



Semana 03:

Comandos de desvio de fluxo. Expressões lógicas.

Material Didático Unificado.

1

BCC701
2012/01

Agenda

- Introdução;
- Comandos de desvio de fluxo;
- Expressões lógicas;
- Exercícios.

2

BCC701
2012/01

Introdução:

Comandos de desvio de fluxo;
Expressões lógicas;
Exercícios.

INTRODUÇÃO

3

BCC701
2012/01

Introdução

Programação estruturada

- Relembrando o conceito de programação estruturada:
 - **Programação estruturada** é uma forma de programação de computadores que preconiza que todos os programas possíveis podem ser reduzidos a apenas **três estruturas: sequência, decisão e iteração.**

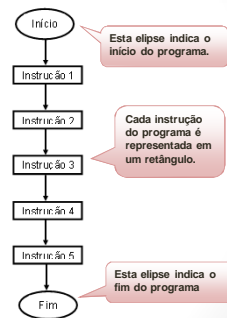
4

BCC701
2012/01

Introdução

Sequência

- Nas aulas anteriores os programas continham apenas a primeira estrutura;
- Eles eram formados por uma **sequência de instruções**, ou comandos, **executados sequencialmente**, conforme o fluxograma ao lado.



5

BCC701
2012/01

Introdução

Decisão: Desvio de fluxo

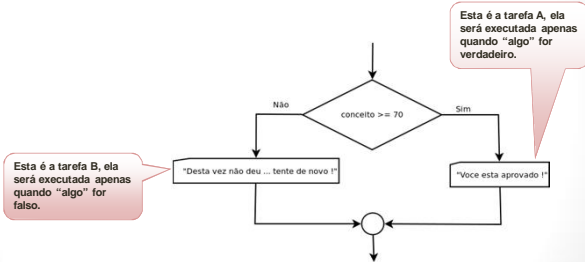
- A segunda estrutura é utilizada quando é necessário realizar um **desvio de fluxo**, realizado com base em uma **decisão**;
- **Decisão:**
 - Se algo for verdadeiro:
 - Faça a tarefa A;
 - Caso contrário:
 - Faça a tarefa B;
- O **desvio de fluxo** é caracterizado pela “escolha” (decisão) entre executar a **tarefa A** ou executar a **tarefa B**.

6

BCC701
2012/01

Decisão: Desvio de fluxo

- Fluxograma para a decisão;



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

7

BCC701
2012/01

Introdução;
Comandos de desvio de fluxo;
Expressões lógicas;
Exercícios.

COMANDOS DE DESVIO DE FLUXO

8

BCC701
2012/01

Comando condicional if

- Vamos retornar ao exemplo da equação de segundo grau:
 - $ax^2 + bx + c = 0$;
- Para que esta equação seja de fato de segundo grau, o valor de **a** não pode ser **0 (zero)**, pois acarretaria em um erro;

```

Defina um valor para a: 0
Defina um valor para b: 5
Defina um valor para c: 6
x1 = (-b + sqrt(delta)) / (2*a);
                                     |--error 27
Divisão por zero...
at line 10 of exec file called by :
o de 2o grau IF.sce', -1
  
```

- Assim, vamos aprimorar nosso exemplo, para fazer uma verificação do valor de **a**.

9

BCC701
2012/01

Comando condicional if

- Novo exemplo para a equação de segundo grau:

```

a = input("Defina um valor para a: ");
if a == 0 then
    printf("O coeficiente a deve ser diferente de 0.\n");
    a = input("Defina um valor para a: ");
end
b = input("Defina um valor para b: ");
c = input("Defina um valor para c: ");
delta = (b*b)-4*a*c;
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (-b - sqrt(delta)) / (2*a);
printf("A raiz x1 é %g.\n", x1);
printf("A raiz x2 é %g.", x2);
  
```

10

BCC701
2012/01

Comando condicional if

- Resultado:

```

Defina um valor para a: 0
O coeficiente a deve ser diferente de 0.
Defina um valor para a: -5
Defina um valor para b: 5
Defina um valor para c: 6
A raiz x1 é -0.704159.
A raiz x2 é 1.70416.
-->
  
```

11

BCC701
2012/01

Comando condicional if

- Problema:** E se o usuário digitar **0** novamente?

```

Defina um valor para a: 0
O coeficiente a deve ser diferente de 0.
Defina um valor para a: 0
Defina um valor para b: 5
Defina um valor para c: 6
x1 = (-b + sqrt(delta)) / (2*a);
                                     |--error 27
Divisão por zero...
at line 10 of exec file called by :
o de 2o grau IF.sce', -1
-->
  
