Introdução à Ciência da Computação

Unidade III

Programação de Computadores com FreeMat

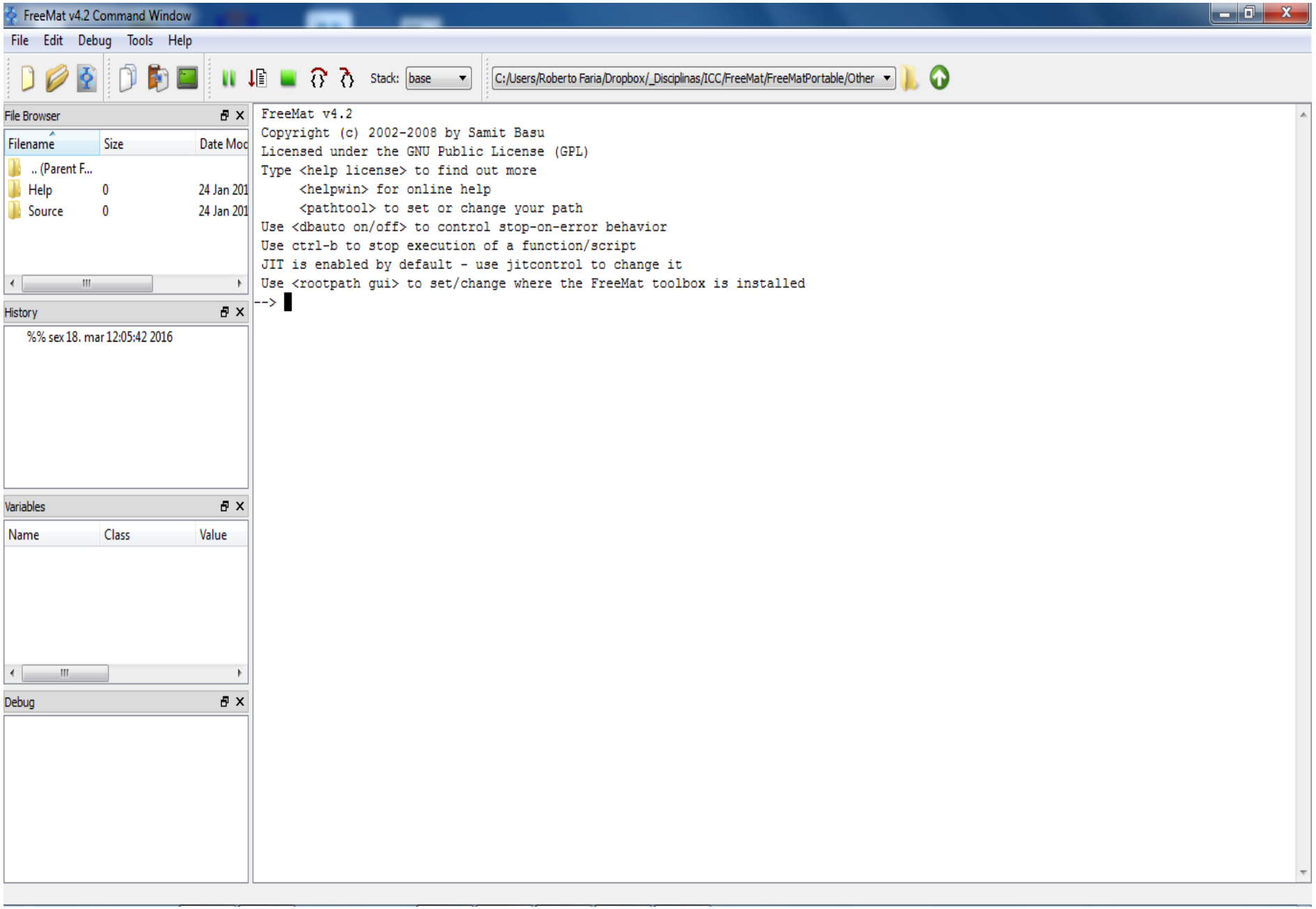
Prof. Roberto M. de Faria/DSC/UFCG

FreeMat

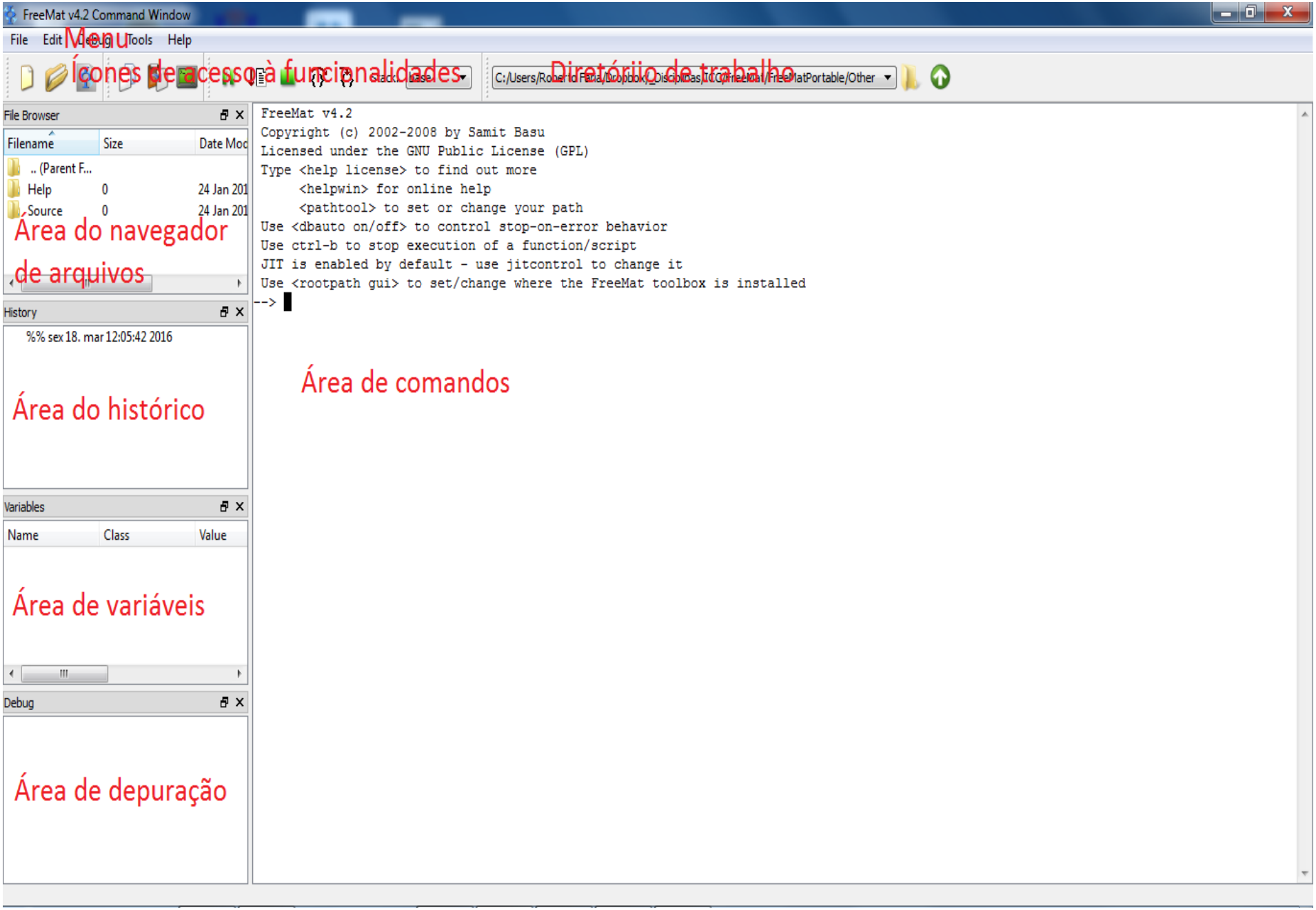
- FreeMat é um ambiente livre para engenharia rápida, prototipagem científica e processamento de dados. Ele é semelhante à sistemas comerciais tais como MATLAB da Mathworks e IDL do Research Systems, mas é um sistema de código aberto (Open Source). FreeMat é disponível sob a licença GPL (GNU Public License).

(http://freemat.sourceforge.net)

Tela Principal



Componentes da Tela Principal



Componentes da Tela Principal

- O Menu dá acesso às funcionalidades do software.
- Os Ícones dão acesso a algumas funcionalidades do software utilizadas com mais frequência
- O Diretório de Trabalho mostra o caminho do diretório atual
- A Área do Navegador de Arquivos permite a mudança do diretório atual

Componentes da Tela Principal

- A Área de Histórico é uma janela que mostra, em ordem cronológica, todos os comandos que foram inseridos e os resultados. Os comandos mais novos estão listados na parte inferior e os mais velhos na superior.
- A Área de Variáveis contém informações sobre a variáveis utilizadas
- A Área de Depuração contém informações para eliminação de erros

Componentes da Tela Principal

- A Área de Comandos é uma janela onde é possível inserir variáveis, funções, programas e comandos. Quando você inicia o programa Freemat, as primeiras linhas da janela de comando irão fornecer algumas informações básicas sobre o programa e sobre alguns comandos básicos.

Conceito Fundamental - Variáveis

- Uma variável é simplesmente uma porção (parte) da memória do computador usada para armazenar algum dado.
- Uma variável possui um nome (um único caractere ou um conjunto de caracteres) que é por meio dele que se referencia aquela parte da memória. Um nome de variável pode conter letras, números e/ou um sublinhado, mas deve começar com uma letra. Além disso, os nomes de variáveis diferenciam maiúsculas de minúsculas. Ex.: a variável "`X`" é diferente da variável "`x`".

Variáveis

- Em geral, uma variável pode armazenar:
  - um número
  - uma cadeia de caracteres (texto)
  - uma matriz de números, sequências de caracteres e/ou outras matrizes
  - um apontador para uma função anônima.

