

Introdução a TI

Victor Machado da Silva, MSc
victor.silva@professores.ibmec.edu.br



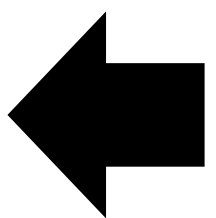
Índice

Introdução ao curso

- Afinal, o que é programar?
- Linguagens de programação

Algoritmos e Lógica de Programação

- O que são algoritmos?
- Como representar algoritmos
- Estruturas de seleção
- Estruturas de repetição



Introdução ao curso

Apresentação do curso

Informações gerais:

- Contato: victor.silva@professores.ibmec.edu.br
- Aulas às sextas-feiras, de 16:30 às 18:00
- Site com materiais: <http://victor0machado.github.io>

Apresentação do curso

Agenda de aulas:

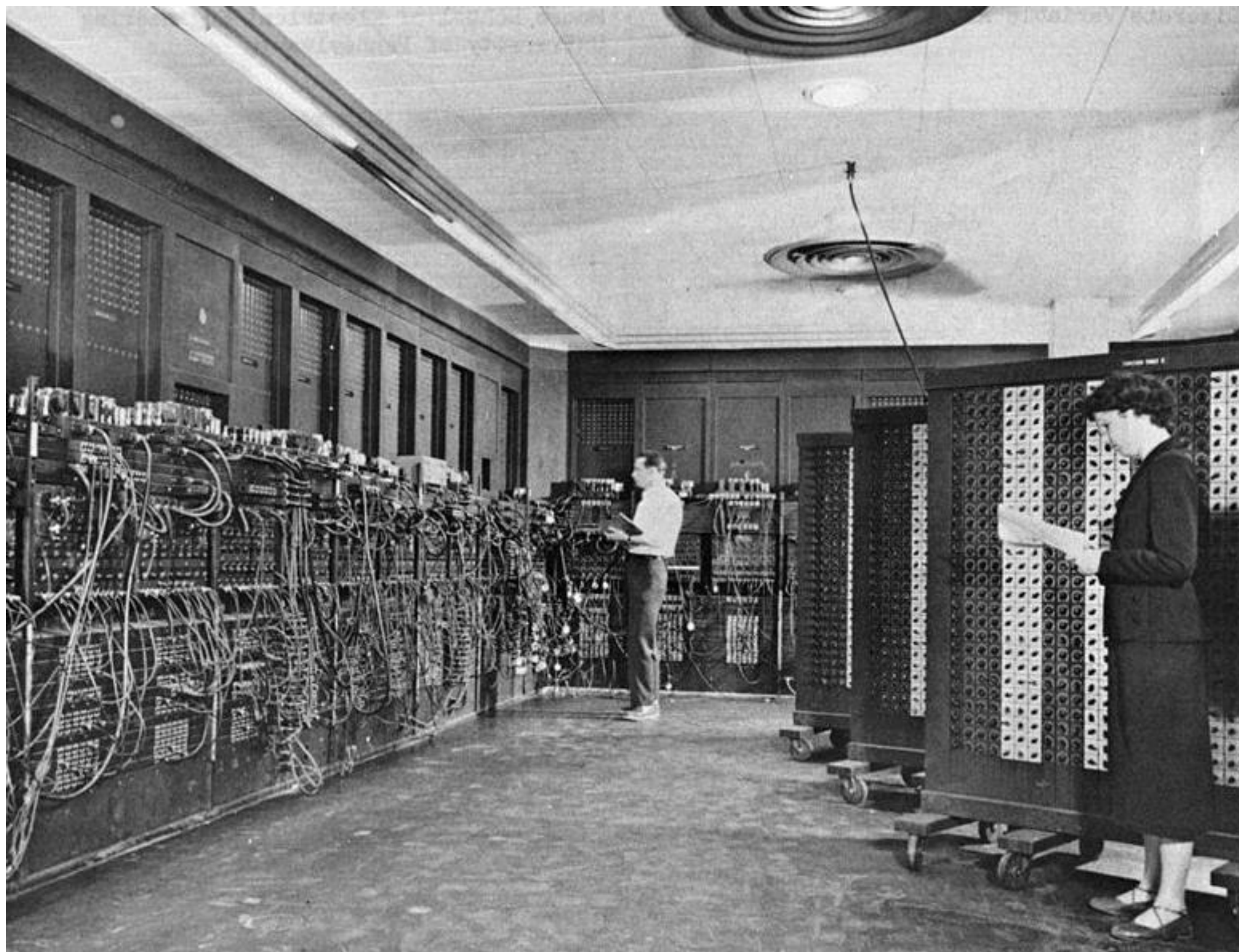
- Introdução à computação
- Algoritmos e Lógica de Programação
- Programação com Scratch
- Desenvolvendo jogos com Scratch
- Apresentação dos trabalhos

Projeto final:

- Construção de um jogo com Scratch, usando os conhecimentos desenvolvidos durante o curso

Afinal, o que é “programar”?