```

12

BCC701
2012/01

Comando condicional if

- Mais um aprimoramento:

```
a = input("Defina um valor para a: ");
if (a == 0) then // O parêntese é bom para facilitar a leitura
    printf("O coeficiente a deve ser diferente de 0.\n");
else
    b = input("Defina um valor para b: ");
    c = input("Defina um valor para c: ");
    delta = (b*b)-4*a*c;
    x1 = (-b + sqrt(delta)) / (2*a);
    x2 = (-b - sqrt(delta)) / (2*a);
    printf("A raiz x1 é %g.\n", x1);
    printf("A raiz x2 é %g.", x2);
end
```

13

Comando condicional if

- Resultado 1:

```
Defina um valor para a: 0
O coeficiente a deve ser diferente de 0.
-->
```

- Resultado 2:

```
Defina um valor para a: -5
Defina um valor para b: 5
Defina um valor para c: 6
A raiz x1 é -0.704159.
A raiz x2 é 1.70416.
-->
```

14

Comando condicional if

- Mais um problema: E quando o delta for menor que 0 (zero)?
- Executando nosso programa o resultado seria:

```
Defina um valor para a: 1
Defina um valor para b: 2
Defina um valor para c: 3
A raiz x1 é -1.
A raiz x2 é -1.
-->
```

- Estes valores para x1 e x2 estão corretos?

15

Comando condicional if

- Mais um aprimoramento:

```
:
:
delta = (b*b)-4*a*c;
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (-b - sqrt(delta)) / (2*a);
if (delta >= 0) then
    printf("A raiz x1 e %g.\n", x1);
    printf("A raiz x2 e %g.", x2);
else
    printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
    printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
end
:
:
```

16

Comando condicional if

- Mais um aprimoramento:

```
:
:
delta = (b*b)-4*a*c;
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (-b - sqrt(delta)) / (2*a);
if (delta >= 0) then
    printf("A raiz x1 e %g.\n", x1);
    printf("A raiz x2 e %g.", x2);
else
    printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
    printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
end
:
:
```

O Scilab possibilita manipular números complexos de forma simples.

real(x1) => retorna a parte real.

imag(x1) => retorna a parte imaginária.

Exemplos de números complexos:

z1 = 3 + 4 * %i;

z2 = 1 - %i;

z3 = z1 + z2;

z4 = z1 * z2;

17

Comando condicional if

- Resultado 1:

```
Defina um valor para a: -5
Defina um valor para b: 5
Defina um valor para c: 6
A raiz x1 é -0.704159.
A raiz x2 é 1.70416.
-->
```

- Resultado 2:

```
Defina um valor para a: 1
Defina um valor para b: 2
Defina um valor para c: 3
A raiz x1 e -1 + 1.41421*i.
A raiz x2 e -1 + -1.41421*i.
-->
```

Ainda poderíamos melhorar a formatação, verificando se a parte imaginária é menor do que zero. Façamos isso como exercício.

18

Comando condicional if

- Sintaxe:

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

A <condição> deve ser uma expressão lógica, ou seja, deve ser uma expressão que resulta em um valor verdadeiro (%t) ou falso (%f).

O <bloco "então"> será executado apenas quando <condição> resultar em verdadeiro (%t).

- O **else** não é obrigatório, mas quando for necessário, resulta em:

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

O <bloco "senão"> será executado apenas quando <condição> resultar em falso (%f).

- if, then, else e end**, são palavras reservadas do Scilab, não podem ser utilizadas para dar nome a variáveis.

19

BCC701
2012/01

Comando condicional if

- Exemplo:

```
:
:
if (delta >= 0) then
  printf("A raiz x1 e %g.\n", x1);
  printf("A raiz x2 e %g.", x2);
else
  printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
  printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
end
:
:
```

<condição> resultará em verdadeiro (%t) quando o valor de delta for maior ou igual a 0.

Estes dois comandos representam o <bloco "então">.

Estes dois comandos representam o <bloco "senão">.

20

BCC701
2012/01

Comando condicional if

- Observe que em nosso exemplo ainda existe uma situação que pode ser considerada:

- Quando **delta** é igual a **zero**, as duas raízes são iguais, ou seja, existe apenas uma raiz;
- Podemos melhorar a saída do programa, considerando esta situação.

