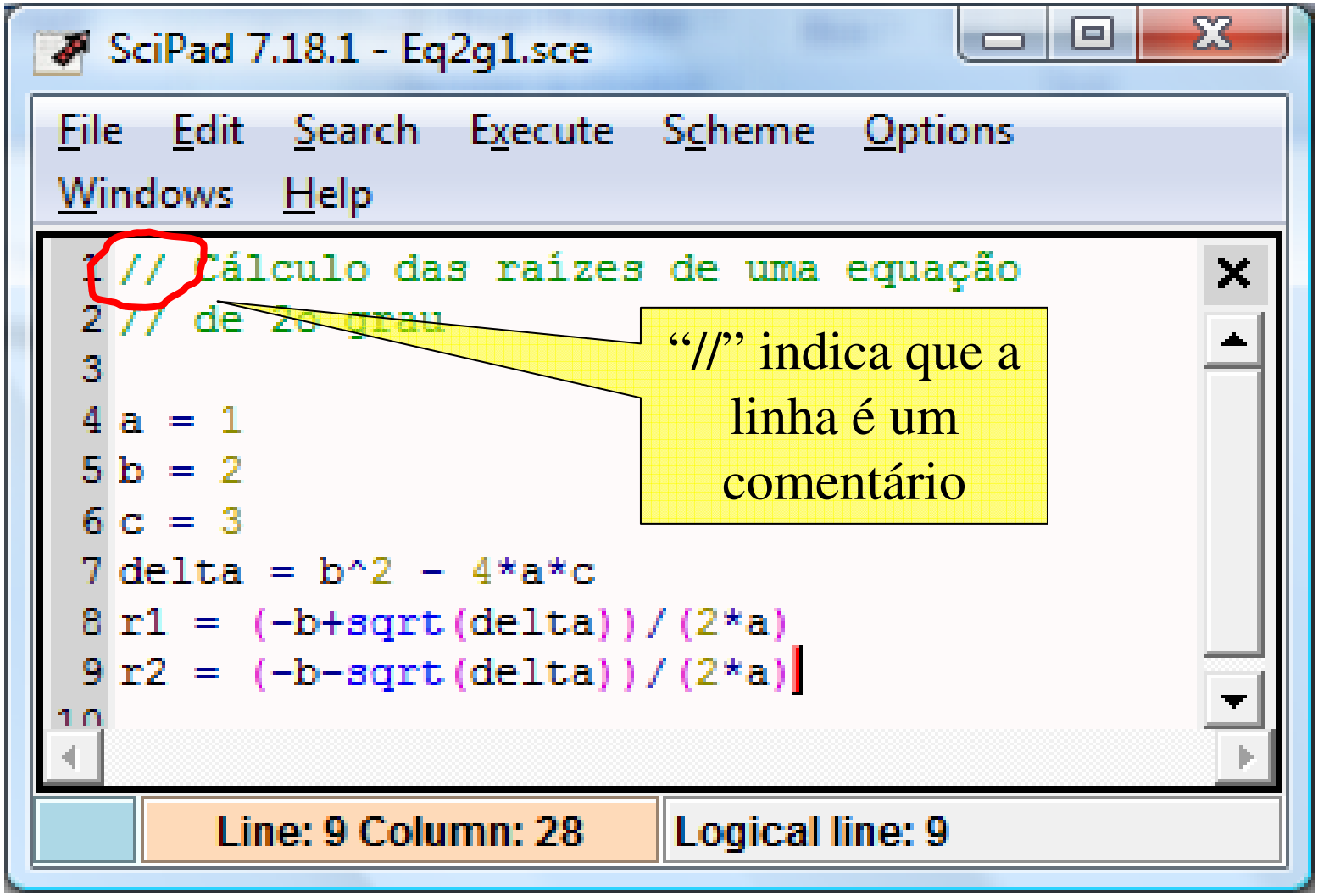
- Programas são arquivos ASCII (caracteres sem formatação) com a terminação **.sce**
- Um arquivo-programa contém comandos Scilab
- Um programa é construído usando o editor SciPad
- Um programa é executado seguindo o menu Execute/Load into Scilab do editor Scipad
- Sua execução equivale à digitação na console dos comandos no arquivo