Modos de Operação do FreeMat

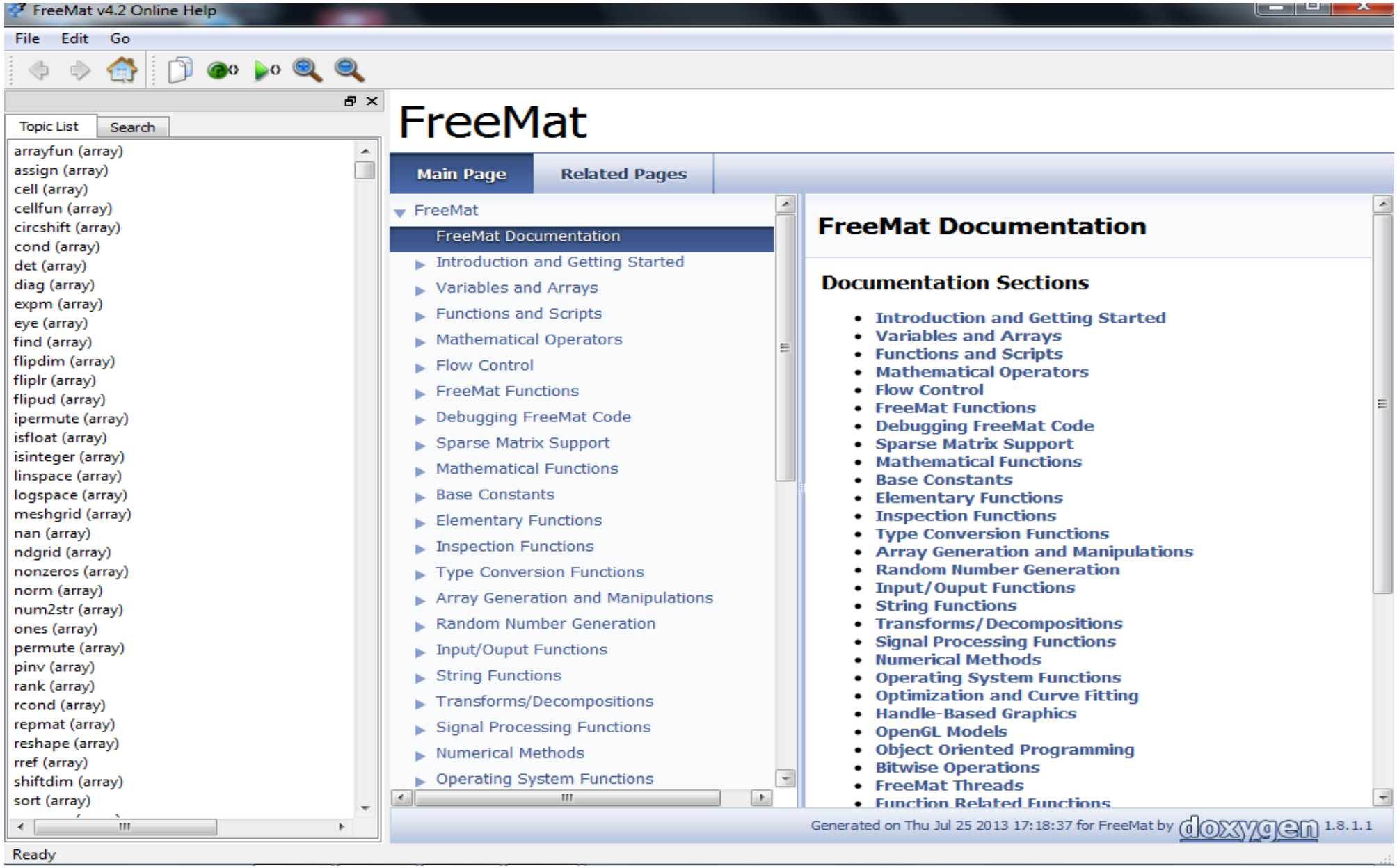
- O FreMat pode ser operado de dois modos: ***interativo*** e ***de programação***
- No ***modo interativo***, o usuário digita comandos diretamente na Área de Comandos e, na medida que cada comando é digitado, vai sendo executado
- No ***modo de programação***, o usuário armazena comandos do FeeMat num arquivo de extensão "**.**m" (*arquivo-m*) e posteriormente o executa.
- A sequência de execução dos comandos contidos no *arquivo-m* pode ser modificada, usando comandos de controle de fluxo

Parte I
Operação do FreeMat no Modo Interativo

Ajuda para o FreeMat

- Para ter acesso à documentação *on line* do FreeMat, digite na Área de Comandos o comando "`helpwin`" seguido de um `<Enter>`

Tela de Ajuda



Comandos Básicos do FreeMat

- Operadores Aritméticos (por precedência)
  - “^” potenciação ou exponenciação
  - “*” e “/” multiplicação e divisão
  - “+” e “–” soma e subtração
- Numa expressão aritmética, usa-se parênteses para alterar a precedência dos operadores
- No FreeMat, usa-se ponto decimal “ .” para separar as casas decimais de números reais

Comandos Básicos do FreeMat

- Comandos para cálculos básicos
  - Encerra-se um comando com a tecla `<Enter>`
  - Ex:
    - `--> 2 + 5`
    - `--> 7.8 + 3.5`
    - `--> 6 * 5`
    - `--> 2.5 / 5`
    - `--> 2 ^ 7`
    - `--> (2 + 5) * 7`
  - O resultado da execução de um comando, não atribuído a uma variável, fica armazenado na variável "`ans`" (answer – resposta) do FreeMat

Reedição dos Comandos já Submetidos

- Utilizando-se as teclas "`seta-para-cima`" e "`seta-para-baixo`", pode-se acessar comandos já submetidos à execução anterior
- Um comando já executado pode ser reeditado com modificações e re-submetido à execução ou simplesmente ser reexecutado sem modificações

Exercícios

- Calcule a energia cinética de uma partícula que tem a massa de 0,0023 kg e a velocidade de 3235 m/s
- Calcule o IMC de uma pessoa de peso 84,5 kg e altura de 1,72 m
- Calcule a área de um círculo e sua circunferência, sabendo que seu raio é 60 cm
- Calcule o valor final, após um mês, de um investimento de R$ 1.000,00 numa caderneta de poupança, sabendo que a inflação do período foi de 2,37% e o juro mensal é de 0,5%

Relembrando Variáveis

- Uma variável é simplesmente uma porção (parte) da memória do computador usada para armazenar algum dado.
- Uma variável possui um nome (um único caractere ou um conjunto de caracteres) que é por meio dele que se referencia aquela parte da memória. Um nome de variável pode conter letras, números e/ou um sublinhado, mas deve começar com uma letra. Além disso, os nomes de variáveis diferenciam maiúsculas de minúsculas. Ex.: a variável "`X`" é diferente da variável "`x`".

