Semana 0: Pensamento Computacional

Fundamentos da Computação - 2023/1

Prof. Abrantes Araújo Silva Filho

Você sabe por que está aqui?

Projetar e implementar soluções para **resolver problemas**!

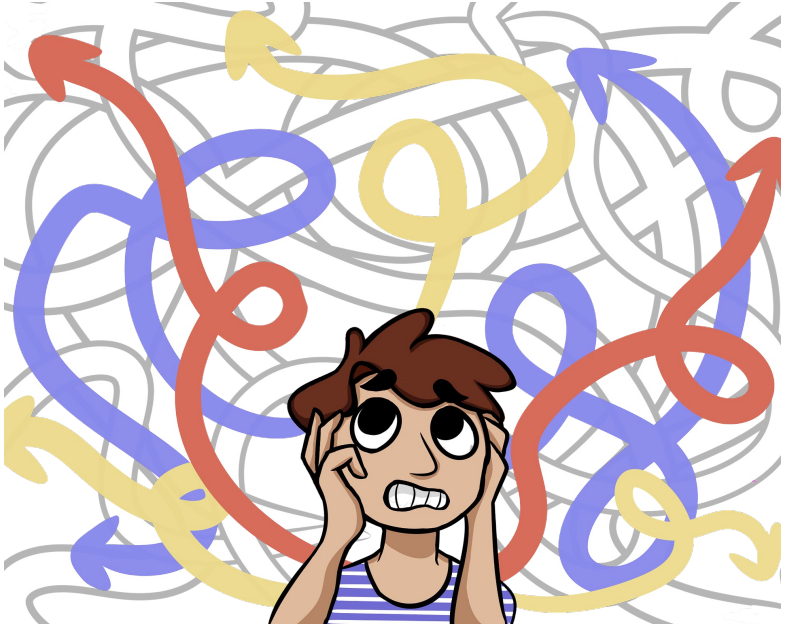
Impacto gigante na sociedade, em todas as áreas!

Beleza matemática e de idéias.

Computação não é sobre computadores ou programação, é sobre **como resolver problemas** importantes para nós.

Esta disciplina: **conceitos fundamentais** + **grandes idéias da computação** + **grandes idéias da programação** + **visão geral** + **visão futura**:

- Pensamento computacional
- Abstração, algoritmos, recursão
- Estruturas de dados, funções como dados
- Representação de dados
- Bancos de dados
- Programação web
- Idéias fundamentais da computação
- …



“Drink from a Fire Hose”

“**Getting an Education from MIT is like drink from a Fire Hose**”.

Jorome Weisner,
Former MIT President
(71-80)

Estudantes do MIT, em 1991, transformaram um hidrante em um bebedor.
http://hacks.mit.edu/Hacks/by_year/1991/fire_hydrant/

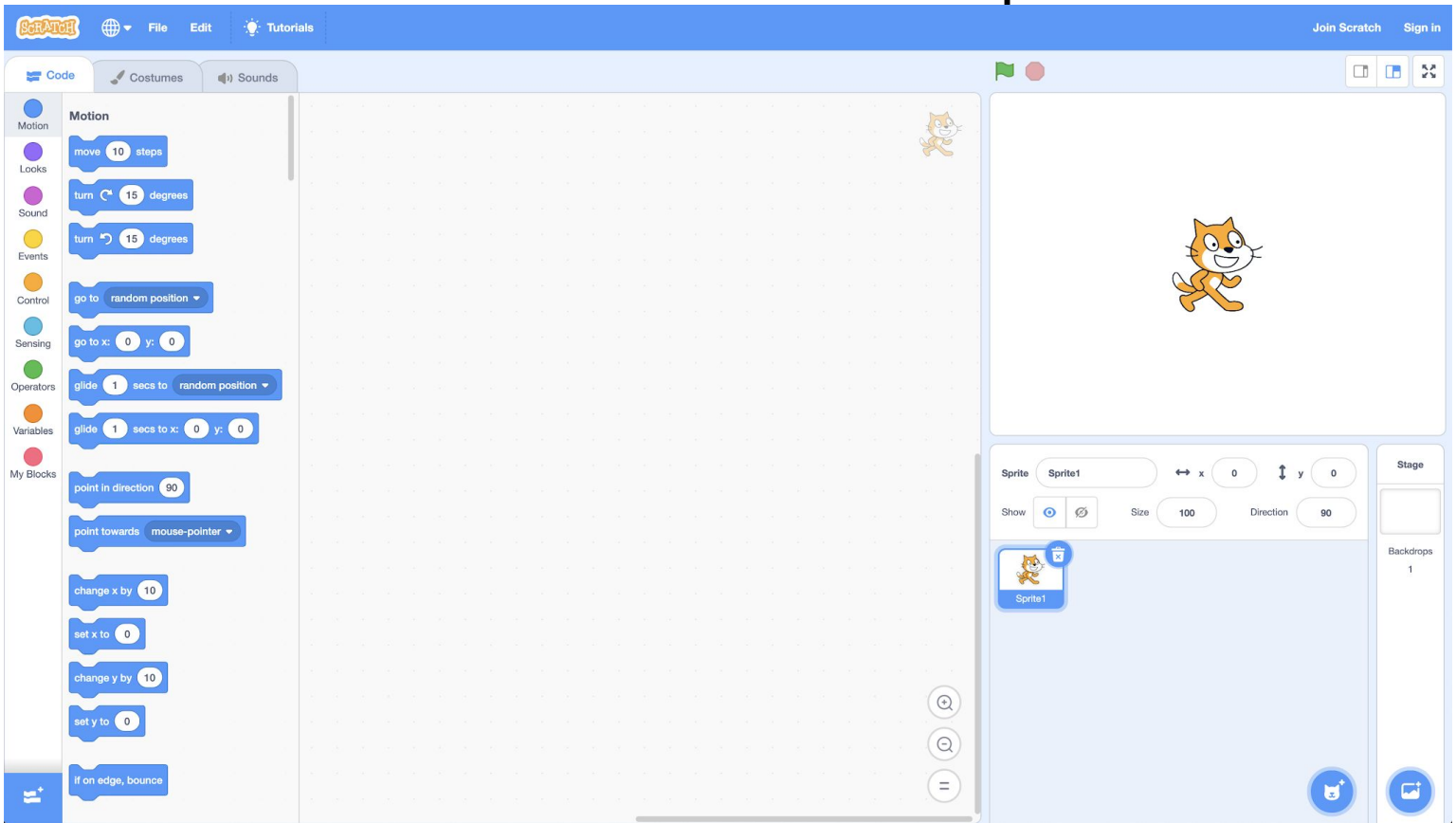
O que é necessário?

- **Coragem**: vença seu medo!
- **Resiliência**: não desanime com os fracassos iniciais!
- **Estudo**: fundamental!
- **Sair da zona de conforto**: quanto maior o desafio, maior a aprendizagem!

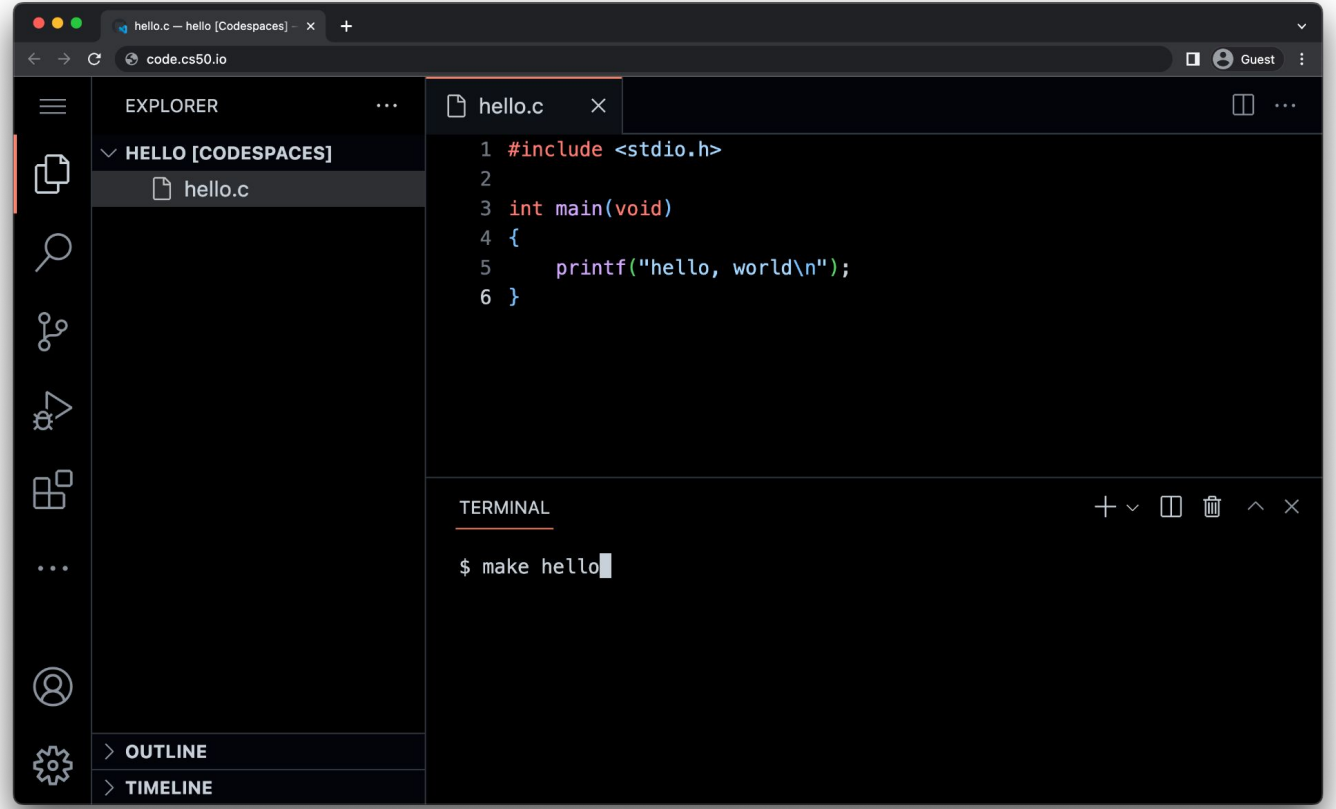
⅔ dos alunos NUNCA FIZERAM UM CURSO DE COMPUTAÇÃO ANTES. E, mesmo os que já estudaram computação, irão se beneficiar: preencheremos diversas lacunas no conhecimento! Ao final, todos estarão “na mesma página”.

Ao final do curso, não importa tanto onde você está em relação aos seus colegas, mas **onde você chegou em relação a onde estava no começo**!

O que veremos?
Semana 0: Pensamento Computacional



O que veremos?
Semanas 1 e 2: C e Arrays



O que veremos?
Semanas 1 e 2: C e Arrays

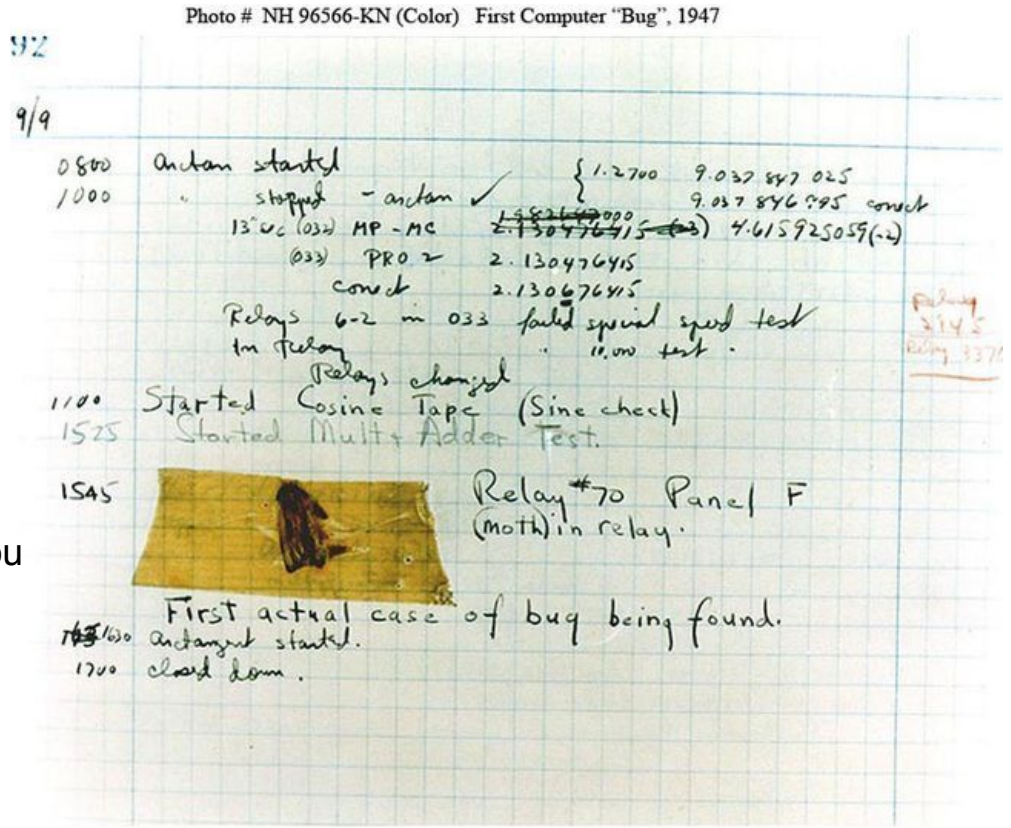


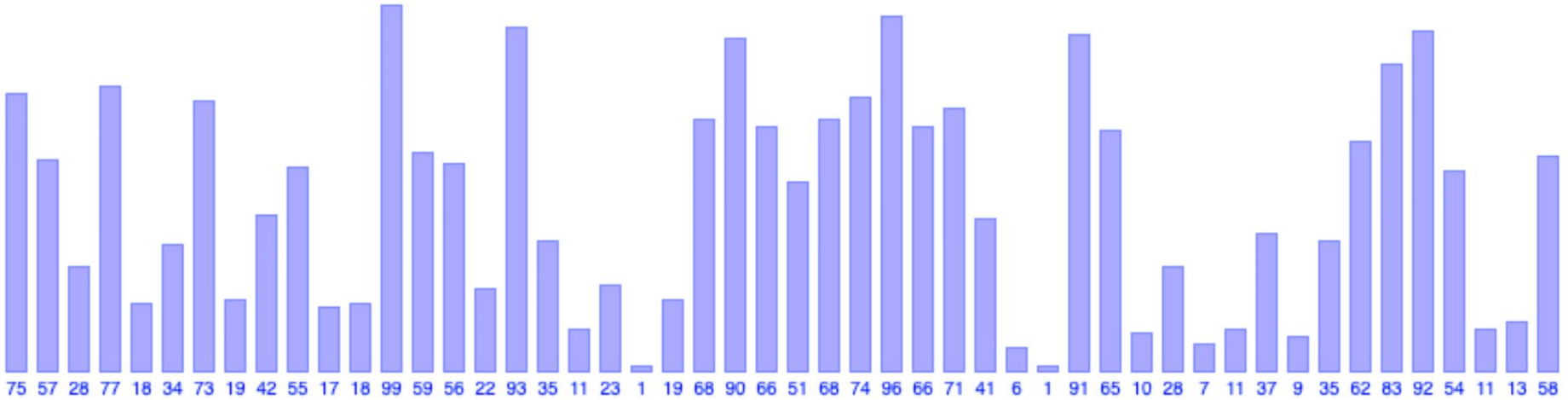
Photo # NH 96566-KN (Color) First Computer "Bug", 1947

9/9

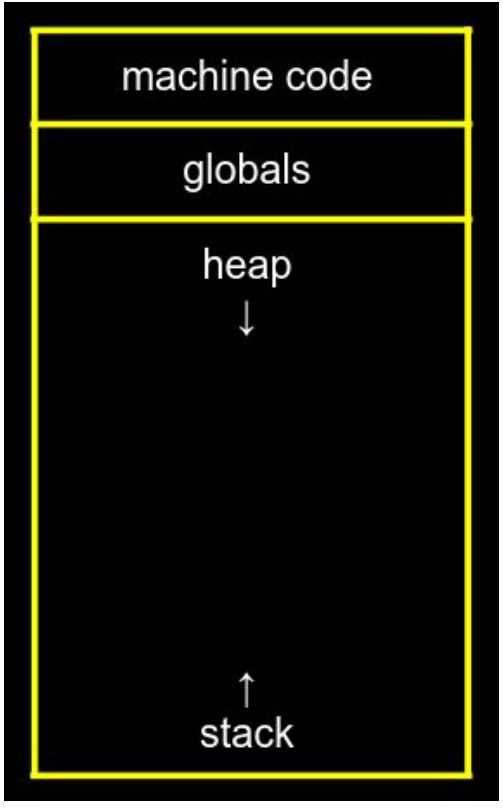
0800 Arctan started
1000 " stopped - arctan ✓
13" vc (033) MP - MC
(033) PRO 2
correct
Relays 6-2 in 033 failed special speed test
In relay
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 closed down.

09/09/1947, 15:45h, Grace Murray Hopper registrou o “Primeiro Bug em um Computador”, o Harvard Mark II. O “bug” foi encontrado por outras pessoas de sua equipe, mas Hopper fez o registro em seu Engineering Notebook.
https://www.computerhistory.org/tdih/september/9/

O que veremos?
Semana 3: Algoritmos

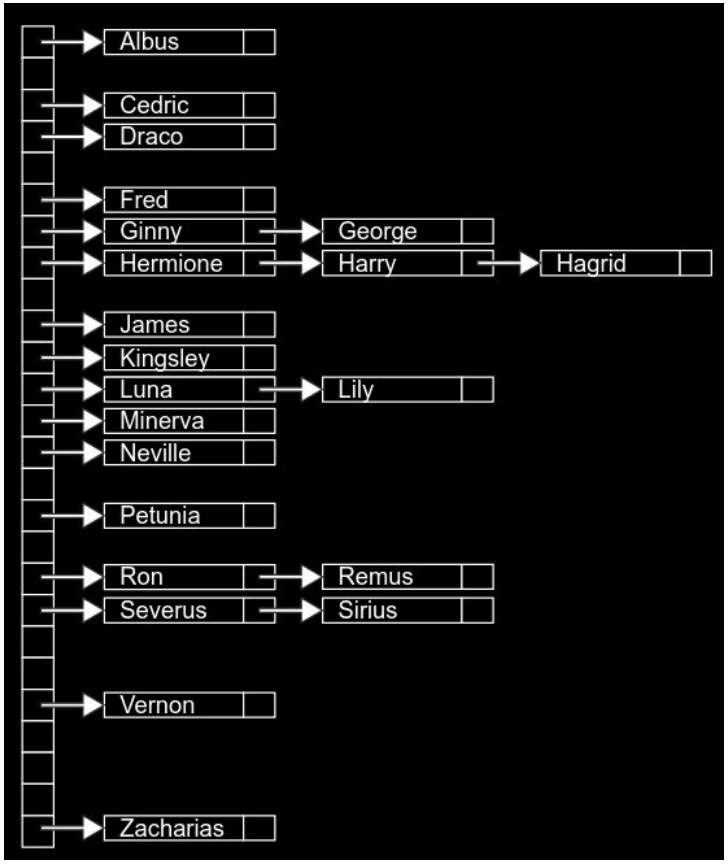


O que veremos?
Semana 4: Memória



stackoverflow

O que veremos?
Semana 5: Estruturas de Dados



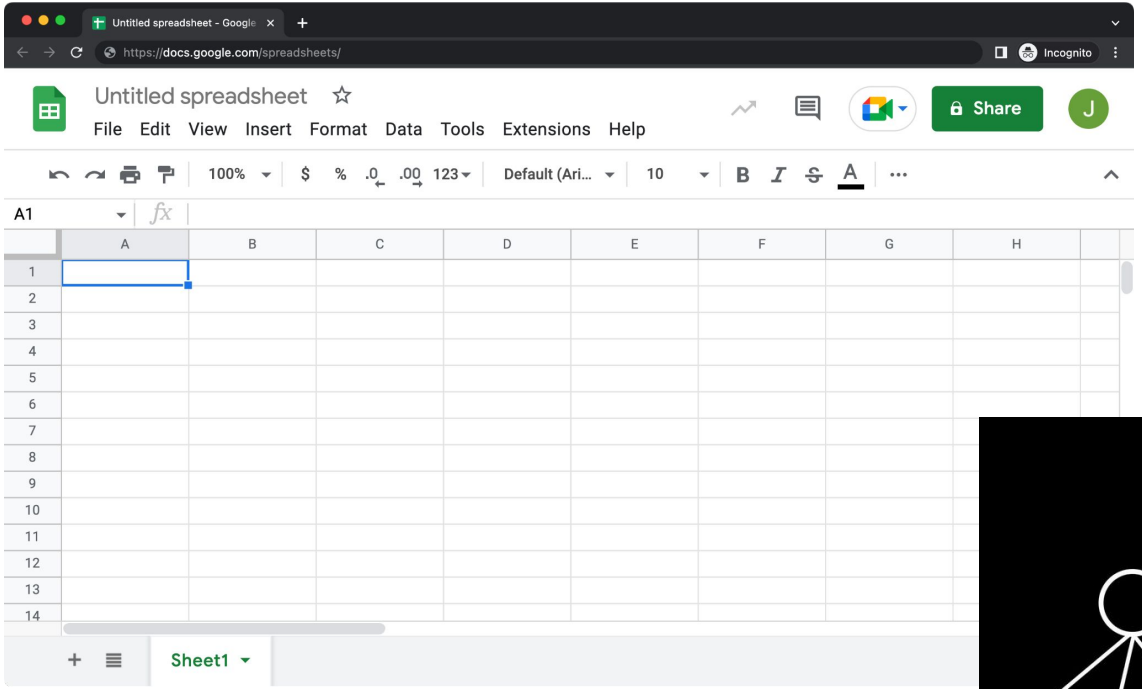
O que veremos?
Semana 6: Python

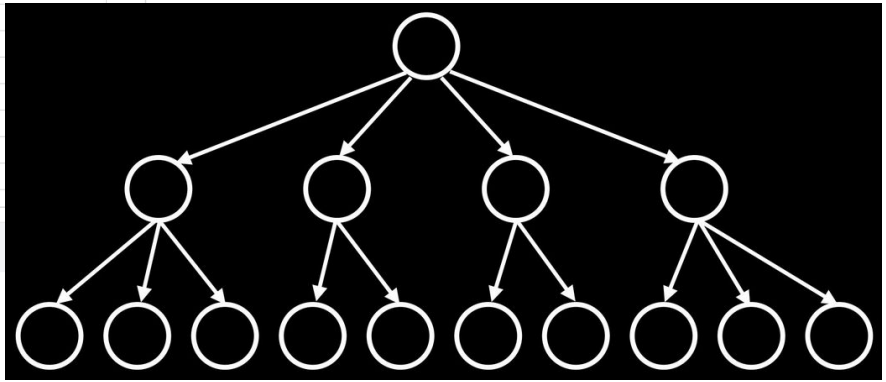
```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```python
print("hello, world")
```