O Editor SciPad

- *Use sempre o SciPad para construir programas*
- *Nunca use o Word, pois ele introduz bytes de formatação*



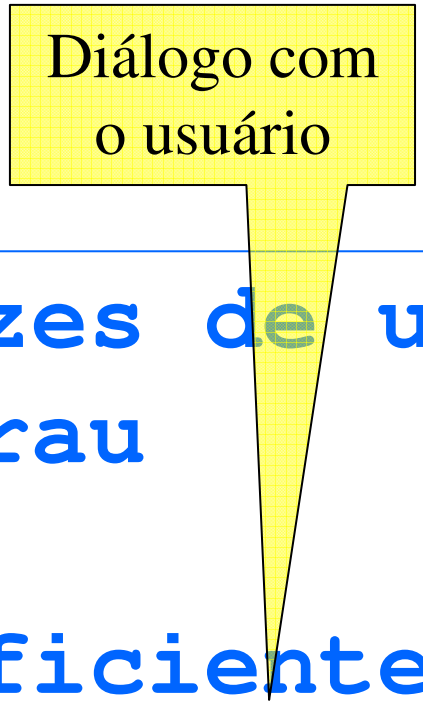
Equações de Segundo Grau: Programa Scilab – Eq2g1.sce



Equações de Segundo Grau: Programa Scilab – Eq2g_1_0.sce

- Para uma nova equação, basta substituir no programa os valores dos novos coeficientes
- As chances de erros de digitação são consideravelmente diminuídas
- Entretanto, a prática de modificar programas a cada execução não é recomendada
- O melhor é fazer com que o programa *leia* os valores dos coeficientes a cada execução

Equações de Segundo Grau: Programa Scilab – Eq2g_2_0.sce - 1




```
// Cálculo das raízes de uma
// equação de 2o grau

// Entrada dos coeficientes
a = input("Valor de a:");
b = input("Valor de b:");
c = input("Valor de c:");
```

Equações de Segundo Grau: Programa Scilab – Eq2g_2_0.sce - 2

```
// Cálculo das raízes de uma equação
// de 2o grau

a = input("Digite o valor de a:")
b = input("Digite o valor de b:")
c = input("Digite o valor de c:")

delta = b^2 - 4*a*c
r1 = (-b+sqrt(delta))/(2*a)
r2 = (-b-sqrt(delta))/(2*a)
```

Execução do Programa Eq2g2.sce

```
Digite o valor de a:1
 a   =
    1.
Digite o valor de b:2
 b   =
    2.
Digite o valor de c:3
 c   =
    3.
 delta   =
  - 8.
 r1   =
  - 1. + 1.4142136i
 r2   =
  - 1. - 1.4142136i
```

O Programa Eq2g3.sce – Especificação

- O programa só deverá calcular as raízes quando elas forem reais
- A saída do programa deverá ser uma frase como “As raízes são xxxx e xxxx”, quando as raízes forem reais, e senão, “As raízes são complexas.”