Relembrando Variáveis

- Em geral, uma variável pode armazenar:
  - um número
  - uma cadeia de caracteres (texto)
  - uma matriz de números, sequências de caracteres e/ou outras matrizes
  - um apontador para uma função anônima.

Ainda sobre Variáveis

- Armazenamento numa Variável
  - Quando atribui-se um valor a uma variável, ela passa a existir e é mostrada na Área de Variáveis
    - Ex:

```
- --> a = 2
- --> b = 9
- --> c = 7.5
- --> delta = b ^ 2 - 4 * a * c
- --> X1 = (-b + sqrt(delta)) / (2 * a)
- --> X2 = (-b - sqrt(delta)) / (2 * a)
```

  - Obs: "`sqrt()`" é uma função pré-definida do FreeMat

Ainda sobre Variáveis

- Uma variável pode armazenar também um texto
  - Ex: `--> titulo = 'Raízes de uma Equação do Segundo Grau'`
- O nome de uma variável, como comando, faz com que seu valor seja apresentado
  - Ex: `--> delta`
- O comando "`who`" mostra as variáveis já definidas
  - Ex: `--> who`
- O comando "`clear`" seguido do nome da variável remove a variável da memória
  - Ex: `--> clear delta`
- O comando "`clear all`" remove todas as variáveis definidas da memória
  - Ex: `--> clear all`

Variáveis do FreeMat

- O FreeMat possui algumas variáveis com valores pré-definidos:
  - `nan` → not-a-number
  - `inf` → infinito
  - `pi` → 3.1416
  - `e` → 2.7183 – base do logaritmo natural (constante de Euler)
  - `j` → 0.0000 + 1.0000i – raiz quadrada de -1
  - `eps` → 2.2204e-016 – epsilon – constante da máquina de dupla precisão em ponto flutuante
  - `true` → valor lógico verdadeiro
  - `false` → valor lógico falso

Tipos de Variáveis

- Toda a variável tem um tipo a ela associado
- Os tipos para as variáveis do FreeMat são:
  - `'cell'` → arrays de células
  - `'struct'` → arrays de estruturas
  - `'logical'` → arrays lógicos
  - `'int8'` → inteiros com 1 byte
  - `'int16'` → inteiros com 2 bytes
  - `'int32'` → inteiros com 4 bytes
  - `'int64'` → inteiros com 8 bytes
  - `'float' ou 'single'` → real de ponto flutuante com 4 bytes (4 decimais)
  - `'double'` → real de ponto flutuante com 8 bytes (14 decimais)

Mais sobre Tipos de Variáveis

- Outros tipos para as variáveis do FreeMat são:
  - `'uint8'` → inteiros sem sinal com 1 byte
  - `'uint16'` → inteiros sem sinal com 2 bytes
  - `'uint32'` → inteiros sem sinal com 4 bytes
  - `'uint64'` → inteiros sem sinal com 8 bytes
  - `'char'` → para arrays de string
- Criação de variável usando tipo:
  - `--> x = int32(123)`
- Mudança de tipo com a função "`cast`"
  - `--> y = cast(x, 'int64')`

Múltiplos Comandos

- FreeMat permite que se coloque múltiplos comandos numa mesma linha de comando separados por “;”
  - Ex: `--> x = 7; y = x * 5; z = x + y`
- O “**;**” no final de um comando, faz com que o FreeMat não mostre a resposta do cálculo
  - Ex:
    - `--> raio = 30;`
    - `--> pi * raio ^ 2`

Comentários e Limpeza da Área de Comandos

- Uma linha de comando que começa com “%” é um comentário
- Um comentário é colocado para acrescentar alguma informação sobre os comandos que estão sendo executados
- Um comentário não é executado, ou seja não interfere na execução de outros comandos nem no armazenamento de dados
- O comando “`clc`” executa a limpeza da Área de Comandos

Funções Pré-definidas

- Uma função é um conjunto de comandos que possui um nome e que, quando executados, produzem um resultado
- O FreeMat possui muitas funções pré-definidas
  - Exemplos de algumas funções:
    - `abs()` → valor absoluto
    - `max()` → máximo
    - `min()` → mínimo
    - `sum()` → soma
    - `imag()` → parte imaginária de um número complexo
    - `round()` → arredonda para o inteiro mais próximo

Mais Funções Pré-definidas

- Exemplos de algumas funções trigonométricas (usa radianos):
  - `sin()` → seno
  - `cos()` → cosseno
  - `tan()` → tangente
  - `sec()` → secante
  - `csc()` → cossecante
  - `cot()` → cossecante

Mais Funções Pré-definidas

- Exemplos de mais algumas funções trigonométricas (retorna radianos):
  - `asin()` → seno
  - `acos()` → cosseno
  - `atan()` → tangente
  - `asec()` → secante
  - `acsc()` → cossecante
  - `acot()` → cossecante

Mais Funções Pré-definidas

- Exemplos de mais algumas funções trigonométricas (usa graus):
  - `sind()` → seno
  - `cosd()` → cosseno
  - `tand()` → tangente
  - `secd()` → secante
  - `cscd()` → cossecante
  - `cotd()` → cossecante
- Existem outras funções tais como: logaritmo, exponencial, raiz quadrada, etc., que podem ser pesquisadas na documentação do FreeMat

Formatos

- O FreeMat pode trabalhar com formatos de precisão (número de dígitos após o ponto decimal) diferentes
- A precisão pode ser alterada com os comandos "`format long`" e "`format short`"
  - Ex:

```
--> format short
--> pi

ans =
    3.1416

--> format long
--> pi

ans =
   3.141592653589793
```

Formato de Notação Científica

- O FreeMat pode utilizar o formato de notação científica com diferentes precisões
  - Ex:

```
--> format short e
--> pi
ans =
 3.1416e+000

--> format long e
--> pi
ans =
 3.14159265358979e+000
```