O que veremos?
Semana 7: SQL e Bancos de Dados

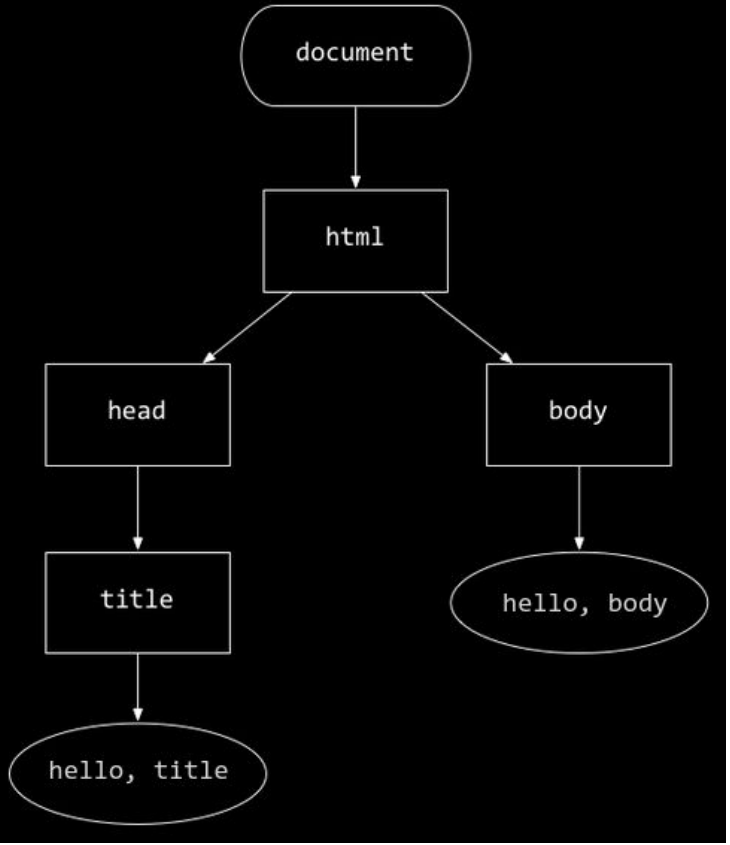




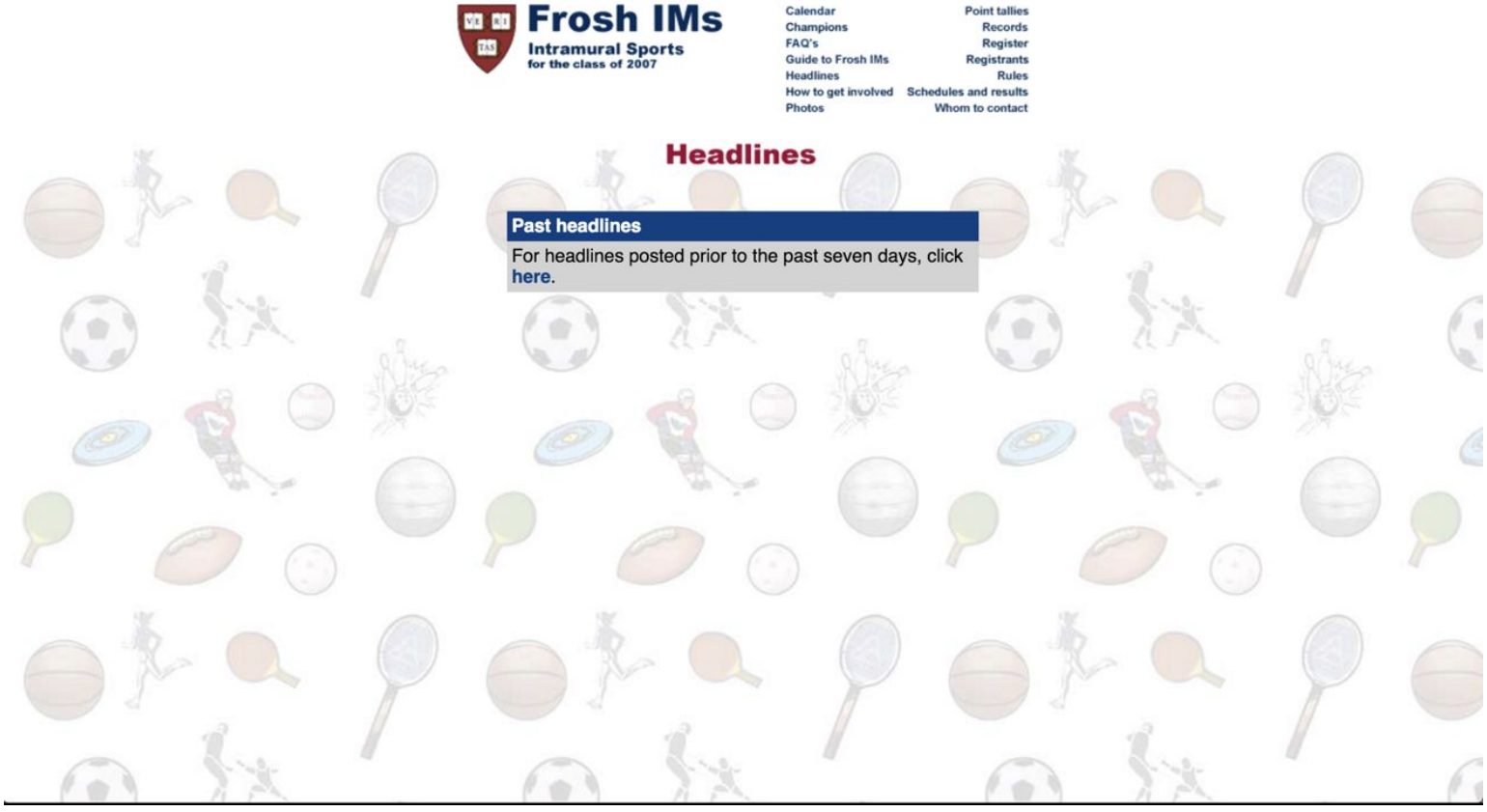
O que veremos?
Semana 8: HTML, CSS, JS

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>
            hello, title
        </title>
    </head>
    <body>
        hello, body
    </body>
</html>
```



O que veremos?
Semana 9: Flask Web Applications



O que veremos?
Semana 10: Juntando tudo e Projeto Final

O que é **computação**?

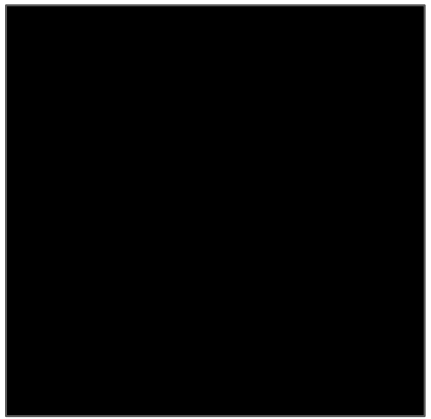
O que é computação?

$$\sqrt{33} = ?$$

O que é computação?

input
(problema)

33



output
(solução)

$5.74456\ldots$

$\sqrt{33} = ?$

Conhecimento DECLARATIVO:

$$\sqrt{x} = y \mid y \geq 0 \wedge y^2 = x$$

Nos diz **<u>O QUE É</u>** alguma coisa, define algo.

"Não ajuda" a resolver o problema, pelo menos de forma direta…

O que é computação?

$$\sqrt{33} = ?$$

Método de Newton:

1. Chute um valor para y;
2. Calcule o valor de y^2;
3. Se $y^2 = x$ (ou um valor próximo o suficiente), você achou a raiz. Termine o procedimento.
4. Se $y^2 \neq x$ (ou não está próximo o suficiente), melhore a estimativa de y fazendo o seguinte cálculo: novo $y = \dfrac{y + \frac{x}{y}}{2}$;
5. Retorne ao segundo passo até que você encontre a raiz ou uma aproximação suficiente.

$$\sqrt{x} = y \mid y \geq 0 \wedge y^2 = x$$

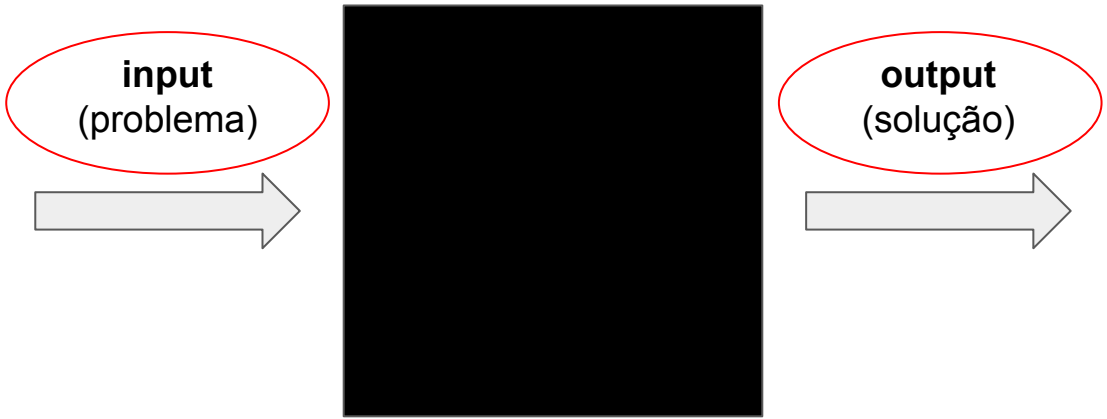
Conhecimento IMPERATIVO:

Nos diz **COMO** resolver um problema.

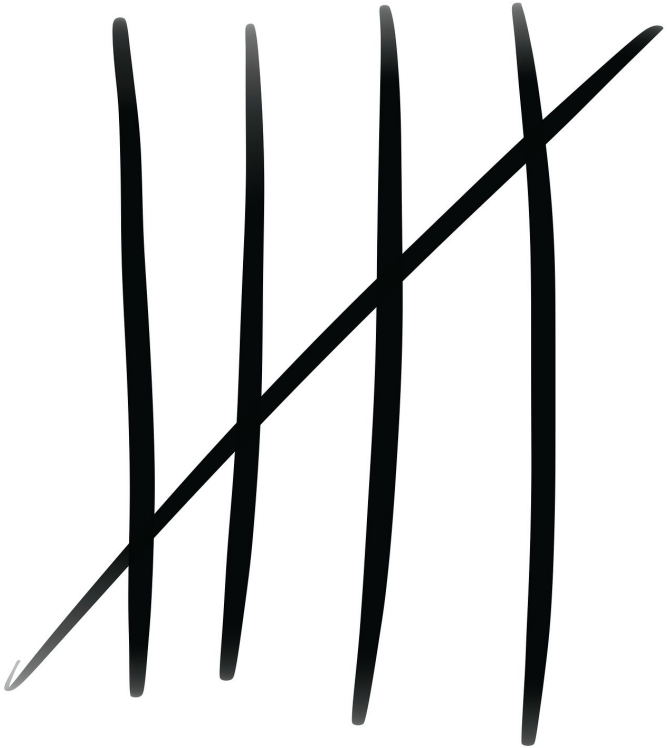
A computação está interessada neste tipo de conhecimento, em **COMO PROJETAR E IMPLEMENTAR SOLUÇÕES PARA PROBLEMAS**!

Como **REPRESENTAR** um problema?
(como representar dados e informações?)

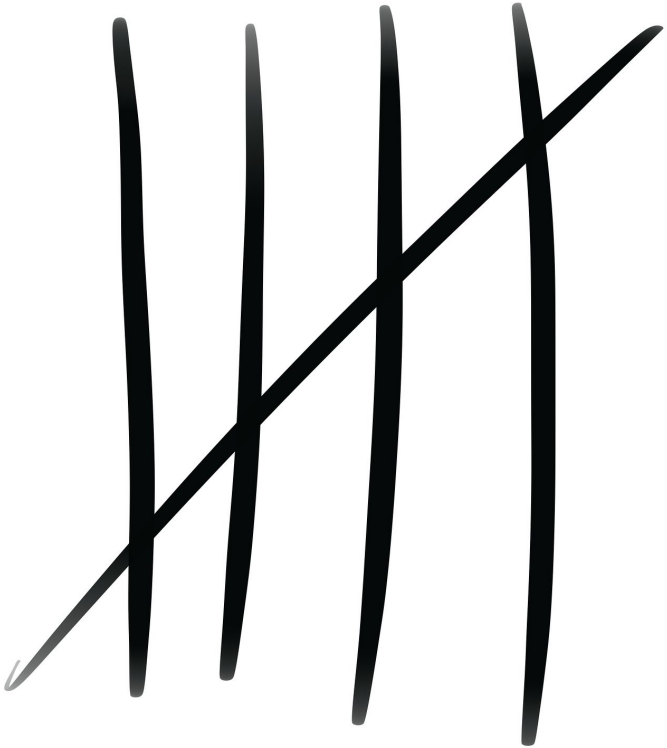
Como **REPRESENTAR** um problema? (como representar dados e informações?)



Um problema simples: quantos alunos estão aqui? Como representar a contagem de alunos?



Sistema unário! Temos um algarismo: |
É possível fazer melhor?



Mudando a representação do problema, podemos fazer melhor! Quantos alunos posso contar…

… com 1 mão?

… com 2 mãos?

Sistema binário! Temos dois algarismos: 0 e 1

do 0 ao 31

do 0 ao 1023

Sistema binário! Temos dois algarismos: 0 e 1

0

Sistema binário! Temos dois algarismos: 0 e 1

1

Sistema binário! Temos dois algarismos: 0 e 1

2

Sistema binário! Temos dois algarismos: 0 e 1

3

Sistema binário! Temos dois algarismos: 0 e 1

4

Sistema binário! Temos dois algarismos: 0 e 1

5

Sistema binário! Temos dois algarismos: 0 e 1

6

Sistema binário! Temos dois algarismos: 0 e 1

7

Sistema binário! Temos dois algarismos: 0 e 1

8

Sistema binário! Temos dois algarismos: 0 e 1

9

Sistema binário! Temos dois algarismos: 0 e 1

10

Sistema binário! Temos dois algarismos: 0 e 1

11

Sistema binário! Temos dois algarismos: 0 e 1

12

Sistema binário! Temos dois algarismos: 0 e 1

13

Sistema binário! Temos dois algarismos: 0 e 1

14

Sistema binário! Temos dois algarismos: 0 e 1

15

Sistema binário! Temos dois algarismos: 0 e 1

16

Sistema binário! Temos dois algarismos: 0 e 1

17

Sistema binário! Temos dois algarismos: 0 e 1

18

Sistema binário! Temos dois algarismos: 0 e 1

19

Sistema binário! Temos dois algarismos: 0 e 1

20

Sistema binário! Temos dois algarismos: 0 e 1

21

Sistema binário! Temos dois algarismos: 0 e 1

22

Sistema binário! Temos dois algarismos: 0 e 1

23

Sistema binário! Temos dois algarismos: 0 e 1

24

Sistema binário! Temos dois algarismos: 0 e 1

25

Sistema binário! Temos dois algarismos: 0 e 1

26

Sistema binário! Temos dois algarismos: 0 e 1

27

Sistema binário! Temos dois algarismos: 0 e 1

28

Sistema binário! Temos dois algarismos: 0 e 1

29

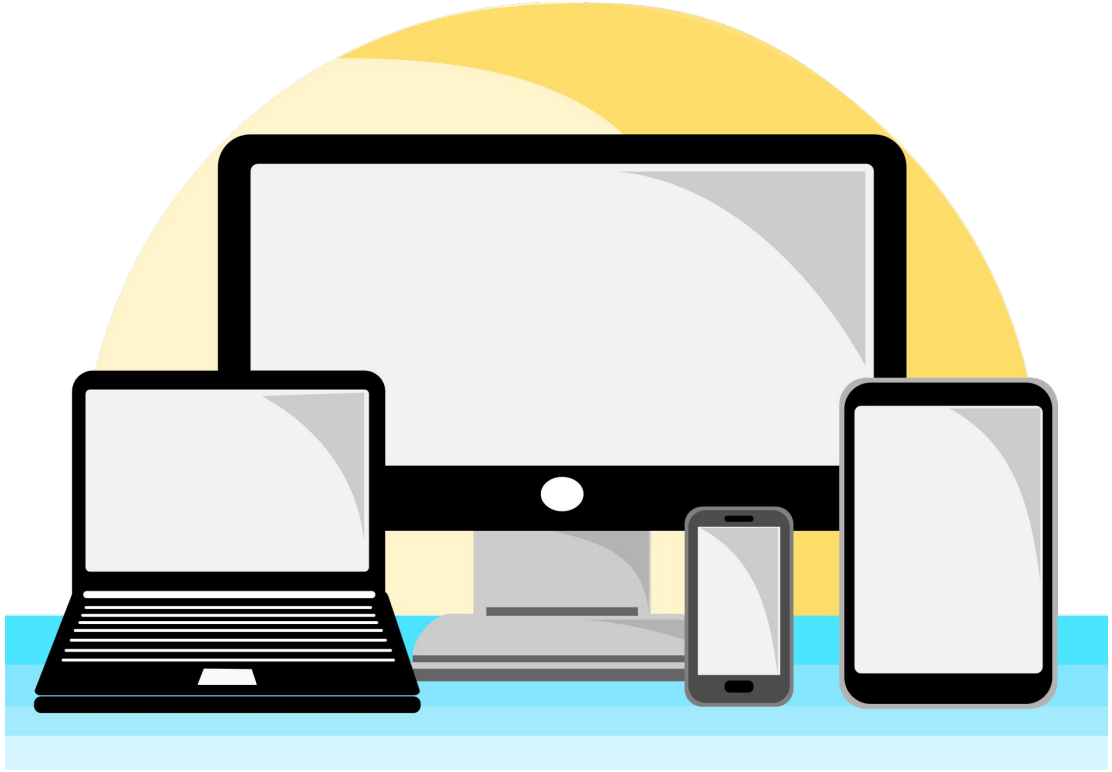
Sistema binário! Temos dois algarismos: 0 e 1

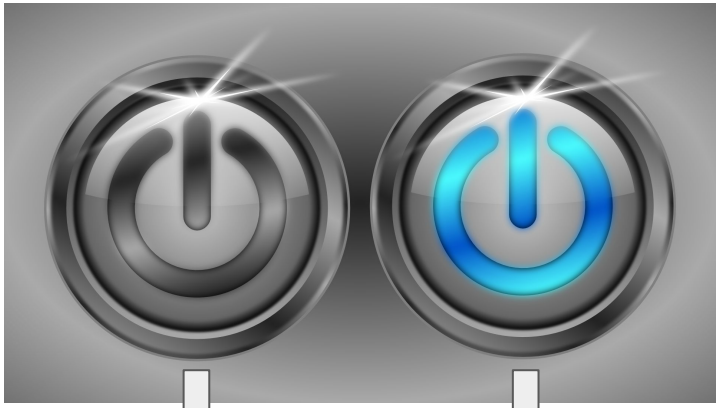
30

Sistema binário! Temos dois algarismos: 0 e 1

31

E qual a relação da representação binária com os computadores?

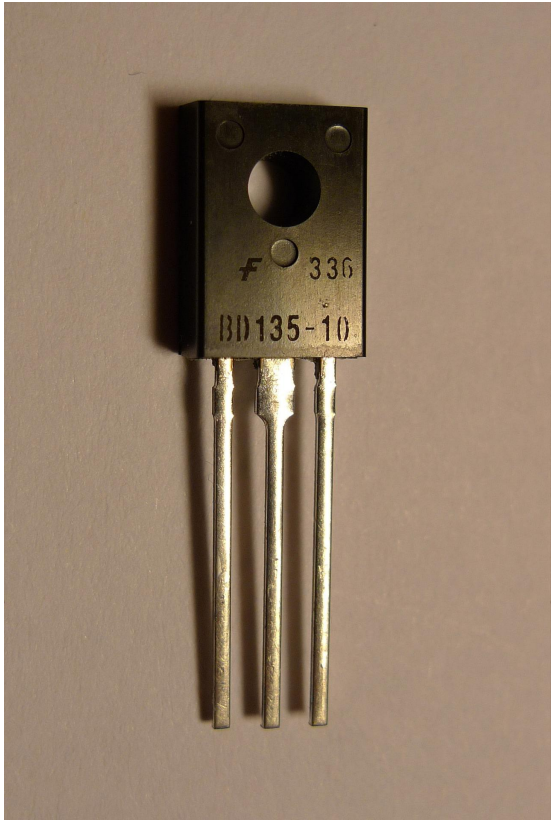




sem energia
0

com energia
1

E qual a relação da representação binária com os computadores?

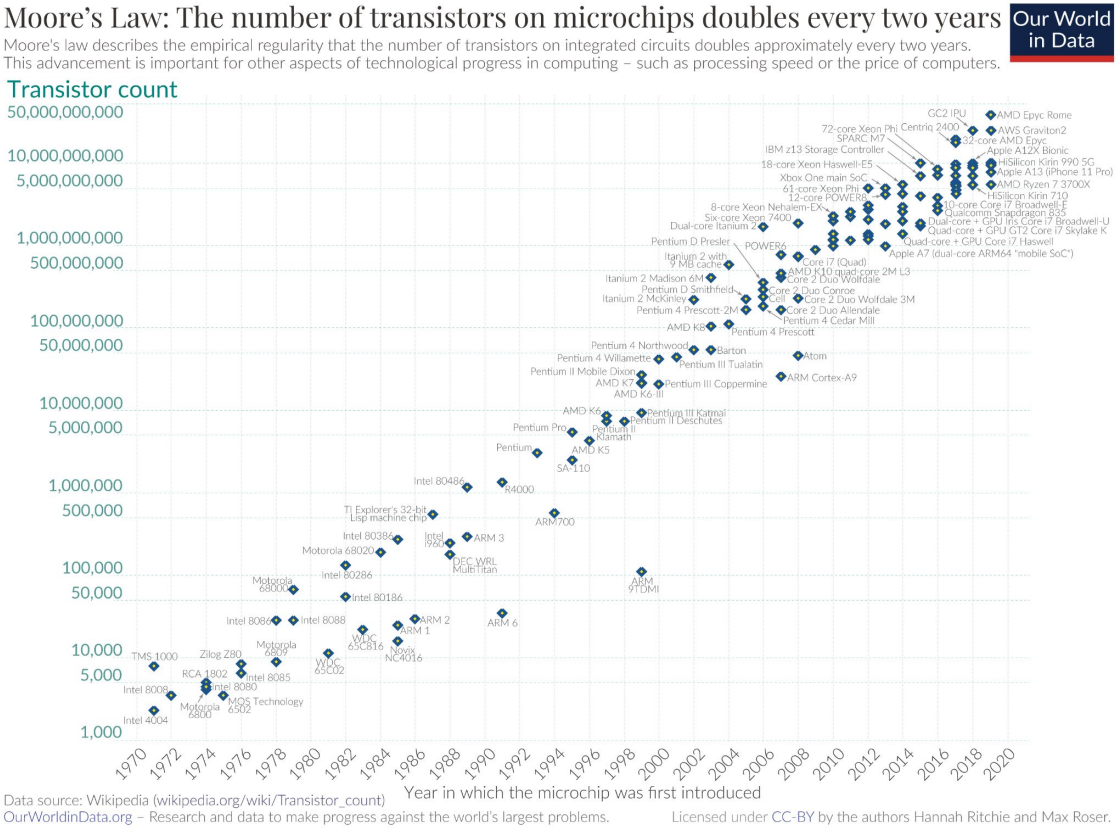


O componente eletrônico responsável pelos “zeros” e “uns” é o TRANSÍSTOR.

- 0: quando ele interrompe a passagem de energia
- 1: quando ele permite a passagem de energia

Quantos transistores existem em uma CPU?

E qual a relação da representação binária com os computadores?



Como o sistema binário funciona para representar números?

Agora você vai descobrir que quando seu professor de matemática da 5ª série falava que uma coisa era importante e seria usada no futuro, ele não estava mentindo…

Representação da informação:

- Base do sistema numérico (quantidade de algarismos)
- Valor posicional
- Decomposição numérica

O sistema decimal!

Representação de números:

- Base do sistema numérico: 10
- Valor posicional de um algarismo: $n_i \times 10^i$
  n = algarismo
  i = posição (**contada da direita para esquerda, iniciando em 0**)
- Decomposição numérica:

$$(n_i \times 10^i) + (n_{i-1} \times 10^{i-1}) + (n_{i-2} \times 10^{i-2}) + \ldots + (n_0 \times 10^0)$$

O sistema decimal!

$$(n_i \times 10^i) + (n_{i-1} \times 10^{i-1}) + (n_{i-2} \times 10^{i-2}) + \ldots + (n_0 \times 10^0)$$

$$\begin{aligned} 364 &= (3 \times 10^2) + (6 \times 10^1) + (4 \times 10^0) \\ &= 300 + 60 + 4 \end{aligned}$$

O sistema decimal!

Quadro de Valor Posicional

Unidades de Trilhões			Unidades de Bilhões			Unidades de Milhões			Unidades de Milhar			Unidades Simples			← Classes
15ª	14ª	13ª	12ª	11ª	10ª	9ª	8ª	7ª	6ª	5ª	4ª	3ª	2ª	1ª	← Ordens
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	← Posição
Centenas de trilhões	Dezenas de trilhões	Unidades de trilhões	Centenas de bilhões	Dezenas de bilhões	Unidades de bilhões	Centenas de milhões	Dezenas de milhões	Unidades de milhões	Centenas de milhar	Dezenas de milhar	Unidades de milhar	Centenas	Dezenas	Unidades	
								1	4	8	3	0	0	7	

(1 x 10^6) + (4 x 10^5) + (8 x 10^4) + (3 x 10^3) + (0 x 10^2) + (0 x 10^1) + (7 x 10^0)
1.000.000 + 400.000 + 80.000 + 3.000 + 0 + 0 + 7
1.483.007

O sistema binário!

Representação de números:

- Base do sistema numérico: 2

- Valor posicional de um algarismo: $n_i \times 2^i$
  n = algarismo
  i = posição (**contada da direita para esquerda, iniciando em 0**)

- Decomposição numérica:

$$(n_i \times 2^i) + (n_{i-1} \times 2^{i-1}) + (n_{i-2} \times 2^{i-2}) + \ldots + (n_0 \times 2^0)$$

O sistema binário!

$$(n_i \times 2^i) + (n_{i-1} \times 2^{i-1}) + (n_{i-2} \times 2^{i-2}) + \ldots + (n_0 \times 2^0)$$

$$\begin{aligned} 10011 &= (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) \\ &= 16 + 0 + 0 + 2 + 1 \\ &= 19 \end{aligned}$$

O sistema binário!

Quadro de Valor Posicional

2								1								← Bytes
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bits
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	← Posição
(32768) Trinta e dois mil setecentos e sessenta e oito	(16384) Dezesseis mil trezentos e oitenta e quatro	(8192) Oito mil dento e noventa e dois	(4096) Quatro mil e noventa e seis	(2048) Dois mil e quarenta e oito	(1024) Mil e vinte e quatro	(512) Quinhentos e doze	(256) Duzentos e cinqüenta e seis	(128) Cento e vinte e oito	(64) Sessenta e quatro	(32) Trinta e dois	(16) Dezesseis	(8) Oito	(4) Quatro	(2) Dois	(1) Um	
						1	0	0	0	0	1	0	1	0	1	