O Comando if

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

Cláusula else vazia

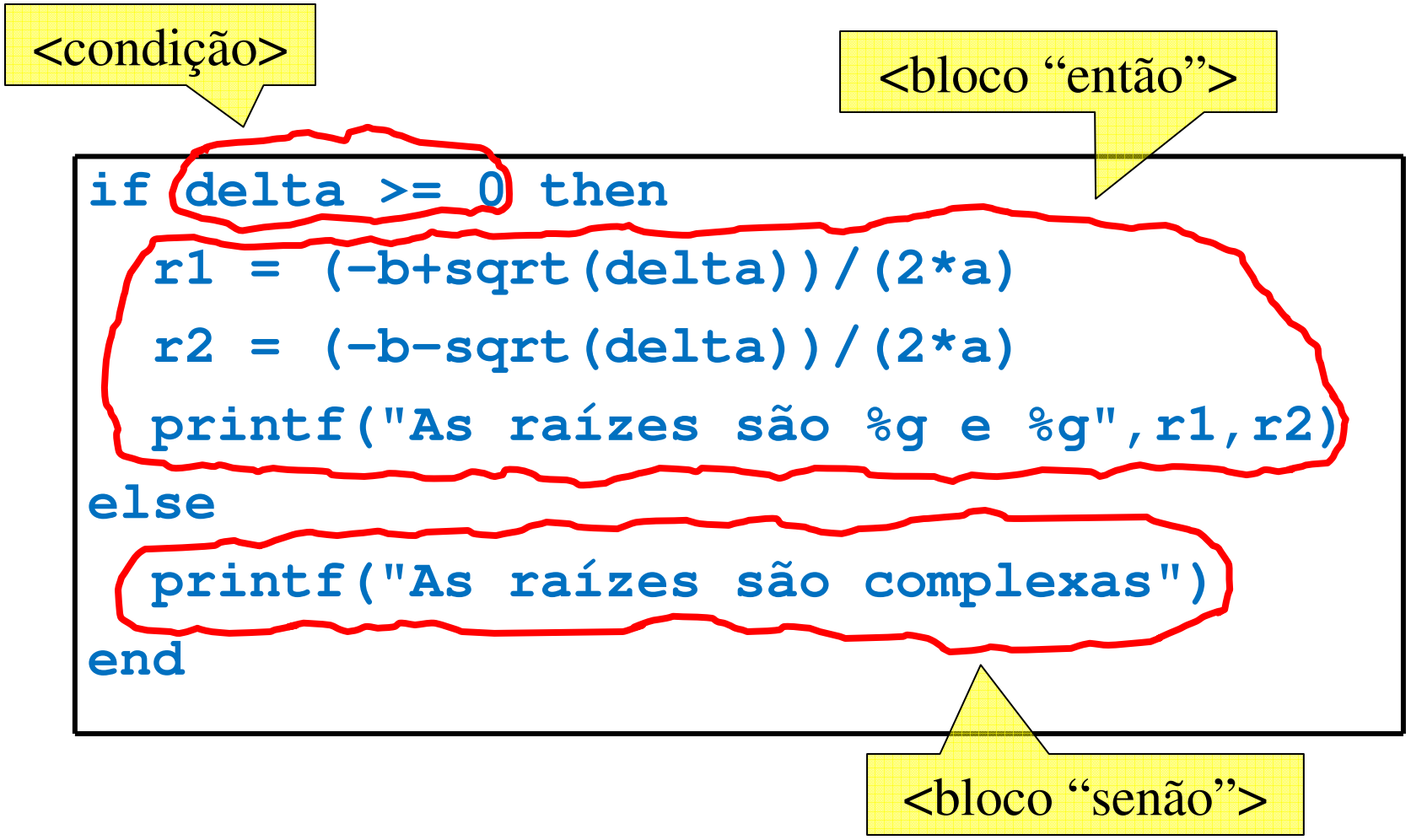
```
if <condição> then
   <bloco "então">
else
   // Nenhum comando aqui
end
```