- O FreeMat pode retornar à formatação padrão digitando-se:

```
--> format short
```

Vetores e Matrizes

- Um vetor ou uma matriz é uma coleção de valores (elementos) de mesmo tipo que possui um nome e um conjunto de índices associados
- Os elementos de um vetor ou matriz são acessíveis por meio de um conjunto de índices
- Criação de um vetor:

```
--> vet = [-5, 20, -33, 15, 0]
```

- Criação de uma matriz:

```
--> mat = [-5, 7, -12, 0; 20, 100,
-3, 0; 88, -33, 15, 0]
```

Mais sobre Vetores e Matrizes

- O acesso a um elemento de um vetor é realizado por meio do nome do vetor ou matriz seguido do conjunto de índices do elemento entre parênteses
- Acesso a um elemento de um vetor:

```
--> vet(2) * 10
```

- Acesso a um elemento de uma matriz:

```
--> mat(2, 3) = 10
```

Mais sobre Vetores e Matrizes

- Criando vetores sequenciais:
  - Para criar vetores sequenciais, utiliza-se o operador “**:**”
    - Informando o valor inicial e o final:

```
--> vetor = 1 : 10
```

    - Informando o valor inicial, o incremento e o final:

```
--> sequencia = 1 : 3 : 10
```

    - Informando o valor inicial, o incremento e o final:

```
--> radianos = 0 : pi/4 : 2 * pi
```

Mais sobre Vetores e Matrizes

- Criando vetores randômicos:
  - Para criar vetores e matrizes randômicas, utiliza-se a função "`rand()`"
    - Matriz com uma linha e dez colunas (vetor):

      ```
      --> vetor = rand(1, 10)
      ```

    - Matriz 10x10:

      ```
      --> matriz_1 = rand(10, 10)
      ```

    - Matriz 10x10:

      ```
      --> matriz_2 = rand(10)
      ```

    - Matriz de 3 dimensões (2x3x2):

      ```
      --> matriz_3 = rand(2, 3, 2)
      ```

Algumas Operações com Matrizes

- Criação de matrizes:

```
--> A = [10, 22, -3, 14; -1, 12, 37, 4;
9, -8, 61, 4]
```

```
--> B = rand(3, 4)
```

```
--> C = [12, 27; -22, 14; -1, 0; 7, 4]
```

- Soma de matrizes:

```
--> D = A + B
```

- Subtração de matrizes:

```
--> E = A - B
```

Algumas Operações com Matrizes

- Multiplicação de matrizes:

```
--> F = A * C
```

- Multiplicação de matriz por escalar:

```
--> G = A * 2
```

- Cálculo do determinante de uma matriz:

```
--> D = [22, -3, 14; -1, 37, 4; 9,
-8, 61]

--> determinante = det(D)
```

Algumas Operações com Matrizes

- Maior elemento de uma matriz:

```
--> maior = max(A)
```

- Maior elemento de uma matriz:

```
--> menor = min(A)
```

- Soma dos elementos de uma matriz:

```
--> soma = sum(A)
```

- Soma acumulativa elementos de uma matriz:

```
--> somaAc = cumsum(D)
```

- Diferença entre elementos de uma matriz:

```
--> diff([2, 4, 8; 15, 31, 22])
```

Mais Algumas Operações com Vetores

- Remover um elementos de um vetor:

```
--> vetor = [6, 9, 0, 7, -4, 10, 22, -3]
--> vetor(4) = []
```

- Remover vários elementos de um vetor:

```
--> vetor(2:4) = []
```

- Acrescentar um elemento a um vetor:

```
--> vetor(5) = 10
```

- Acrescentar vários elementos a um vetor:

```
--> vetor(6:10) = 1:5
```

Mais Algumas Operações com Matrizes

- Remover uma linha de uma matriz:

```
--> matriz = [6, 9, 0; 7, -4, 10; 22,
-3, 12]
```

```
--> matriz(1, 1:3) = []
```

- Acrescentar varias colunas a uma matriz:

```
--> matriz(3:5, 1:3) = 0
```

- Remover varia colunas de uma matriz:

```
--> matriz(1:5, 2:3) = []
```

- Acrescentar várias linhas a uma matriz:

```
--> matriz(6:10, 1) = -1
```

Parte II
Operação do FreeMat no Modo de Programação

Criando Programas no FreeMat

- Arquivos M
- Saída de Dados
- Entrada de Dados
- Comandos Condicionais
- Comandos de Repetição

Arquivos-M

- Para criar um *arquivo-m* no FreeMat, utiliza-se um editor de linhas muito semelhante ao "*Bloco de Notas*" do *Windows*, que pode ser ativado de duas maneiras: com o comando "`edit`" na Área de Comandos, clicando no ícone  do FreeMat ou pressionando-se "`Ctrl-E`"
- Um *arquivo-m*, quando criado, é alocado no "Diretório de Trabalho" mostrado na parte de cima da tela do FreeMat
- Para se realizar a mudança do Diretório de Trabalho, utiliza-se o Navegador de Arquivos ou o comando "`cd`"

Saída de Dados

- FreeMat possui várias funções para a apresentação de valores
  - A função `disp()` apresenta o resultado de um conjunto variável de expressões

    ```
    --> disp('Os valores das raízes da equação do segundo grau é ', x1, ' e ', x2)
    ```

  - A função `printf()` imprime valores na saída

    ```
    --> printf('O valor da Variável é %d\n', var)
    ```

Saída de Dados

- Especificadores de conversão mais comuns
- `%d` ou `%i` → inteiro
- `%f` → real de ponto flutuante
- `%c` → caractere
- `%s` → string
- `%e` → notação científica
- O caractere "`\n`" provoca o cursor ir para a linha seguinte