21

BCC701
2012/01

Comando condicional if

- Mais um aprimoramento:

```
:
:
if (delta == 0) then
  printf("Existe apenas uma raiz: %g.", x1);
else
  if (delta > 0) then
    printf("A raiz x1 e %g.\n", x1);
    printf("A raiz x2 e %g.", x2);
  else
    printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
    printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
  end
end
:
:
```

22

BCC701
2012/01

if's aninhados

- Mais um aprimoramento:

```
:
:
if (delta == 0) then
  printf("Existe apenas uma raiz: %g.", x1);
else
  if (delta > 0) then
    printf("A raiz x1 e %g.\n", x1);
    printf("A raiz x2 e %g.", x2);
  else
    printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
    printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
  end
end
:
:
```

Quando existe este tipo de disposição do comando **if**, ocorre o que denominamos de **if's aninhados**.

23

BCC701
2012/01

if's aninhados

- Sintaxe para if's aninhados (1):

```
if <condição1> then
  <bloco "então1">
elseif <condição2>
  <bloco "então2">
elseif <condição3>
  <bloco "então3">
:
elseif <condiçãoN>
  <bloco "entãoN">
else
  <bloco "senão">
end
```

24

BCC701
2012/01

if's aninhados

• Exemplo:

```

:
:
:
  if (delta == 0) then
    printf("Existe apenas uma raiz: %g.", x1);
  elseif (delta > 0) then
    printf("A raiz x1 e %g.\n", x1);
    printf("A raiz x2 e %g.", x2);
  else
    printf("A raiz x1 e %g + %g*i.\n", real(x1), imag(x1));
    printf("A raiz x2 e %g + %g*i.", real(x2), imag(x2));
  end
:
:

```

- Melhora a formatação e a leitura do código, principalmente para um grande número de if's aninhados.

25

BCC701
2012/01

Introdução;
Comandos de desvio de fluxo;
Expressões lógicas.
Exercícios.

EXPRESSÕES LÓGICAS

26

BCC701
2012/01

Introdução

- Uma parte importante do comando **if** é a **<condição>**;
- Como vimos, a **<condição>** deve ser uma **expressão lógica**, ou seja, deve ser uma expressão que resulta em um valor verdadeiro ou falso;
 - **Verdadeiro** e **falso** são valores lógicos, representados no Scilab como %t (ou %T) e %f (ou %F), respectivamente;
- Tais expressões são criadas a partir do uso de **operadores relacionais** e **operadores lógicos**.

27

BCC701
2012/01

Operadores relacionais

- Já manipulamos operadores relacionais nos exemplos anteriores;
- Eles são usados para comparar valores de duas expressões:

<expressão 1> <operador relacional> <expressão 2>

28

BCC701
2012/01

Operadores relacionais

- Operadores relacionais definidos no Scilab:

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

29

BCC701
2012/01

Operadores relacionais

- Exemplos:
 - $5 < 3$, resulta %f;
 - $5 < 3 * 2$, resulta %t;
 - $5 + 1 > 3 * 2$, resulta %f;
 - $5 + 1 >= 3 * 2$, resulta %t;
 - $5 \sim= 3 * 2$, resulta %t;
 - $5 \sim= 3 * 2 < 10$, resulta em **ERRO**;
 - Não faz sentido dois operadores relacionais neste tipo de expressão;
 - Veremos adiante que expressões relacionais podem ser combinadas através de operadores lógicos;
- Observe que primeiro as expressões aritméticas são avaliadas, para que depois o operador relacional seja avaliado;
 - Ou seja, operadores relacionais têm precedência **menor** sobre os operadores aritméticos;

30

BCC701
2012/01

Operadores Lógicos

- As expressões lógicas podem ser combinadas através de **operadores lógicos**;
- Eles são usados para construir expressões mais complexas, que combinam condições múltiplas:
 - `<expressão lógica 1> <operador lógico> <expressão lógica 2>`
- Ou para negar uma condição:
 - `<operador de negação> <expressão lógica>`

31

BCC701
2012/01

Operadores Lógicos

- Operadores lógicos definidos no Scilab:

Operador	Descrição
&	E (conjunção).
	OU (disjunção não exclusiva).
~	Não (negação).

- E (Conjunção):
 - Resulta verdadeiro se as duas expressões forem verdadeiras;
- OU (disjunção não exclusiva):
 - Resulta verdadeiro se **pelo menos** uma das duas expressões forem verdadeiras;
- Não (negação):
 - Inverte o valor lógico da expressão, se ela for verdadeira o resultado é falso, se ela for falsa o resultado é verdadeiro.

