

# ESTRUTURA SEQUENCIAL

O objetivo deste material é **introduzir** as alunas à estrutura sequencial. Será feita uma abordagem focando na linguagem Java.

Estrutura sequencial é um conjunto de instruções no qual cada instrução será executada em sequência. Essa sequência é executada da seguinte maneira:

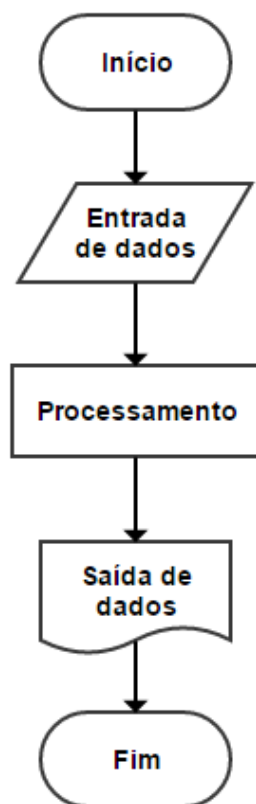


Figura 1: Fluxograma Lógica de Programação

## Cocada

- **INGREDIENTES:**

500 g de coco fresco ralado  
1 lata de leite condensado  
2 latas de açúcar  
1 colher (sopa) de manteiga

**PREPARO:**

1. Em uma panela, coloque os ingredientes e leve ao fogo baixo, mexendo até soltar da panela.
2. Retire do fogo e coloque em uma superfície lisa e untada.
3. Alise com uma espátula e deixe esfriar. Corte em pedaços e sirva.
4. Dica: se desejar regue com leite condensado.

## Estrutura básica de um programa em Java

Como qualquer linguagem de programação, a linguagem Java tem sua própria estrutura, regras de sintaxe e paradigma de programação. O paradigma de programação da linguagem Java baseia-se no conceito de OOP, que os recursos da linguagem suportam.

Estruturalmente, a linguagem Java começa com *pacotes*. Um pacote é o mecanismo de namespace da linguagem Java. Dentro dos pacotes estão as classes e dentro das classes estão métodos, variáveis, constantes e mais.

```
package code.girls.estruturaSequencial;

public class Hello {

    public static void main(String[] args) {

        String nome = "Code Girls";
        System.out.println("Hello, " + nome);

    }
}
```

O diagrama mostra o código Java com várias anotações em português explicando as partes:   
- **pacote**: aponta para o `code.girls.estruturaSequencial`.   
- **Nome da classe**: aponta para `Hello`.   
- **Nome do metodo**: aponta para `main`.   
- **Metodo (Neste caso, ponto de entrada da aplicacao)**: aponta para o `main` e seus parâmetros.   
- **Tipo da variavel**: aponta para `String`.   
- **variavel**: aponta para `nome`.   
- **Valor atribuido a variavel**: aponta para `"Code Girls"`.   
- **Comando de saida para o console**: aponta para `System.out.println`.   
- **Argumentos que serao exibidos**: aponta para `"Hello, " + nome`.

Figura 2 Estrutura básica de um programa Java

### Método para Construção de Algoritmos:

1. Ler atentamente o enunciado
2. Descobrir no enunciado o conjunto de entradas (dados que são fornecidos)
3. Descobrir no enunciado o conjunto de saídas (resultados que se deseja obter)
4. Determinar o que deve ser feito para transformar as entradas em saídas desejadas
5. Construir o algoritmo
6. Executar o algoritmo verificando se produz os resultados esperados (teste de mesa)

# Variável

Um algoritmo e, posteriormente um programa, recebe dados. Tais dados precisam ser armazenados na memória do computador para serem utilizados no processamento. Para isso, utilizam-se as variáveis, que nada mais são que espaços reservados na memória RAM do computador. Para cada variável, atribuímos um nome e definimos seu tipo.

**Palavras reservadas:** Como qualquer linguagem de programação, a linguagem Java designa determinadas palavras que o compilador reconhece como especiais. Por essa razão, você não pode usá-las para nomear suas construções Java. Ex: `package`, `public`, `class`, `static`, `void` e etc.