Um pouco de história...



- ENIAC, primeiro computador eletrônico da história, entrou em operação logo após a II Guerra Mundial
- 30 toneladas, em uma sala de 10m x 15m
- Dezenas de pessoas operando ao mesmo tempo, como telefonistas

Programação era configurar o computador para que ele realize operações específicas

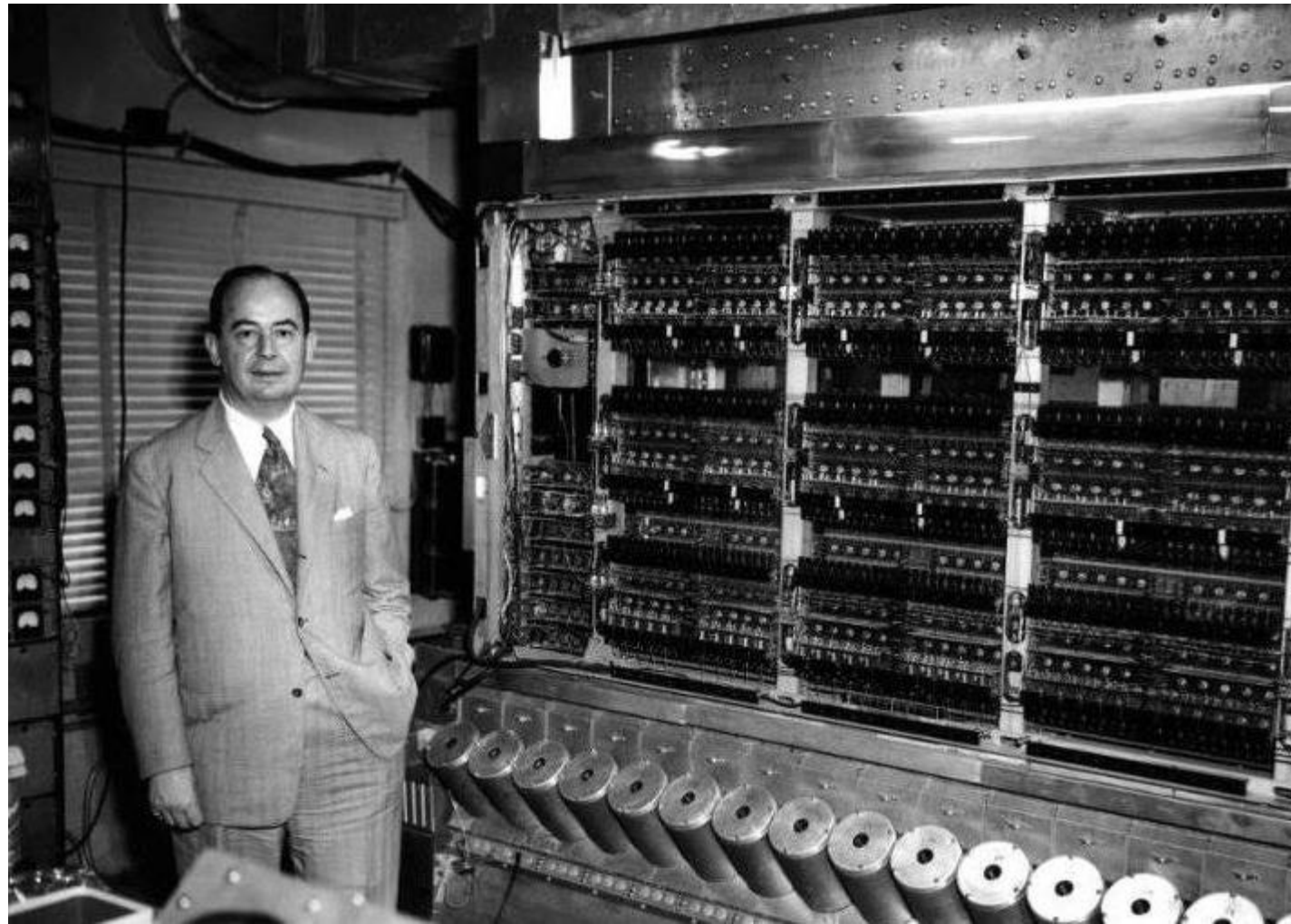
Afinal, o que é “programar”?

Limitações da época:

- Computadores com programas fixos
 - Uma alteração simples no programa do ENIAC poderia demorar três semanas ou mais para que ele voltasse a trabalhar!
 - Computadores eram como calculadoras de mesa...
- Processo de programação era altamente complexo, envolvia configurar sinais específicos em linguagem binária, ou seja, utilizando 0s e 1s
- Os computadores ficavam restritos a grandes corporações ou a instituições governamentais

Afinal, o que é “programar”?