Entrada de Dados

- A função "`input()`" recebe um valor digitado do teclado pelo usuário e possui dois formatos:
  - Neste formato o usuário digita uma expressão que avaliada e o resultado enviado para a variável

    ```
    --> variavel = input('Digite o valor para a variável:')
    ```

  - Neste outro formato o usuário digita um valor e este é enviado para a variável

    ```
    --> variavel = input('Digite o valor para a variável:', 's')
    ```

Exemplo de Programa

- Programa para calcular a área de um círculo:

```
% Este programa calcula a área de um
% circulo a partir de seu raio em
% centímetros
disp(' ');
disp('Cálculo da área de um círculo:');
disp(' ');
raio = input('Informe o comprimento do raio em cm: ');
area = pi * raio ^ 2;
printf('\nA área do círculo de raio %f cm é %f
centímetros quadrados!\n\n', raio, area);
```

Exercícios

Escreva vários programas que:

1) Calcule a área de um triângulo a partir de suas base e altura
2) Calcule a distância entre dois pontos no plano a partir das coordenadas de cada ponto
3) Calcule o volume de um cone a partir de seus raio da base e altura
4) Calcule a superfície de um cilindro a partir de seus raio da base e altura
5) Calcule o volume de uma esfera oca a partir dos raios das paredes interna e externa

Expressões Relacionais

- ***Expressões relacionais*** são aquelas cujas avaliações recebem o valor ***verdadeiro*** (*diferente de 0*) ou ***falso*** (*igual a 0*)
- São chamadas também de ***expressões Booleanas*** ou ***expressões Lógicas***
- Essas expressões podem usar ***operadores relacionais*** e/ou ***operadores lógicos***, que operam sobre ***operandos lógicos***
- Um ***operando lógico*** possui valor ***verdadeiro*** (*diferente de 0*) ou ***falso*** (*igual a 0*)
- As ***expressões relacionais*** definem ***condições*** para comandos do FreeMat

Operadores Relacionais

- Operadores relacionais
  - **==** → igual a
  - **~=** → diferente de
  - **<** → menor que
  - **>** → maior que
  - **<=** → menor ou igual a
  - **>=** → maior ou igual a
- Exemplos

```
--> 3 < 5
```

Operadores Relacionais

- Mais exemplos

```
--> 2 > 9
--> class(ans)
--> result = 5 < 7
--> result + 1
--> 5 + 3 > 7 - 2
--> a = 10
- --> b = 20
- --> a + 10 ~= b
--> b - 10 <= 10 + a
```

Operadores Lógicos

- Operadores lógicos
  - **&&** → e
  - **||** → ou
  - **~** → não
- Exemplos

```
--> 3 > 5 && 9 >= 3
--> 33 > 5 || 99 <= 23
--> 9 || 7 - (5 + 2)
--> 0 && 10 - (5 + 2)
```

Operadores Lógicos

- Tabelas verdade

E (&&)	V	F
V	V	F
F	F	F

Ou (||)	V	F
V	V	V
F	V	F

Não (~)	
V	F
F	V

Regras de Precedência de Operadores

Operadores	**Precedência**
parênteses `()`	***maior***
potência `^`	
negação `-`, não `~` (*unários*)	
multiplicação `*`, divisão `/`, `\`	
adição `+`, subtração `-`	
relacionais `<`, `<=`, `>`, `>=`, `==`, `~=`	
and `&&`	
ou `||`	
atribuição `=`	***menor***

Comandos Condicionais

- Comando **`if`**

```
if condição
   ações
end
```

  - As `ações` serão executadas se a `condição` for ***verdadeira*** (*diferente de 0*)

- Exemplo

```
a = input('Digite o valor de a: ');
b = input('Digite o valor de b: ');
if a ~= 0
   c = b / a;
end
```

Comandos Condicionais

- Comando **`if`** com a cláusula **`else`**

```
if condição
   ações_1
else
   ações_2
end
```

  - Serão executadas as `ações_1` após a `condição`, se a `condição` for ***verdadeira*** (*diferente de 0*), caso contrário, se for ***falsa*** (*igual a 0*), serão executadas as `ações_2` após o `else`

Comandos Condicionais

- Exemplo de **`if`** com a cláusula **`else`**

```
a = input('Digite o coeficiente a: ');
b = input('Digite o coeficiente b: ');
c = input('Digite o coeficiente c: ');
delta = b ^ 2 - 4 * a * c;
if delta < 0
   printf('Não há raízes reais!\n');
else
   printf('As raízes são reais!\n');
end
```

Comandos Condicionais

- Comandos **`if`**'s aninhados

```
sexo = input('Informe o sexo da pessoa(M ou F): ');
idade = input('Informe a idade da pessoa: ');
if sexo == 'f' || sexo == 'F'
   printf('Dispensada de alistamento militar!\n');
else
   if idade == 18
      printf('Precisa alistar-se!\n');
   else
      printf('Não precisa alistar-se!\n');
   end
end
```

Comandos Condicionais

- A cláusula **`elseif`**

```
if condição_1
   ações_1
elseif condição_2
   ações_2
elseif condição_3
   ações_3
% etc. ... qualquer quantidade de elseif's
else
   ações_n
end
```

Comandos Condicionais

- Exemplo

```
conta_maiusc = 0;
conta_minusc = 0;
conta_outro = 0;
carac = input('Digite um caractere: ');
if carac >= 'A' && carac <= 'Z'
   conta_maiusc = conta_maiusc + 1
elseif carac >= 'a' && carac <= 'z'
   conta_minusc = conta_minusc + 1
else
   conta_outro = conta_outro + 1
end
printf('Número de maiúsculas = %d\n', conta_maiusc);
printf('Número de maiúsculas = %d\n', conta_minusc);
printf('Número de outros = %d\n', conta_outros);
```