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

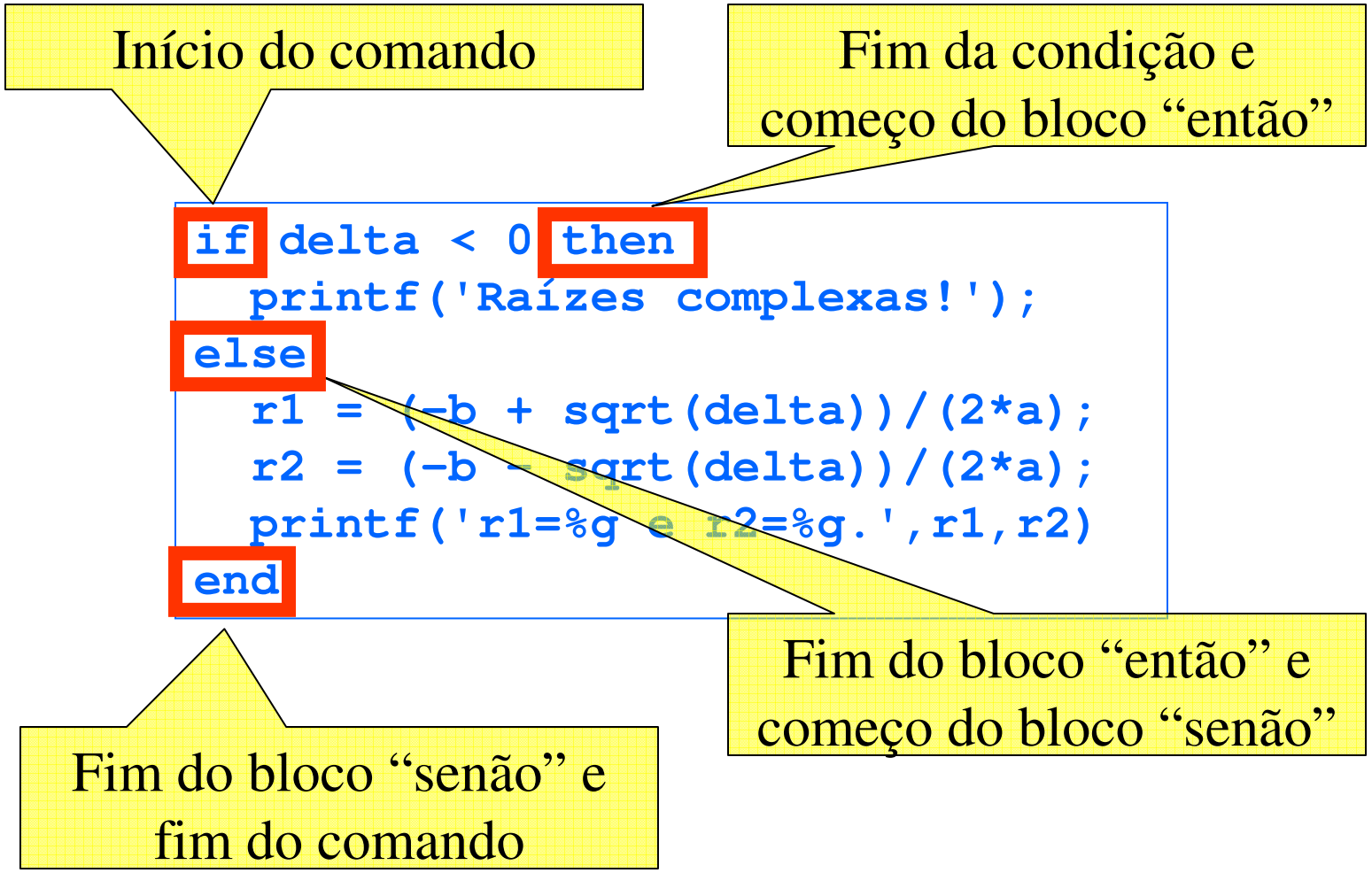
Equações de Segundo Grau: Programa Scilab – Eq2g_3.sce - 2

```
//Cálculo e impressão das raízes
delta = b^2 - 4*a*c;
if delta > 0 then
  r1 = (-b+sqrt(delta))/(2*a);
  r2 = (-b-sqrt(delta))/(2*a);
  printf("Raízes: %g e %g.",r1,r2);
else
  printf("Raízes complexas.")
end
```

Partes de um comando If



Palavras-chave de um Comando if



Operadores Relacionais

>	maior que
>=	maior ou igual a
<	menor que
<=	menor ou igual a
==	igual a
<> ou ~=	diferente de

Problema: Cálculo do Fatorial

```
-->fat = 1
 fat  =
    1.
-->fat = fat*2
 fat  =
    2.
-->fat = fat*3
 fat  =
    6.
```

- Outra coisa chata, repetitiva e propensa a erros
- Solução? Programa Scilab!