Duas pessoas fundamentais para a mudança de paradigma



John von Neumann, 1903-1957

- Um dos construtores do ENIAC
- Desenvolveu a **arquitetura de von Neumann**, usada até hoje
- Permitiu a construção de computadores de programas armazenados



Grace Hopper, 1906-1992

- Uma das construtoras do UNIVAC I
- Criadora do primeiro compilador da história, o A-0
- Desenvolveu uma das primeiras linguagens de programação de alto nível, o Flow-Matic
- Possível criadora do termo “bug” para indicar defeitos no software

Afinal, o que é “programar”?

Com a criação dos compiladores e a constante evolução dos hardwares e softwares, o processo de programação se tornou muito mais simples.

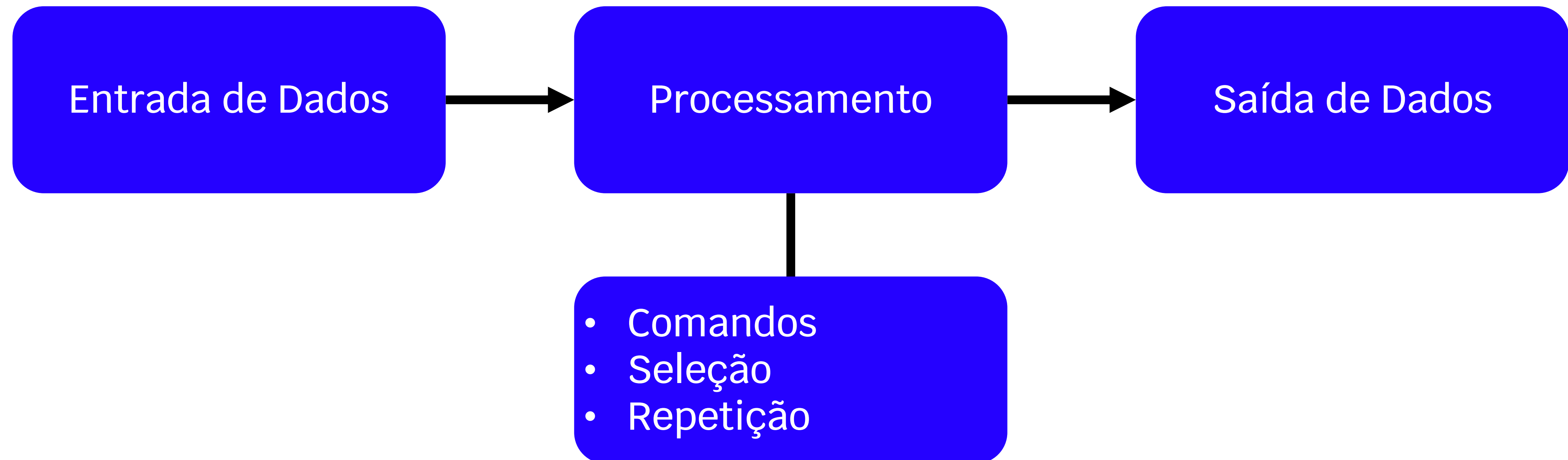
Em essência, sua definição continua a mesma:

Programar um computador significa fornecer à máquina um conjunto de instruções que indica o que você quer que ela faça

Quando você pressiona o botão “enviar” no seu aplicativo de e-mail, está comandando a máquina para que ela produza um determinado resultado. Ela reage a esse comando executando um conjunto de instruções previamente programadas pelos criadores do aplicativo, as quais realizam a tarefa de enviar um e-mail.

Afinal, o que é “programar”?

A programação se resume a uns poucos conceitos fundamentais



Linguagens de programação

O conceito de compiladores criado por Grace Hopper trouxe uma nova perspectiva sobre como programar. A partir de então, já não era mais necessário escrever código em linguagem de máquina, ou seja, com 0s e 1s.

Os conjuntos pré-definidos de instruções, que são traduzidos pelo compilador, passaram então a ser chamados de **linguagens de programação**.