## Tipos de Dados Primitivos

Tipo	Tamanho	Valor padrão	Faixa de valores
<code>boolean</code>	n/a	false	true ou false
<code>byte</code>	8 bits	0	-128 a 127
<code>char</code>	16 bits	(não designado)	\u0000' \u0000' a \uffff' ou 0 a 65535
<code>short</code>	16 bits	0	-32768 a 32767
<code>int</code>	32 bits	0	-2147483648 a 2147483647
<code>long</code>	64 bits	0	-9223372036854775808 a 9223372036854775807
<code>float</code>	32 bits	0,0	1.17549435e-38 a 3.4028235e+38
<code>double</code>	64 bits	0,0	4.9e-324 a 1.7976931348623157e+308

Figura 3: Tipos de dados primitivos

**STRING:** Na linguagem Java, variáveis string são objetos de primeira classe do tipo `String`, com métodos que ajudam a manipulá-los.

# Operadores Aritméticos da linguagem Java

Operador	Uso	Descrição
<b>+</b>	<code>a + b</code>	Inclui <code>a</code> e <code>b</code>
<code>+</code>	<code>++a</code>	Promove <code>a</code> para <code>int</code> se ele for um <code>byte</code> , <code>short</code> ou <code>char</code>
<b>-</b>	<code>a - b</code>	Subtrai <code>b</code> de <code>a</code>
<code>-</code>	<code>-a</code>	Nega aritmeticamente <code>a</code>
<b>*</b>	<code>a * b</code>	Multiplica <code>a</code> e <code>b</code>
<b>/</b>	<code>a / b</code>	Divide <code>a</code> por <code>b</code>
<code>%</code>	<code>a % b</code>	Retorna o restante da divisão de <code>a</code> por <code>b</code> (o operador de módulo)
<code>++</code>	<code>a++</code>	Incrementa <code>a</code> por 1; calcula o valor de <code>a</code> antes de incrementar
<code>++</code>	<code>++a</code>	Incrementa <code>a</code> por 1; calcula o valor de <code>a</code> depois de incrementar
<code>--</code>	<code>a--</code>	Decrementa <code>a</code> por 1; calcula o valor de <code>a</code> antes do decremento de
<code>--</code>	<code>--a</code>	Decrementa <code>a</code> por 1; calcula o valor de <code>a</code> após decremento de
<code>+=</code>	<code>a += b</code>	Abreviação de <code>a = a + b</code>
<code>-=</code>	<code>a -= b</code>	Abreviação de <code>a = a - b</code>
<code>*=</code>	<code>a *= b</code>	Abreviação de <code>a = a * b</code>
<code>%=</code>	<code>a %= b</code>	Abreviação de <code>a = a % b</code>

Figura 4: Operadores Aritméticos da linguagem Java

# Operadores Relacionais e Condicionais

Operador	Uso	Retorna true se...
>	a > b	a for maior que b
>=	a >= b	a é maior que ou igual a b
<	a < b	a é menor que b
<=	a <= b	a é menor que ou igual a b
==	a == b	a é igual a b
!=	a != b	a não é igual a b
&&	a && b	a e b são true, condicionalmente avalia b (se a for false, b não será avaliado)
`a` & `b`		a ou b é true; condicionalmente avalia b (se a for true, b não será avaliado)
!	!a	a é false
&	a & b	a e b são true; sempre avalia b
`a`   `b`		a ou b é true; sempre avalia b
^	a ^ b	a e b são diferentes

Operador ( || )    Uso (int idade > 18 || idade < 65)  
É muito utilizado quando queremos decidir o fluxo de uma estrutura de decisão, por exemplo.

Figura 5: Operadores Relacionais e Condicionais

# Precedência de Operadores

As regras de precedências de operadores especificam a ordem que a linguagem C utiliza para avaliar expressões aritméticas, sendo, geralmente, iguais às regras da álgebra.

Ordem de precedência (do maior para o menor)	Operadores
1	++ e -- pós-fixados
2	+ e - unários, ++ e -- pré-fixados, !
3	*, /, %
4	+ e - binários
5	<, >, <=, >=
6	=, !=
7	& &
8	

SEBESTA, R. W. **Conceitos de linguagens de programação**.  
Porto Alegre: Bookman, 2011 (adaptado).

## Referências:

[https://gabrielbueno072.github.io/rea-aed/aula\\_seq.html](https://gabrielbueno072.github.io/rea-aed/aula_seq.html)

<https://developer.ibm.com/br/tutorials/j-introtojava1/>

[https://www.inf.pucrs.br/~emoreno/undergraduate/EC/alproI/class\\_files/Aula02-sequencial.pdf](https://www.inf.pucrs.br/~emoreno/undergraduate/EC/alproI/class_files/Aula02-sequencial.pdf)