(1 x 2^9) + (0 x 2^8) + (0 x 2^7) + (0 x 2^6) + (0 x 2^5) + (1 x 2^4) + (0 x 2^3) + (1 x 2^2) + (0 x 2^1) + (1 x 2^0)
512 + 0 + 0 + 0 + 0 + 16 + 0 + 4 + 0 + 1
533

O sistema octal!

Representação de números:

- Base do sistema numérico: 8

- Valor posicional de um algarismo: $n_i \times 8^i$
  n = algarismo
  i = posição (**contada da direita para esquerda, iniciando em 0**)

- Decomposição numérica:

$$(n_i \times 8^i) + (n_{i-1} \times 8^{i-1}) + (n_{i-2} \times 8^{i-2}) + \ldots + (n_0 \times 8^0)$$

O sistema octal!

$$(n_i \times 8^i) + (n_{i-1} \times 8^{i-1}) + (n_{i-2} \times 8^{i-2}) + \ldots + (n_0 \times 8^0)$$

$$\begin{aligned} 17034 &= (1 \times 8^4) + (7 \times 8^3) + (0 \times 8^2) + (3 \times 8^1) + (4 \times 8^0) \\ &= 4096 + 3584 + 0 + 24 + 4 \\ &= 7708 \end{aligned}$$

O sistema octal!

Quadro de Valor Posicional

2								1								← Bytes
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bits
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	← Posição
(35.184.372.088.832) Trinta e cinco trilhões cento ...	(4.398.046.511.104) Quatro trilhões trezentos e ...	(549.755.813.888) Quinhentos e quarenta e nove ...	(68.719.476.736) Sessenta e oito bilhões setecentos ...	(8.589.934.592) Oito bilhões quinhentos e oitenta...	(1.073.741.824) Um bilhão setenta e três milhões...	(134.217.728) Cento e trinta e quatro milhões ...	(16.777.216) Dezesseis milhões setecentos e setenta...	(2.097.152) Dois milhões noventa e sete mil cento ...	(262.144) Duzentos e sessenta e dois mil cento e ...	(32.768) Trinta e dois mil setecentos e sessenta e ...	(4.096) Quatro mil e noventa e seis	(512) Quinhentos e doze	(64) Sessenta e quatro	(8) Oito	(1) Um	
							5	7	0	0	3	4	1	5	3	

(5 x 8^8) + (7 x 8^7) + (0 x 8^6) + (0 x 8^5) + (3 x 8^4) + (4 x 8^3) + (1 x 8^2) + (5 x 8^1) + (3 x 8^0)
86.889.080 + 14.680.064 + 0 + 0 + 12.288 + 2.048 + 64 + 40 + 3
98.580.587

O sistema hexadecimal!

Hexadecimal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Representação de números:

- Base do sistema numérico: 16
- Valor posicional de um algarismo: $n_i \times 16^i$
  n = algarismo
  i = posição (**contada da direita para esquerda, iniciando em 0**)
- Decomposição numérica:

$$(n_i \times 16^i) + (n_{i-1} \times 16^{i-1}) + (n_{i-2} \times 16^{i-2}) + \ldots + (n_0 \times 16^0)$$

O sistema hexadecimal!

$$(n_i \times 16^i) + (n_{i-1} \times 16^{i-1}) + (n_{i-2} \times 16^{i-2}) + \ldots + (n_0 \times 16^0)$$

$$\begin{aligned} \text{F90B4} &= (15 \times 16^4) + (9 \times 16^3) + (0 \times 16^2) + (11 \times 16^1) + (4 \times 16^0) \\ &= 983040 + 36864 + 0 + 176 + 4 \\ &= 1020084 \end{aligned}$$

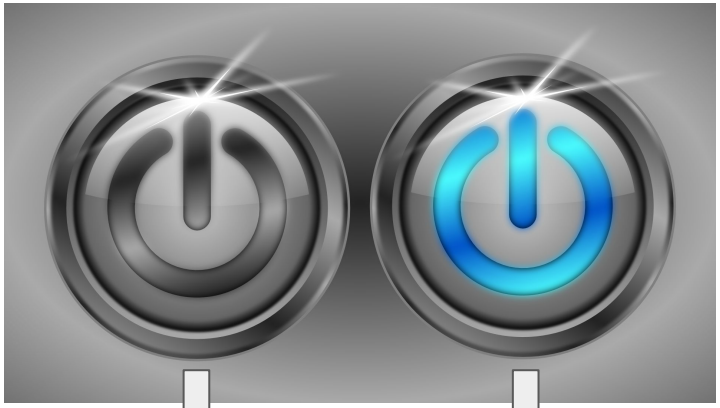
O sistema hexadecimal!

Quadro de Valor Posicional

2								1								← Bytes
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bits
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	← Posição
(1.152.921.504.606.850.000) Um quintilhão, cento e...	(72.057.594.037.927.900) Setenta e dois quatrilhões...	(4.503.599.627.370.496) Quatro quatrilhões quinhentos...	(281.474.976.710.656) Duzentos e oitenta e um trilhões...	(17.592.186.044.416) Dezessete trilhões quinhentos ...	(1.099.511.627.776) Um trilhão noventa e nove bilhões...	(68.719.476.736) Sessenta e oito bilhões setecentos ...	(4.294.967.296) Quatro bilhões duzentos e noventa...	(268.435.456) Duzentos e sessenta e oito milhões ...	(16.777.216) Dezesseis milhões cetecentos e setenta ...	(1.048.576) Um milhão quarenta e oito mil quinhentos...	(65.536) Sessenta e cinco mil quinhentos e trinta ...	(4.096) Quatro mil e noventa e seis	(256) Duzentos e cinqüenta e seis	(16) Dezesseis	(1) Um	
							1	B	0	0	0	A	9	E		

(1 x 16^8) + (11 x 16^7) + (0 x 16^6) + (0 x 16^5) + (0 x 16^4) + (0 x 16^) + (10 x 16^2) + (9 x 16 ^1) + (14 x 16^0)
4.294.967.296 + 2.952.790.016 + 0 + 0 + 0 + 0 + 2.560 + 144 + 14
7.247.760.030

O computador não entende 0 e 1, entende apenas se tem ou não energia! Nós é que “mapeamos” a falta ou presença de energia para números binários 0 ou 1.



sem energia
0

com energia
1

E como representar um problema de “texto”? Como representar letras se tudo o que o computador entende é eletricidade que mapeamos para os números 0 e 1?

Oi!

E como representar um problema de “texto”? Como representar letras?

0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

ASCII: American Standard Code for Information Interchange

E como representar um problema de “texto”?
Como representar letras?

Oi!

79	**105**	**33**
1001111	**1101001**	**100001**

Se a representação é a mesma (binário), como os computadores diferenciam letras de números?

Contexto!

ABSTRAÇÃO

Abstração: **focar no essencial**, **esconder os detalhes**!

- Complexidade x Abstração:
  - Programar é muito fácil se o programa é pequeno e pouco complexo
  - Complexidade é nosso “inimigo”
  - Abstração é a ferramenta para vencer a complexidade

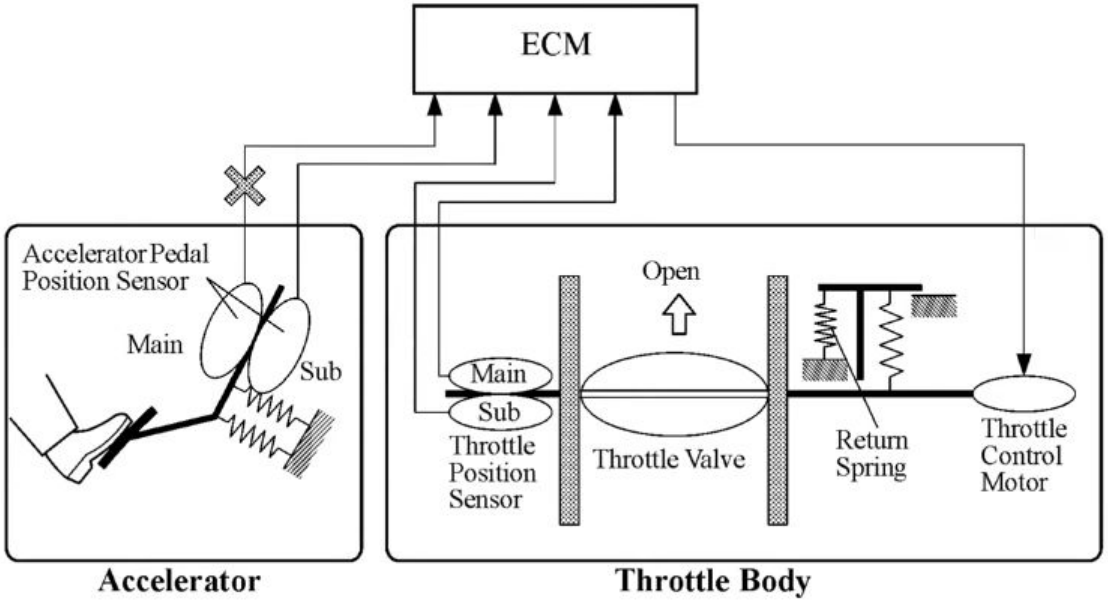
- Aprender a escolher e justificar o uso da ferramenta mais apropriada de abstração é fundamental na computação

Abstração: **focar no essencial**, **esconder os detalhes**!

- **Esconder os detalhes para os usuários**:
  - Qual a abstração para dirigirmos um carro?
  - Essa abstração nos fornece qual interface?
    - Chave, volante, acelerador, freio, embreagem
  - Mesmo que a tecnologia “dentro” da abstração mude, a abstração em si (e a interface) não muda:
    - Injeção eletrônica, carro elétrico
    - ABS
  - Colocar mais controles poderia ser ruim:
    - Alguém quer um controle para o ABS?

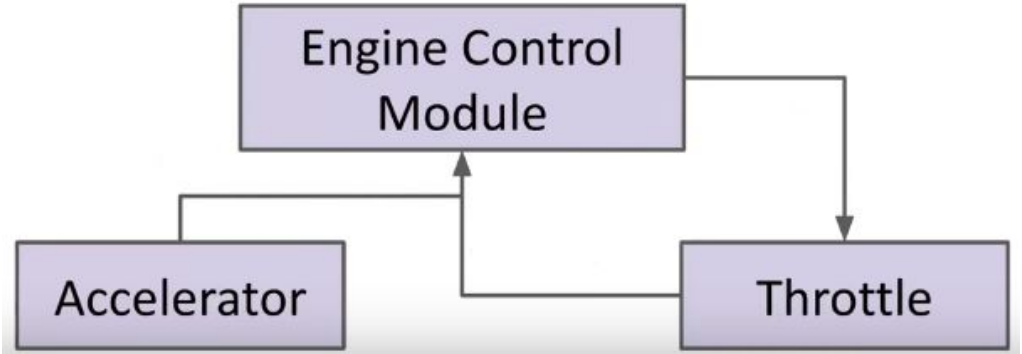
Abstração: **focar no essencial**, **esconder os detalhes**!

- **Esconder os detalhes para outros projetistas**:
  - O projetista do “Engine Control Module” (ECM) não precisa entender como a “Return Spring” funciona



Abstração: **focar no essencial**, **esconder os detalhes**!

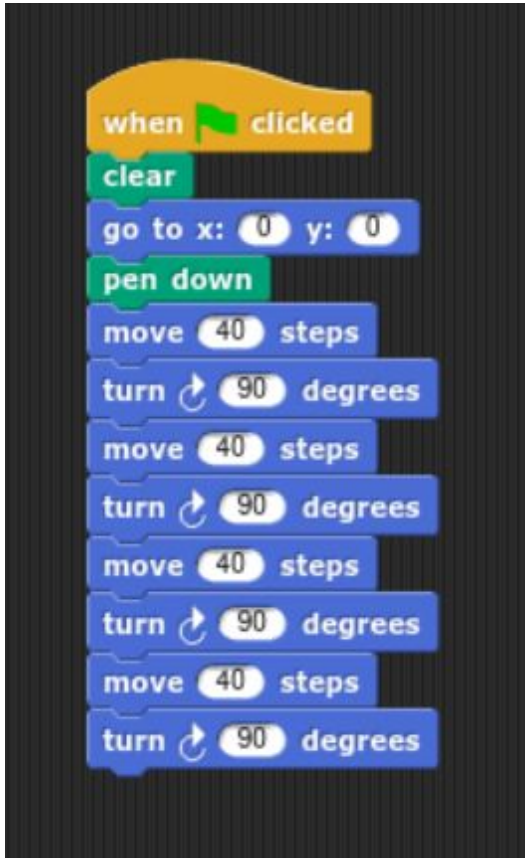
- **Esconder os detalhes para outros projetistas**:
  - O projetista do “Engine Control Module” (ECM) não precisa entender como a “Return Spring” funciona
  - Imagine o sistema todo como uma “hierarquia” de “partes” bem definidas, com funcionalidade precisa e com uma interface para outros utilizarem. Essas “partes” abstraem/escondem a complexidade interna e permitem que outros as utilizem sem se preocupar com os detalhes, que ficam escondidos.



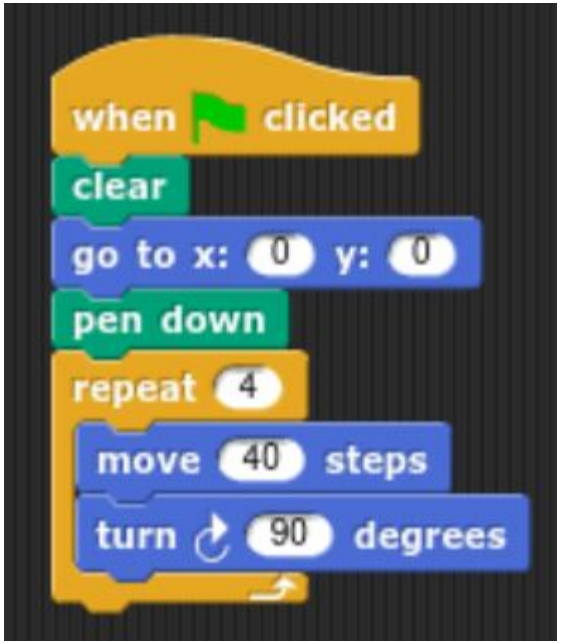
Abstração: **construir “peças” generalizáveis**!

- **Generalização nos permite evitar trabalho repetitivo**:

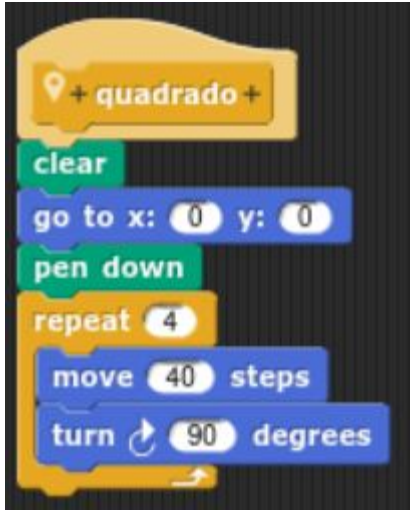
Abstração: **construir “peças” generalizáveis**!



Abstração: **construir “peças” generalizáveis**!



Abstração: **construir “peças” generalizáveis**!



Abstração: **“peças” generalizáveis** e **esconder detalhes**!

O novo “bloco” **esconde os detalhes** do usuário, e age como uma **ferramenta geral** para desenhar quadrados.

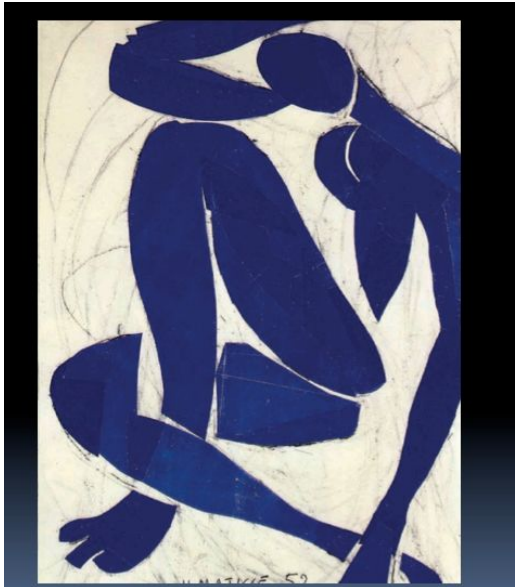
Identificar a abstração correta para resolver um problema é uma habilidade essencial que você deve adquirir.

Uma das maiores diferenças entre programadores bons e ruins é a habilidade de identificar e criar as abstrações necessárias.

Abstração: **generalização** e **esconder detalhes**!

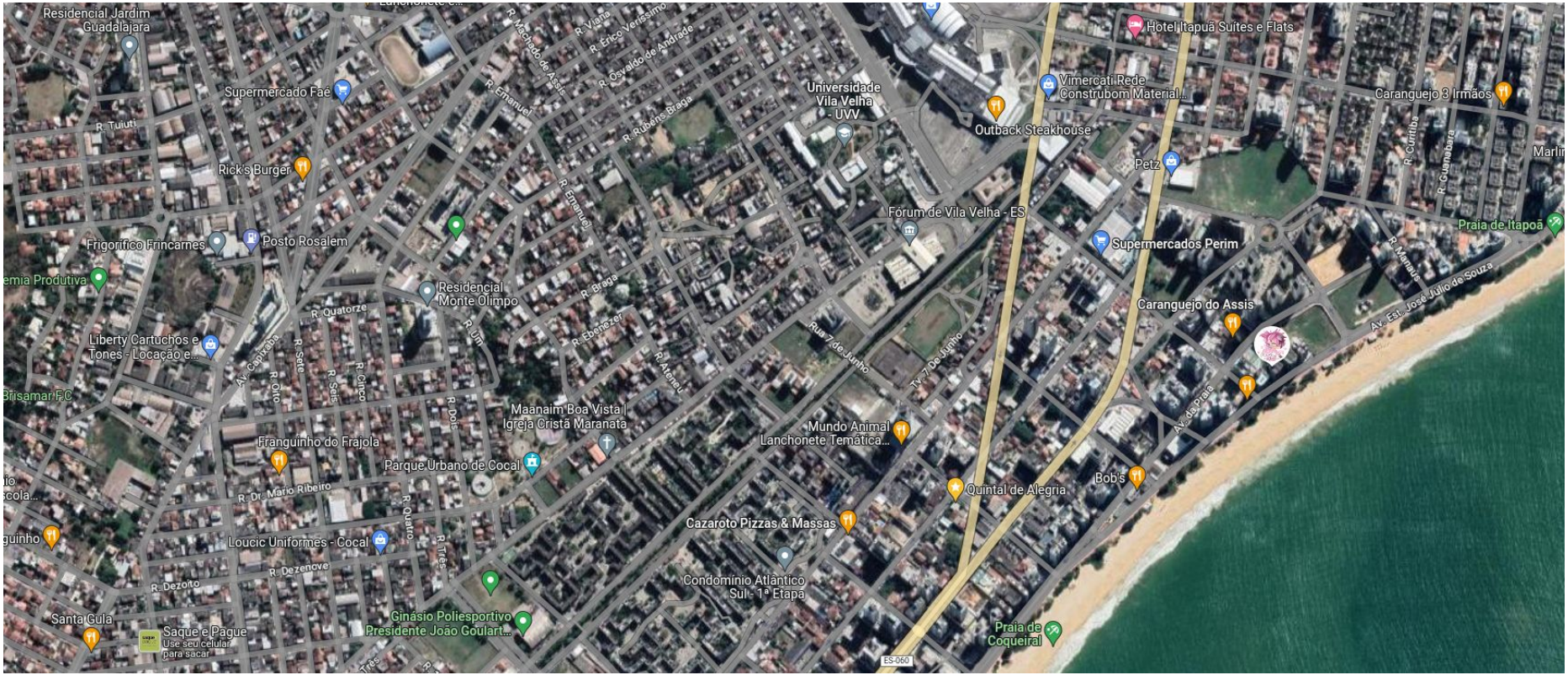
- Esconder detalhes:
  - O processo de ignorar uma ou mais propriedades de um objeto complexo para se focar no que é essencial.

- Generalização:
  - O processo de formular conceitos gerais abstraindo-se as propriedades, características e funcionalidades comuns.

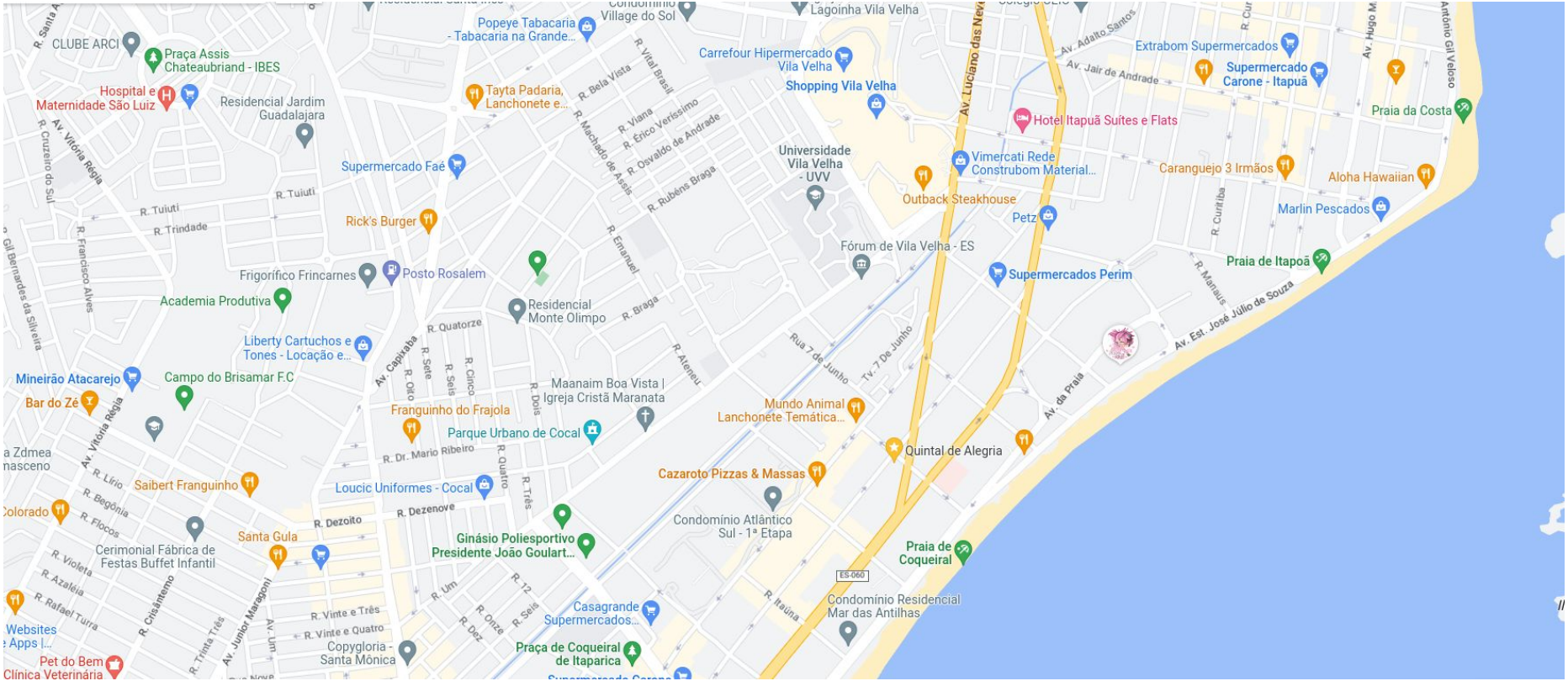


Naked Blue IV *(Henri Matisse, 1952)*

Abstração: **generalização** e **esconder detalhes**!



Abstração: **generalização** e **esconder detalhes**!



Abstração: **generalização** e **esconder detalhes**!

Abstração: **generalização** e **esconder detalhes**!

Generalization Example

- You have a farm with many different animals.
- Different food for each
- You have directions:
  - To feed dog, put dog food in dog dish
  - To feed chicken, put chicken food in chicken dish
  - To feed rabbit, put rabbit food in rabbit dish
  - Etc…
- How could you do better?
  - To feed <animal>, put <animal> food in <animal> dish

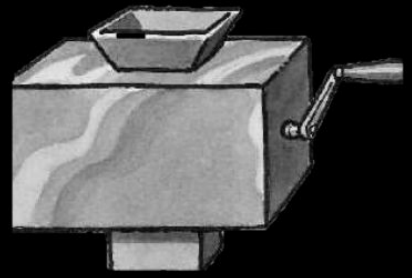
Abstração: **generalização** e **esconder detalhes**!

Generalization (in BJC) … foreshadowing

- You are going to learn to write functions, like in math class:

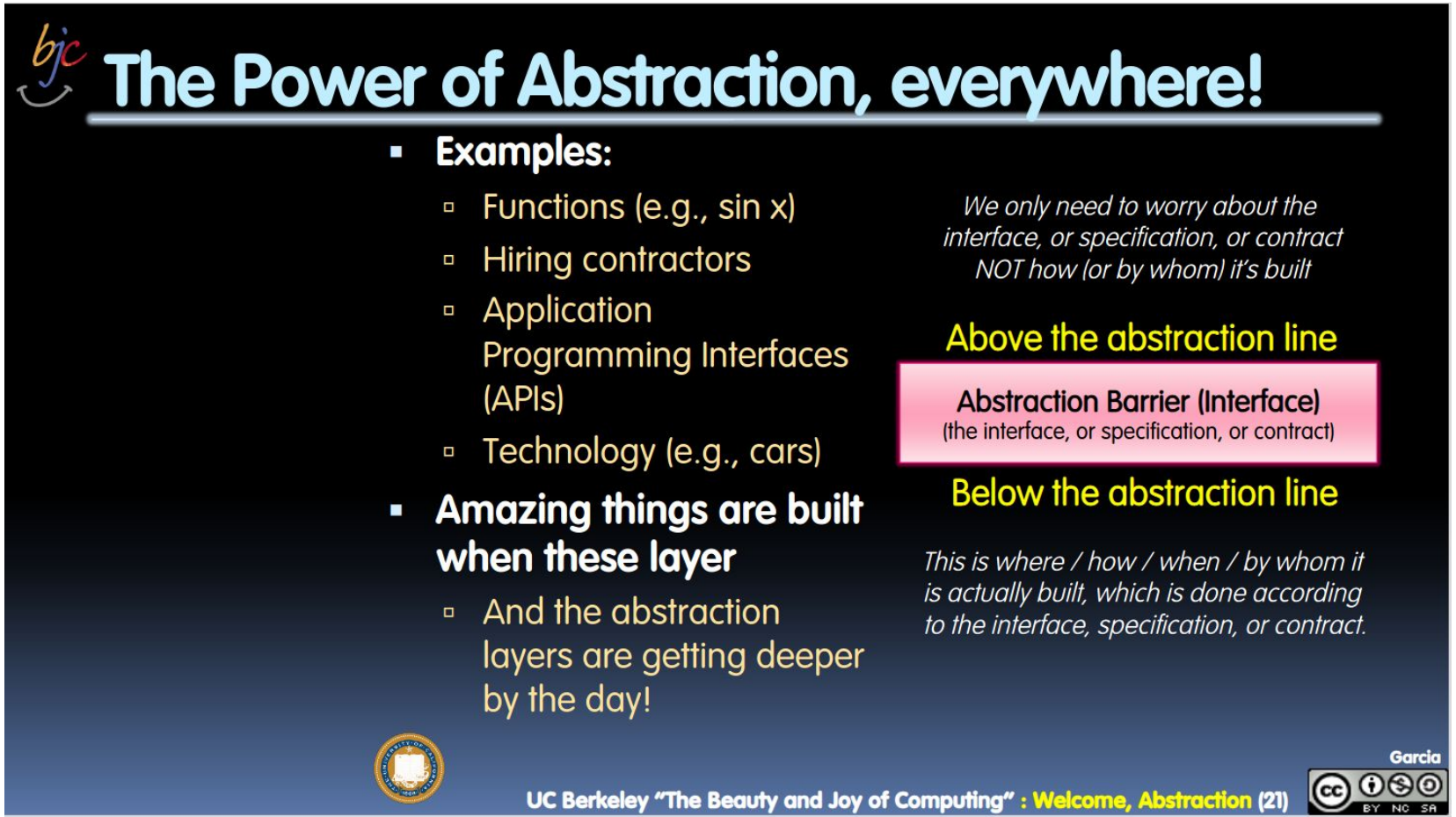
```
y = sin(x)
```

- You should think about **what inputs make sense to use so you don't have to duplicate code**



Function machine
(*Simply Scheme*, Harvey)

Abstração: **generalização** e **esconder detalhes**!



Abstração: **generalização** e **esconder detalhes**!

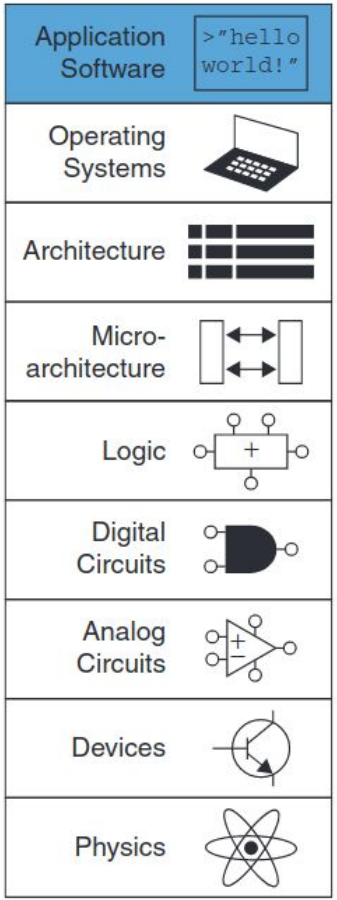
Summary

- Abstraction is one of the big ideas of computing!
- It's how mankind has engineered some of the greatest structures and managed the complexity
- Two definitions
  - Detail Removal
  - Generalization

Someone who drove in 1930 could still drive a car today because they've kept the same Abstraction!
(right pedal faster, left pedal slow)

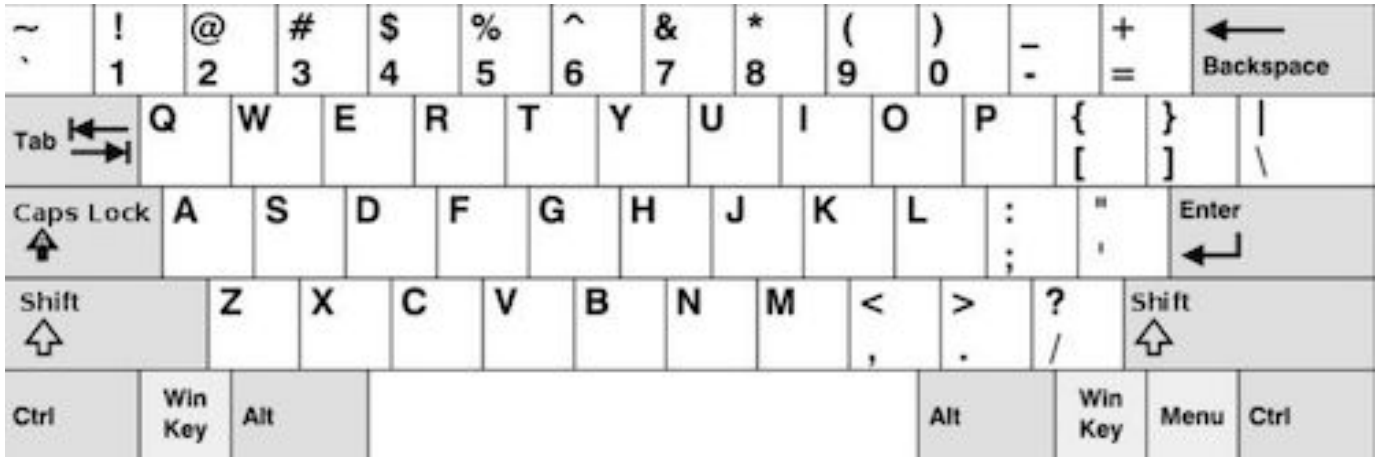
Garcia

Abstração: **generalização** e **esconder detalhes**!

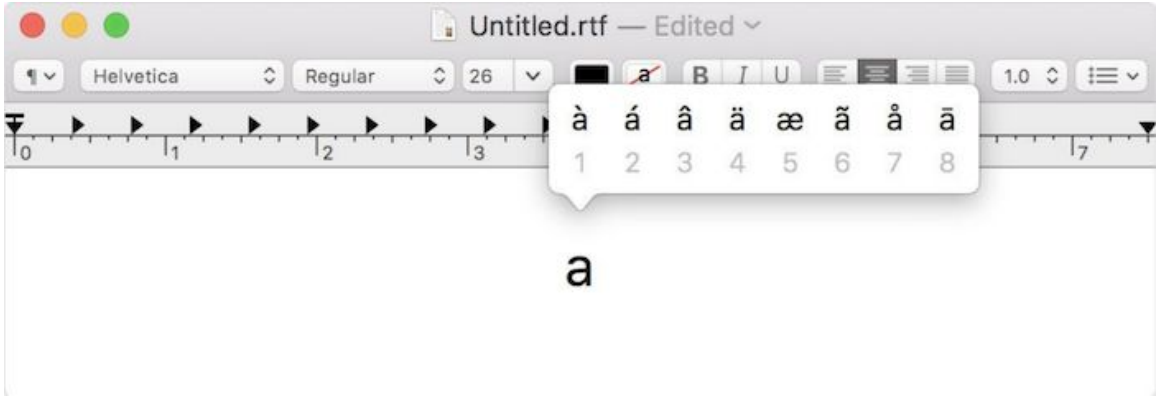


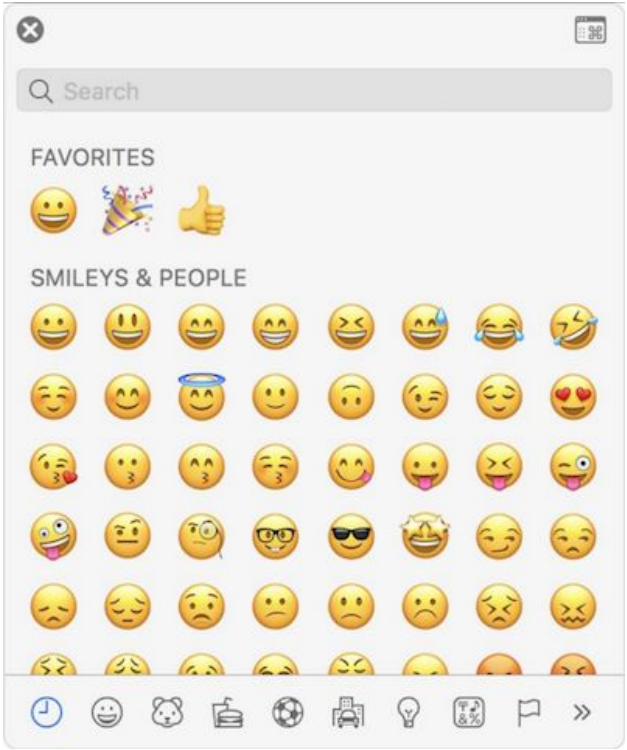
Revisitando a **REPRESENTAÇÃO** dos dados

Revisitando a **REPRESENTAÇÃO** dos dados



Revisitando a **REPRESENTAÇÃO** dos dados





ASCII x Unicode

ASCII: American Standard Code for Information Interchange

- Original: 7 bits, representando 128 caracteres
- Estendido: 8 bits (1 Bytes), representando 256 caracteres
- https://www.ascii-code.com

Unicode:

- Define 2 mapeamentos para armazenar caracteres:
  - Unicode Transformation Format (UTF)
  - Universal Coded Character Set (UCS)

- Principais mapeamentos:
  - UTF-8: usa de 1 até 4 Bytes para cada caractere, maximizando compatibilidade com ASCII (padrão)
  - UTF-16: usa 1 ou 2 códigos de 16 bits para cada caractere
  - UTF-32: use 1 código de 32 bits para cada caracteres
- Outros:
  - UTF-EBCDIC
  - UCS-2
  - UCS-4
- https://home.unicode.org

Unicode x Emoji

1F602	hexadecimal
128514	decimal
111110110000000010	binário

Como representar cores? RGB!

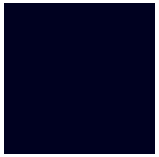
Para representar cores em um monitor, padronizou-se que cada pixel da tela teria sua cor representada por 3 Bytes:

- R: a quantidade de vermelho (red), de 0 a 255
- G: a quantidade de verde (green), de 0 a 255
- B: a quantidade de azul (blue), de 0 a 255

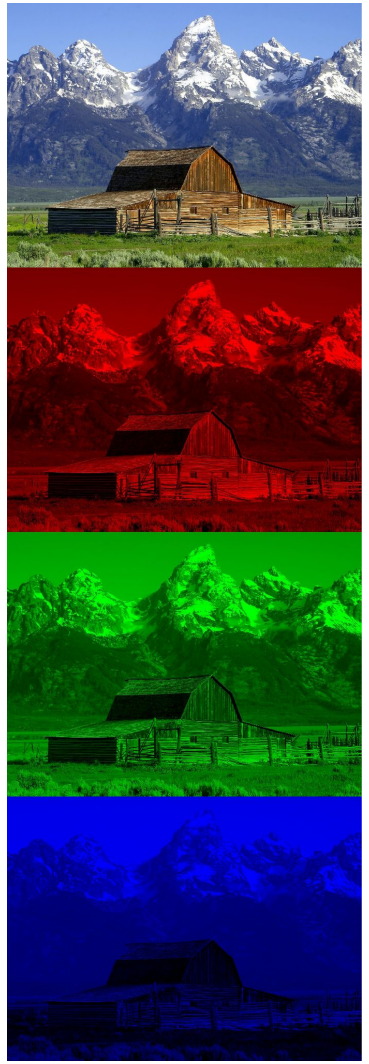
79,0,0

0,105,0

0,0,33



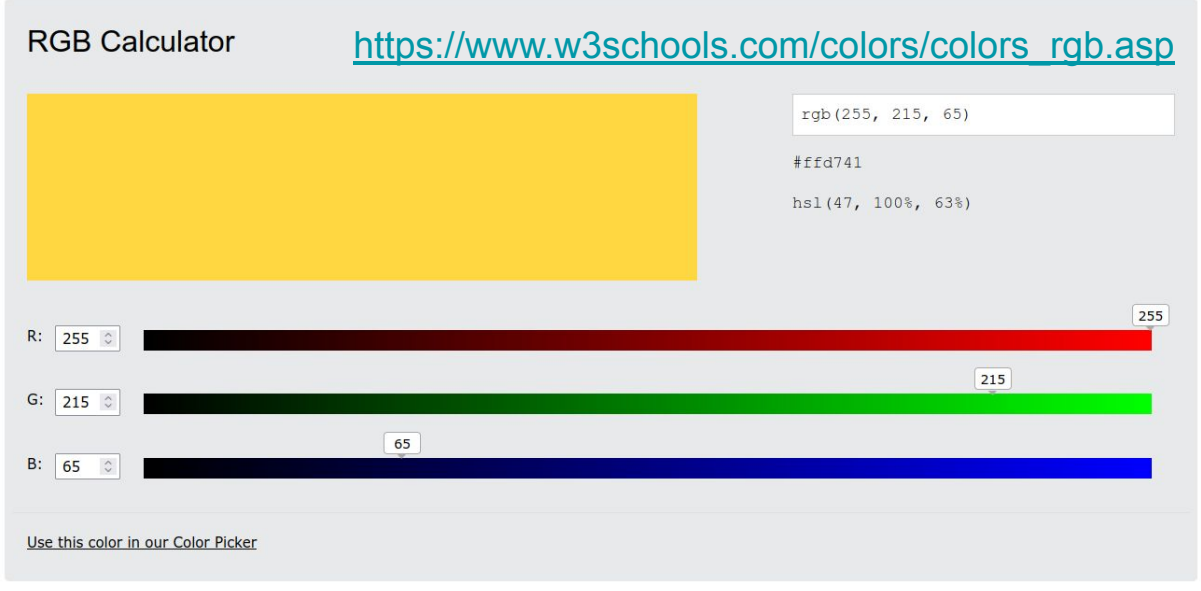
79,105,33

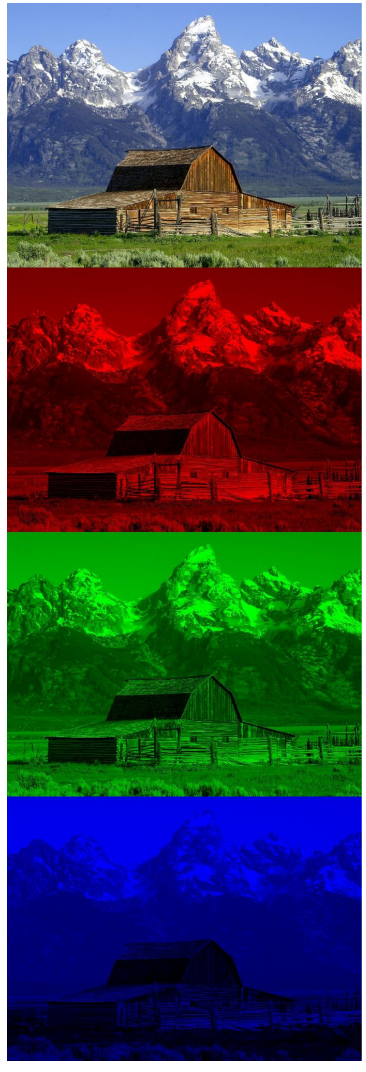


Como representar cores? RGB!

Para representar cores em um monitor, padronizou-se que cada pixel da tela teria sua cor representada por 3 Bytes:

- R: a quantidade de vermelho (red), de 0 a 255
- G: a quantidade de verde (green), de 0 a 255
- B: a quantidade de azul (blue), de 0 a 255

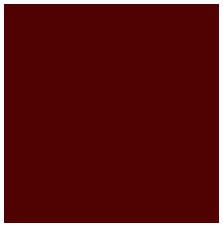




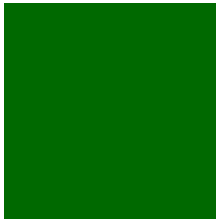
Como representar cores? RGB!

O contexto importa na representação de dados!

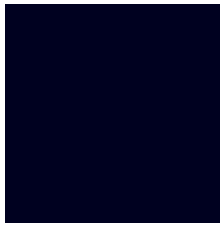
79,0,0



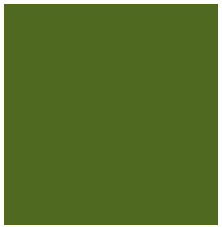
0,105,0



0,0,33



79,105,33



Oi!

79 105 33

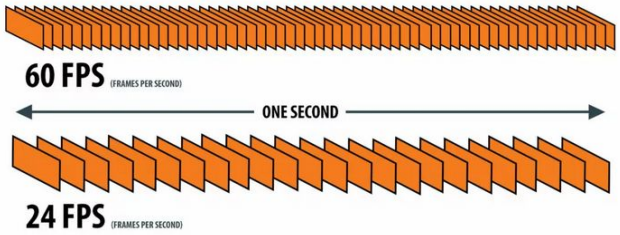
Como representar vídeos?

Grumpy Cloud, por Andymation
https://www.youtube.com/watch?v=p3q9MM__h-M

Cinema: 24 frames/segundo (FPS)

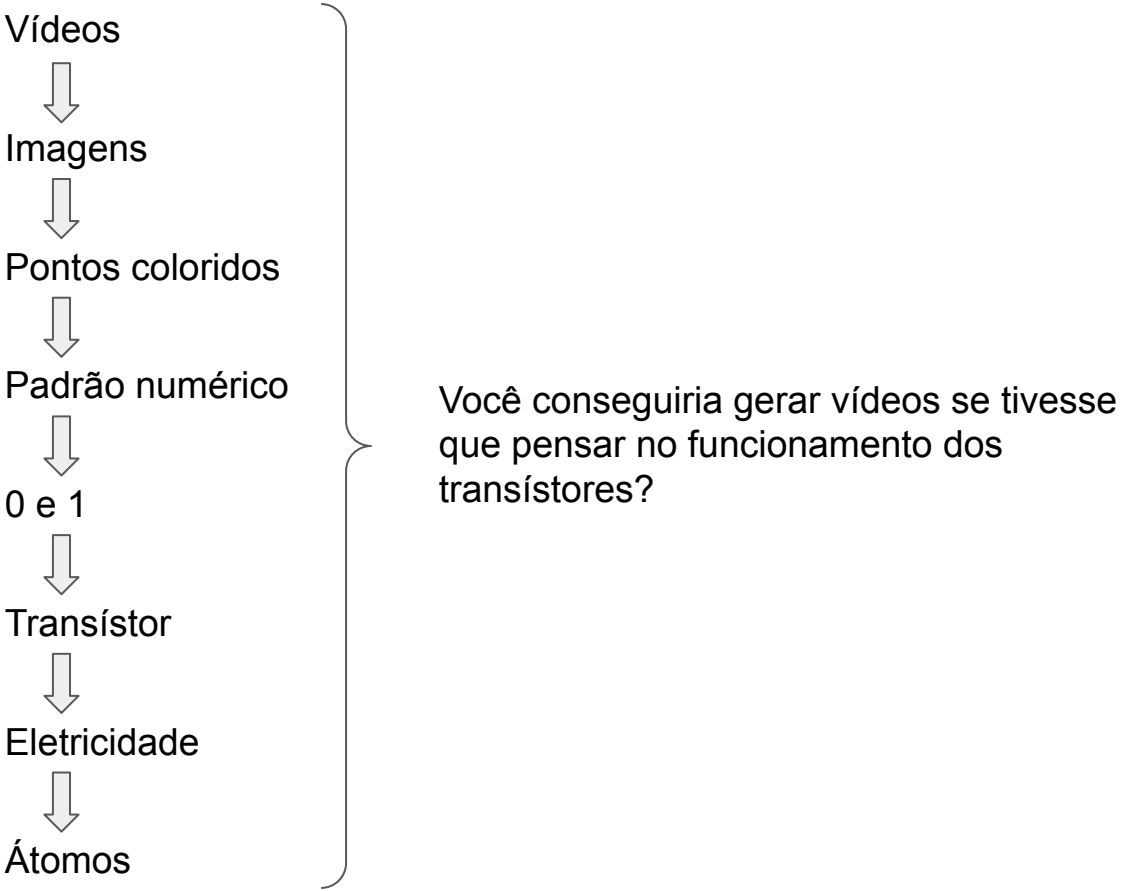
Vídeos Youtube: 30 ou 60 FPS

Games: 60+ FPS

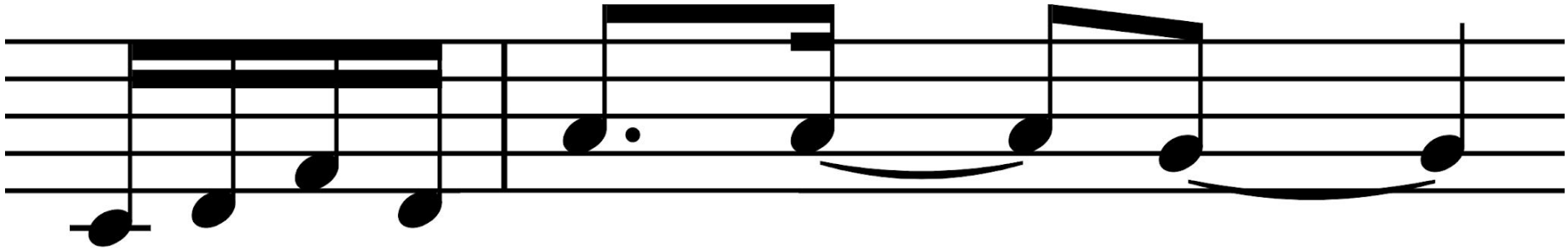


Como representar vídeos?

Representação x Abstração



Como representar músicas?



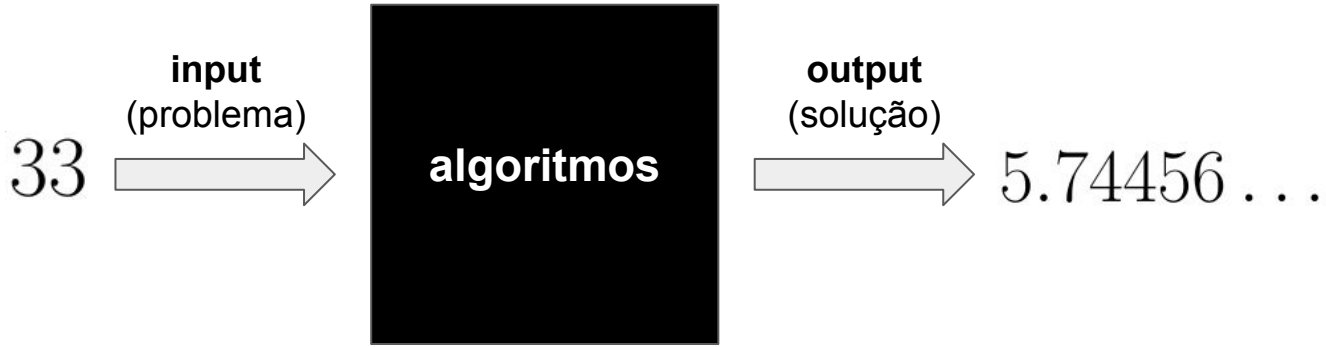
Cada nota é uma freqüência e podemos representar

- Freqüencia (Hz)
- Altura
- Duração
- …

Sistemas: MIDI, MP3, AAC…

ALGORITMOS

Algoritmo: “receita” para resolver um problema.



Método de Newton:

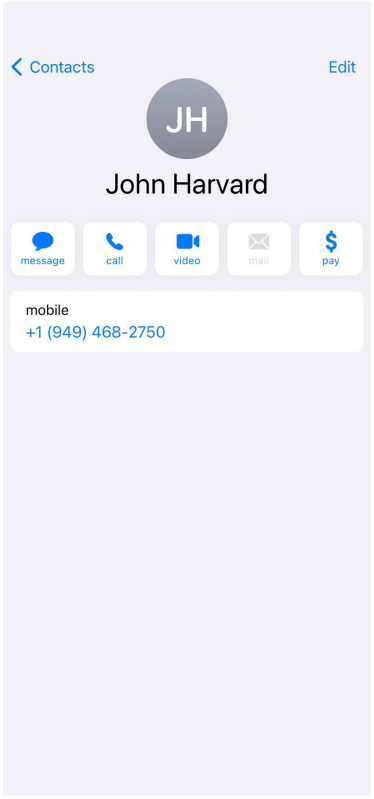
1. Chute um valor para y;
2. Calcule o valor de y^2;
3. Se $y^2 = x$ (ou um valor próximo o suficiente), você achou a raiz. Termine o procedimento.
4. Se $y^2 \neq x$ (ou não está próximo o suficiente), melhore a estimativa de y fazendo o seguinte cálculo: novo $y = \dfrac{y + \frac{x}{y}}{2}$;
5. Retorne ao segundo passo até que você encontre a raiz ou uma aproximação suficiente.

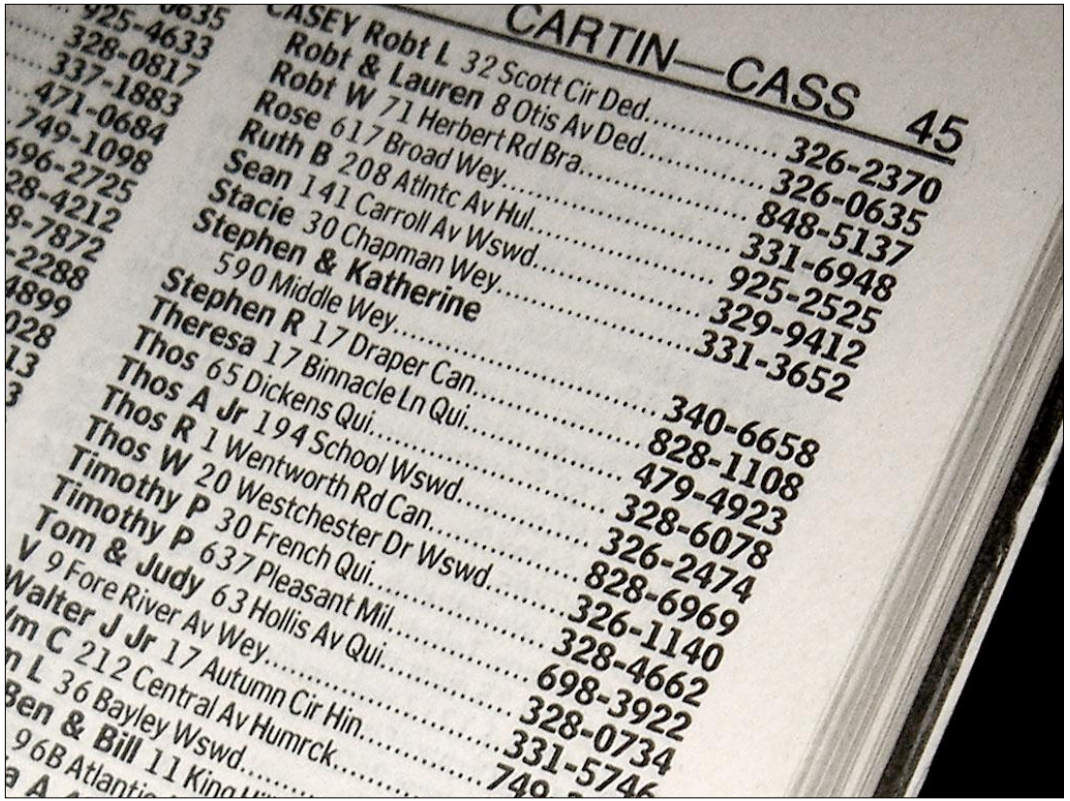
Conhecimento IMPERATIVO:

Nos diz **COMO** resolver um problema.
Essa “receita” é chamada de **ALGORITMO**, e deve ser:

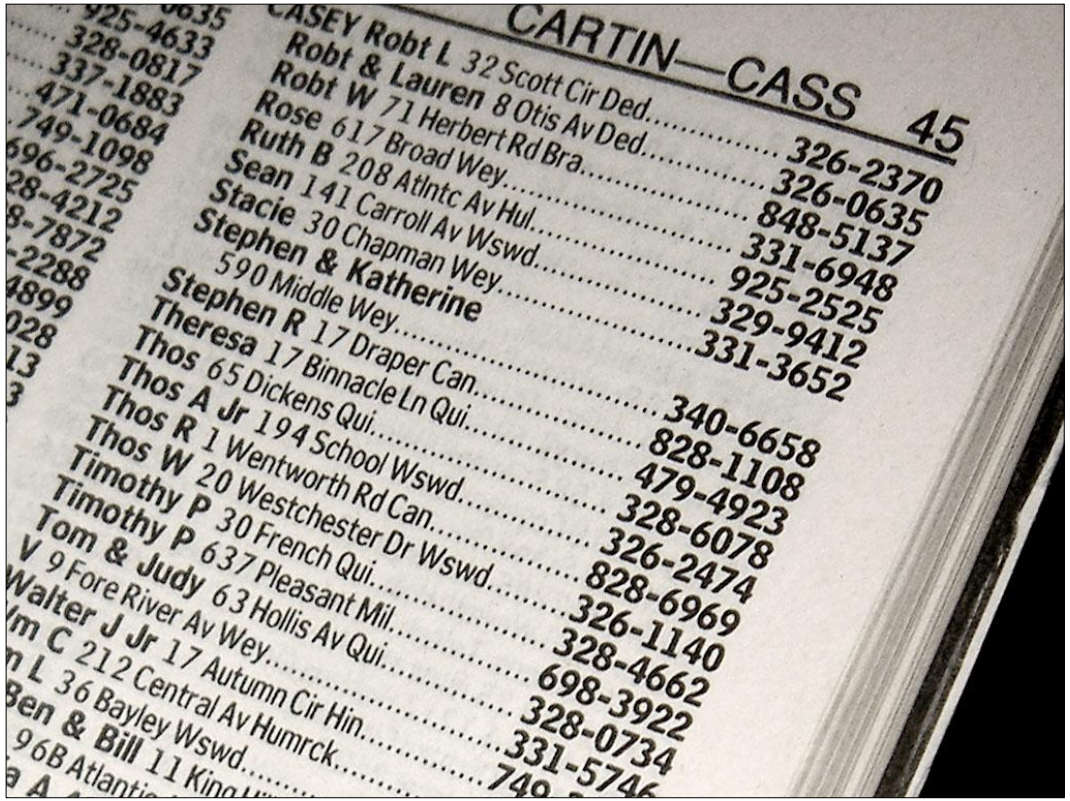
- Correto
- Eficiente
  - Tempo
  - Espaço

Algoritmo: “receita” para resolver um problema.





Algoritmo: vários para o mesmo problema… qual o melhor?



1º Algoritmo:

- Comece a pesquisar a partir da 1ª página, lendo todos os nomes, um por um, até encontrar.

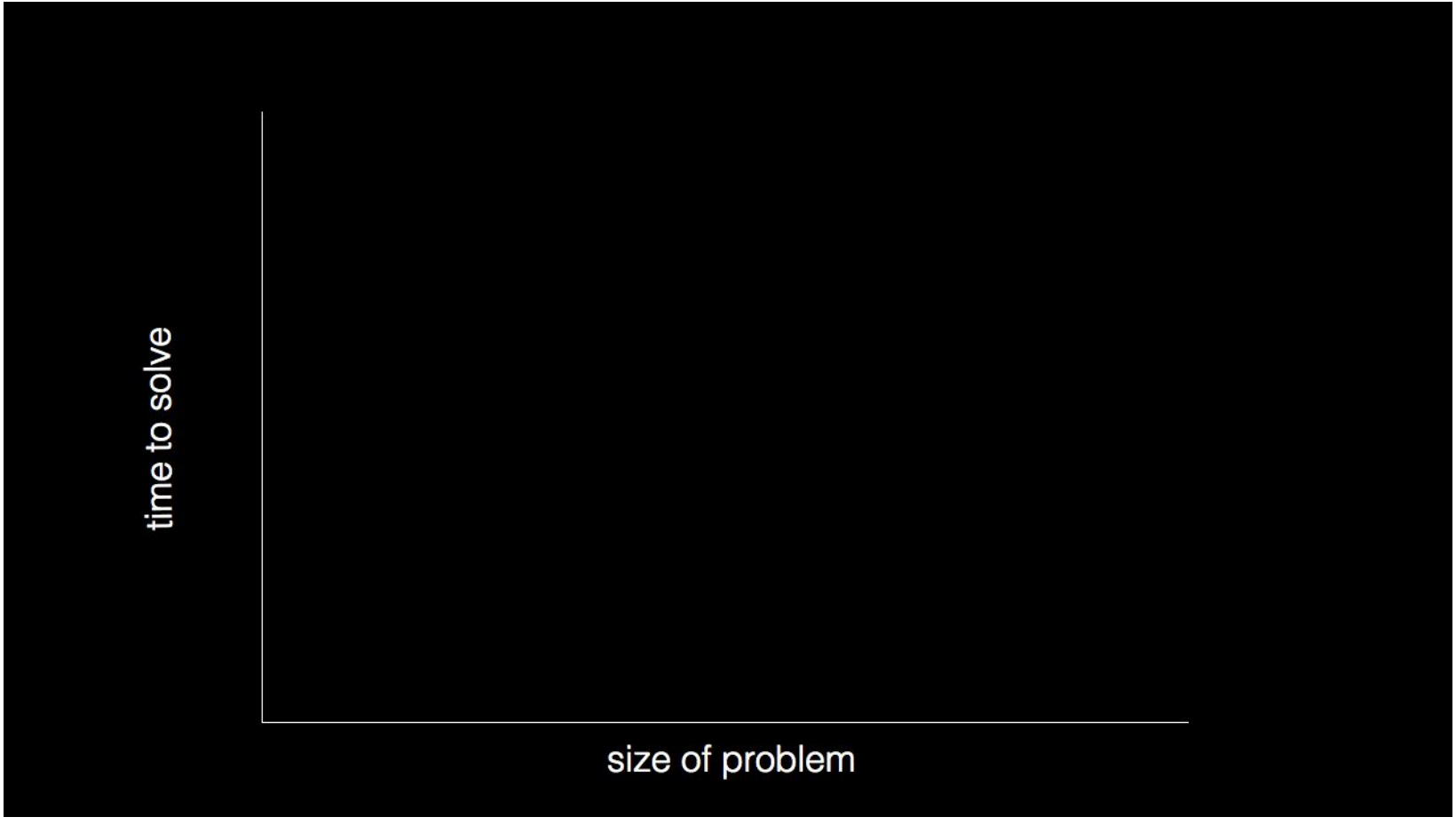
2º Algoritmo:

- Comece a pesquisar a partir da 1ª página, lendo todos os nomes, mas passe duas páginas de cada vez.

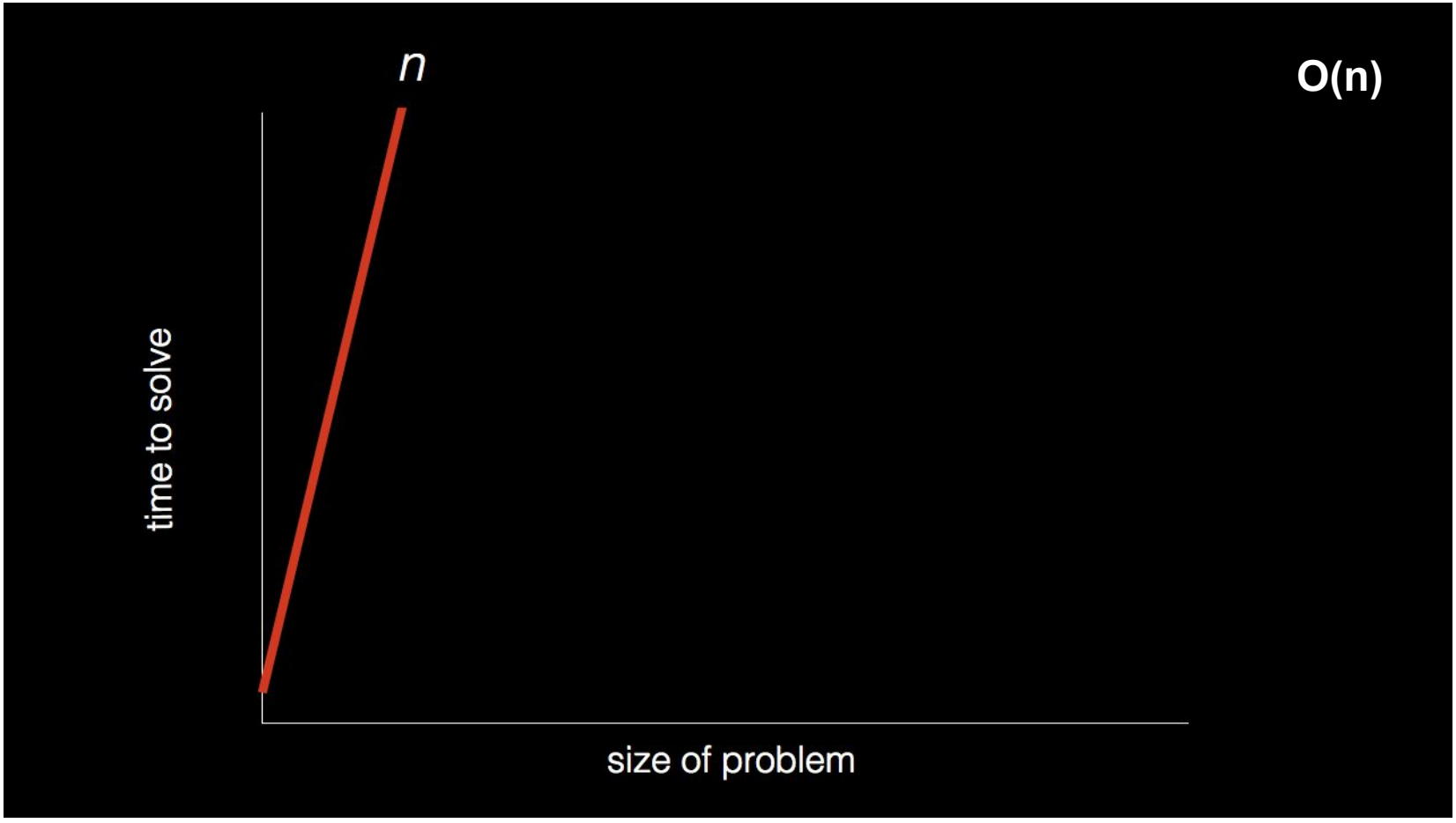
3º Algoritmo:

- Abra no meio. Verifique se o nome está lá. Se não estiver, descarte a metade onde o nome não está. Repita.

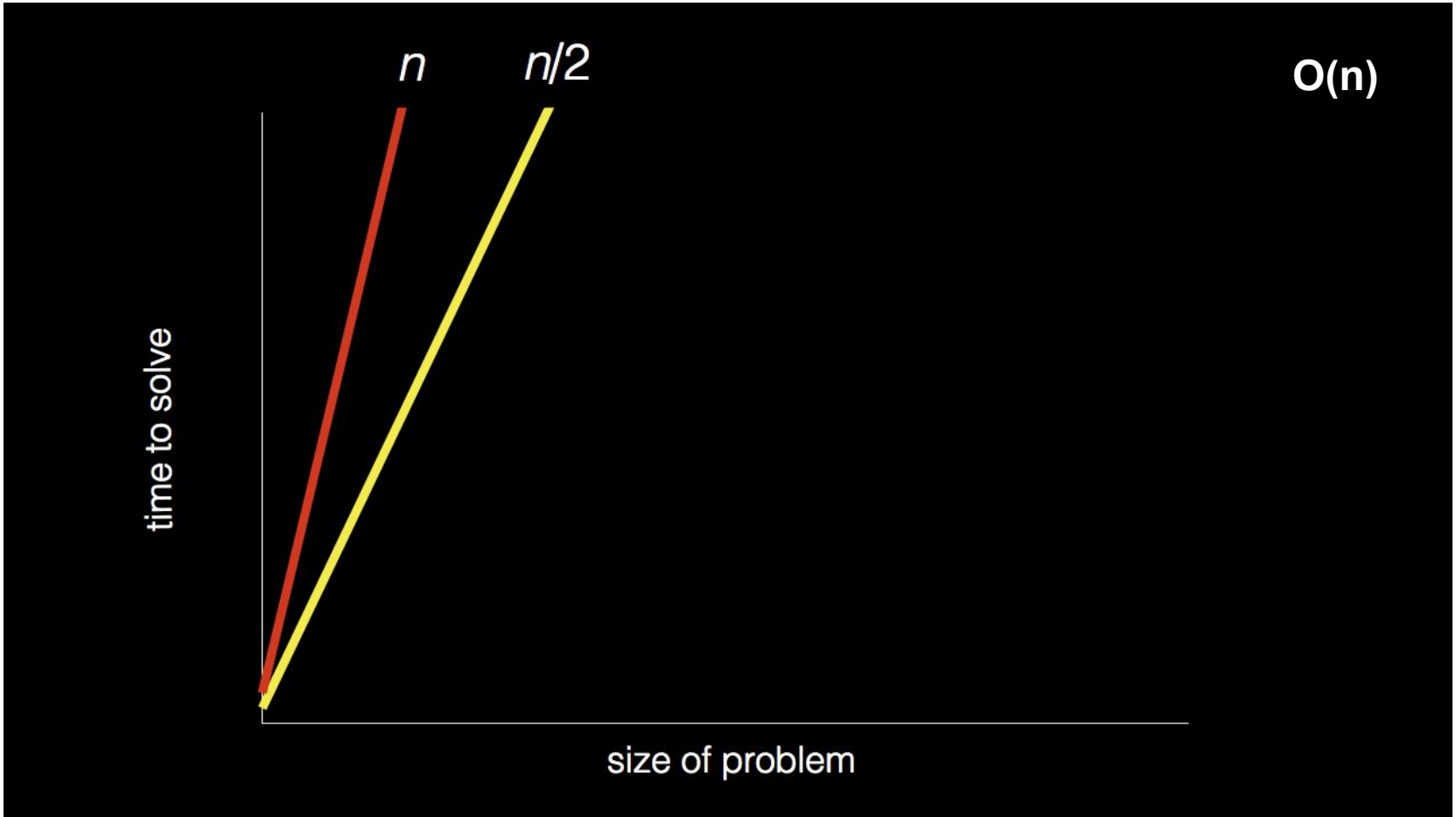
Algoritmo: vários para o mesmo problema… qual o melhor?



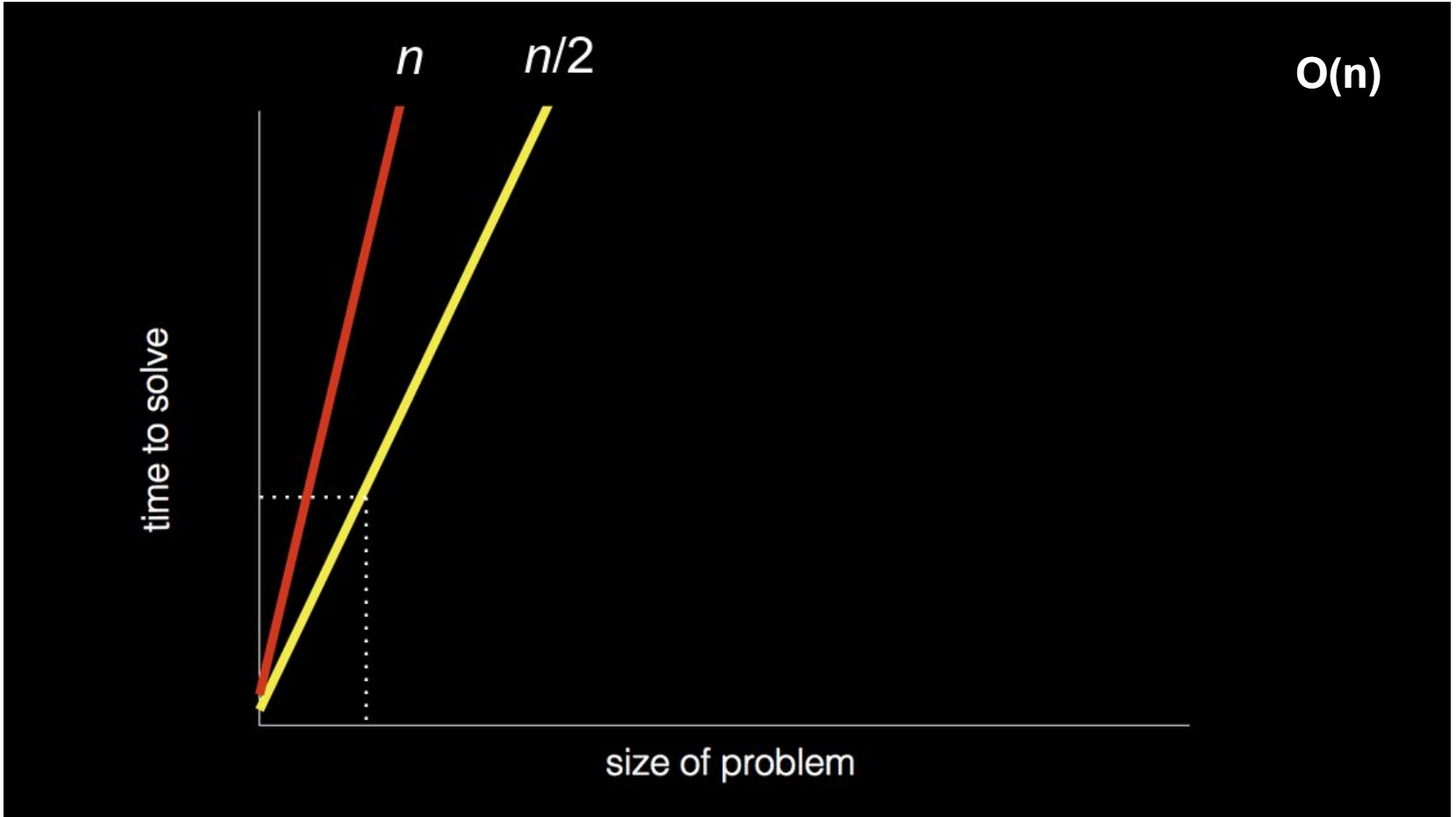
Algoritmo: vários para o mesmo problema… qual o melhor?



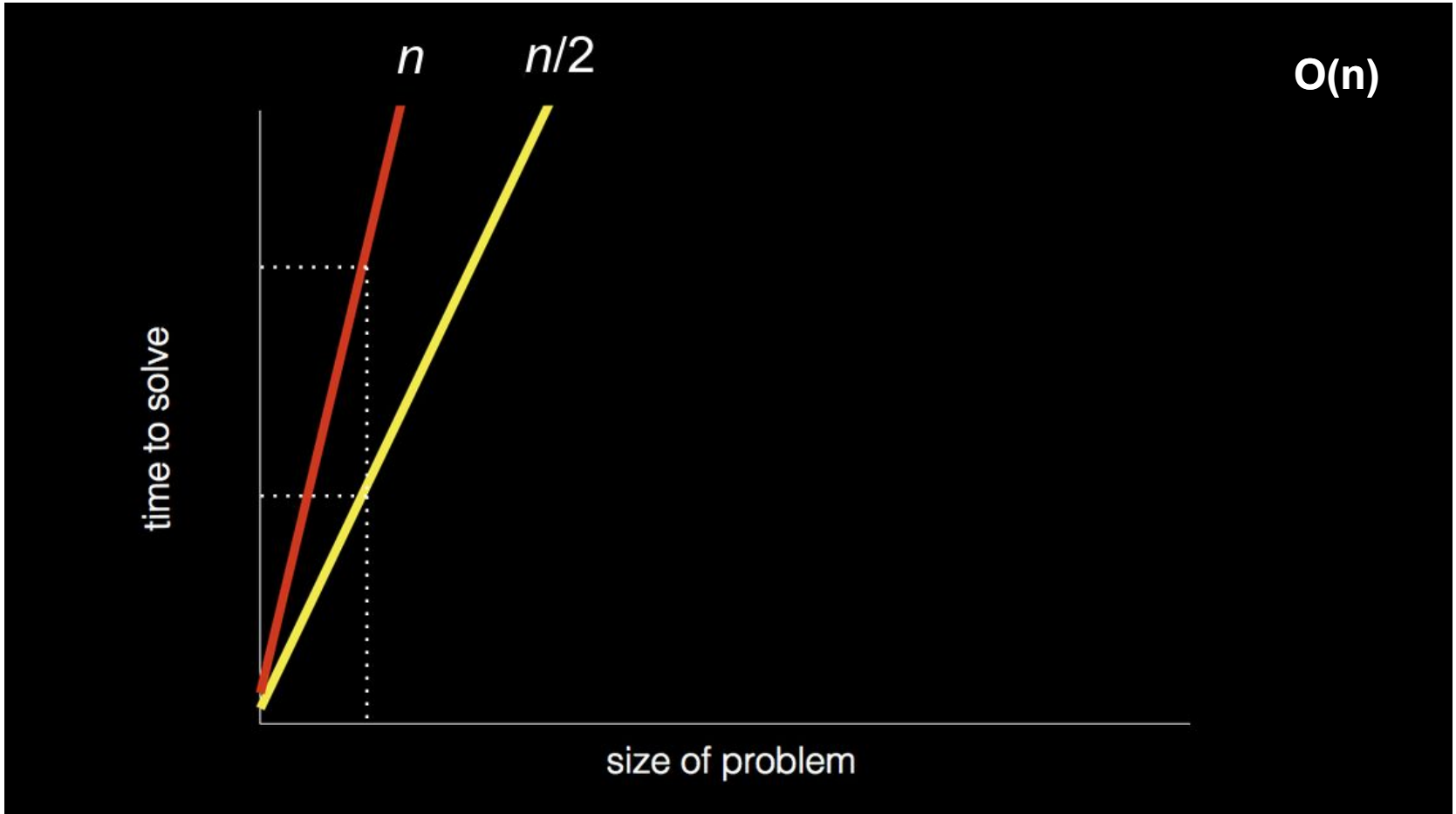
Algoritmo: vários para o mesmo problema… qual o melhor?



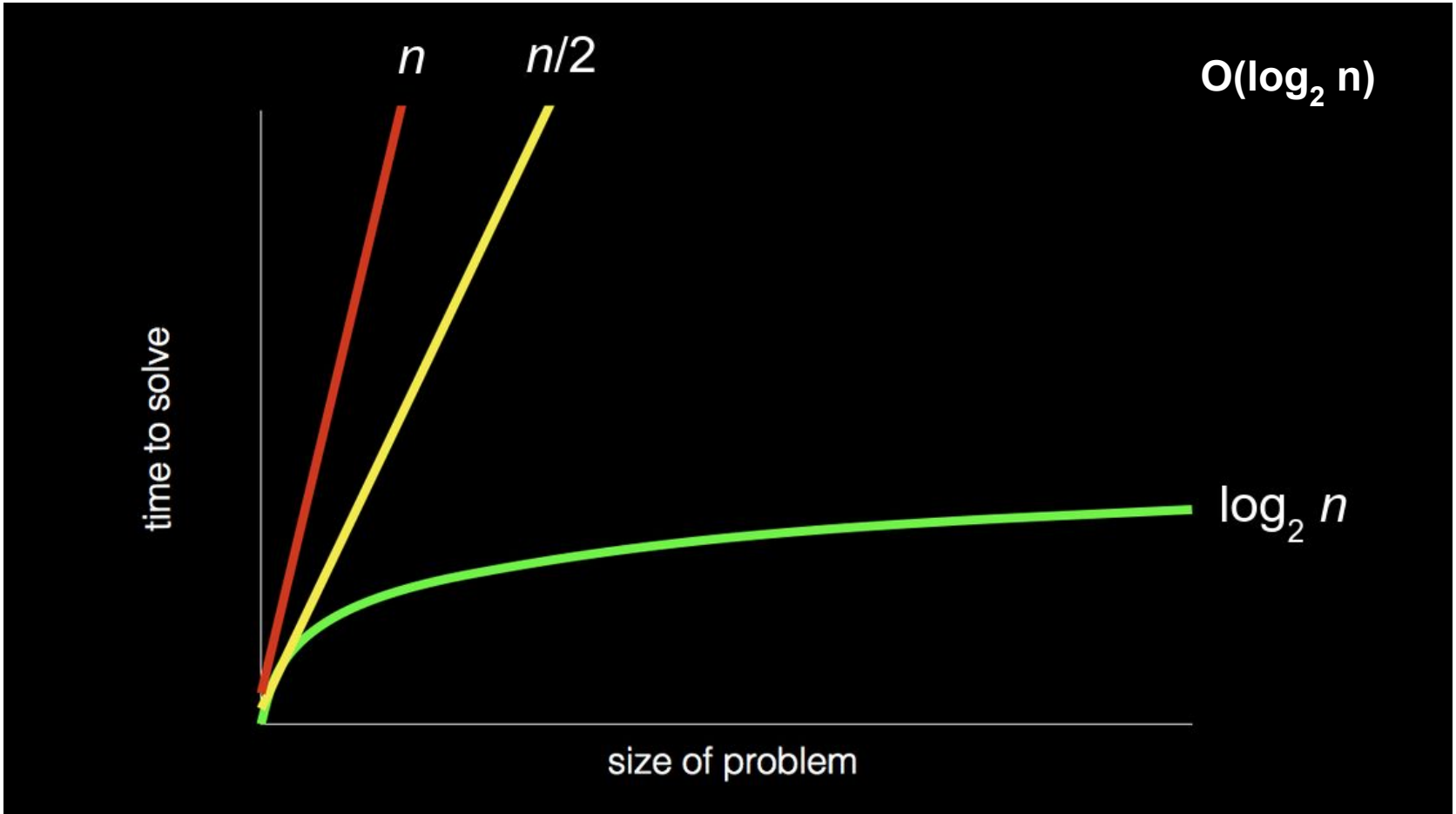
Algoritmo: vários para o mesmo problema… qual o melhor?



Algoritmo: vários para o mesmo problema… qual o melhor?



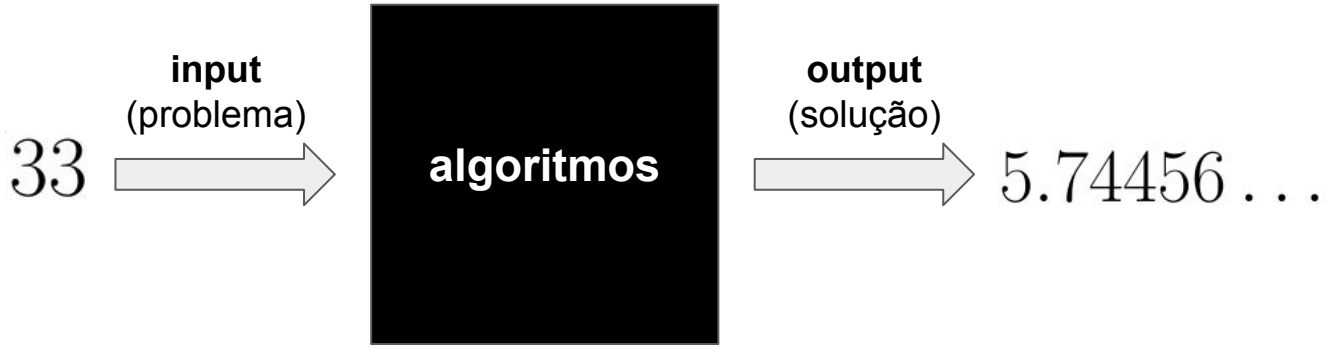
Algoritmo: vários para o mesmo problema… qual o melhor?



Algoritmo: vários para o mesmo problema… qual o melhor?

nomes	n	n/2	$\log_2 n$
500	500	250	9
1.000	1.000	500	10
2.000	2.000	1.000	11
4.000	4.000	2.000	12
8.000	8.000	4.000	13
16.000	16.000	8.000	14
32.000	32.000	16.000	15
64.000	64.000	32.000	16
128.000	128.000	64.000	17
256.000	256.000	128.000	18
512.000	512.000	256.000	19
1.024.000	1.024.000	512.000	20

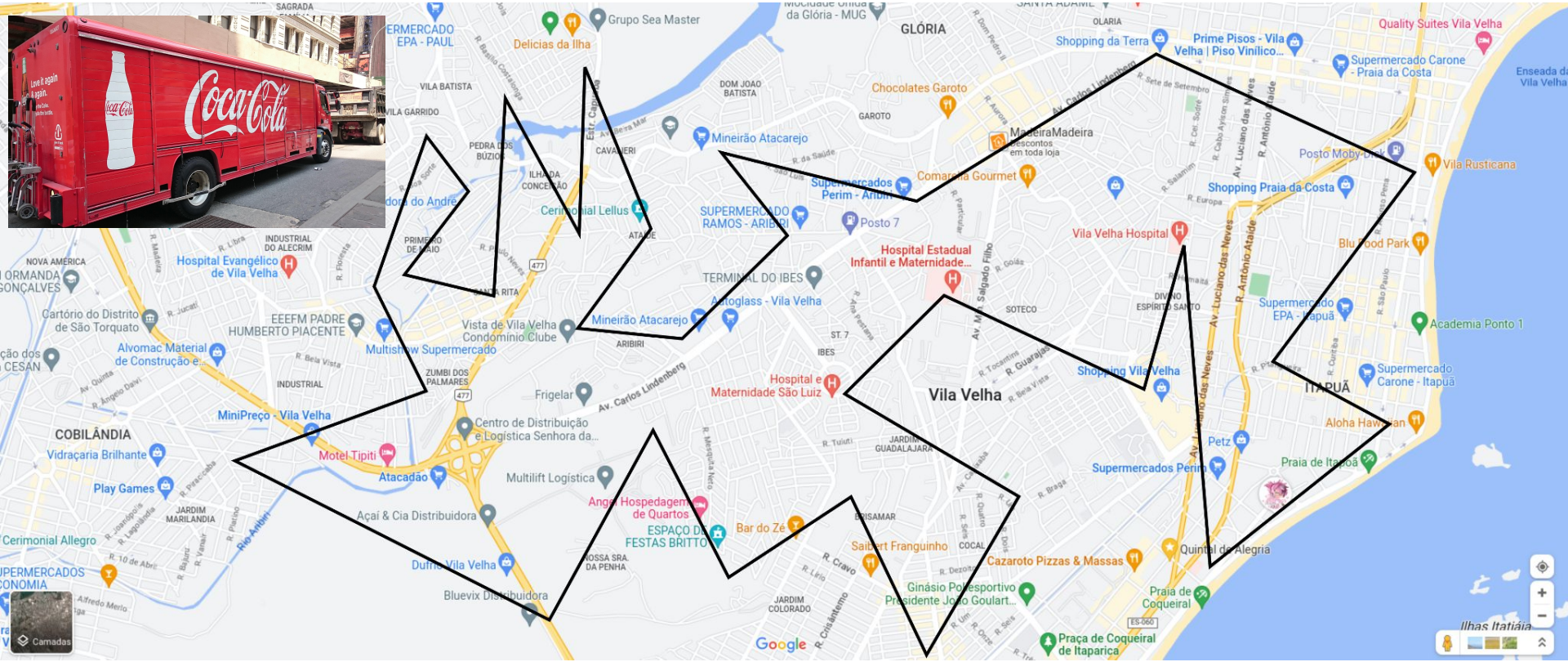
Algoritmo: “receita” para resolver um problema.



Os **ALGORITMOS** são a “alma” da computação, pois são eles que nos dizem **COMO RESOLVER UM PROBLEMA**, de forma:

- Correta
- Eficiente
  - Tempo
  - Espaço

Algoritmo: é fácil encontrar solução para tudo?



Algoritmo: é fácil encontrar solução para tudo?

O(n!)

O Caixeiro Viajante

Bares	Intel Core i9-13900KS (6 GHz)	Supercomputador Frontier (1,1E18 FLOPS)
5	insignific.	insignific.
10	0,001 seg	insignific.
15	4 min	insignific.
20	13 anos	2 seg
25	82 milhões de anos	6 meses
30	1,5 quatrilhões de anos	7 milhões de anos
35	54 sextilhões de anos	297 trilhões de anos

Como escrever um algoritmo? **PSEUDOCÓDIGO!**

PSEUDOCÓDIGO:
instruções detalhadas em português