Alan Turing, 1912-1954

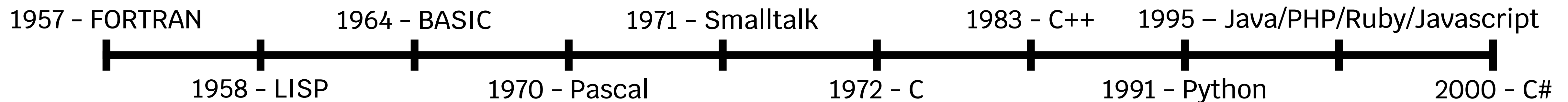
- Considerado o pai da computação moderna e da inteligência artificial
- Responsável por quebrar a segurança da máquina Enigma, usada pelo exército nazista na II Guerra Mundial
- Propôs a máquina de Turing, computador teórico que é considerado até hoje como o melhor modelo de um computador de uso geral
- Propôs tabelas de instruções que automaticamente converteriam a escrita decimal em códigos numéricos, permitindo que computadores fizessem muito mais que operações matemáticas

Linguagens de programação

As linguagens que utilizavam os conceitos propostos por Hopper (com os compiladores) e Turing (com as tabelas de instruções) passaram a ser chamadas de **linguagens de alto nível**, ou seja, linguagens que são facilmente lidas e interpretadas por seres humanos.

As **linguagens de baixo nível** são aquelas legíveis apenas para máquinas. Atualmente, temos pouquíssimos programadores de linguagens de baixo nível.

Com o avanço da capacidade dos hardwares, começaram a surgir algumas linguagens de alto nível **interpretadas**, ao invés de compiladas.



Linguagens de programação

Existe uma diferença fundamental entre a linguagem humana (ou natural) e a linguagem de computadores. Os seres humanos são capazes de compreender expressões mesmo que elas não estejam escritas corretamente.

Olá! Td bem c vc?

Um tópico fundamental em programação é:

Programas são escritos em uma linguagem formal. Cada letra e cada sinal matemático ou de pontuação têm um significado muito preciso. Mudar qualquer um destes sinais pode, muitas vezes, mudar o sentido do programa, ou até fazê-lo parar de funcionar.

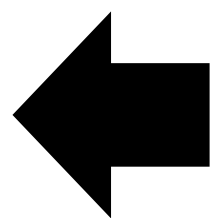
```
1  for cont in range(10):  
2  |  print(str(cont))  
3  print(str(cont))
```

```
1  for cont in range(10):  
2  |  print(str(cont))  
3  |  print(str(cont))
```

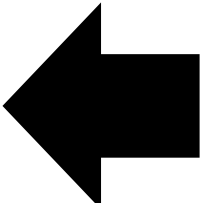
O que é um programa?

Um *programa* consiste em um texto escrito em uma determinada linguagem de programação, seja de alto ou baixo nível, que por sua vez é processada de forma a comunicar ao computador comandos que devem ser executados.

O texto escrito do programa (ou código) pode ser armazenado no computador com extensões que variam conforme a linguagem de programação que estiver sendo utilizada.



Algoritmos e Lógica de Programação



O que são algoritmos?

Uma resposta geral seria "um conjunto de etapas para executar uma tarefa". Todos nós executamos algoritmos na nossa vida diária, desde escovar os dentes ao dirigir um carro.

Para fritar um ovo, por exemplo, temos uma sequência de etapas:

- Pegar um ovo na geladeira;
- Selecionar uma frigideira e colocá-la no fogão;
- Acender a chama do fogão;
- Colocar um pouco de óleo na frigideira;
- Quebrar o ovo e temperar;
- Aguardar N segundos e servir.

O que são algoritmos?

Basicamente, para qualquer tarefa do dia-a-dia, precisamos de uma sequência de etapas, que normalmente já está internalizado nas nossas cabeças.

Da mesma forma como precisamos de algoritmos para termos um funcionamento básico no nosso dia, os computadores também o precisam.

- Rotas de aplicativos como o Waze;
- Compra de produtos na Amazon;
- Uso de um micro-ondas.

A principal diferença entre os algoritmos dos humanos e os de computador é que, nos humanos, conseguimos tolerar uma pouca precisão nas descrições.

- Dizer "quebrar um ovo" pode ser extremamente ambíguo!

O que são algoritmos?

Portanto, quando definimos um algoritmo de computador, precisamos descrever como sendo **"um conjunto de etapas para executar uma tarefa descrita com precisão suficiente para que um computador possa executá-la"**.

Usualmente, um bom algoritmo de computador precisa de algumas características:

- É preciso: todas as etapas e pré-condições precisam ser bem definidas;
- Deve possuir um nível adequado de corretude;
- Precisa ser eficiente.

Como representar algoritmos

Terminologia usada:

- Procedimentos
- Chamada
- Parâmetros
- Retorno

```
PROCEDIMENTO incremento(valor)
-----
# Entrada: parâmetro valor é um número.
# Saída: o parâmetro valor acrescido de uma unidade.
-----
1. Encerra o procedimento, retornando valor + 1.
```

```
PROCEDIMENTO algoritmo_inutil()
-----
# Entrada: Nada.
# Saída: Nada.
-----
1. Atribui 1 à variável numero.
2. Atribui à variável numero o retorno do procedimento incremento(numero).
3. Encerra o procedimento.
```