O Comando Repetitivo for

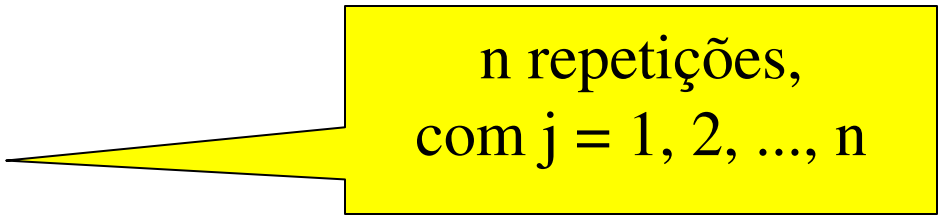
- O comando

```
for j = 1:5
   // corpo do "for"
end
```

- resulta em 5 execuções do corpo do **`for`**, com j = 1, 2, 3, 4 e 5.

Programa - Fatorial

```
n = input("Por favor entre com o valor de n");
f = 1;
for j=1:n
  f = f*j;
end;
printf("O fatorial de %d é igual a %d",n,f)
```



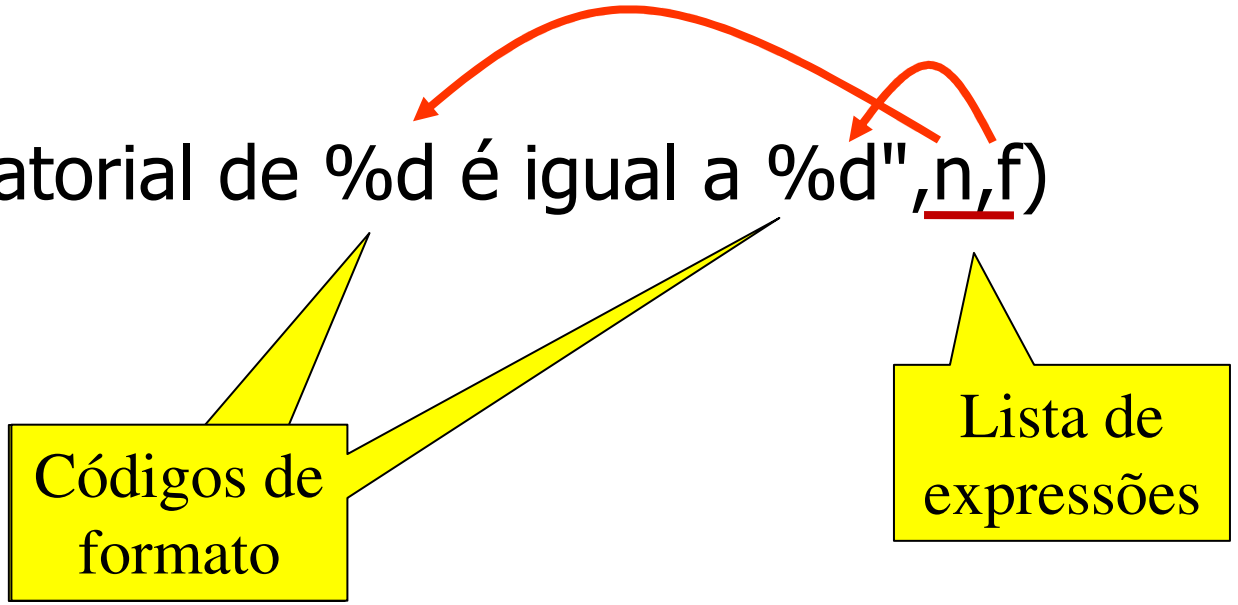


Tabela de Senos

x	seno(x)
0.0	0.0000
0.2	0.1987
0.4	0.3894
0.6	0.5646
0.8	0.8415

Forma Geral de um comando for

```
for <variável> = <inicial>:<passo>:<limite>
   <bloco for>;
end
```

