

**GRUPO  
SEMEAR**

# **Aula 1: Programação**

---

02/10/2021

```
document.getElementById(div).im  
else if (i==2)  
divs[1];  
var atpos=inputs[i].indexOf(  
uts[1])  
var dotpos=inputs[i].lastIn  
if (atpos<1 || dotpos<atp  
(1)  
document.getElementById(  
2)  
else  
document.getElementById(  
inputs[1].lastIndex  
atpos+2 || dotpos  
1)  
document.getElementById("password")  
document.getElementById("password")
```

# 1

## Introdução

# Algoritmo

São sequências finitas de ações executáveis a fim de solucionar um problema.

- Finitos
- Passos bem definidos
- Resolver problema

Entradas  
(inputs)



Sequência de  
Instruções



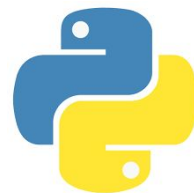
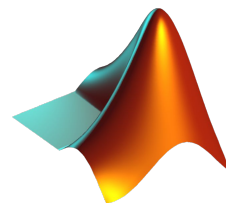
Saídas  
(outputs)

# Linguagem de programação

Algoritmo que através de métodos padronizados – com regras sintáticas e semânticas – pode ser traduzido para informar instruções a um processador

As linguagens podem ser divididas em:

- Grau de abstração
- Paradigma
- Tipagem
- Compilada/Interpretada



# Linguagem de programação

Linguagem  
visual

Linguagem de alto  
nível

Linguagem de montagem

Linguagem de máquina

O grau de abstração de uma linguagem é dividido em:

Baixo nível - mais próximo da linguagem de máquina (instruções diretas)

Alto nível - mais próximo da linguagem humana (instruções abstratas)

# Linguagem de programação

Paradigmas representam o estilo, estrutura em que o código será escrito.

Existem diversos tipos de paradigmas cada um tendo suas vantagens e desvantagens:

- Procedural
- Funcional
- Orientado a Objetos
- Declarativo

As linguagens  
podem ser  
multi-paradigma

# Linguagem de programação

**Dinâmica**

**Estática**

**Sistemas  
de tipo**

Não exige a  
definição do  
tipo de dado

Exige a  
definição do  
tipo de dado

**Forte**

**Fraca**

Não realiza  
conversões de tipos  
de dados  
automaticamente

Realiza conversões  
de tipos de dados  
automaticamente

**Tipagem**

# Linguagem de programação

## COMPILAÇÃO

Processo de conversão do código fonte para a linguagem de máquina para ser executado posteriormente

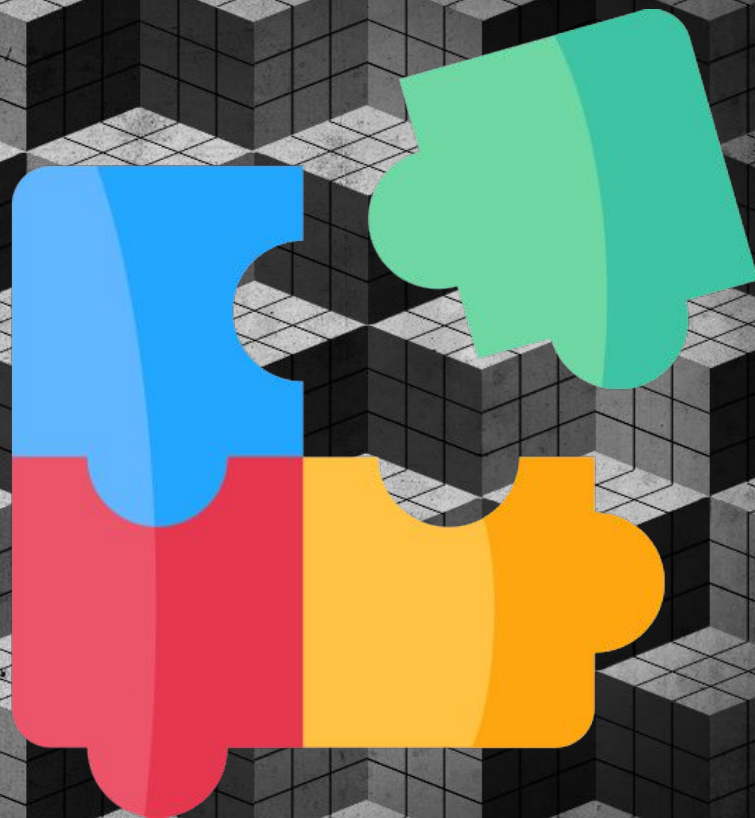
## INTERPRETAÇÃO

Processo em que o código fonte é traduzido e executado simultaneamente após ser interpretado por um interpretador (software)



