Loops, strings e números complexos

Manipulação de loops, texto e fórmula de báskara

Matheus Roos

Universidade Federal de Santa Maria

9 de agosto de 2022

1. Loops
   - Tipos de Loops
   - Alguns detalhes
2. Atributos do Fortran redundantes, obsoletos e deletados
3. Manipulando palavras
   - Funções intrísecas para caracteres
4. Aplicação - Fórmula de Baskara
   - Panorama geral
   - Construindo a necessidade de incluir os números complexos
   - Definindo um número complexo (rapidamente)
   - Variável do tipo COMPLEX
5. Bibliografia

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.
  - A iteração é automática, não precisamos fazê-la explícitamente.

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.
  - A iteração é automática, não precisamos fazê-la explícitamente.
- Existem duas formas de fazer o loop WHILE:

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.
  - A iteração é automática, não precisamos fazê-la explícitamente.
- Existem duas formas de fazer o loop WHILE:
  - "Loop DO vazio", instrução análoga ao GO TO do Fortran 77;

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.
  - A iteração é automática, não precisamos fazê-la explícitamente.
- Existem duas formas de fazer o loop WHILE:
  - "Loop DO vazio", instrução análoga ao GO TO do Fortran 77;
  - Loop DO WHILE, que é uma forma análoga ao loop WHILE padrão do Fortran 90/95;

Loops e lógica

- Em Fortran temos dois tipos de loops, DO (faça) e WHILE (enquanto).
- Características do loop DO:
  - Contagem explícita, conhecemos o valor inicial e final que o loop deve percorrer.
  - A iteração é automática, não precisamos fazê-la explícitamente.
- Existem duas formas de fazer o loop WHILE:
  - "Loop DO vazio", instrução análoga ao GO TO do Fortran 77;
  - Loop DO WHILE, que é uma forma análoga ao loop WHILE padrão do Fortran 90/95;
- Veja o Exemplo 1.

Alguns detalhes

- Identação;

Alguns detalhes

- Identação;
- Modificação do índice **i**;

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;
- Valor de **i** depois de fechar o loop. Veja o Exemplo 3;

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;
- Valor de **i** depois de fechar o loop. Veja o Exemplo 3;
  - Recalculando

$$index = index + incre,$$

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;
- Valor de **i** depois de fechar o loop. Veja o Exemplo 3;
  - Recalculando

$$index = index + incre,$$

  - Verificando

$$index \times incre \leqslant iend \times incre.$$

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;
- Valor de **i** depois de fechar o loop. Veja o Exemplo 3;
  - Recalculando

$$index = index + incre,$$

  - Verificando

$$index \times incre \leqslant iend \times incre.$$

- **Filtrando** valores de um loop com o atributo CYCLE. Veja o Exemplo 4;

Alguns detalhes

- Identação;
- Modificação do índice **i**;
- O número de iterações deve ser maior que zero:

$$\text{iteração} = \frac{\text{iend} - \text{istart} + \text{incre}}{\text{incre}} > 0$$

- Loops regressivos;
- Valor de **i** quando o loop DO for interrompido. Veja o Exemplo 2;
- Valor de **i** depois de fechar o loop. Veja o Exemplo 3;
  - Recalculando

$$index = index + incre,$$

  - Verificando

$$index \times incre \leqslant iend \times incre.$$

- **Filtrando** valores de um loop com o atributo CYCLE. Veja o Exemplo 4;
- **Nomeando** loops. Veja o Exemplo 5;

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;
  - Mas pode-se fazer o mesmo com instruções WRITE e READ, com bem mais flexibilidade;

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;
  - Mas pode-se fazer o mesmo com instruções WRITE e READ, com bem mais flexibilidade;
  - A instrução PAUSE foi excluída do FORTRAN 95.