```
 1   Pegue a lista telefônica
 2   Abra na página do meio
 3   Procure o nome na página
 4   Se a pessoa procurada está na página
 5      Ligue para a pessoa e vá para a linha 13
 6   Ou se a pessoa está na parte anterior da lista:
 7      Abra na página do meio da metade esquerda da lista
 8      Volte para a linha 3
 9   Ou se a pessoa está na parte posterior da lista:
10      Abra na página do meio da metade direita da lista
11      Volte para a linha 3
12   Caso contrário:
13      Saia
```

PSEUDOCÓDIGO:
instruções detalhadas em português

```
 1   Pegue a lista telefônica
 2   Abra na página do meio
 3   Procure o nome na página
 4   Se a pessoa procurada está na página
 5      Ligue para a pessoa e vá para a linha 13
 6   Ou se a pessoa está na parte anterior da lista:
 7      Abra na página do meio da metade esquerda da lista
 8      Volte para a linha 3
 9   Ou se a pessoa está na parte posterior da lista:
10      Abra na página do meio da metade direita da lista
11      Volte para a linha 3
12   Caso contrário:
13      Saia
```

Ações: FUNÇÕES

PSEUDOCÓDIGO:
instruções detalhadas em português

```
 1   Pegue a lista telefônica
 2   Abra na página do meio
 3   Procure o nome na página
 4   Se a pessoa procurada está na página
 5      Ligue para a pessoa e vá para a linha 13
 6   Ou se a pessoa está na parte anterior da lista:
 7      Abra na página do meio da metade esquerda da lista
 8      Volte para a linha 3
 9   Ou se a pessoa está na parte posterior da lista:
10      Abra na página do meio da metade direita da lista
11      Volte para a linha 3
12   Caso contrário:
13      Saia
```

Decisões: CONDIÇÕES

PSEUDOCÓDIGO:
instruções detalhadas em português

```
 1   Pegue a lista telefônica
 2   Abra na página do meio
 3   Procure o nome na página
 4   Se a pessoa procurada está na página
 5      Ligue para a pessoa e vá para a linha 13
 6   Ou se a pessoa está na parte anterior da lista:
 7      Abra na página do meio da metade esquerda da lista
 8      Volte para a linha 3
 9   Ou se a pessoa está na parte posterior da lista:
10      Abra na página do meio da metade direita da lista
11      Volte para a linha 3
12   Caso contrário:
13      Saia
```

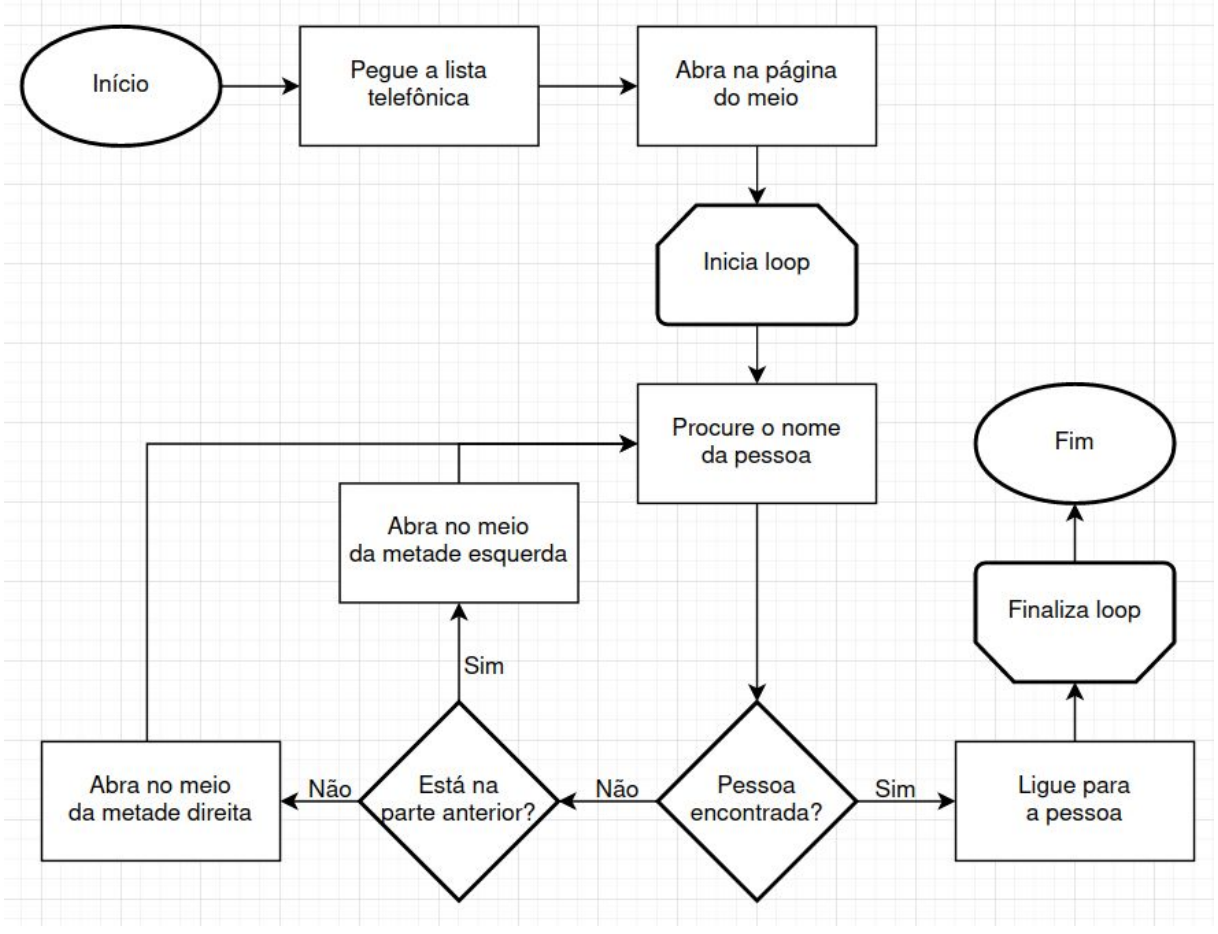
Perguntas: EXPRESSÕES BOOLEANAS

PSEUDOCÓDIGO:
instruções detalhadas em português