# 2

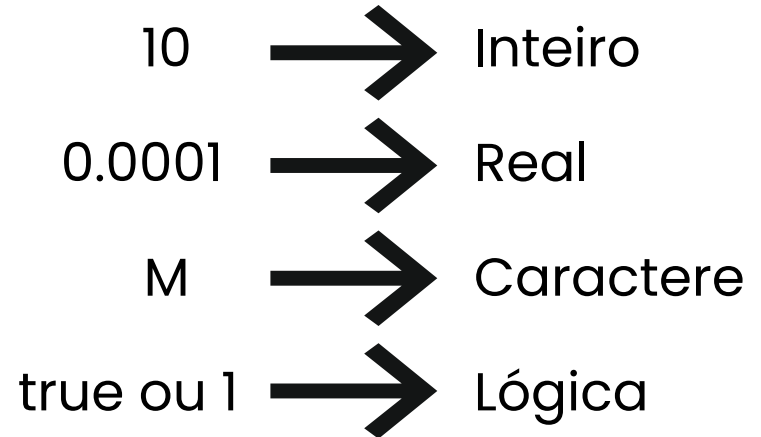
**Lógica de  
programação**



# Variáveis e constantes

As linguagens de programação possuem formas de armazenar dados, sejam eles variáveis ou constantes:

- Inteiro
- Real
- Caractere (texto)
- Lógico



# Variáveis e constantes

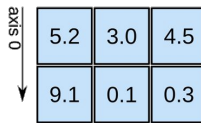
1D array



axis 0

shape: (4,)

2D array

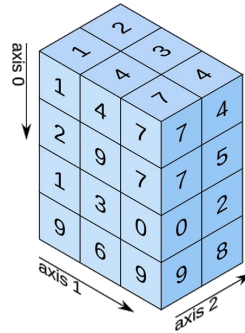


0 sixte

axis 1

shape: (2, 3)

3D array



0 sixte

axis 1

axis 2

shape: (4, 3, 2)

Existem cadeias de dados chamadas arrays, podendo armazenar dados do mesmo tipo em uma única variável/constante.

Esses arrays podem ser multi-dimensionais, sendo muito útil para armazenar dados relacionados.

Ex: Idade de um grupo, Texto, Matriz

# Estrutura de seleção

## If / Else

Utilizado para verificar se determinado parâmetro satisfaz ou não a condição imposta

Utilizado para verificar em qual caso o parâmetro satisfaz a condição imposta

## Switch

# Estrutura de repetição

## **FOR**

Laço de repetição utilizado quando é conhecido o intervalo de repetições a serem realizadas

Laço de repetição utilizado quando é aplicada uma condição para a repetição continuar sendo realizada

## **WHILE**

# Operadores

## Operadores aritméticos:

- + -> Soma
- - -> Subtração
- \* -> Multiplicação
- / -> Divisão
- % -> Módulo

## Operador ternário:

- ?: -> Verifica o argumento e retorna dependendo se for verdadeiro ou falso

## Operadores Lógicos:

- && -> E (And)
- || -> OU (Or)
- ! -> NÃO (Not)

## Operadores de comparação:

- == -> Igual a
- != -> Diferente de
- > -> Maior que
- >= -> Maior ou igual que
- < -> Menor que
- <= -> Menor ou igual que