Comandos Condicionais

- Comando **`switch`**

```
switch expressão
   case valor_1
      ações_1
   case valor_2
      ações_2
% etc. ... qualquer quantidade de case's
   otherwise
      ações_n
end
```

Comandos Condicionais

- Funcionamento do comando **`switch`**
  - No início do `switch` a `expressão` é comparada com cada `valor_i` de `case`
  - Se houver coincidência do valor da `expressão` com algum `valor_i` de case, a execução inicia na `ação_i` e encerra antes do próximo `case` ou do `otherwise`
  - Se não houver coincidência com nenhum `valor_i` de case, as `ações_n` após o `otherwise` são executadas

Comandos Condicionais

- Exemplo

```
ordem = input('Digite a ordem(1, 2 ou 3): ');
switch ordem
   case 1
      printf('Primeiro!\n');
   case 2
      printf('Segundo!\n');
   case 3
      printf('Terceiro!\n');
   otherwise
      printf('Ordem inválida!\n');
end
```

Exercícios

1) Faça um programa que receba valores para as variáveis ***a*** e ***b***. Se o valor de ***a*** for maior que o valor de ***b***, troque os valores dessas variáveis, entre si. Use um comando `if` na solução.

Exercícios

2) As leituras de pressão arterial sistólica e diastólica são encontradas quando o coração está bombeando e o coração está em repouso, respectivamente. Um experimento biomédico está sendo realizado apenas para os participantes cuja pressão arterial é ideal. Esta é definida como uma pressão arterial sistólica menor ou igual a 120 e uma pressão arterial diastólica menor ou igual a 80. Escreva um programa que irá pedir as pressões sistólica e diastólica de uma pessoa e, em seguida, imprima uma mensagem dizendo se essa pessoa é, ou não, um candidato para este experimento. Use um comando `if` com a cláusula `else` na solução.

Exercícios

3) Faça um programa que receba os coeficientes (***a***, ***b*** e ***c***) de uma equação do segundo grau e mostre, dependendo do seu ***delta***: uma raiz real, duas raízes reais ou uma mensagem informando que não existe raízes reais para esta equação. Use comandos `if`'s aninhados na solução.

Exercícios

4)Faça um programa que calcule a área de uma figura geométrica plana (círculo, triângulo, quadrado ou retângulo). O usuário escolherá a figura por meio de um menu, com opções numéricas, e, em seguida, o programa solicitará os dados necessários, para então, mostrar o valor da área da figura. Use cláusulas `elseif`'s na solução.

Exercícios

5) Faça um programa que receba um número entre 1 e 99, inclusive, e mostre seu numeral ordinal correspondente. Use comandos `switch`'s na solução.

Comandos de Repetição

- Existem dois comandos de repetição (laços) diferentes no FreeMat: o comando `for` e o comando `while`
- O `for` é usado como um laço contado, e o `while` é usado como um laço condicional
- Em muitas linguagens de programação, laços para manipulação de elementos em um vetor ou matriz é um conceito fundamental
- No FreeMat, laços para processar elementos de vetores e matrizes, geralmente não é necessário, em vez disso, "código vetorizado" é usado, o que significa substituir os laços de manipulação de vetores e matrizes pelo uso de operadores e funções internas

Comandos de Repetição

- O comando `for` ,ou o laço `for`, é usado quando é necessário repetir comandos em um programa ou função, quando é conhecido previamente quantas vezes os comandos deverão ser repetidos
- As instruções que são repetidas são chamadas de ***ações do laço***.
- Chama-se a variável que é usada para a contagem das iterações do laço, de ***variável do laço*** ou ***variável do iterador***.
- Por exemplo, a variável pode iterar os inteiros de 1 a 5 (por exemplo, 1, 2, 3, 4 e em seguida 5)

Comandos de Repetição

- A forma geral do laço `for` é:

```
for variável_do_laço = intervalo
    ações_do_laço
end
```

- `intervalo` é o intervalo de valores através do qual a `variável_do_laço` itera as `ações_do_laço`, que consiste de todas as instruções até o `end`
- As `ações_do_laço` são recuadas para torná-las mais fácil de ver
- O `intervalo` pode ser especificado mais facilmente, utilizando o ***operador de dois pontos***.

Comandos de Repetição

- Por exemplo, será impressa uma coluna de números de 1 a 10:

```
for contador = 1:10
    printf('%d\n', contador);
end
```

- O que imprimirá este laço?

```
for controle = 20:-3:1
    printf('Eu não devo desobedecer ');
    printf('meus pais!\n');
end
```

Exercícios

1) Faça um programa que calcule e imprima o IMC de 5 pessoas
2) Modifique o programa anterior para que ele execute para um número determinado de pessoas
3) Faça um programa que calcule e mostre o fatorial de um numero natural
4) Faça um programa que some os números naturais até 20

Exercícios

5) Modifique o programa anterior para que some os números naturais até N

6) Faça um programa que some os números naturais no intervalo de M a N, inclusive

7) Faça um programa que calcule e mostre ao soma de 10 números inteiros

8) Modifique o programa anterior para que calcule e mostre a soma de N números inteiros

Exercícios

9) Faça um programa que crie um vetor com 10 valores reais aleatórios, mostre os dados desse vetor com um valor em cada linha

10) Estenda o programa anterior para ordenar os dados do vetor em ordem crescente e mostrá-los ordenados