```
 1   Pegue a lista telefônica
 2   Abra na página do meio
 3   Procure o nome na página
 4   Se a pessoa procurada está na página
 5      Ligue para a pessoa e vá para a linha 13
 6   Ou se a pessoa está na parte anterior da lista:
 7      Abra na página do meio da metade esquerda da lista
 8      Volte para a linha 3
 9   Ou se a pessoa está na parte posterior da lista:
10      Abra na página do meio da metade direita da lista
11      Volte para a linha 3
12   Caso contrário:
13      Saia
```

Repetições: LOOPS

Outra maneira: **FLUXOGRAMA**



PSEUDOCÓDIGO E/OU FLUXOGRAMA: tradução para linguagem de programação

Agora que você já conhece os “blocos” básicos de construção de programas, sabe **escrever o algoritmo em pseudocódigo ou com um fluxograma**, podemos começar a traduzir esse algoritmo para uma linguagem de programação real!

- Funções (ações)
  - parâmetros, argumentos, retorno
- Condições (tomar decisões)
- Expressões booleanas (perguntas)
- Loops (repetições)

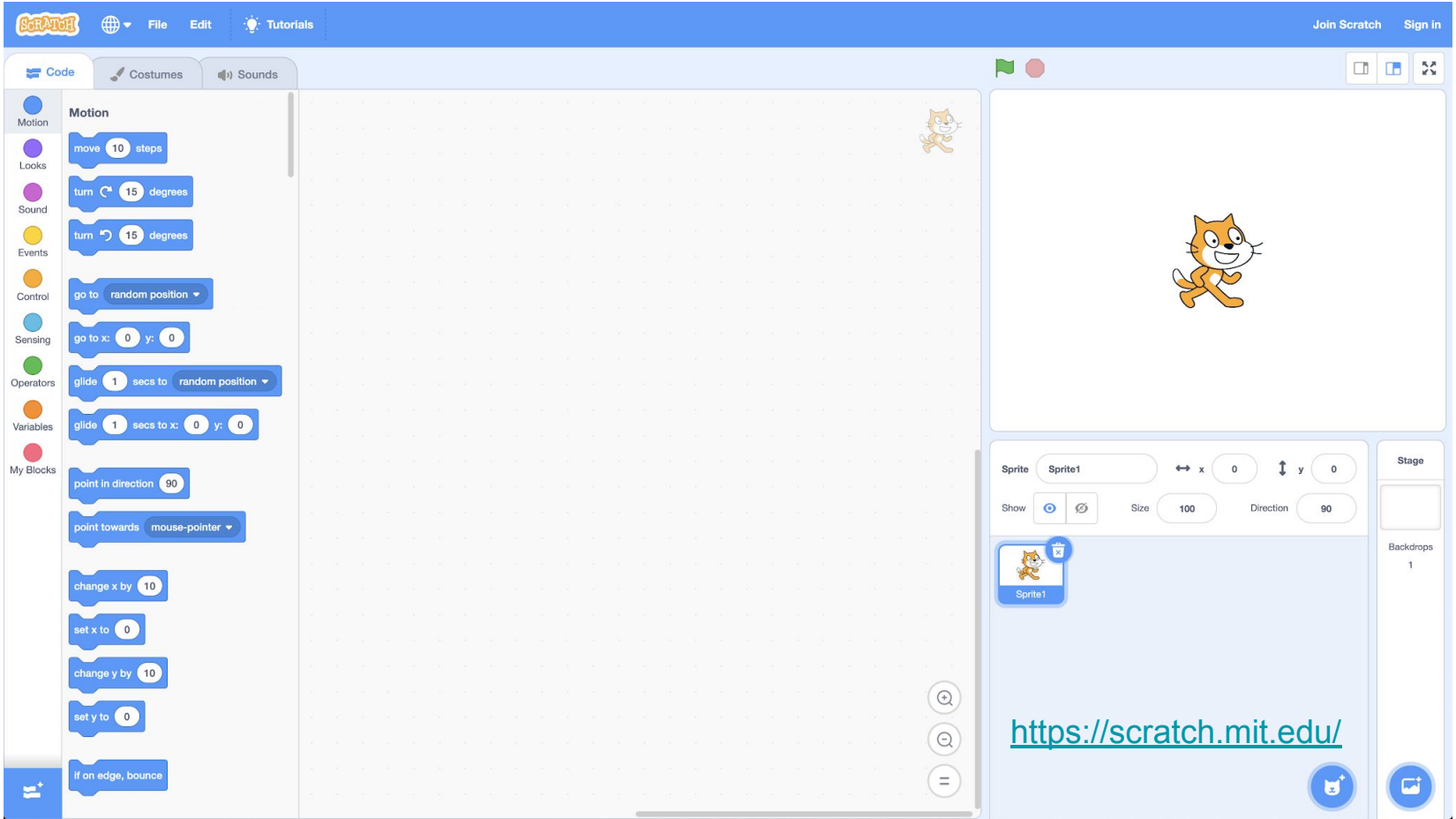
- Variáveis
- Threads
- Eventos
- …

A primeira linguagem de programação que veremos é o Scratch (ou o Sanp):

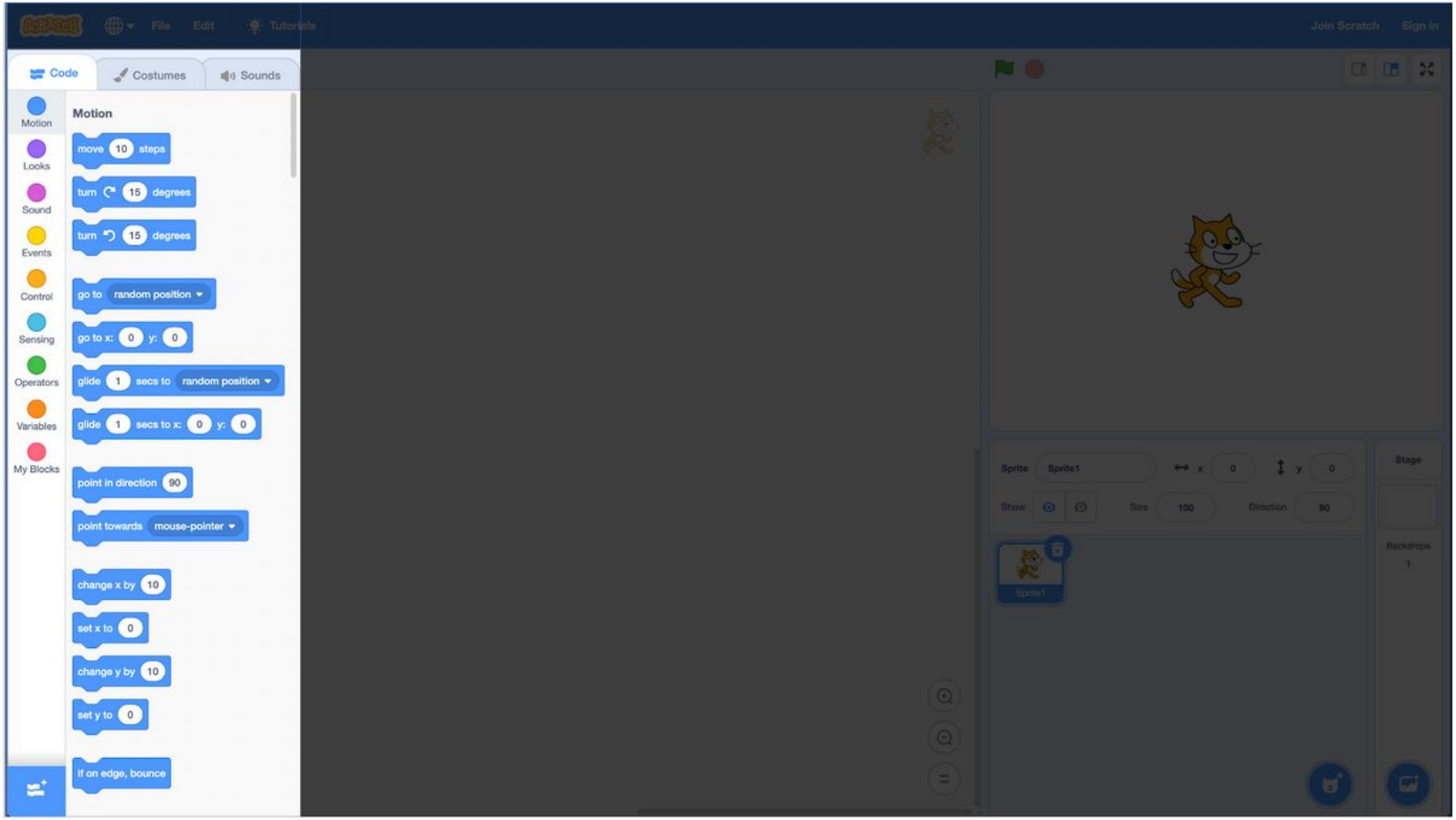
- Scratch: https://scratch.mit.edu/
- Snap!: https://snap.berkeley.edu/

Programação visual: **SCRATCH**

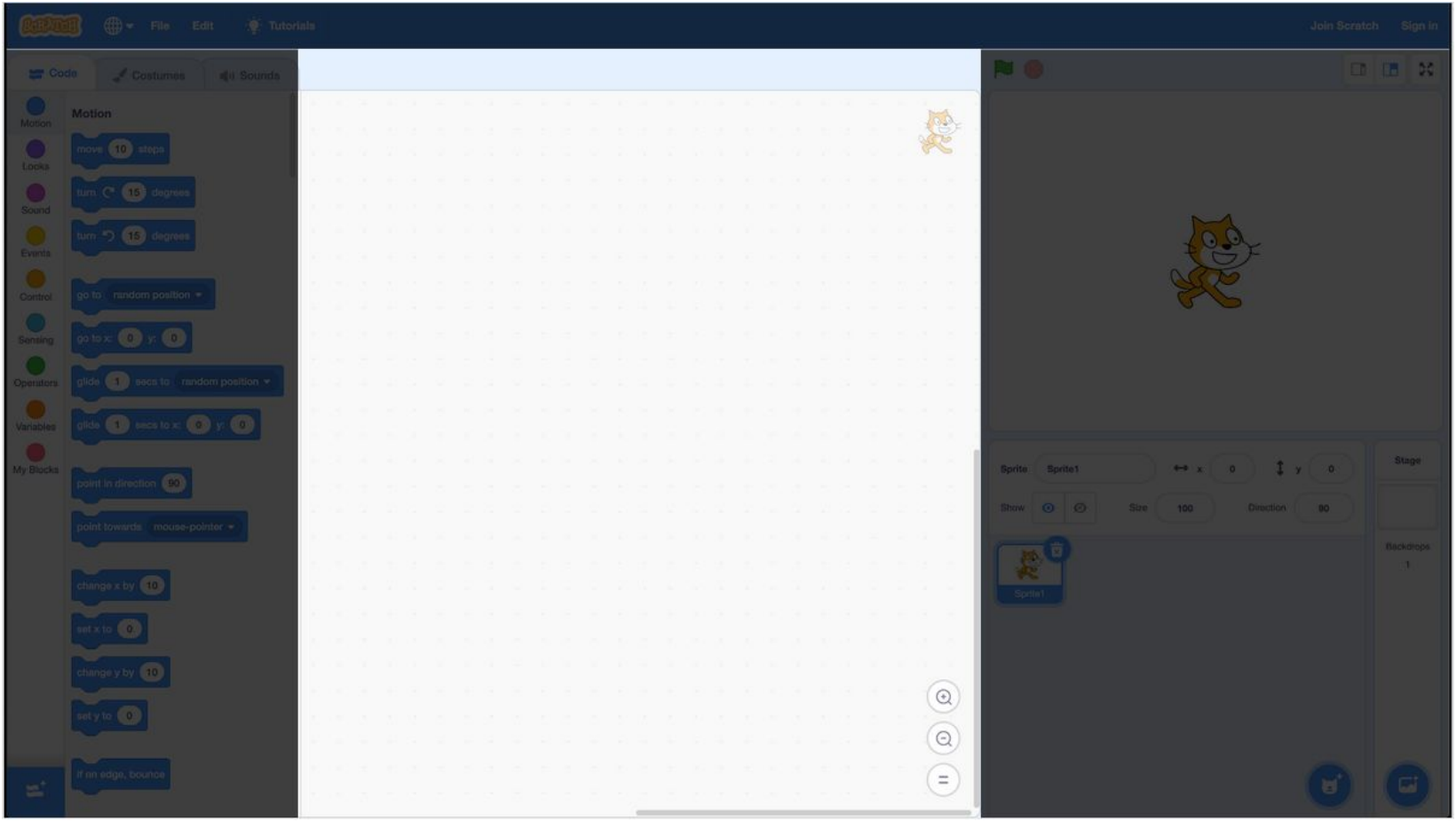
Programação visual: **SCRATCH**



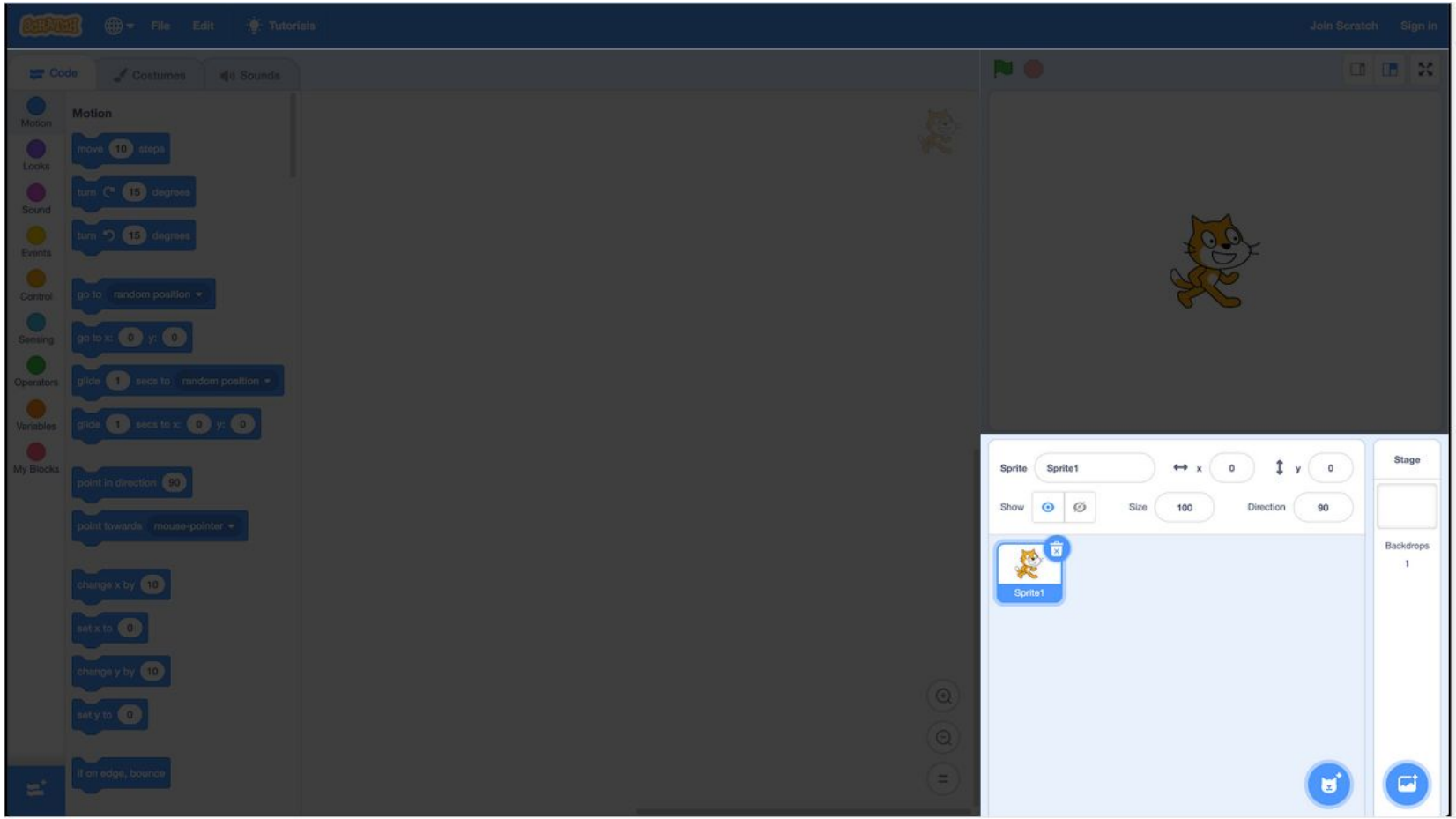
Área dos blocos, personagens e sons



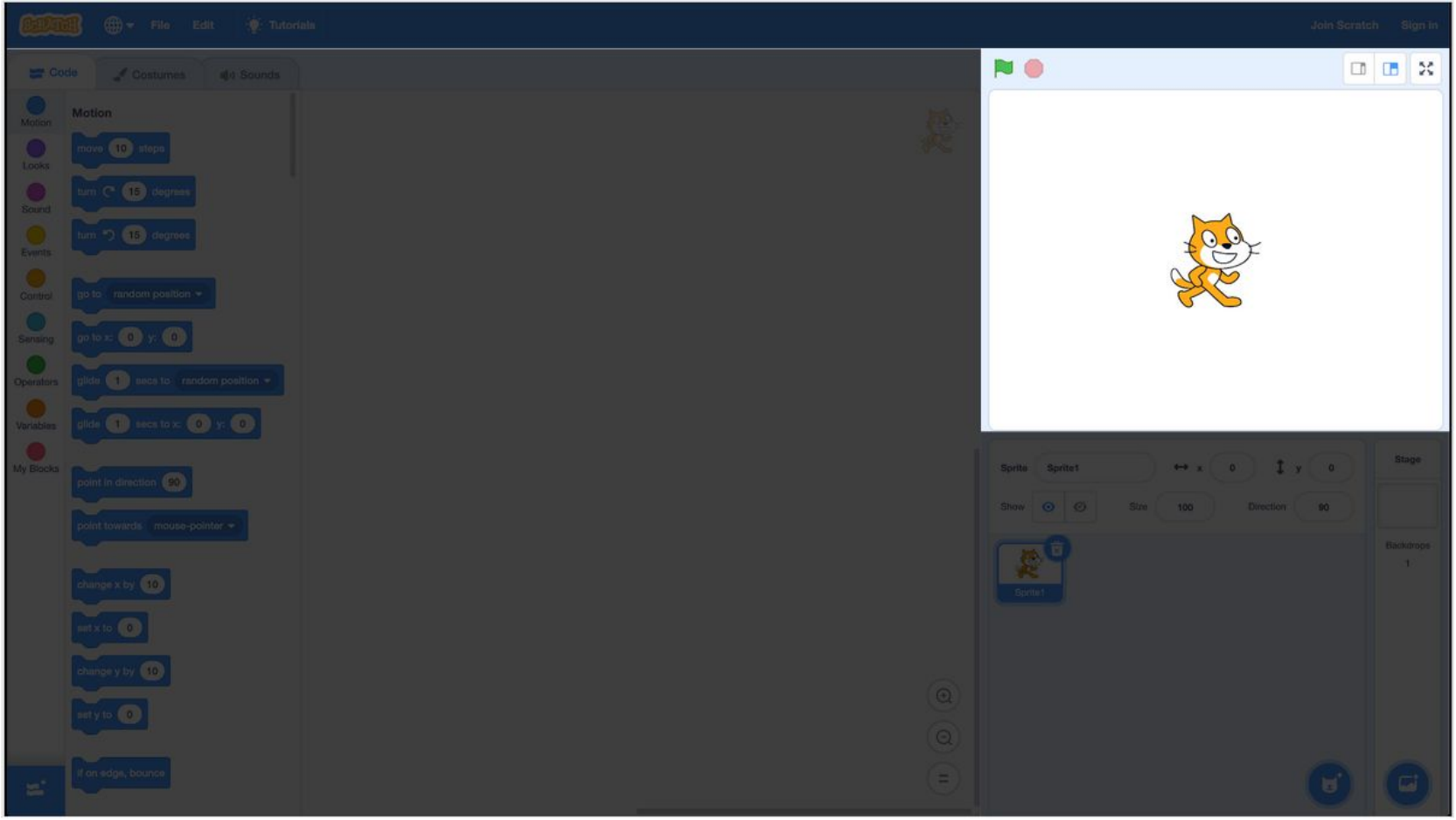
Área da programação de cada “sprite” (personagem)



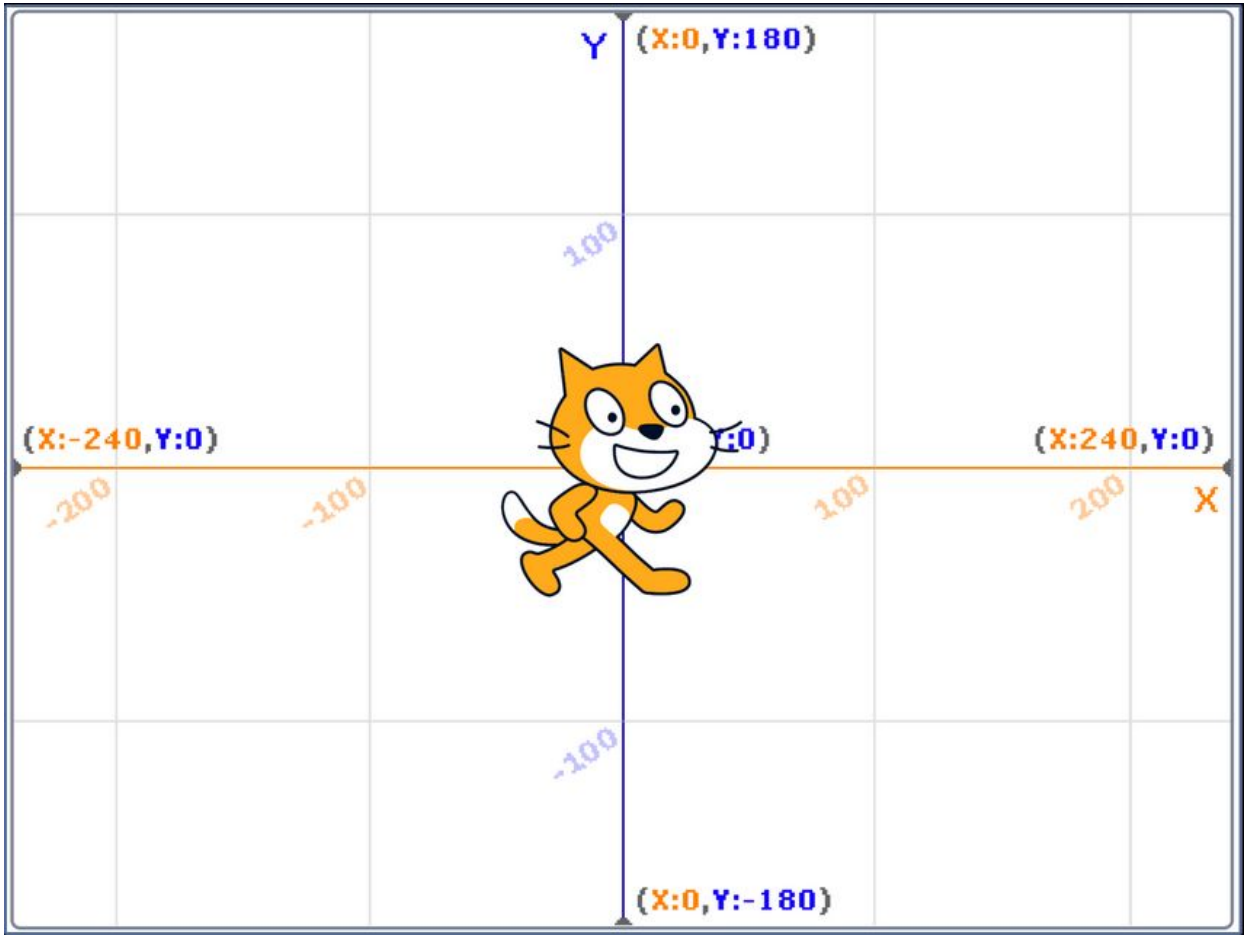
Área de seleção do sprite



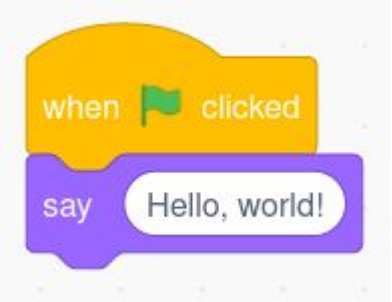
“Mundo” do Scratch

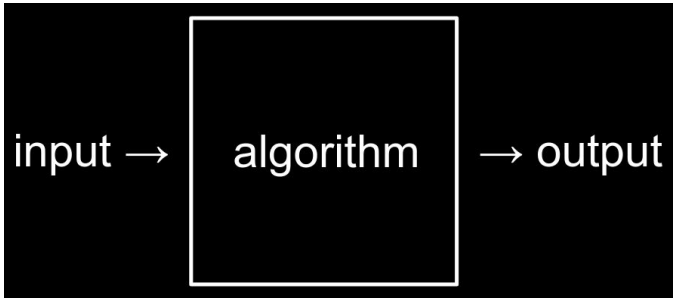


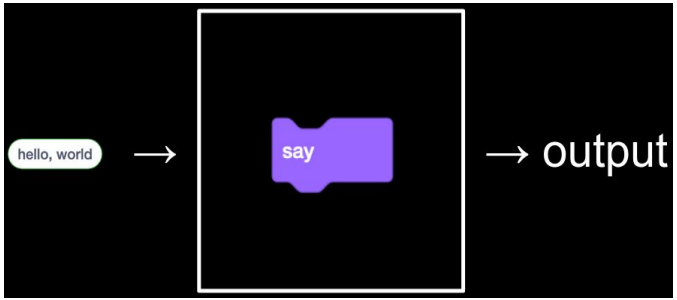
“Mundo” do Scratch

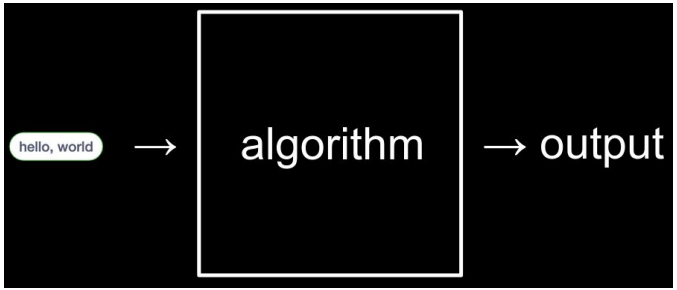


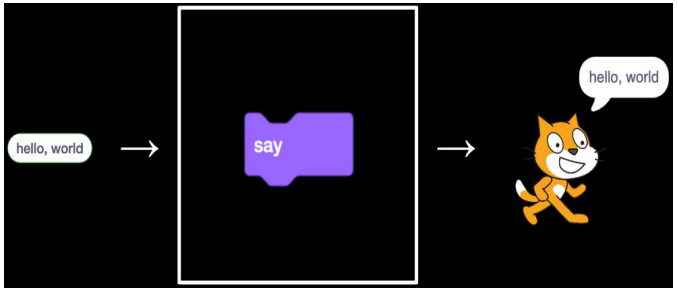
Exemplo: uma função simples



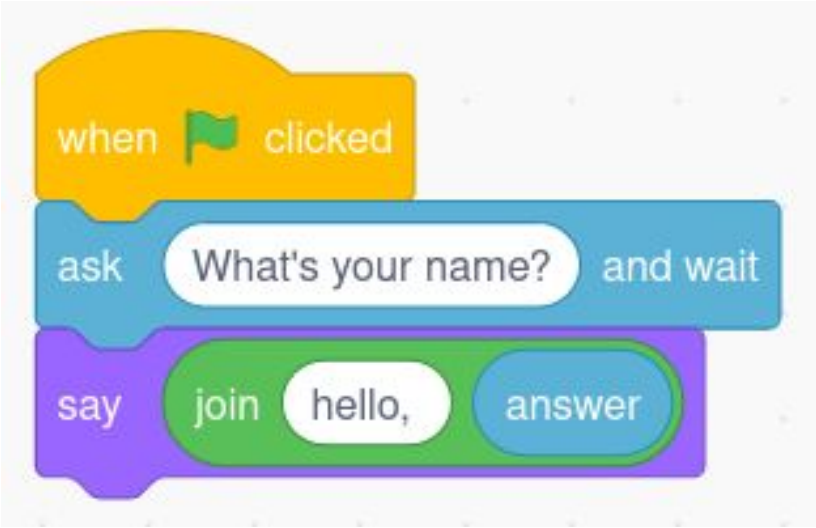




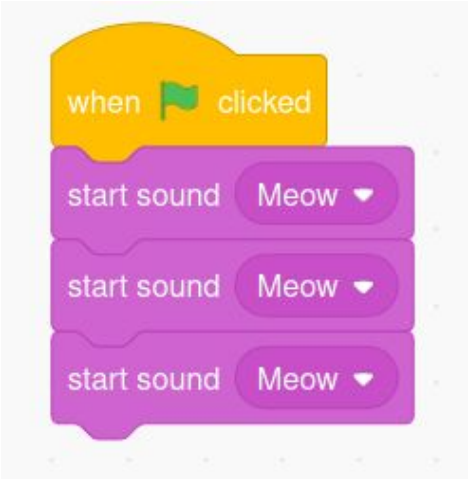




Exemplo: combinação de funções

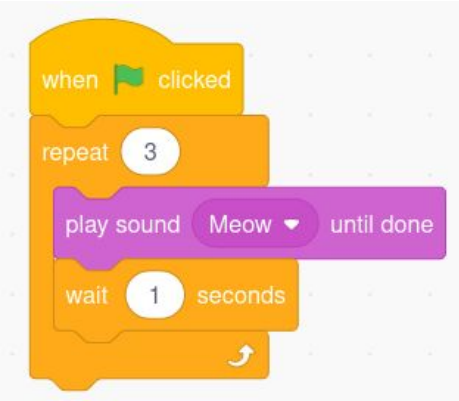


Exemplo: bug e má programação

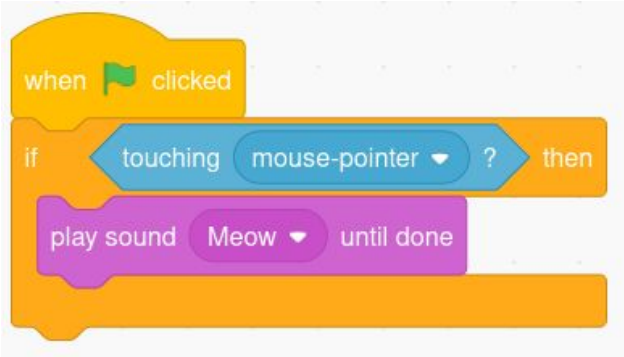


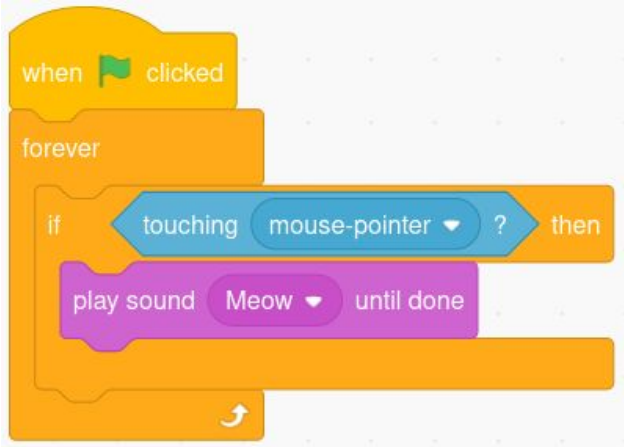
Correção!

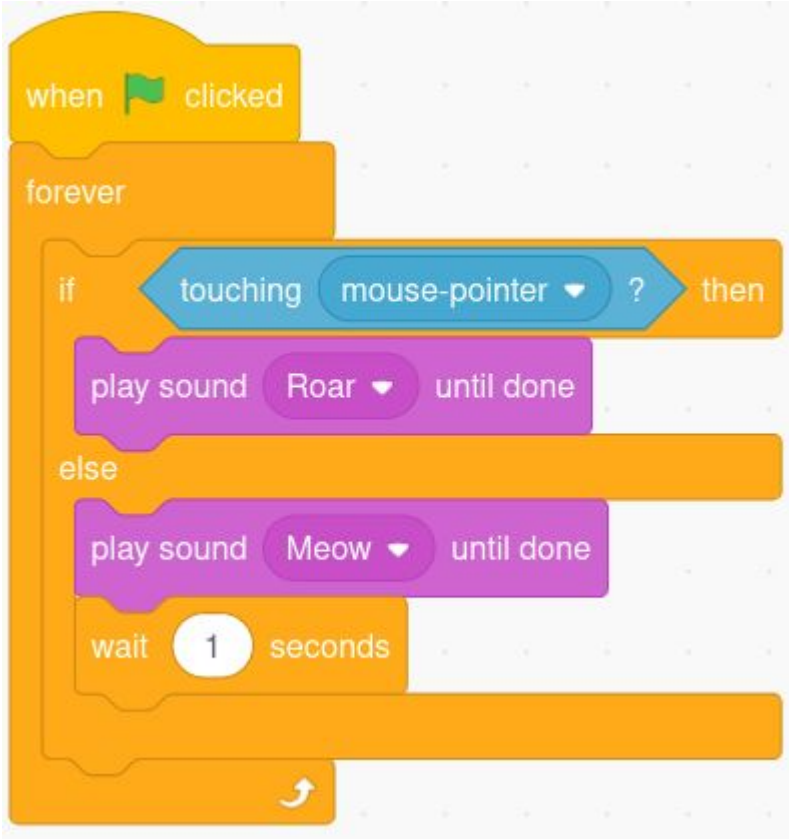
Melhoria!



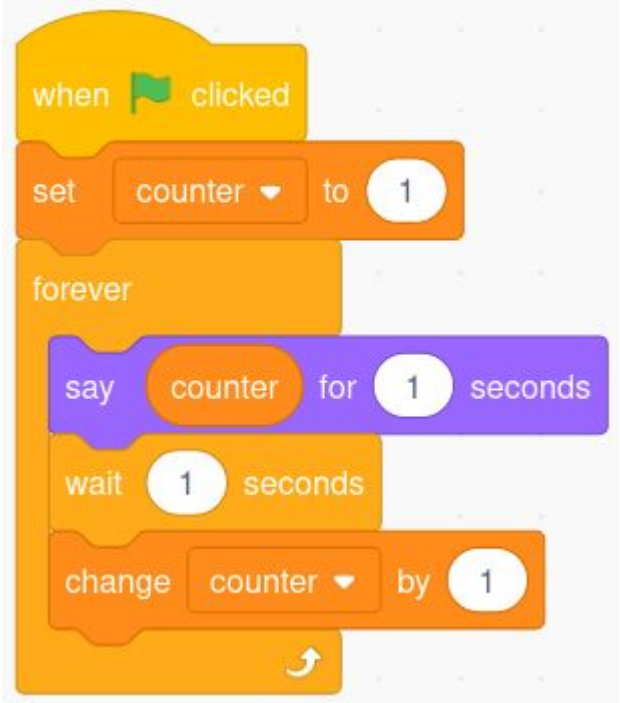
Exemplo: bug, correção, condicionais

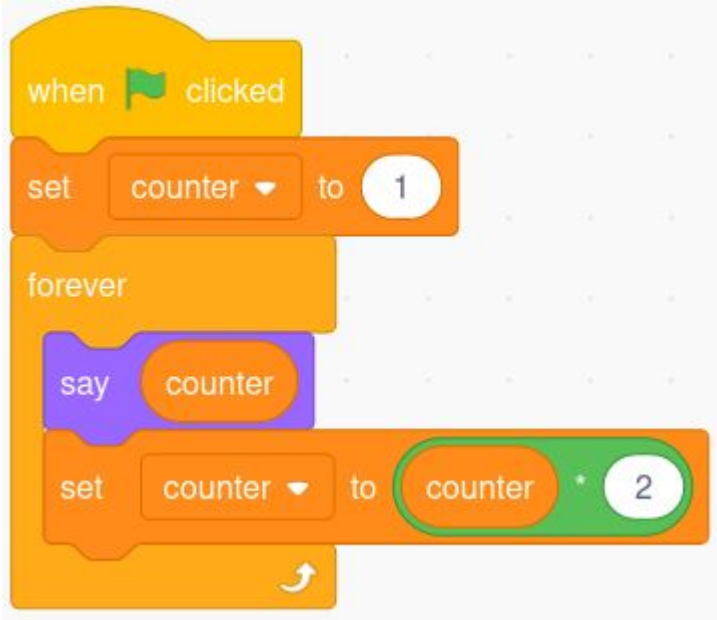




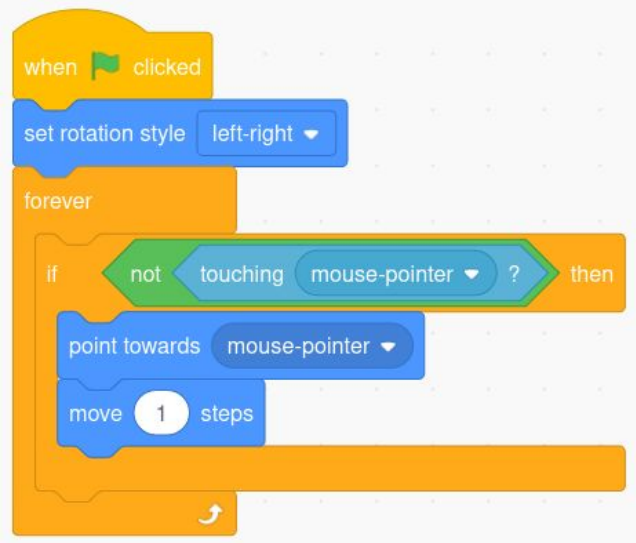


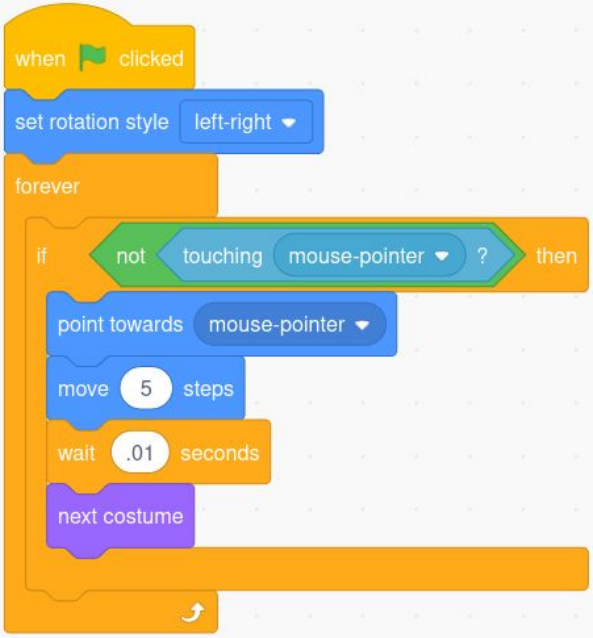
Exemplo: variáveis, overflow



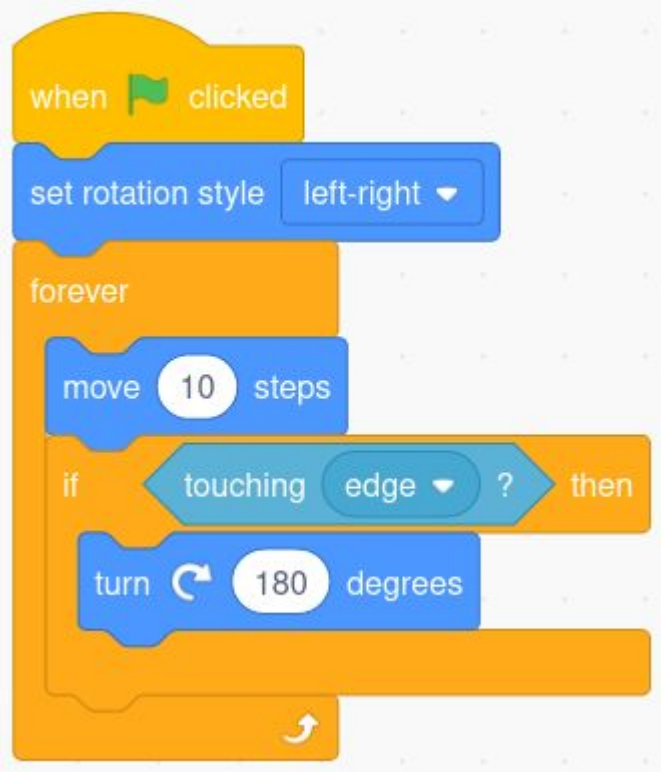


Exemplo: movimentação, “costumes”

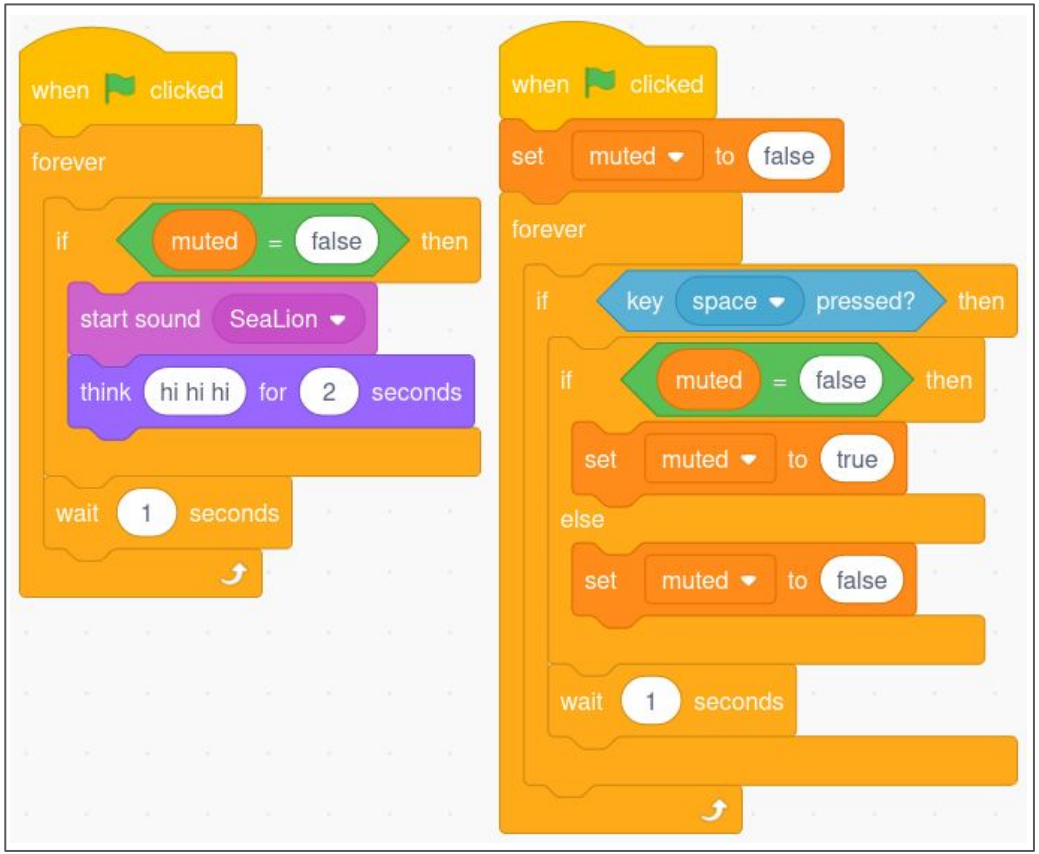




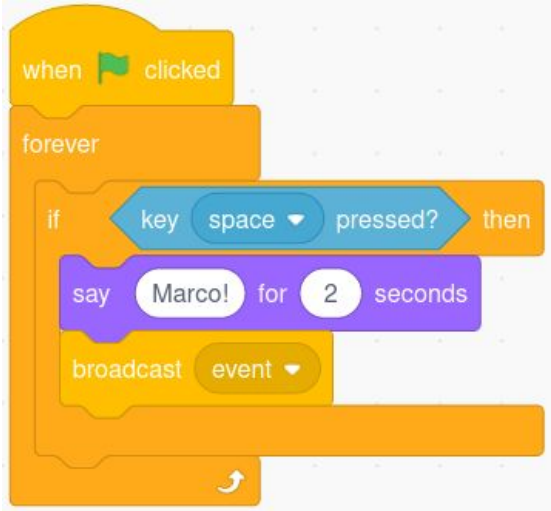
Exemplo: limites do mundo

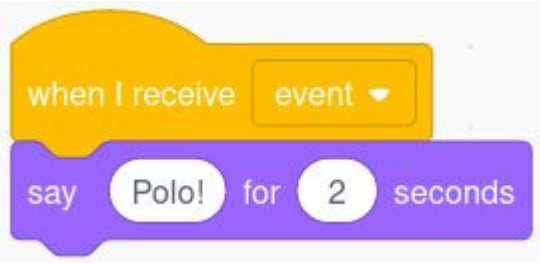


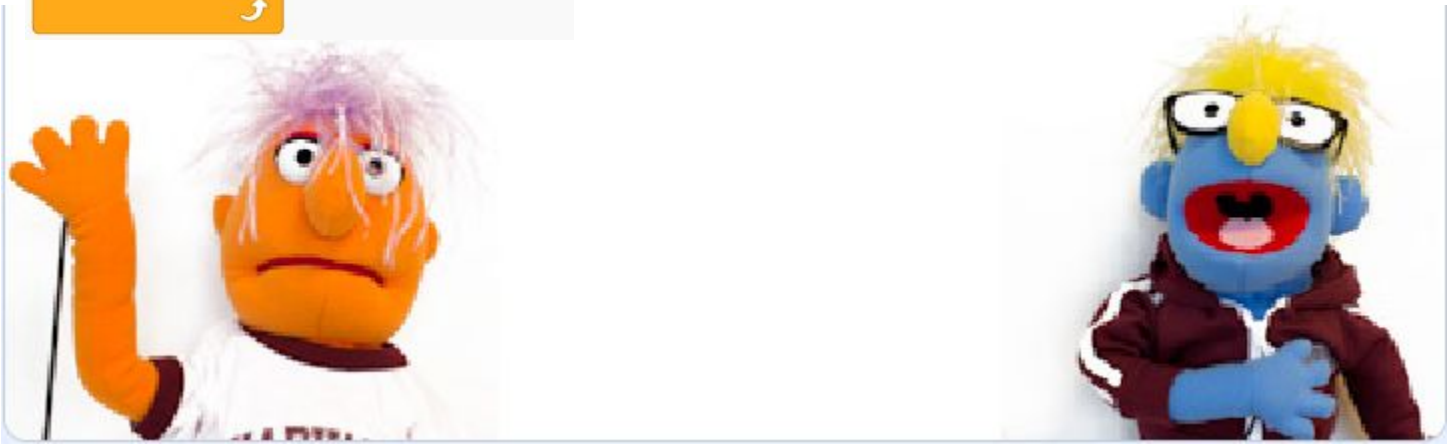
Exemplo: múltiplos programas simultâneos (threads)



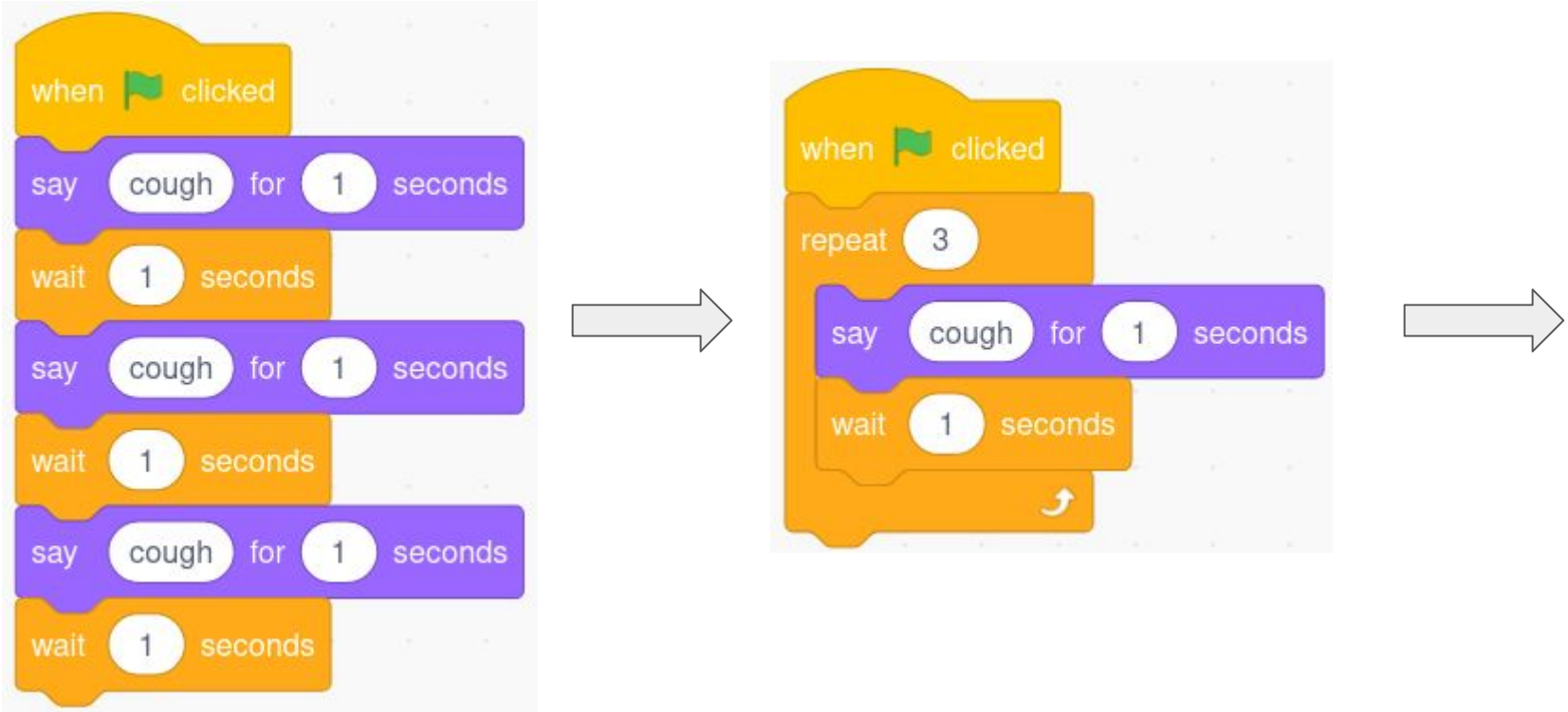
Exemplo: signals e listeners!



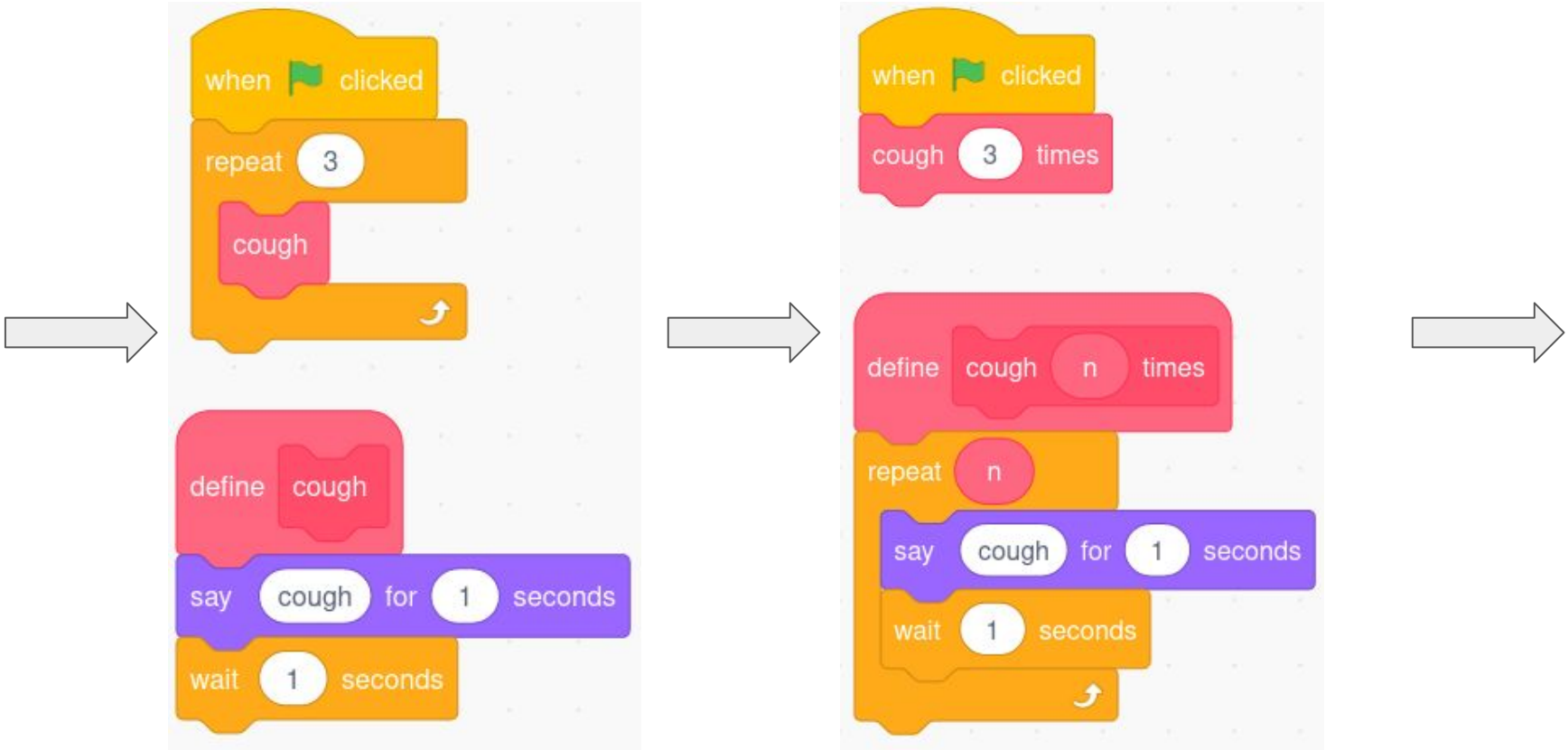




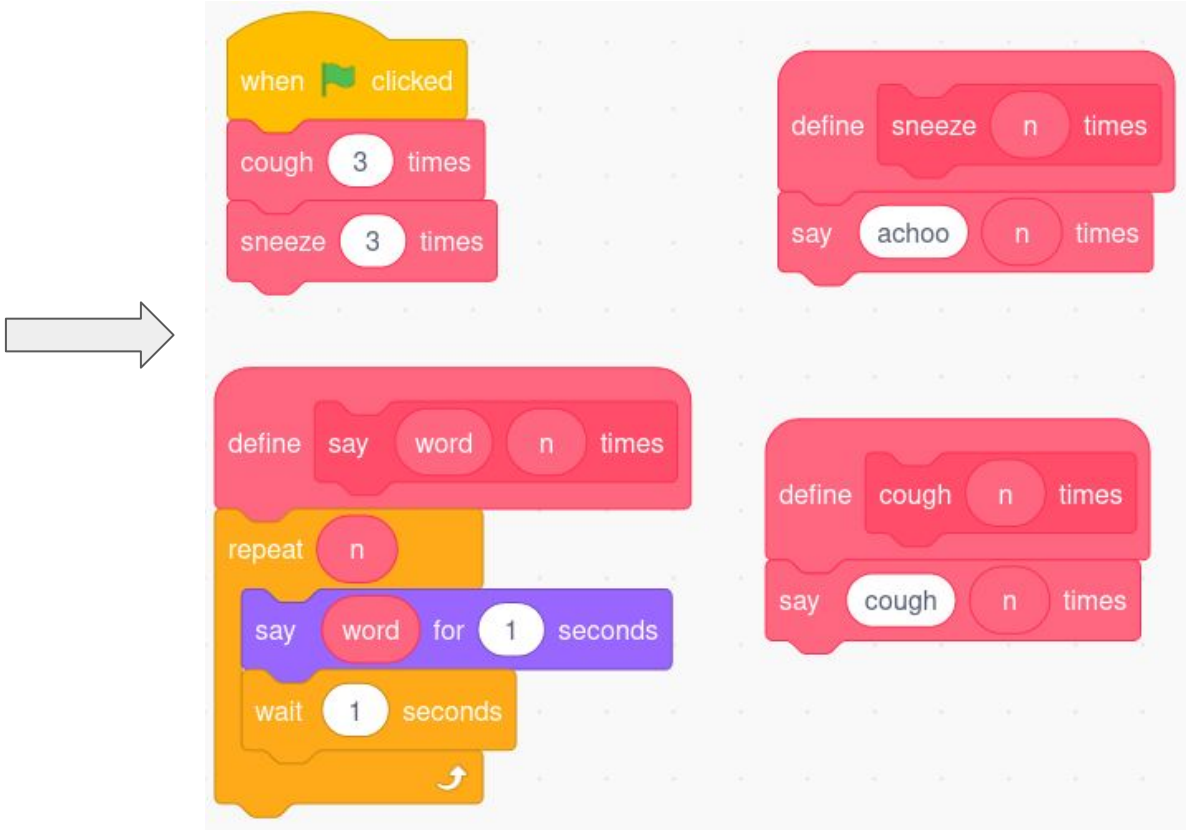
Exemplo: funções (suas próprias abstrações!)



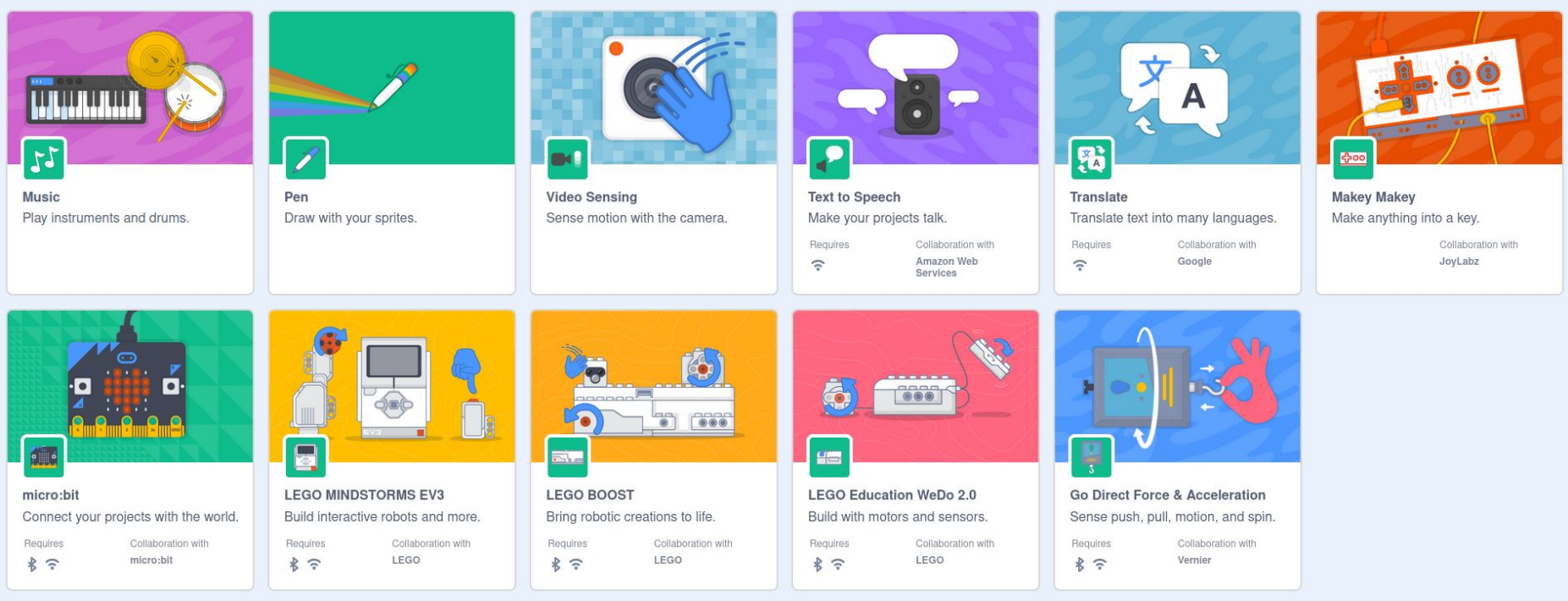
Exemplo: funções (suas próprias abstrações!)



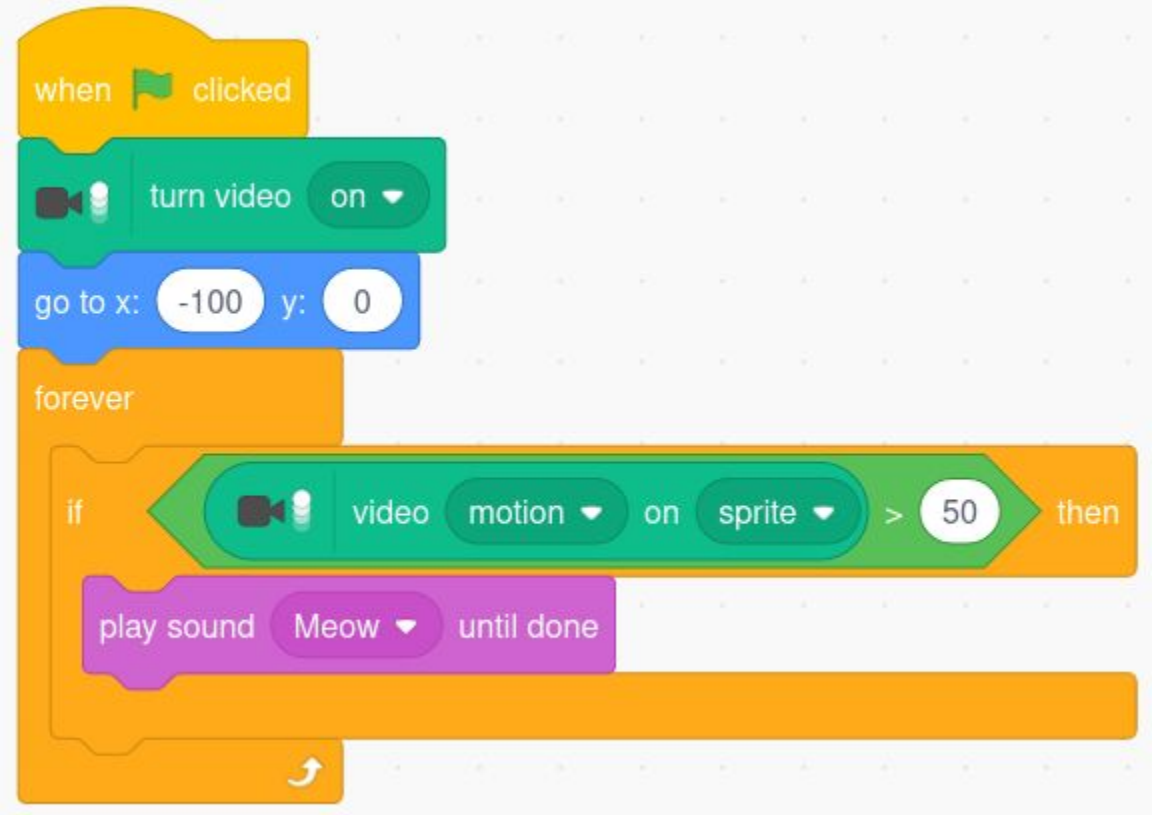
Exemplo: funções (suas próprias abstrações!)



Exemplo: blocos avançados

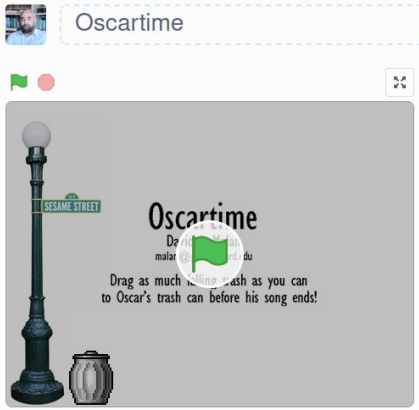


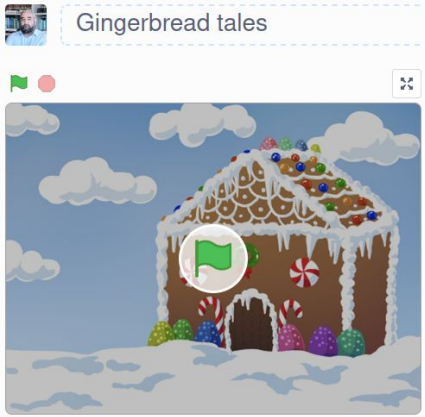
Exemplo: blocos avançados

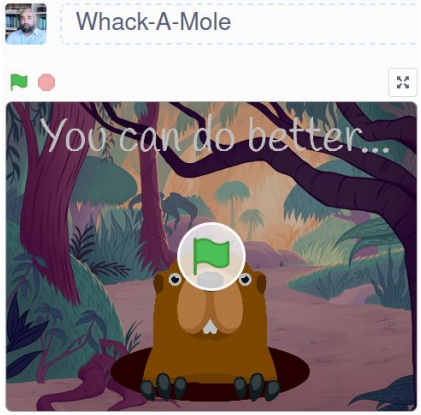


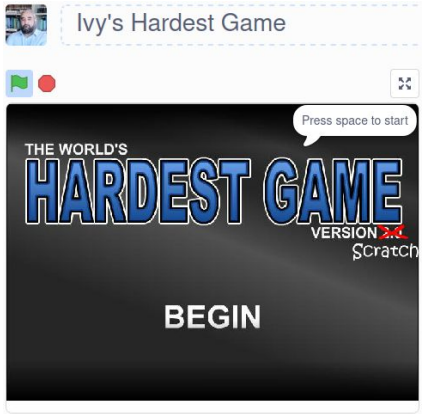
Exemplo: blocos avançados

Exemplos de outros alunos:



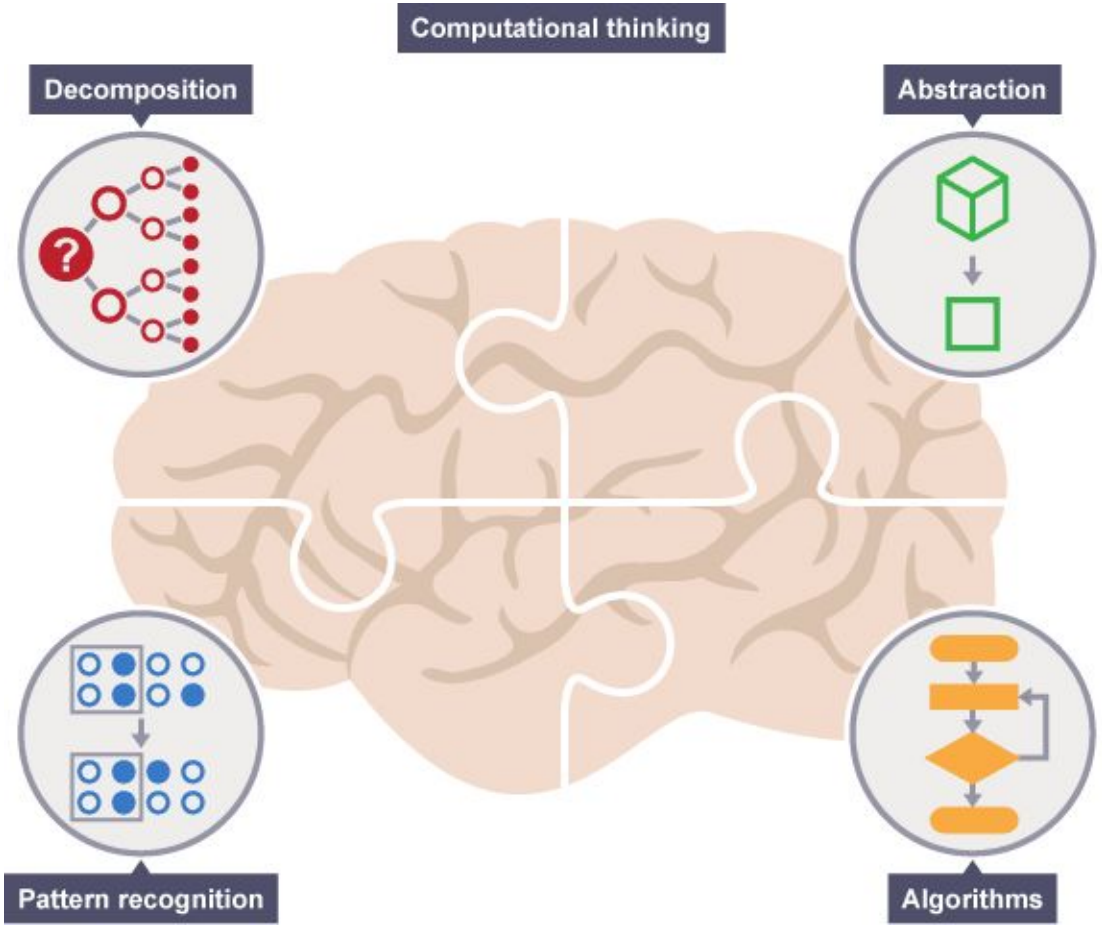






Resolvendo problemas:
PENSAMENTO COMPUTACIONAL

PENSAMENTO COMPUTACIONAL



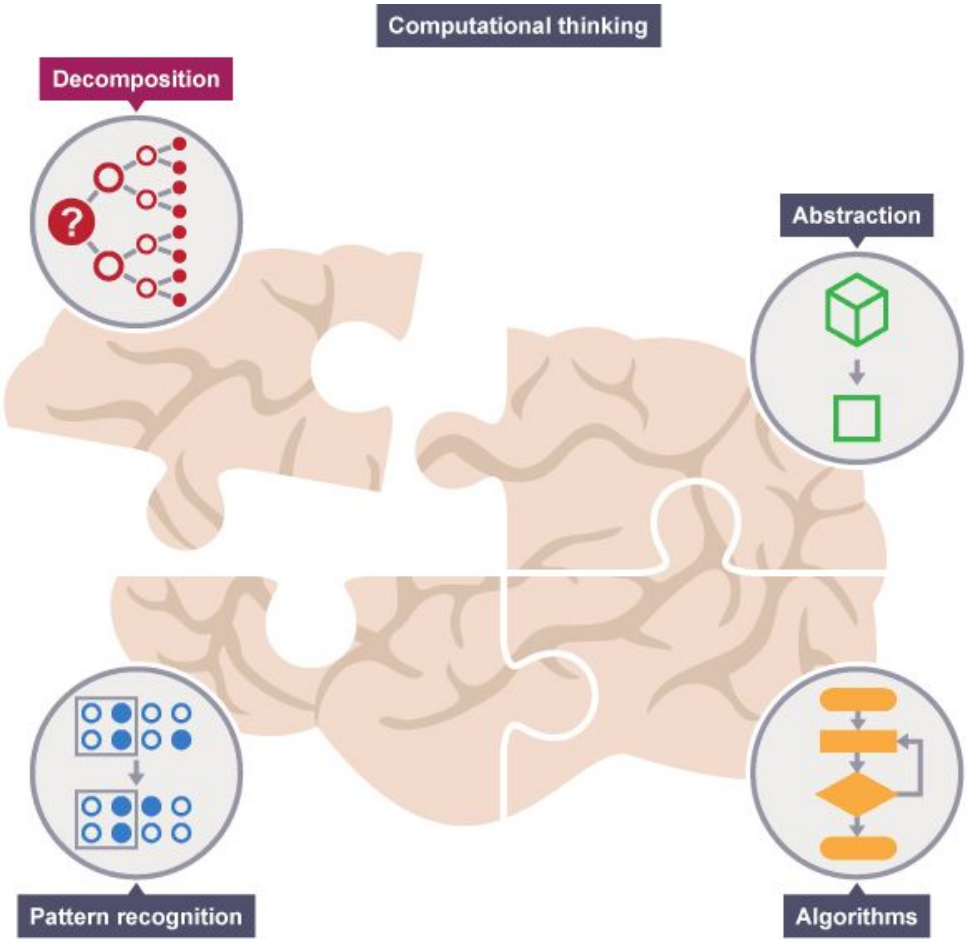
Nos permite pegar um problema complexo, compreender o que o problema é, projetar e implementar soluções.

Tem 4 componentes fundamentais:

- decomposição
- reconhecimento de padrões
- abstração
- algoritmos

https://www.bbc.co.uk/bitesize/topics/z7tp34j