32

BCC701
2012/01

Operadores Lógicos

- Os operadores lógicos podem ser definidos por **tabelas verdade**:

A	B	A & B	A B	~A
%t	%t	%t	%t	%f
%t	%f	%f	%t	%f
%f	%t	%f	%t	%t
%f	%f	%f	%f	%t

33

BCC701
2012/01

Operadores Lógicos

- Primeiro os operadores aritméticos e relacionais são avaliados, para depois serem avaliados os operadores lógicos;
- Operadores E (&) e OU (|) possuem a mesma precedência, que é inferior à precedência da negação (~), ou seja, $\sim X \& Y$, corresponde a $(\sim X) \& Y$;
- Exemplos (1):**
 - Sejam as seguintes atribuições: $A = 5$; $B = 1$; $C = 2$; $D = 8$; $E = 3$;
 - $A > B \mid D > E$, resulta em verdadeiro;
 - $\sim A > B$, resulta em falso;
 - $A + 3 == 8 \& A > B$, resulta em verdadeiro;
 - Atenção:** um erro comum de programação costuma ocorrer pela confusão entre o operador de atribuição (representado por =) e o operador de igualdade ("igual a", representado por ==);

34

BCC701
2012/01

Operadores Lógicos

- Exemplos (2):**
 - Em um parque de diversões apenas pessoas entre 18 e 50 anos podem andar na montanha russa, considerando a existência de uma variável chamada **idade**, a expressão lógica para permitir a entrada no brinquedo seria:
 - `(idade >= 18) & (idade <= 50)`;
 - Mais uma vez, o uso dos parênteses é bom para facilitar a leitura;**
 - Atenção:** a expressão: `idade >= 18 & <= 50`, está incorreta pela sintaxe do Scilab;
 - A expressão `18 <= idade <= 50` também está incorreta.

35

BCC701
2012/01

Introdução;
Comandos de desvio de fluxo;
Expressões lógicas;
Exercícios.

EXERCÍCIOS

36

BCC701
2012/01

Exercícios (lista 2 do prof. David)

- (4) Escreva um programa para determinar se um dado número N (recebido através do teclado) é POSITIVO, NEGATIVO ou NULO.
- (8) Escreva um programa que leia um número e informe se ele é ou não divisível por 5.
- (12) A prefeitura de Contagem abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um programa que permita entrar com o salário bruto e o valor da prestação, e informar se o empréstimo pode ou não ser concedido.

37

BCC701
2012/01

Exercícios (lista 2 do prof. David)

- (16) Construa um programa, que receba três valores, A, B e C, e armazene-os em três variáveis com os seguintes nomes: MAIOR, INTER e MENOR (os nomes correspondem aos valores ordenados).
- (20) Criar um programa que leia dois números e imprima o quadrado do menor número e raiz quadrada do maior número, se for possível.
- (24) Crie um programa que leia a idade de uma pessoa e informe a sua classe eleitoral:
 - Não eleitor (abaixo de 16 anos);
 - Eleitor obrigatório (entre a faixa de 18 e menor de 65 anos);
 - Eleitor facultativo (de 16 até 18 anos e maior de 65 anos, inclusive).

38

BCC701
2012/01

Exercícios (lista 2 do prof. David)

- (28) Um comerciante calcula o valor da venda, tendo em vista a tabela a seguir:

Valor da Compra	Valor da Venda
Valor < R\$ 10,00	Lucro de 70%
R\$ 10,00 <= Valor < R\$ 30,00	Lucro de 50%
R\$ 30,00 <= Valor < R\$ 50,00	Lucro de 40%
Valor >= R\$ 50,00	Lucro de 30%

Criar um programa que leia o valor da compra e imprima o valor da venda.

39

BCC701
2012/01

Exercícios (lista 2 do prof. David)

- (32) Dados três valores A, B e C, construa um programa para verificar se estes valores podem ser valores dos lados de um triângulo, e se for um triângulo retângulo, determinar (imprimir) os seus ângulos internos.
- (36) Construir um programa para calcular as raízes de uma equação do 2º grau, sendo que os valores dos coeficientes A, B, e C devem ser fornecidos pelo usuário através do teclado.

40

BCC701
2012/01

Exercícios (lista 2 do prof. David)

- (40) Criar um programa que leia o destino do passageiro, se a viagem inclui retorno (ida e volta) e informar o preço da passagem conforme a tabela a seguir:

Condição	Ida	Ida e Volta
Região Norte	R\$ 500,00	R\$ 900,00
Região Nordeste	R\$ 350,00	R\$ 650,00
Região Centro-Oeste	R\$ 350,00	R\$ 600,00
Região Sul	R\$ 300,00	R\$ 550,00

- (44) Criar um programa que leia um número inteiro entre 1 e 12 e escrever o mês correspondente. Caso o usuário digite um número fora desse intervalo, deverá aparecer uma mensagem informando que não existe mês com este número.

41

BCC701
2012/01

Próxima aula prática: resolução de exercícios com o Scilab.
Próxima aula teórica: mais exercícios para fixação.

FIM!
DÚVIDAS?

42

BCC701
2012/01