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;
  - Mas pode-se fazer o mesmo com instruções WRITE e READ, com bem mais flexibilidade;
  - A instrução PAUSE foi excluída do FORTRAN 95.
- A declaração END isolada:

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;
  - Mas pode-se fazer o mesmo com instruções WRITE e READ, com bem mais flexibilidade;
  - A instrução PAUSE foi excluída do FORTRAN 95.
- A declaração END isolada:
  - Comum antes do FORTRAN 90;

Atributos Fortran redundantes, obsoletos e deletados

- A instrução PAUSE:
  - Quando o usuário precisa ler muitos valores na tela, seria necessário pausar de tempos em tempos a execução para que ele pudesse ler os dados;
  - A instrução PAUSE fazia isso;
  - Mas pode-se fazer o mesmo com instruções WRITE e READ, com bem mais flexibilidade;
  - A instrução PAUSE foi excluída do FORTRAN 95.
- A declaração END isolada:
  - Comum antes do FORTRAN 90;
  - Novo padrão: END PROGRAM, END FUNCTION,...

- Declaração GO TO:

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";
  - Podemos substituí-los, enquanto loop, por um DO vazio.

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";
  - Podemos substituí-los, enquanto loop, por um DO vazio.
  - Veja o Exemplo 10.

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";
  - Podemos substituí-los, enquanto loop, por um DO vazio.
  - Veja o Exemplo 10.
- Declaração IF artimético:

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";
  - Podemos substituí-los, enquanto loop, por um DO vazio.
  - Veja o Exemplo 10.
- Declaração IF artimético:
  - Originário de 1954, nunca deve ser usado em programas modernos, foi tornado obsoleto no Fortran 95;

- Declaração GO TO:
  - Era utilizado para criar ramificações, loops;
  - É recomendado evitar o seu uso, pois seu uso em excesso torna-se um "spaghetti code";
  - Podemos substituí-los, enquanto loop, por um DO vazio.
  - Veja o Exemplo 10.
- Declaração IF artimético:
  - Originário de 1954, nunca deve ser usado em programas modernos, foi tornado obsoleto no Fortran 95;
  - Veja o Exemplo 6.

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

- Podemos "fatiar"uma palavra. Veja o Exemplo 7;

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

- Podemos "fatiar"uma palavra. Veja o Exemplo 7;
- Há também a operação de concatenação (juntar palavras). Veja o Exemplo 8;

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

- Podemos "fatiar"uma palavra. Veja o Exemplo 7;
- Há também a operação de concatenação (juntar palavras). Veja o Exemplo 8;
  - Seja word = string1 // string 2

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

- Podemos "fatiar"uma palavra. Veja o Exemplo 7;
- Há também a operação de concatenação (juntar palavras). Veja o Exemplo 8;
  - Seja word = string1 // string 2
  - Devemos ter len(word) $\geqslant$ len(string) + len(string2)

Manipulando palavras

- Podemos trabalhar com caracteres ao atribuirmos a uma variável o tipo **CHARACTER**;
- Devemos também definir o tamanho, LEN, a **quantidade de caracteres**;

```
CHARACTER(len=5) :: texto
texto = "teste"
```

- Podemos "fatiar"uma palavra. Veja o Exemplo 7;
- Há também a operação de concatenação (juntar palavras). Veja o Exemplo 8;
  - Seja word = string1 // string 2
  - Devemos ter len(word) $\geqslant$ len(string) + len(string2)
  - O que fazer com os espaços em branco?

Funções intrísecas para caracteres

Temos algumas funções nativas do Fortran para manipular caracteres, tais como:

Função	Tipo do argumento	Tipo do resultado	Retorna
ACHAR(num)	INTEGER	CHARACTER	Veja o Exemplo 9
IACHAR(str)	CHARACTER	INTEGER	O contrário de ACHAR
LEN(str)	CHARACTER	INTEGER	O comprimento de str
LEN_TRIM(str)	CHARACTER	INTEGER	Sem os espaços em branco
TRIM(str)	CHARACTER	CHARACTER	str sem espaço em branco

- Veja o Exemplo 10 e 11;

Funções intrísecas para caracteres

Temos algumas funções nativas do Fortran para manipular caracteres, tais como:

Função	Tipo do argumento	Tipo do resultado	Retorna
ACHAR(num)	INTEGER	CHARACTER	Veja o Exemplo 9
IACHAR(str)	CHARACTER	INTEGER	O contrário de ACHAR
LEN(str)	CHARACTER	INTEGER	O comprimento de str
LEN_TRIM(str)	CHARACTER	INTEGER	Sem os espaços em branco
TRIM(str)	CHARACTER	CHARACTER	str sem espaço em branco

- Veja o Exemplo 10 e 11;
- Podemos então, automatizar a criação de um arquivo de dados, veja os Exemplo 12;

Funções intrísecas para caracteres

Temos algumas funções nativas do Fortran para manipular caracteres, tais como:

Função	Tipo do argumento	Tipo do resultado	Retorna
ACHAR(num)	INTEGER	CHARACTER	Veja o Exemplo 9
IACHAR(str)	CHARACTER	INTEGER	O contrário de ACHAR
LEN(str)	CHARACTER	INTEGER	O comprimento de str
LEN_TRIM(str)	CHARACTER	INTEGER	Sem os espaços em branco
TRIM(str)	CHARACTER	CHARACTER	str sem espaço em branco

- Veja o Exemplo 10 e 11;
- Podemos então, automatizar a criação de um arquivo de dados, veja os Exemplo 12;
- Para criar automatizar a criação de vários arquivos de dados, primeiro criamos uma função que converte um número inteiro numa string.

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

- Que conforme o resultado do discriminante Δ, teremos os seguintes resultados:

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

- Que conforme o resultado do discriminante Δ, teremos os seguintes resultados:
  - $\Delta > 0$ Duas raízes distintas;

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

- Que conforme o resultado do discriminante Δ, teremos os seguintes resultados:
  - $\Delta > 0$ Duas raízes distintas;
  - $\Delta = 0$ Apenas uma raiz;

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

- Que conforme o resultado do discriminante Δ, teremos os seguintes resultados:
  - $\Delta > 0$ Duas raízes distintas;
  - $\Delta = 0$ Apenas uma raiz;
  - $\Delta < 0$ Não possui raiz real.

Panorama geral

- Podemos encontrar as raízes **reais** de uma equação de ordem 2, de formato

$$ax^2 + bx + c = 0,$$

- através da famosa relação de Baskara, tal que as raízes sejam:

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4\,(a)\,(c)}}{2a} = \frac{-b \pm \Delta}{2a}.$$

- Que conforme o resultado do discriminante Δ, teremos os seguintes resultados:
  - $\Delta > 0$ Duas raízes distintas;
  - $\Delta = 0$ Apenas uma raiz;
  - $\Delta < 0$ Não possui raiz real.
- Veja o fluxograma para construir um programa que calcula as raízes reais de uma equação de 2^o grau.

Porque incluir os números complexos

- Supomos que estejamos nos primórdios da matemática,e só há números inteiros, maiores do que zero e que só haja a operação de soma;

$$2+3=5 \quad ; \quad 1+3=4 \quad .$$

Porque incluir os números complexos

- Supomos que estejamos nos primórdios da matemática,e só há números inteiros, maiores do que zero e que só haja a operação de soma;

$$2+3=5 \quad ; \quad 1+3=4 \quad .$$

- Então incluímos a operação de subtração;

$$5-2=3 \quad ; \quad 7-2=5 \quad .$$

Porque incluir os números complexos

- Supomos que estejamos nos primórdios da matemática,e só há números inteiros, maiores do que zero e que só haja a operação de soma;

$$2+3=5 \quad ; \quad 1+3=4 \quad .$$

- Então incluímos a operação de subtração;

$$5-2=3 \quad ; \quad 7-2=5 \quad .$$

- Mas gostaríamos de mais liberdade nos cálculos, então incluímos os números negativos e o zero;

$$2-5=-3 \quad ; \quad 4-4=0 \quad .$$

Porque incluir os números complexos

- Supomos que estejamos nos primórdios da matemática,e só há números inteiros, maiores do que zero e que só haja a operação de soma;

