



# Aula 03 - Diagrama de Blocos

☰ Ciclo	Ciclo 01: Lógica de Programação
# Aula	3
🕒 Created	@October 19, 2022 2:37 PM
☑ Reviewed	<input type="checkbox"/>
📎 Material PDF	
☑ Finished	<input checked="" type="checkbox"/>
📄 Status	

## ▼ Diagrama de Bloco

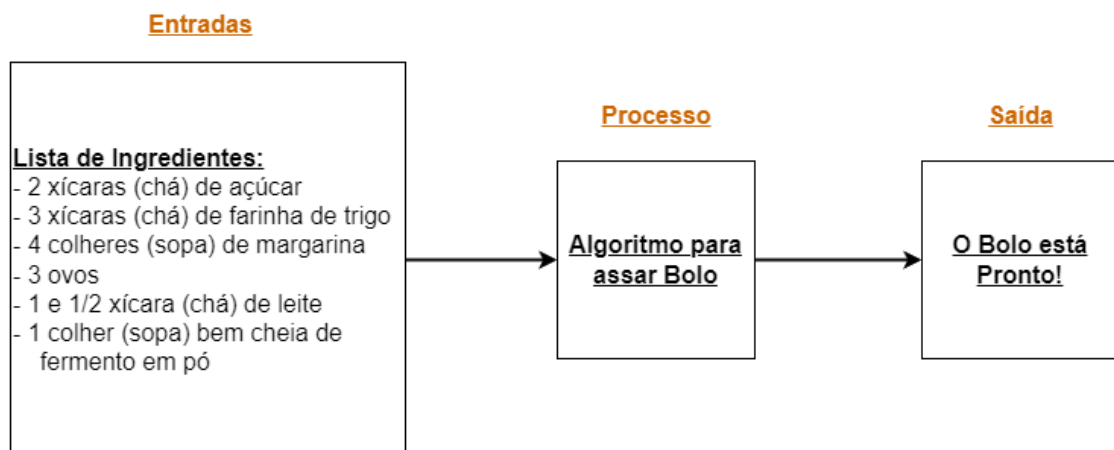
Uma das maneiras de escrevermos **algoritmos**, de **forma visual**, é utilizando **Diagramas de Blocos**. Diagramas de Blocos são como **fluxogramas**, onde **cada bloco representa uma ação** e cada bloco deve seguir um fluxo: Quando um bloco termina, o próximo bloco é executado, até que toda a sequência tenha terminado.

Para exemplificar melhor, vamos criar desenhar o algoritmo de como fazer um bolo, utilizando diagrama de blocos de para exemplificar. Primeiro vamos pensar nas entradas. Elas são a lista de ingredientes:

## Entradas

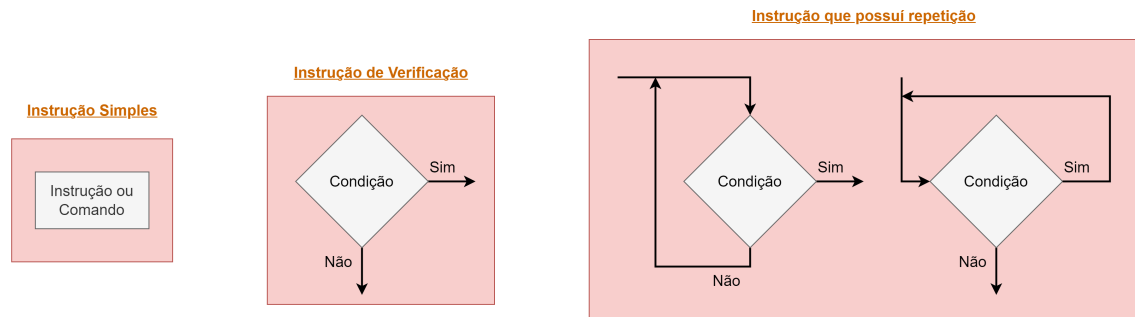
- 2 xícaras (chá) de açúcar
- 3 xícaras (chá) de farinha de trigo
- 4 colheres (sopa) de margarina
- 3 ovos
- 1 e 1/2 xícara (chá) de leite
- 1 colher (sopa) bem cheia de fermento em pó