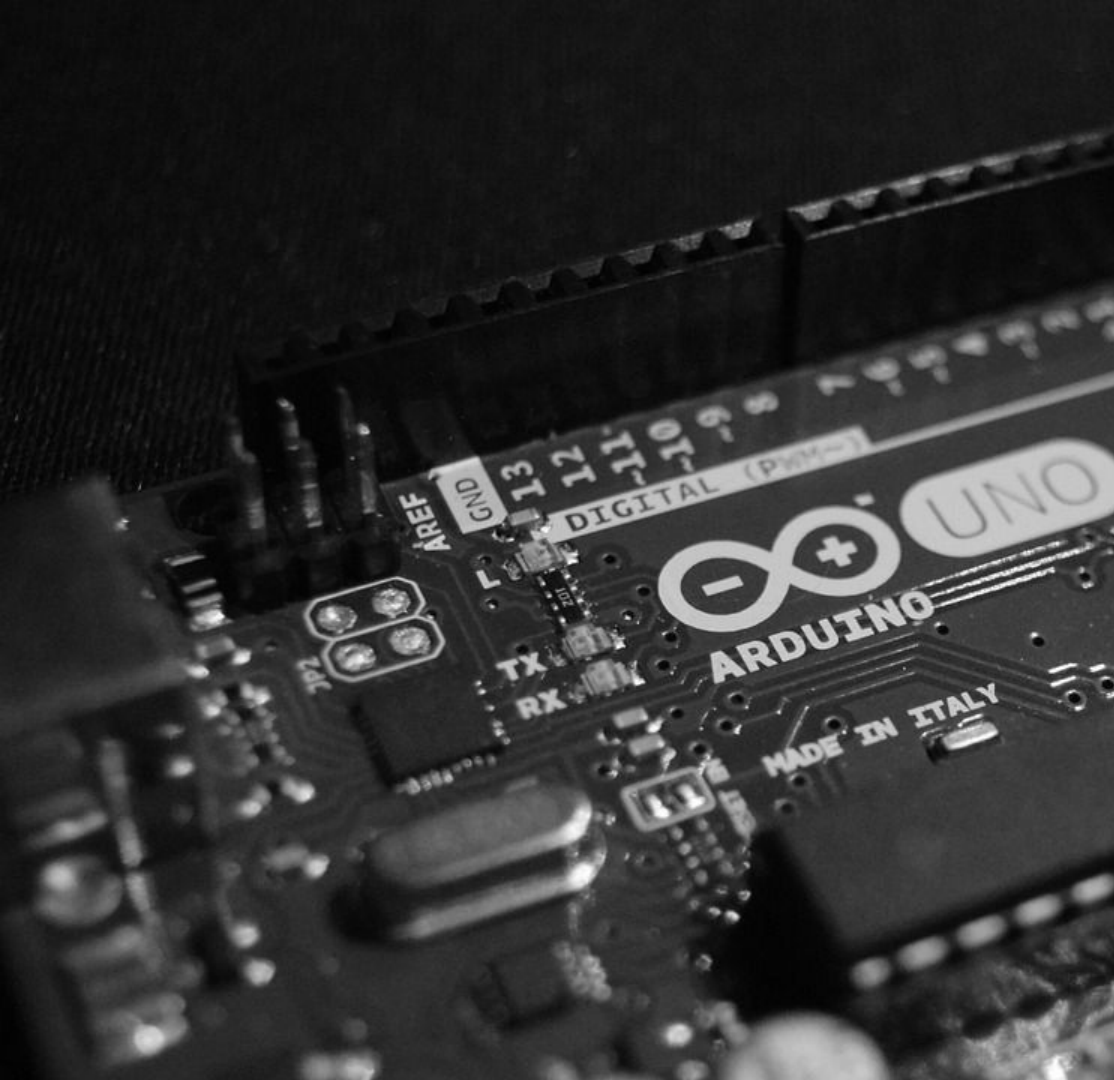
# Framework X Biblioteca

## FRAMEWORK

Fornece uma arquitetura para o desenvolvimento de uma aplicação - integração do seu código a estrutura do framework

## BIBLIOTECA

Fornece funcionalidades prontas para o desenvolvimento de uma aplicação, facilitando sua construção - integração da estrutura da biblioteca ao seu código

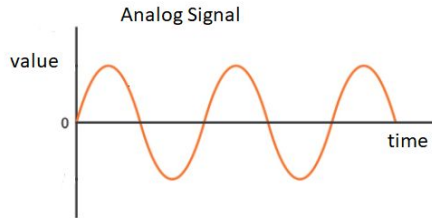


**3**

**Aplicação em  
microcontrolador**

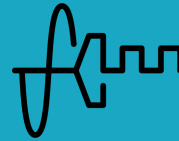


# Sinais



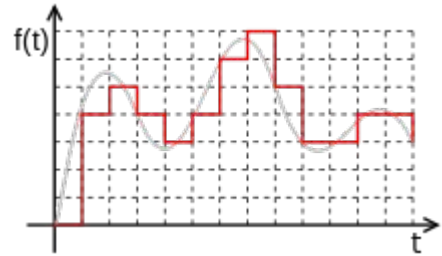
## Analógico

O sinal é transmitido em forma de ondas contínuas que variam ao longo do tempo



## Digital

O sinal é transmitido com valores discretos na amplitude, tornando-o menos complexo



# Arduino

Além dos microcontroladores arduino, existe também o framework Arduino

O qual pode ser utilizado para elaborar a programação dos micros, permitindo utilizar funções para controlar os pinos

Código  
<Arduino>



Compilação



Importação  
para o micro

# Constantes e tempo(Arduino)

## Constantes:

- Input -> Entrada de sinal
- Output -> Saída de sinal
- HIGH -> Acionará tensão > 0
- LOW -> Acionará tensão = 0

## Funções de tempo:

- `millis()`  
Retorna o tempo decorrido desde que o micro foi ligado
- `delay()`  
Paralisa a execução do código por um determinado tempo

# Pinagem (Arduino)

Funções dos pinos:

- `pinMode(pin, mode)`  
Configura o pino como de entrada ou saída -  
input/output
- `digitalWrite(pin, mode)/analogWrite(pin, mode)`  
Aciona uma valor digital/análogo no pino
- `digitalRead(pin)/analogRead(pin)`  
Lê o valor do pino especificado digital/análogo

# Obrigado!

Saiba mais em:



SEMEAR - EESC/USP



@semear.usp



Grupo SEMEAR - EESC USP



Equipe Atena EESC-USP

[www.semear.eesc.usp.br](http://www.semear.eesc.usp.br)