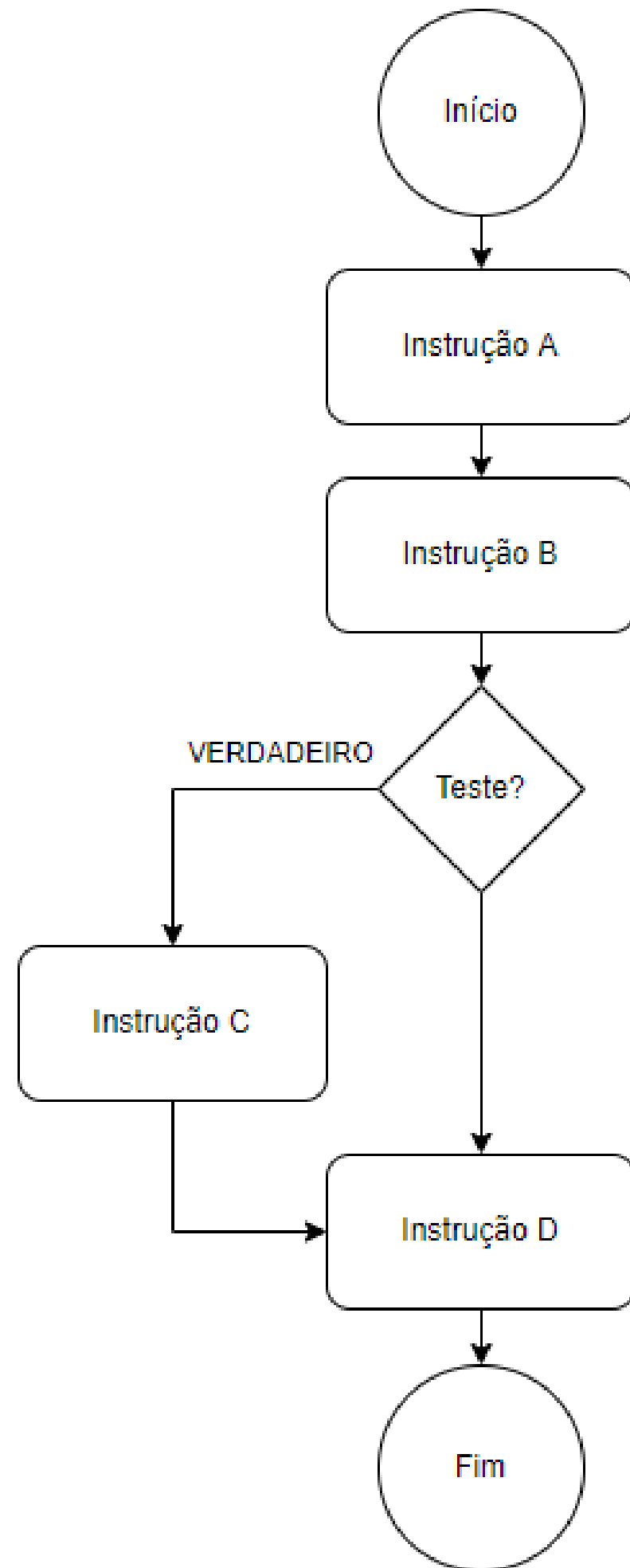
Como representar algoritmos

Noção de arranjos:

- Arranjo agrega dados do mesmo tipo em uma única entidade;
- Considere o arranjo ao lado, que chamamos de `lista_compras`;
- O elemento de índice 3 na tabela é Carne, e representamos por `lista_compras[3]`;
- Arranjos começam pelo índice 0!
- O tempo para localizar qualquer elemento de um arranjo é o mesmo, independentemente de sua posição.

Índice	Item para comprar
0	Pão
1	Queijo
2	Ovo
3	Carne
4	Manteiga

Estruturas de seleção



```
PROCEDIMENTO exibe_nota(nota)
```

```
-----  
# Entrada: parâmetro nota é um número entre 0 e 10.
```

```
# Saída: nenhuma.  
-----
```