Comando for com passo diferente de 1

```
for i = 1:2:10
  printf('\ni = %g',i);
end
```

```
i = 1
i = 3
i = 5
i = 7
i = 9
```

i varia de 2 em 2

Saída

Repare que i não assumiu o limite superior do loop

Comando for com passo negativo

```
for i = 20:-2:16
  printf('\ni = %g',i);
end
```

```
i = 20
i = 18
i = 16
```

Comando for com controle fracionário

A variável de controle pode assumir valores não inteiros

```
for x = 0:0.3:0.7
  printf('\nx = %g',x);
end
```

```
x = 0
x = 0.3
x = 0.6
```

Saída

Tabela de Senos
1ª tentativa

```
// Tabela da função Seno
for x = 0:0.2:2*%pi
  printf("%g %g",x, sin(x))
end
```

```
-->
 0 00.2 0.1986690.4 0.3894180.6 0.5646420.8 0.7173561 0.841471
```

Tabela de Senos
2ª Tentativa

```
// Tabela da função Seno
for x = 0:0.2:2*%pi
  printf("\n %g %g",x, sin(x))
end
```

Tabela de Senos
2ª Tentativa

```
0 0
0.2 0.198669
0.4 0.389418
0.6 0.564642
0.8 0.717356
1 0.841471
1.2 0.932039
```

Tabela de Senos

```
// Tabela da função Seno
// Impressão do cabeçalho
printf("\n x   seno(x)")
// Impressão das linhas da tabela
for x = 0:0.2:2*%pi
  printf("\n%3.1f %7.4f",x, sin(x))
end
```

Tabela de Senos

x	seno(x)
0.0	0.0000
0.2	0.1987
0.4	0.3894
0.6	0.5646
0.8	0.7174
1.0	0.8415
1.2	0.9320

“Indentação”

```
if delta < 0 then
  printf('Raízes complexas!');
else
  r1 = (-b + sqrt(delta))/(2*a);
  r2 = (-b - sqrt(delta))/(2*a);
  printf('r1=%g e r2=%g.',r1,r2)
end
```

Mais legível

```
if delta < 0 then
printf('Raízes complexas!');
else
r1 = (-b + sqrt(delta))/(2*a);
r2 = (-b - sqrt(delta))/(2*a);
printf('r1=%g e r2=%g.',r1,r2)
end
```

Menos legível

“Indentação”

- Para o Scilab, os dois programas são absolutamente equivalentes, mas...
- Para nós, humanos, a disposição do texto do programa afeta (e muito) a legibilidade (o que se aplica à correção de programas pelo professor e pelos monitores: a indentação é exigida)
- Blocos “então” e “senão” são mais facilmente identificados com indentação.
- Os possíveis fluxos de execução ficam mais claros.

Conclusões

- O Scilab oferece um nível de conforto muito superior às linguagens de máquina
- A interface de comandos é muito boa para pequenos cálculos e para o teste de comandos.
- Cálculos mais complexos são (muito) melhor implementados por programas

Conclusões

- Na linguagem Scilab encontramos construções importantes como `if` e `for`, e também detalhes, como a inclusão ou não de um "`;`" ao fim de um comando, ou os códigos `%d` ou `%f` de conversão em um `printf`,
- Operações como salvar ou executar um programa também merecem sua atenção.
- Não tenha medo! Na dúvida, faça experimentos – o Scilab não estraga.