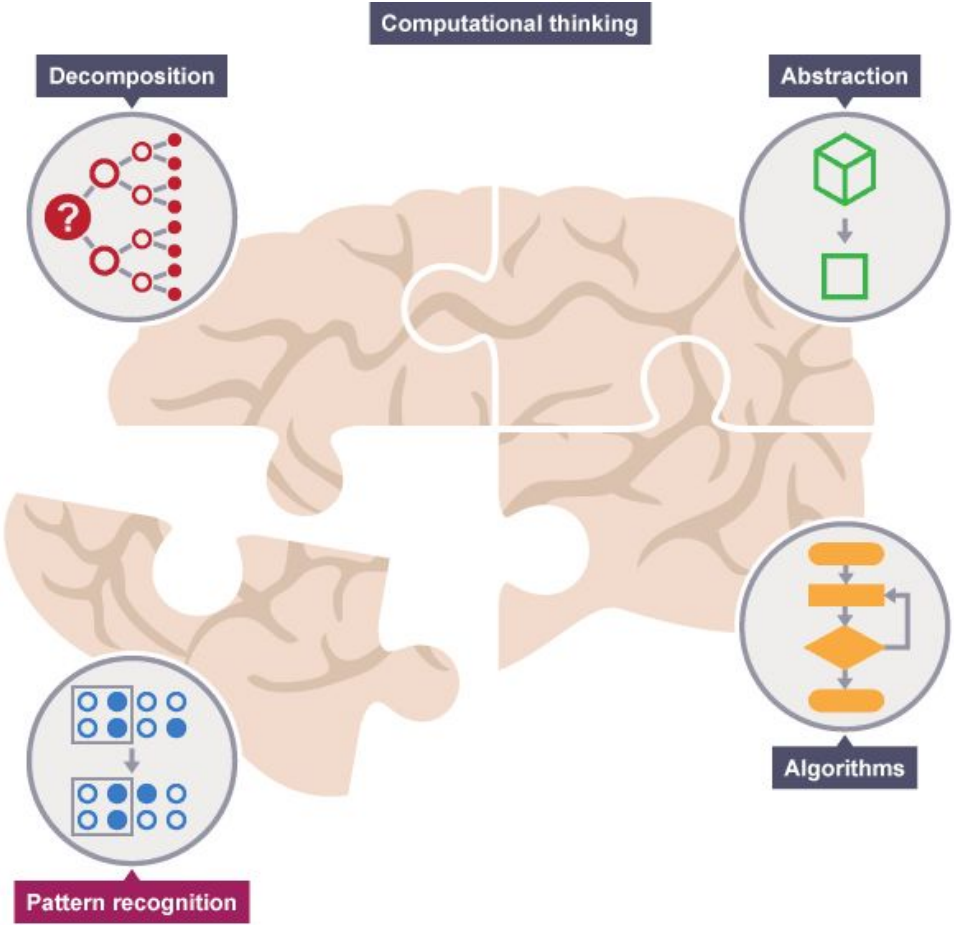
PENSAMENTO COMPUTACIONAL: DECOMPOSIÇÃO



Capacidade de pegar um problema complexo, aparentemente insolúvel, e quebrar (decompor) esse problema em diversas partes menores que podem ser resolvidas.

https://www.bbc.co.uk/bitesize/topics/z7tp34j

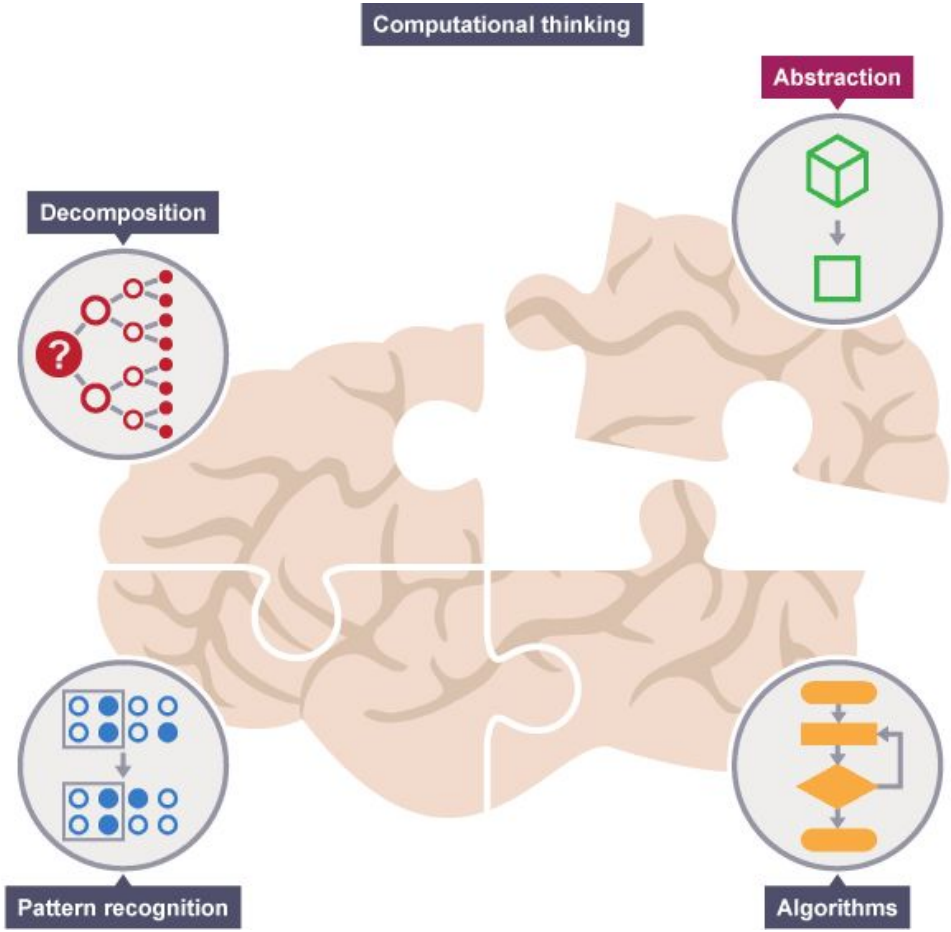
PENSAMENTO COMPUTACIONAL: ACHAR PADRÕES



Capacidade de encontrar padrões ou semelhanças entre os problemas decompostos, ou entre problemas que já foram solucionados antes. Isso nos permitirá resolver os problemas de modo mais eficaz, sem reinventar a roda.

https://www.bbc.co.uk/bitesize/topics/z7tp34j

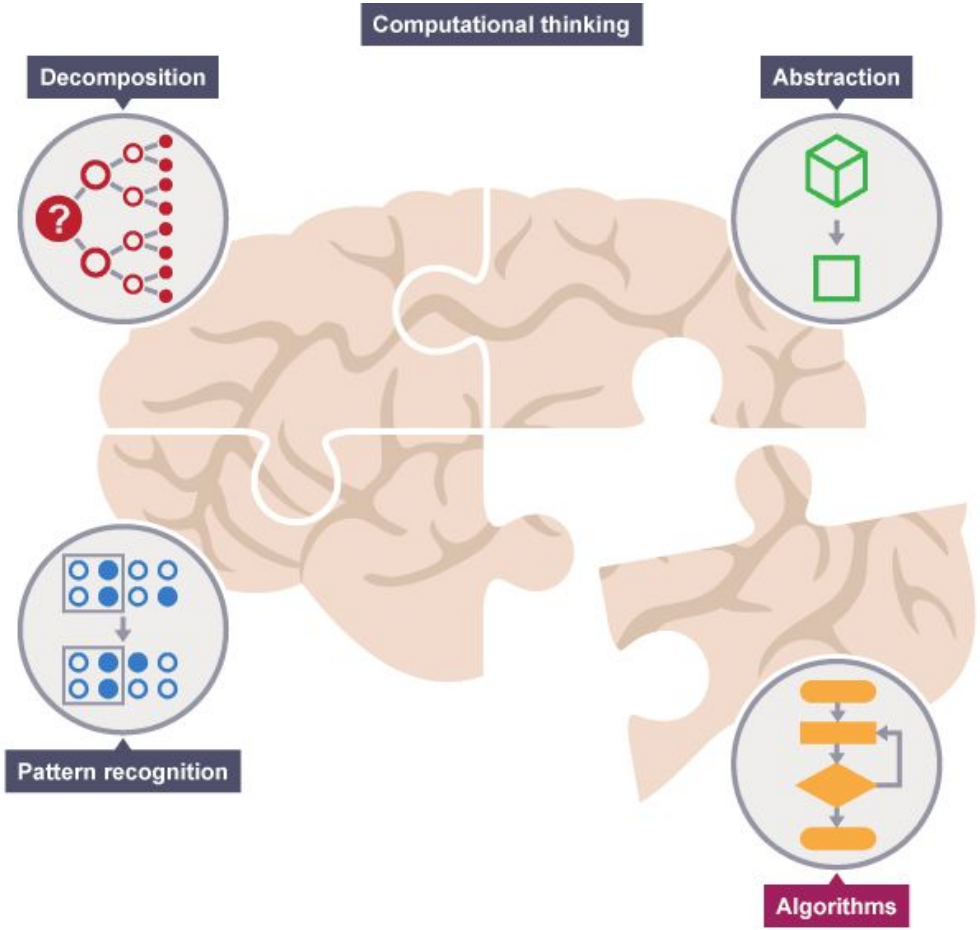
PENSAMENTO COMPUTACIONAL: ABSTRAÇÃO



Capacidade de focar no que é essencial, escondendo ou ignorando os detalhes não importantes, e de generalizar soluções.

https://www.bbc.co.uk/bitesize/topics/z7tp34j

PENSAMENTO COMPUTACIONAL: ALGORITMO



Capacidade de criar uma “receita” passo a passo que indica como resolver um problema. Essa receita pode ser escrita em pseudocódigo ou com um fluxograma para depois ser traduzida em uma linguagem de programação.

https://www.bbc.co.uk/bitesize/topics/z7tp34j

PENSAMENTO COMPUTACIONAL

Ivy's Hardest Game - Harvard 2

Last modified: 3 minutes ago

See inside

Add to

1 0
0 0
0 0

Unshare

Ivy's Hardest Game - Harvard 1

Last modified: 31 minutes ago

See inside

Add to

1 0
0 0
0 0

Unshare

Ivy's Hardest Game - Harvard 0

Last modified: 33 minutes ago

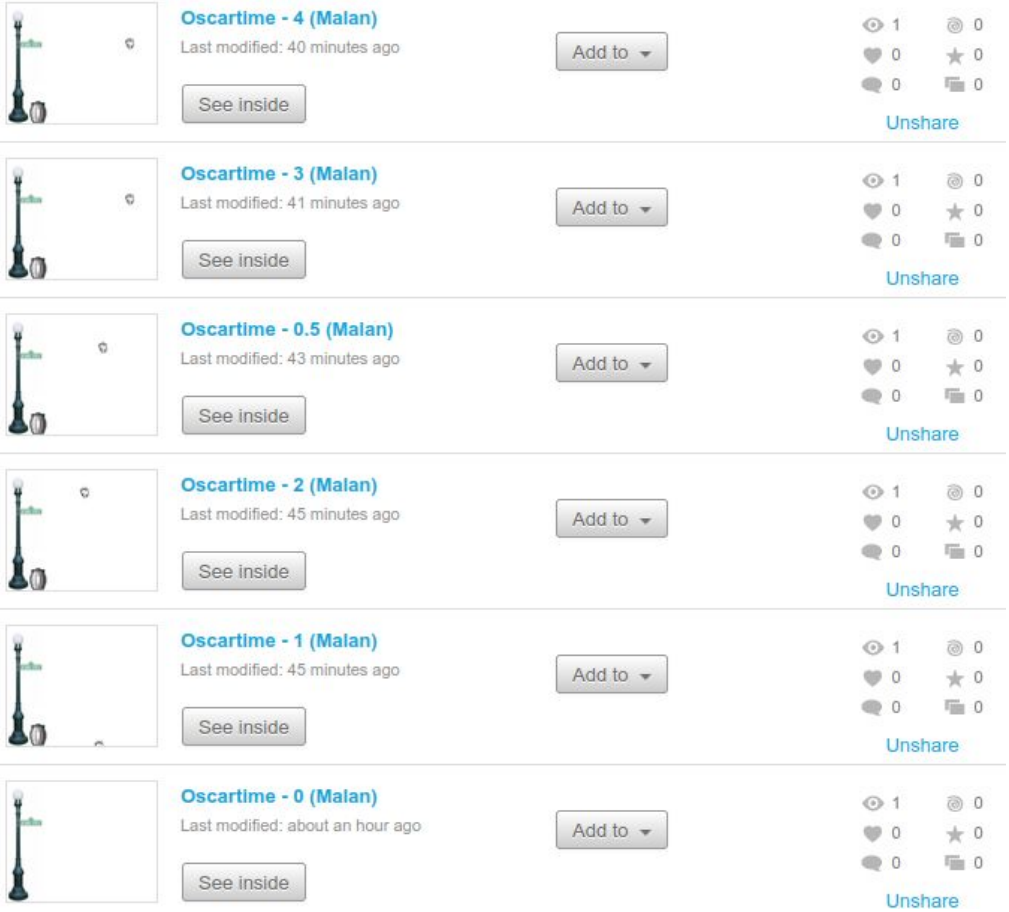
See inside

Add to

1 0
0 0
0 0

Unshare

PENSAMENTO COMPUTACIONAL



PSET 0

https://cs50.harvard.edu/x/2023/psets/0/scratch/

Crie um programa de sua própria imaginação em Scratch, sendo que:

- Deve usar pelo menos 2 sprites, e pelo menos um deles não deve ser um gato
- Deve ter, no mínimo, 3 scripts no total
- Deve usar, pelo menos:
  - 1 condição
  - 1 loop
  - 1 variável
- Deve ter, pelo menos, 1 bloco (função) que você mesmo criou. Esse bloco deve receber, pelo menos, 1 argumento (input)
- Deve ser mais complexo que os mostrados na aula, mas não precisam ser tão complexos como o Oscartime ou o Ivy's Harderst Game.

SIGA AS INSTRUÇÕES DE ENVIO para que os *autograders* de Hardard avaliem seu programa!

Créditos das Imagens

- **Homem com dúvidas**: Image by Tumisu from Pixabay
- **Cubo de Rubick**: Image by rubylia from Pixabay
- **Perdido entre caminhos**: Image by Maddy Mazur from Pixabay
- **Drink for Fire Hose**: http://hacks.mit.edu/Hacks/by_year/1991/fire_hydrant/
- **Pulando o penhasco**: Image by Waldryano from Pixabay
- **Não desista**: Image by Gerd Altmann from Pixabay
- **Balança**: Image by Mediamodifier from Pixabay
- **Ambiente do Scratch**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **VSCode**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Memória Crucial**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Bug**: https://www.computerhistory.org/tdih/september/9/
- **Gráfico de barras**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Arquitetura de memória**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Caixa de correio**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Stack Overvflow logo**: https://stackoverflow.com/
- **Hash table**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Hello word em C e Python**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Planilha e árvore**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Estrutura HTML**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Frosh IMs**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)

Créditos das Imagens

- **Confraternização/programação**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Mulher apontando para cima**: Image by Nebraska Department of Education from Pixabay
- **Marcas de contagem**: Image by L from Pixabay
- **Mãos**: Image by Alexander Lesnitsky from Pixabay