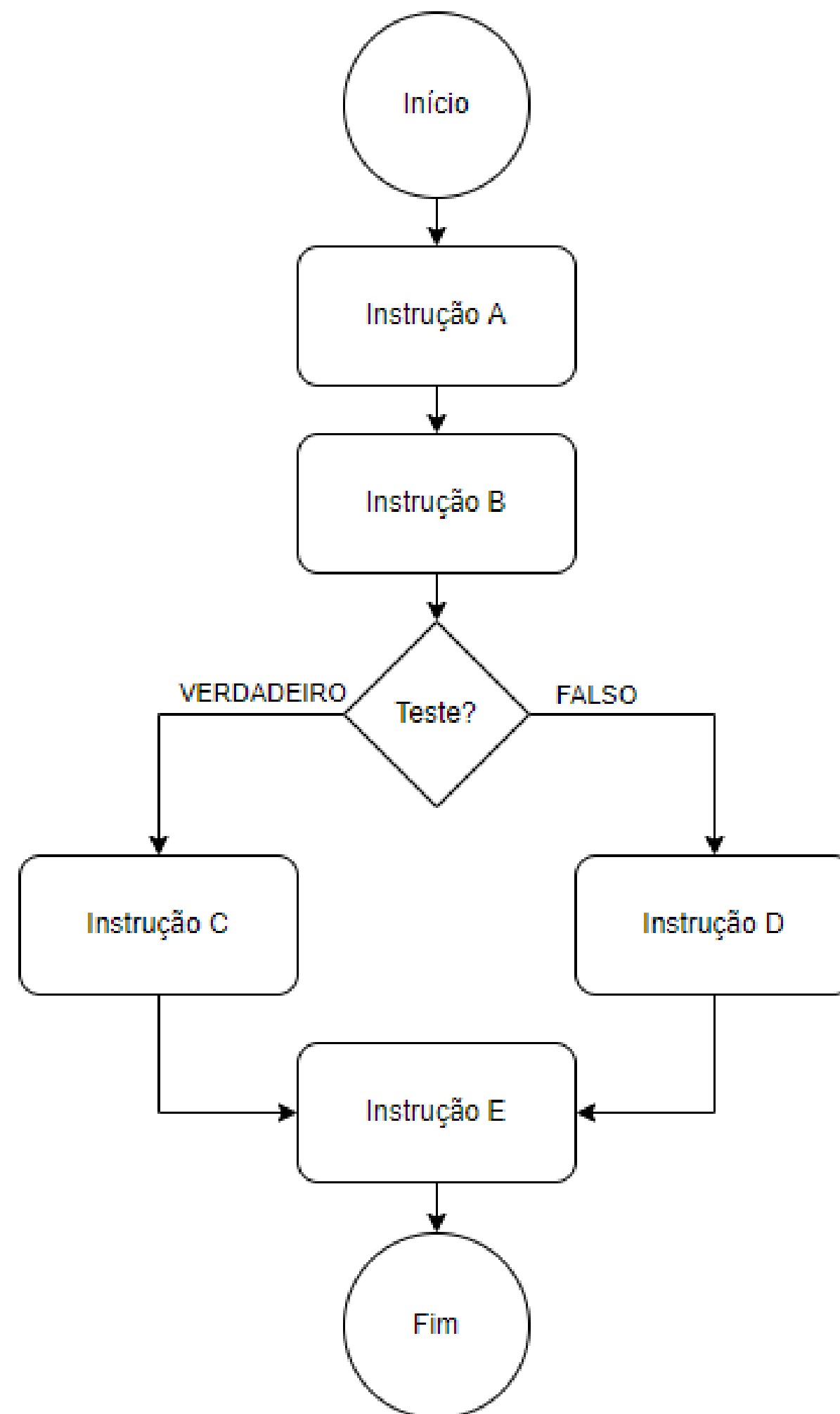
```
1. Se nota > 9.0 faça:
```

```
    1.1. Exiba na tela a mensagem "Parabéns!".
```

```
2. Exiba na tela a mensagem "Sua nota foi ", seguida pelo valor de nota.
```

```
3. Encerre o procedimento.
```

Estruturas de seleção



```
PROCEDIMENTO exibe_nota(nota)
```

```
-----  
# Entrada: parâmetro nota é um número entre 0 e 10.  
# Saída: nenhuma.  
-----
```

```
1. Se nota > 9.0 faça:
```

```
    1.1. Exiba na tela a mensagem "Parabéns!".
```