11) Faça um programa para desenhar as seguintes figuras, dado o número de linhas:

```
a)*        b)    *    c)****    d)****
  **             **       ***          ***
  ***           ***       **            **
  ****         ****       *              *
```

Comandos de Repetição

- O comando `while` é usada como o ***laço condicional*** em FreeMat
- Ele é usado para repetir uma ***ação*** quando, previamente, não se sabe quantas vezes a ***ação*** será repetida
- A forma geral do comando `while` é:

```
while condição
   ação
end
```

- A `ação`, que consiste de qualquer número de comandos, é executada enquanto a `condição` for ***verdadeira***

Comandos de Repetição

- O modo como funciona, é que primeiro a `condição` é avaliada
- Se a `condição` é logicamente ***verdadeira***, a ação é executada
- Até aí, o comando `while` é igual a um comando `if`.
- No entanto, nesse ponto a condição é avaliada novamente
- Se é ***verdadeira***, a `ação` é executada novamente e assim por diante
- Então, eventualmente, algo na `ação` tem que mudar alguma coisa na `condição` para assim, tornar-se ***falsa***
- A `condição` deve, eventualmente, tornar-se ***falsa*** para evitar um ***laço infinito***
- Neste caso, ***Ctrl-b*** pode forçar a interrupção do laço e, em seguida, pressione (Stop), para parar o programa

Comandos de Repetição

- Outro exemplo com `while`:

```
clc
clear all
printf('\nPrograma que recebe um inteiro positivo, ');
printf('mostra-o e,em seguida,\nmostra os seus ');
printf('dígitos separados por espaços em branco.\n\n');
inteiro = input('Digite um número positivo: ');
printf('\nNúmero lido = %d\n\n', inteiro);
printf('Os dígitos são:\n\n');
while inteiro > 0
    printf('%d  ', mod(inteiro, 10));
    inteiro = fix(inteiro / 10);
end
printf('\n\n');
```

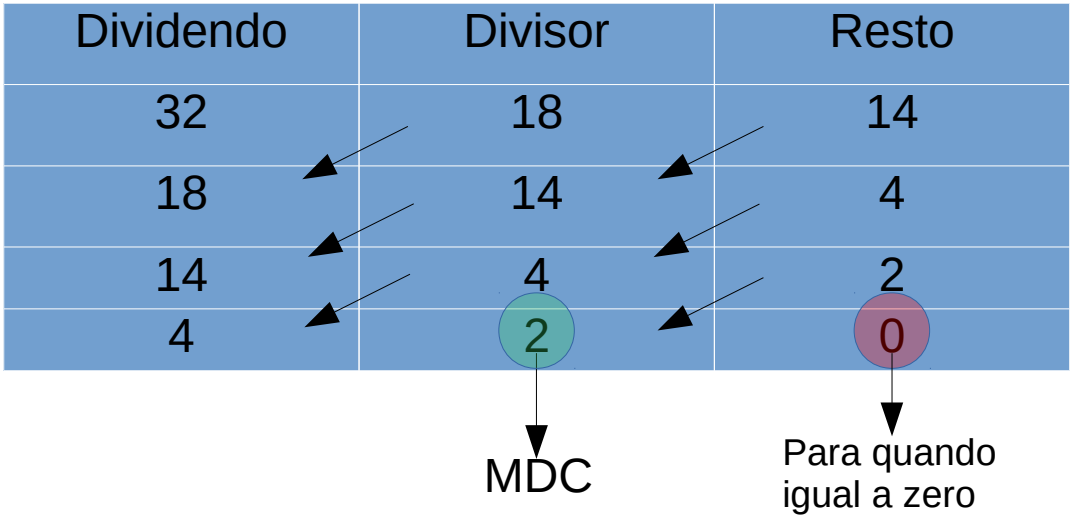
Comandos de Repetição

- Exemplo com `while`:

```
clc
clear all
printf('Programa que soma inteiros positivos e ');
printf('para quando encontra um negativo ou 0:\n');
soma = 0;
inteiro = input('Informe um inteiro a somar:');
while inteiro > 0
    soma = soma + inteiro
    inteiro = input('Informe um inteiro a somar:');
end
printf('\nA soma dos inteiros positivos é %d\n\n');
```

Exercícios

1) Modifique o programa exemplo com `while` para somar valores e parar quando aparecer um múltiplo de 5.
2) Modifique o programa anterior para parar quando aparecer um múltiplo de N
3) Faça um programa que calcule o MDC de dois inteiros positivos (Exemplo do método: MDC de 32 e 18)



Exercícios

4) Modifique o programa anterior para que no mesmo `printf` que mostra o MDC, mostrar também os valores sobre os quais foi calculado o MDC

5) Faça um programa para calcular as raízes de várias equações do 2º. grau e parar quando o coeficiente ***a*** for igual a zero

6) Faça um programa que mostre que receba um inteiro positivo e mostre os seus dígitos separados por espaços em branco

Exercícios

7) Modifique o programa anterior para que mostre a soma de dígitos de vários números recebidos e pare quando encontrar um número par

8) Um “Número de Armstrong” de ***n*** dígitos é aquele que é igual a soma das potências de seus dígitos com expoente ***n***

   Exemplo: $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27$

   Mostre os números de Armstrong menores que M

Exercícios

9) Faça um programa para mostrar se um número positivo é ou não primo, lembrando que 1 é divisor de qualquer número e que os divisores de um número encontra-se entre 1 e a metade desse número

10) Faça um programa que gere um número aleatório entre 1 e 100 e aceite palpites do usuário até que o mesmo acerte esse número

    A cada palpite, o programa informa se o palpite foi maior, menor ou se o usuário acertou o número

Criando Funções

Gerando Gráficos