- **Lâmpadas ligadas e desligadas**: Image by OpenClipart-Vectors from Pixabay
- **Desktop, notebook, tablet, phone**: Image by José Augusto Camargo from Pixabay
- **Botão ligado/desligado**: Image by OpenClipart-Vectors from Pixabay
- **Transístor**: Image by WikimediaImages from Pixabay
- **Gráfico do número de transístores**: https://commons.wikimedia.org/wiki/File:Moore%27s_Law_Transistor_Count_1970-2020.png
- **Escola primária**: Image by David Mark from Pixabay
- **Números dourados**: Image by atevern07 from Pixabay
- **ASCII table**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Árvore no vidro quebrado**: Image by Bela Geletneky from Pixabay
- **Pedal do carro**: Notes from CS10 (https://cs10.org/sp23/)
- **Módulo ECM/Accelarator/Throttle**: Notes from CS10 (https://cs10.org/sp23/)
- **Mangueira de chuveiro**: https://produto.mercadolivre.com.br/MLB-1196303462-5-metros-mangueira-para-chuveiro-grande-mais-o-chuveirinho-_JM
- **Blocos do Snap!** (diversos): Notes from CS10 (https://cs10.org/sp23/)
- **Naked Blue IV**: Notes from CS10 (https://cs10.org/sp23/)
- **Mapas (satélite e terreno)**: https://maps.google.com

Créditos das Imagens

- **Mapa do metrô de SP**: https://www.metro.sp.gov.br/pdf/mapa-da-rede-metro.pdf
- **Generalization Example**: Notes from CS10 (https://cs10.org/sp23/)
- **Generalization in JBC**: Notes from CS10 (https://cs10.org/sp23/)
- **Power of Abstraction**: Notes from CS10 (https://cs10.org/sp23/)
- **Abstraction Summary**: Notes from CS10 (https://cs10.org/sp23/)
- **Ilustração da física ao software**: https://booksite.elsevier.com/9780128000564/index.php
- **Teclado americano, acentos do "a" e lista de emojis**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Unicode logo**: https://home.unicode.org
- **Emoji sorrindo com lágrimas**: https://home.unicode.org
- **Paisagem de montanhas em RGB**: https://commons.wikimedia.org/wiki/File:Barn_grand_tetons_rgb_separation.jpg
- **RGB Calculator**: https://www.w3schools.com/colors/colors_rgb.asp
- **Emoji ampliado**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Ilustração 60 x 30 FPS**: https://blog.emania.com.br/videos-em-24fps-30fps-e-60fps-qual-taxa-de-quadros-usar/
- **Lista de animojis**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Notas musicais**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Agenda de contatos do iPhone**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Lista telefônica**: https://www.mbi.com.br/mbi/produtos/listas/telefonia-cidades-brasil/
- **Gráficos de complexidade de algoritmos**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Detalhes das áreas do Scratch**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)
- **Blocos do Scratch (diversos)**: Notes from CS50x (https://cs50.harvard.edu/x/2023/)

Créditos das Imagens

- **Figuras do pensamento computacional (diversas)**: https://www.bbc.co.uk/bitesize/topics/z7tp34j
- **We can do hard things**: Photo by Jess Zoerb on Unsplash

Este texto tem finalidade acadêmica e educacional, para o ensino de ciência da computação, sem fins lucrativos. Procurei ser o mais criterioso e detalhado possível ao citar as fontes de todas as imagens. Se alguma das imagens estiver com a citação incorreta e/ou não deveria ter sido utilizada, por favor entre em contato através do e-mail abrantesasf@uvv.br para que eu possa fazer a correção e/ou a remoção da imagem.

This text has an non-profit, academic and educational purpose, for the teaching of computer science. I have tried to be as thorough and detailed as possible when citing the sources of all images. If any of the images are incorrectly cited and/or should not have been used, please contact me via email abrantesasf@uvv.br so that I can correct and/or remove the image.

Ilustrações, gráficos, tabelas, fotos e imagens que não foram citadas nos créditos, foram feitas pelo autor.