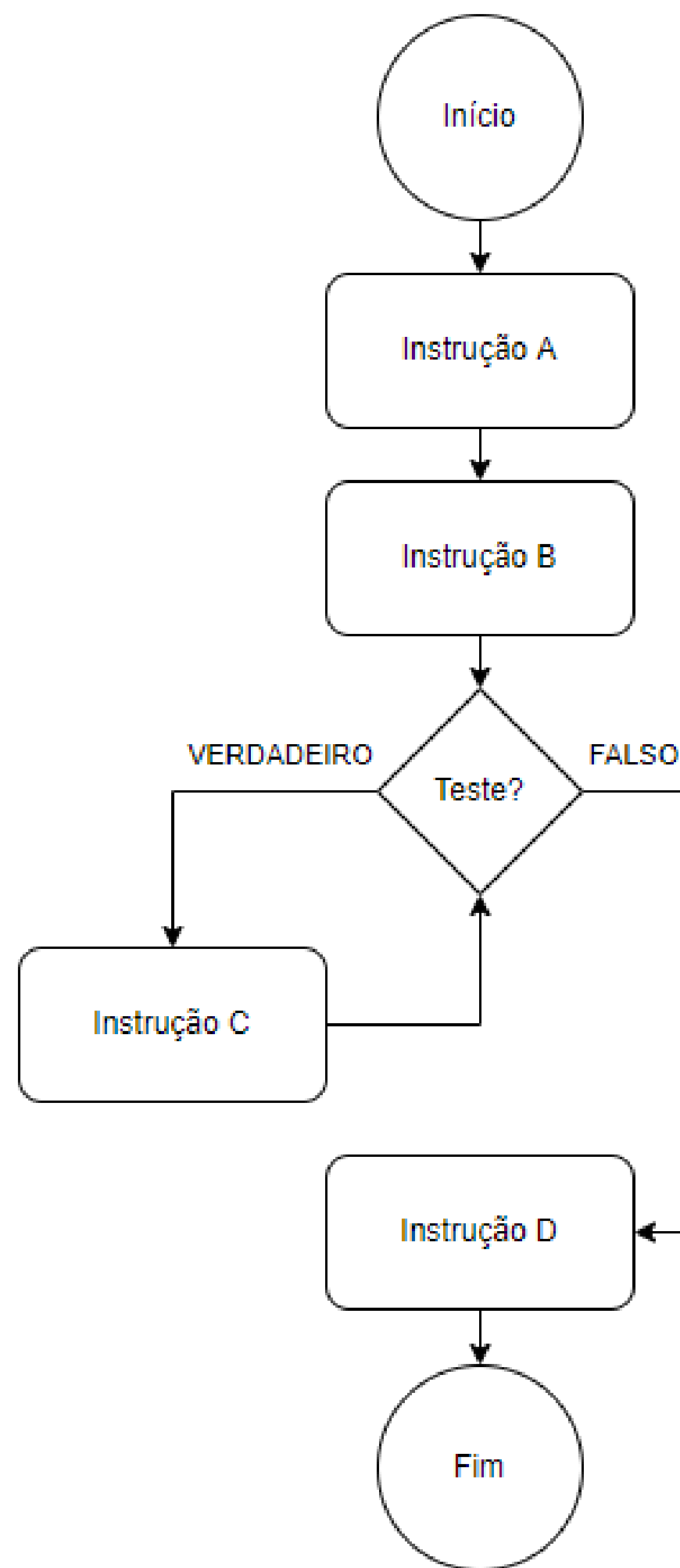
```
2. Senão, faça:
```

```
    2.1. Exiba na tela a mensagem "Continue estudando!".
```

```
3. Exiba na tela a mensagem "Sua nota foi ", seguida pelo valor de nota.
```

```
4. Encerre o procedimento.
```

Estruturas de repetição



```
PROCEDIMENTO progressao_geometrica(base)
```

```
-----  
# Entrada: parâmetro base é um número inteiro.
```

```
# Saída: nenhuma.  
-----
```

```
1. Se  $base < 0$ , encerre o procedimento.
```

```
2. Considere uma variável valor igual a 1, e uma total_elementos igual a 0.
```

```
3. Enquanto  $valor < 1000$ , faça:
```

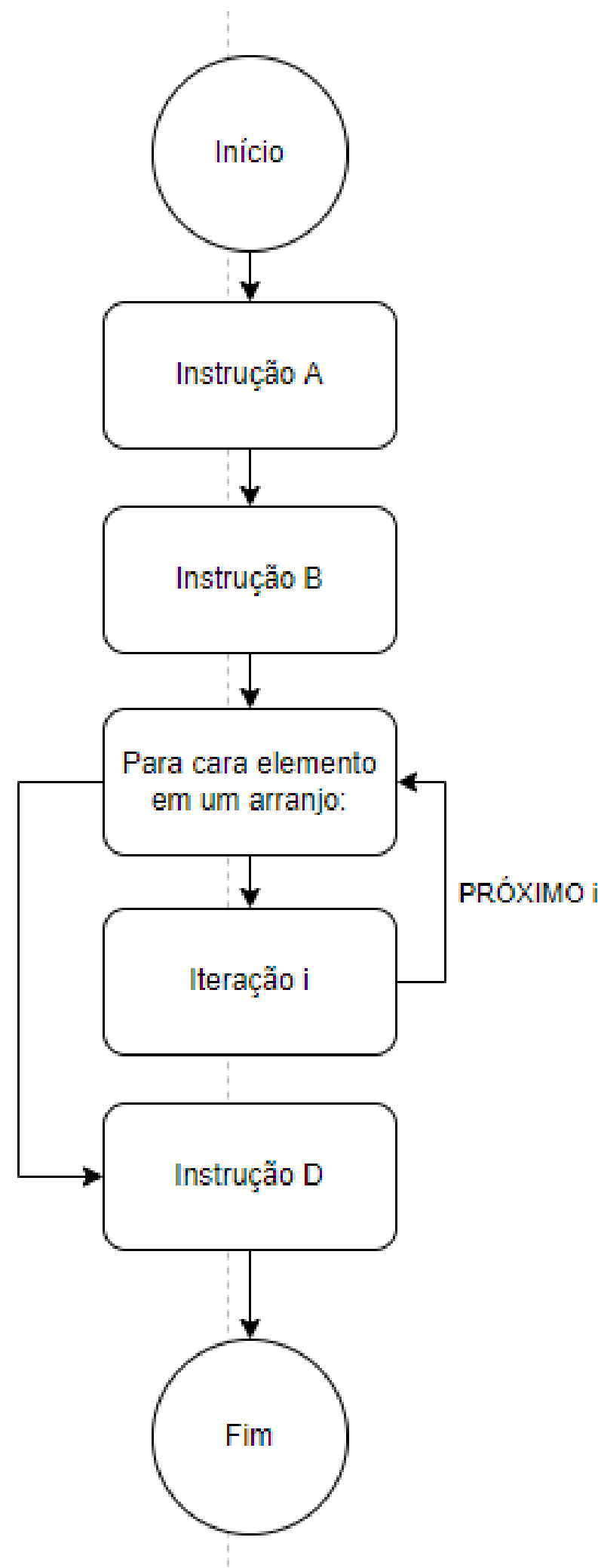
```
    3.1. Exiba valor na tela.
```

```
    3.2. Acrescente 1 a total_elementos.
```

```
    3.3. Atribua para valor o resultado de  $valor * base$ .
```

```
4. Encerre o procedimento, retornando total_elementos como saída.
```