$$2+3=5 \quad ; \quad 1+3=4 \quad .$$

- Então incluímos a operação de subtração;

$$5-2=3 \quad ; \quad 7-2=5 \quad .$$

- Mas gostaríamos de mais liberdade nos cálculos, então incluímos os números negativos e o zero;

$$2-5=-3 \quad ; \quad 4-4=0 \quad .$$

- Então começamos a realizar divisões;

$$\frac{4}{2}=2 \quad ; \quad \frac{9}{3}=3 \quad .$$

- Novamente, queremos expandir as possibilidades, então incluímos números não inteiros

$$\frac{-5+2}{2} = -1,5 \quad ; \quad \frac{1}{3} = 0,3333...$$

- Novamente, queremos expandir as possibilidades, então incluímos números não inteiros

$$\frac{-5+2}{2} = -1,5 \quad ; \quad \frac{1}{3} = 0,3333...$$

- E então fazemos mais algumas operações, tais como a raiz quadrada de um número;

$$\sqrt{4} = \pm 2 \quad ; \quad \sqrt{3} = 1.732\,1.$$

- Novamente, queremos expandir as possibilidades, então incluímos números não inteiros

$$\frac{-5+2}{2} = -1,5 \quad ; \quad \frac{1}{3} = 0,3333...$$

- E então fazemos mais algumas operações, tais como a raiz quadrada de um número;

$$\sqrt{4} = \pm 2 \quad ; \quad \sqrt{3} = 1.732\,1.$$

- Chega uma hora que precisaremos calcular uma raiz quadrada, tal como $\sqrt{-4}$,

$$n^2 \quad < \quad 0$$

- Novamente, queremos expandir as possibilidades, então incluímos números não inteiros

$$\frac{-5+2}{2} = -1,5 \quad ; \quad \frac{1}{3} = 0,3333...$$

- E então fazemos mais algumas operações, tais como a raiz quadrada de um número;

$$\sqrt{4} = \pm 2 \quad ; \quad \sqrt{3} = 1.732\,1.$$

- Chega uma hora que precisaremos calcular uma raiz quadrada, tal como $\sqrt{-4}$,

$$n^2 < 0$$

- Mas sabemos que

$$\begin{aligned} (a)\, a &> 0 \\ (-1)\,(-1) &> 0. \end{aligned}$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i , tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i , tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i, tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

- E é a partir daqui que começamos a construir os números complexos.

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i, tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

- E é a partir daqui que começamos a construir os números complexos.
- Definimos que um número complexo c, como composto de duas partes:

$$c = a + ib.$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i, tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

- E é a partir daqui que começamos a construir os números complexos.
- Definimos que um número complexo c, como composto de duas partes:

$$c = a + ib.$$

- Definimos o quadrado de um número complexo

$$c^2 = (a + ib)^2 = a^2 + 2i\,(ab) + b^2.$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i, tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

- E é a partir daqui que começamos a construir os números complexos.
- Definimos que um número complexo c, como composto de duas partes:

$$c = a + ib.$$

- Definimos o quadrado de um número complexo

$$c^2 = (a + ib)^2 = a^2 + 2i\,(ab) + b^2.$$

- e o módulo através do produto com seu complexo conjugado

$$cc^* = (a + ib)\,(a - ib) = a^2 + b^2.$$

Definindo um número complexo (rapidamente)

- Começamos definindo que exista um número especial i, tal que,

$$i^2 = -1 \qquad \rightarrow \qquad i = \sqrt{-1}$$

- Dessa forma,

$$\sqrt{-4} = \sqrt{4\,(-1)} = \sqrt{(-1)}\sqrt{4} = 2i.$$

- E é a partir daqui que começamos a construir os números complexos.
- Definimos que um número complexo c, como composto de duas partes:

$$c = a + ib.$$

- Definimos o quadrado de um número complexo

$$c^2 = (a + ib)^2 = a^2 + 2i\,(ab) + b^2.$$

- e o módulo através do produto com seu complexo conjugado

$$cc^* = (a + ib)\,(a - ib) = a^2 + b^2.$$

- Que nos fornece a magnitude de um número complexo

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

- Além da possibilidade de variáveis do tipo complexo, temos funções intrísecas para trabalhar com números complexos, tais como:

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

- Além da possibilidade de variáveis do tipo complexo, temos funções intrísecas para trabalhar com números complexos, tais como:
  - CMPLX(a,b): converte em um número real, tal que **a** é a parte real e **b** a parte imaginária. Veja o Exemplo 15;

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

- Além da possibilidade de variáveis do tipo complexo, temos funções intrísecas para trabalhar com números complexos, tais como:
  - CMPLX(a,b): converte em um número real, tal que **a** é a parte real e **b** a parte imaginária. Veja o Exemplo 15;
  - CABS(c): retorna o valor absoluto de um número complexo. Veja o Exemplo 16;

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

- Além da possibilidade de variáveis do tipo complexo, temos funções intrísecas para trabalhar com números complexos, tais como:
  - CMPLX(a,b): converte em um número real, tal que **a** é a parte real e **b** a parte imaginária. Veja o Exemplo 15;
  - CABS(c): retorna o valor absoluto de um número complexo. Veja o Exemplo 16;
  - Podemos tomar apenas a parte real, ou apenas a parte imaginária. Veja o Exemplo 17;

Variável do tipo COMPLEX

- Podemos declarar uma variável do tipo COMPLEX, com c=a+ib , tal que,

$$\text{COMPLEX :: c} = (\text{a, b}).$$

- Além da possibilidade de variáveis do tipo complexo, temos funções intrísecas para trabalhar com números complexos, tais como:
  - CMPLX(a,b): converte em um número real, tal que **a** é a parte real e **b** a parte imaginária. Veja o Exemplo 15;
  - CABS(c): retorna o valor absoluto de um número complexo. Veja o Exemplo 16;
  - Podemos tomar apenas a parte real, ou apenas a parte imaginária. Veja o Exemplo 17;
  - As outras funções matemáticas como sin(x), cos(x), log(x), etc, funcionarão normalmente.

Bibliografia

- Introdução ao Fortran:

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:
  - Elementary Mechanics Using Python, Anders Malthe-Sorenssen (2015). 591p;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:
  - Elementary Mechanics Using Python, Anders Malthe-Sorenssen (2015). 591p;
  - A primer on Scientific Programing With Python, Hahns Peter Langtangen (2016). 942p;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:
  - Elementary Mechanics Using Python, Anders Malthe-Sorenssen (2015). 591p;
  - A primer on Scientific Programing With Python, Hahns Peter Langtangen (2016). 942p;
  - Introdução à Ciência da Computação com Python, Coursera - USP;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:
  - Elementary Mechanics Using Python, Anders Malthe-Sorenssen (2015). 591p;
  - A primer on Scientific Programing With Python, Hahns Peter Langtangen (2016). 942p;
  - Introdução à Ciência da Computação com Python, Coursera - USP;
  - Using Python for Research, EDX - Harvard;

Bibliografia

- Introdução ao Fortran:
  - Fortran 95/2003 for Scientists and Engineers, Stephen K. Chapman (2007). 974p;
  - Numerical Mathematics and Scientific Computation, Michael Metcalf (2004). 434p.
- Aplicação de métodos numéricos utilizando Fortran:
  - Computational Physics, Steven E. Koonin (1998). 653p;
  - Computational Physics, Nicholas J. Giordano (1997). 432p;
  - A First Course in Computational Physics, Paul L. DeVries (1994). 435p.
- Python:
  - Elementary Mechanics Using Python, Anders Malthe-Sorenssen (2015). 591p;
  - A primer on Scientific Programing With Python, Hahns Peter Langtangen (2016). 942p;
  - Introdução à Ciência da Computação com Python, Coursera - USP;
  - Using Python for Research, EDX - Harvard;
  - CS50's Introduction to Programming with Python, EDX - Harvard.