Feito isso, já sabemos o que desejamos, que é o bolo. Então, teremos mais ou menos o seguinte processo:



Ou seja, entramos com os ingredientes, temos o processo, ou algoritmo, para misturar os ingredientes e assar o bolo, e como saída, temos o bolo. Antes de começarmos a desenhar e criar o nosso algoritmo para a construção do bolo,

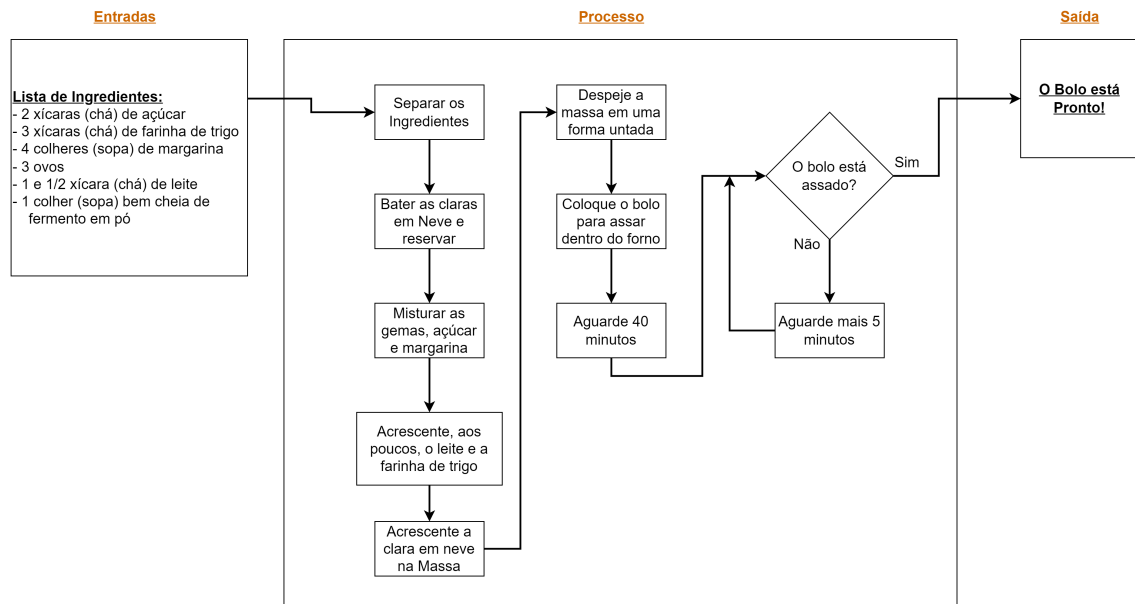
vamos primeiro verificar quais instruções básicas podemos utilizar em um diagrama de blocos:



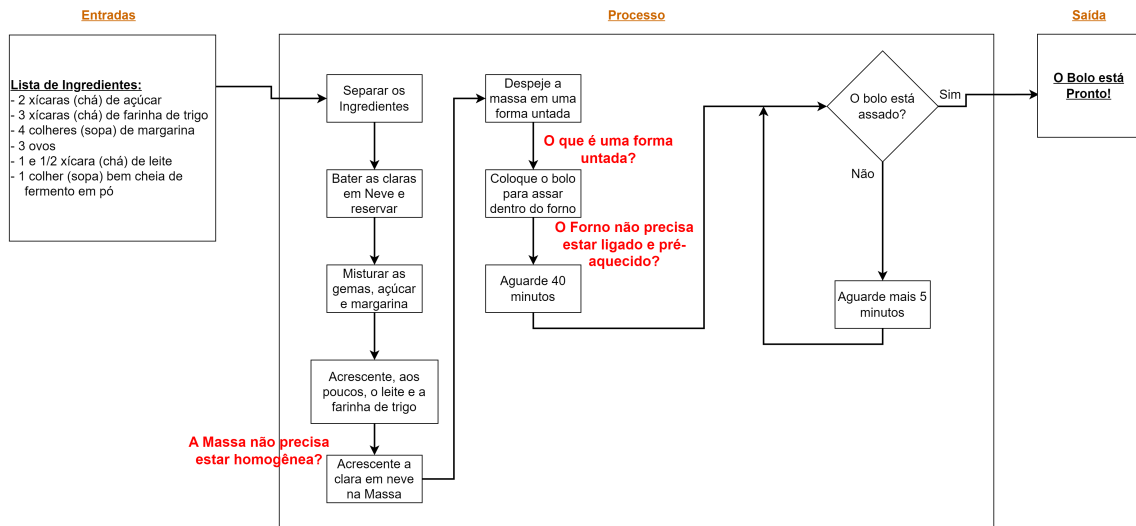
Cada uma dessas instruções serve para algo:

- A **Instrução Simples ou Comando** pode ser utilizado para inserirmos os principais comandos que temos, como misturar os ingredientes da massa, ou colocar a massa na forma.
- A **Instrução de Verificação** pode ser usada quando desejamos verificar algo dentro do fluxo de ações. Ou seja, se precisarmos verificar se o bolo está ou não assado, podemos usar uma Instrução de Verificação.
- E a **Instrução que possui Repetição** serve para criarmos um fluxo que se repita. Novamente, se verificarmos se o bolo está ou não está assado, e constatarmos que ele de fato **NÃO** está assado, temos que esperar mais tempo, afinal, ninguém quer comer um bolo cru. E, uma vez que esse tempo de espera acaba, **temos que voltar para verificar se o bolo está ou não assado**. E faremos esse processo de verificação quantas vezes forem necessárias até que o bolo finalmente esteja assado.

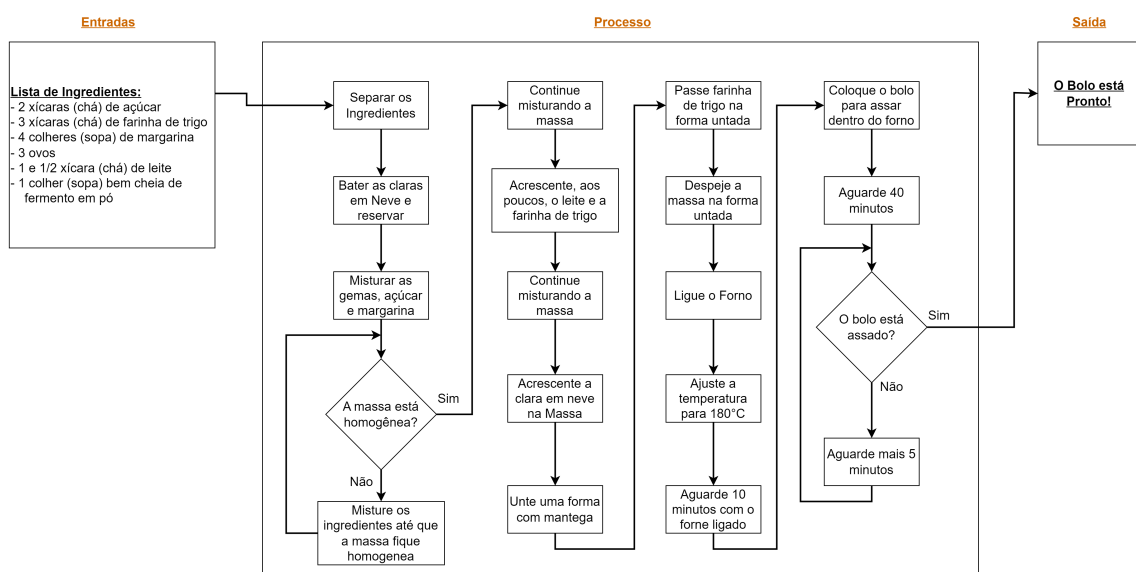
Agora que sabemos os blocos básicos, podemos descrever o algoritmo de feitiço do bolo da seguinte maneira:



Se você entregar esse processo para um **ser humano adulto** e não explicar do que se trata, ele conseguirá identificar que é isso é um processo (**algoritmo**) para fazer um bolo. E se pedirmos ainda para que ele siga esse processo, **muito provavelmente** ele conseguirá produzir esse bolo. Isso porque a **lógica para extrapolar e deduzir informações faltantes está intrínseca na nossa maneira de agir**. Ou seja, por sermos seres pensantes e que conseguimos raciocinar, conseguimos não só pegar as nossas experiências anteriores como também deduzir coisas que não estão tão claras, como por exemplo: O que é uma forma untada? Ou ainda, para colocarmos o bolo para assar o forno deve estar ligado. Abaixo temos alguns exemplos de instruções ou comandos que estão faltando.



Se entregássemos a nossa primeira instrução para uma criança de 5 a 7 anos, ela possivelmente teria dificuldades em interpretar esses passos, e possivelmente não conseguiria assar o bolo. O mesmo ocorre com um computador: **Um computador é extremamente eficaz em executar tarefas, porém ele não consegue deduzir e interpretar nada**, ele não é um ser pensante. Ele precisa que sejamos **extremamente claros e precisos na descrição do que ele precisa fazer**. Dessa forma, um algoritmo mais “completo” para assar o bolo poderia ser como o da imagem abaixo:



Observe que agora, além de descrevermos melhor certas atividades, como o processo de untar a forma ou ligar o forno, adicionamos também mais processos de verificação e de repetição de uma situação, como a verificação se a massa está ou não homogênea!

Esses pontos são extremamente importantes para que um computador possa entender corretamente o que fazer! **Todo erro gerado quando um programa é executado é porque não criamos o algoritmo corretamente ou porque não implementamos, escrevemos, esse algoritmo corretamente.** Ou seja, todo o erro que é gerado tem muito mais responsabilidade nossa do que do próprio computador. E por conta disso, **conseguir criar algoritmos é extremamente importante quando desejamos criar programas que solucionem problemas para nós.** E para isso utilizamos a **lógica**, que é intrínseca no nosso dia a dia e em nossa vida para descrever essas ações o mais detalhado possível.

Por tanto, **Lógica de Programação** nada mais é que a **habilidade de pegarmos uma tarefa**, como assar um bolo, **e quebrarmos ela em vários passos pequenos**, que sejam possíveis de um computador executar para que a tarefa maior, assar um bolo, seja executada. E muitas vezes ela pode parecer difícil pois, como vimos anteriormente, a lógica está intimamente ligada a nós, seres humanos, e por conta disso, **conseguimos extrapolar situações que o computador não consegue.**

Porém, **um computador não entende um Diagrama de Blocos.** Diagramas de Blocos são ferramentas muito úteis para entendermos e visualizarmos o fluxo de execução de tarefas. **O computador precisa que utilizemos uma linguagem para que possamos passar essas instruções pra ele.** E essas linguagens que utilizamos para informar ao computador o que fazer e em qual ordem, são conhecidas como **linguagem de programação!**

Mas antes de falarmos sobre linguagens de programação, vamos aprender a transcrever um Diagrama de Blocos em **Pseudocódigo**, que é uma linguagem que utilizamos para descrever os passos de um algoritmo que temos dentro de um Diagrama de Blocos em palavras, ou comandos!. E esse assunto, a transformação de um Diagrama de Blocos para Pseudocódigo, será o assunto da próxima aula!