Estruturas de repetição



```
PROCEDIMENTO busca_linear(A, n, x)
```

```
-----
```

```
# Entradas:
```

```
# - A: um arranjo.
```

```
# - n: o número de elementos em A no qual procurar.
```

```
# - x: o valor que está sendo procurado.
```

```
# Saída: um índice i para o qual A[i] = x ou o valor -1,
```

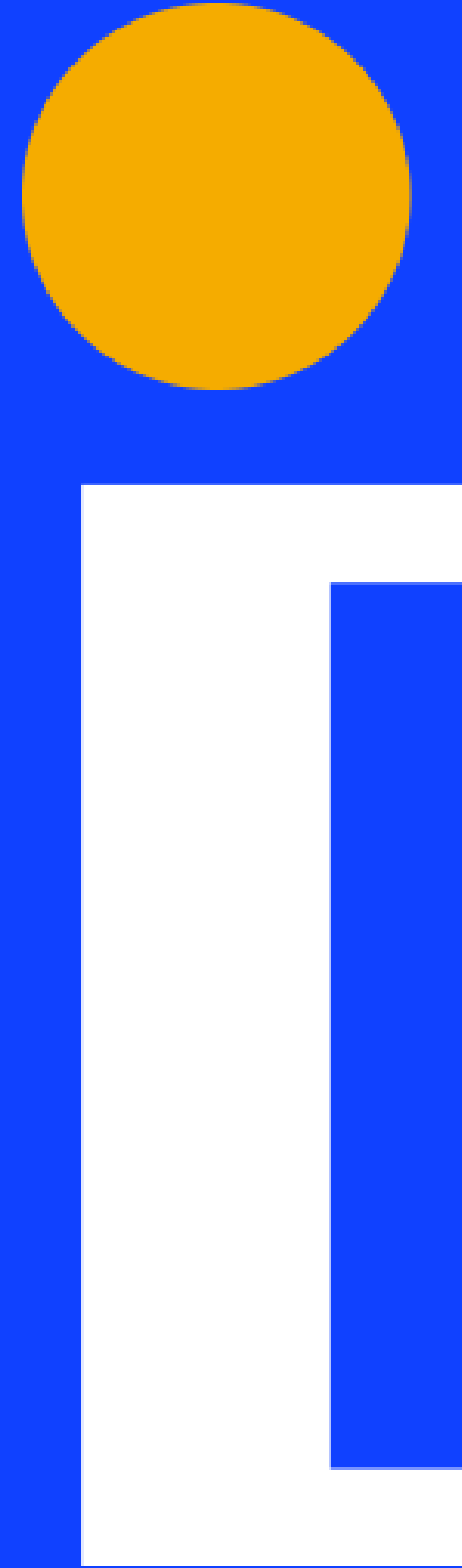
```
# que indica que o valor de x não foi encontrado no arranjo.
```

```
-----
```

```
1. Para i = 0 até n, faça:
```

```
  1.1. Se A[i] = x, então retorne o valor de i como saída.
```

```
2. Retorne -1 como saída.
```

IBMEC.BR

 /IBMEC

 IBMEC

 @IBMEC_OFICIAL

 @IBMEC

 **ibmec**