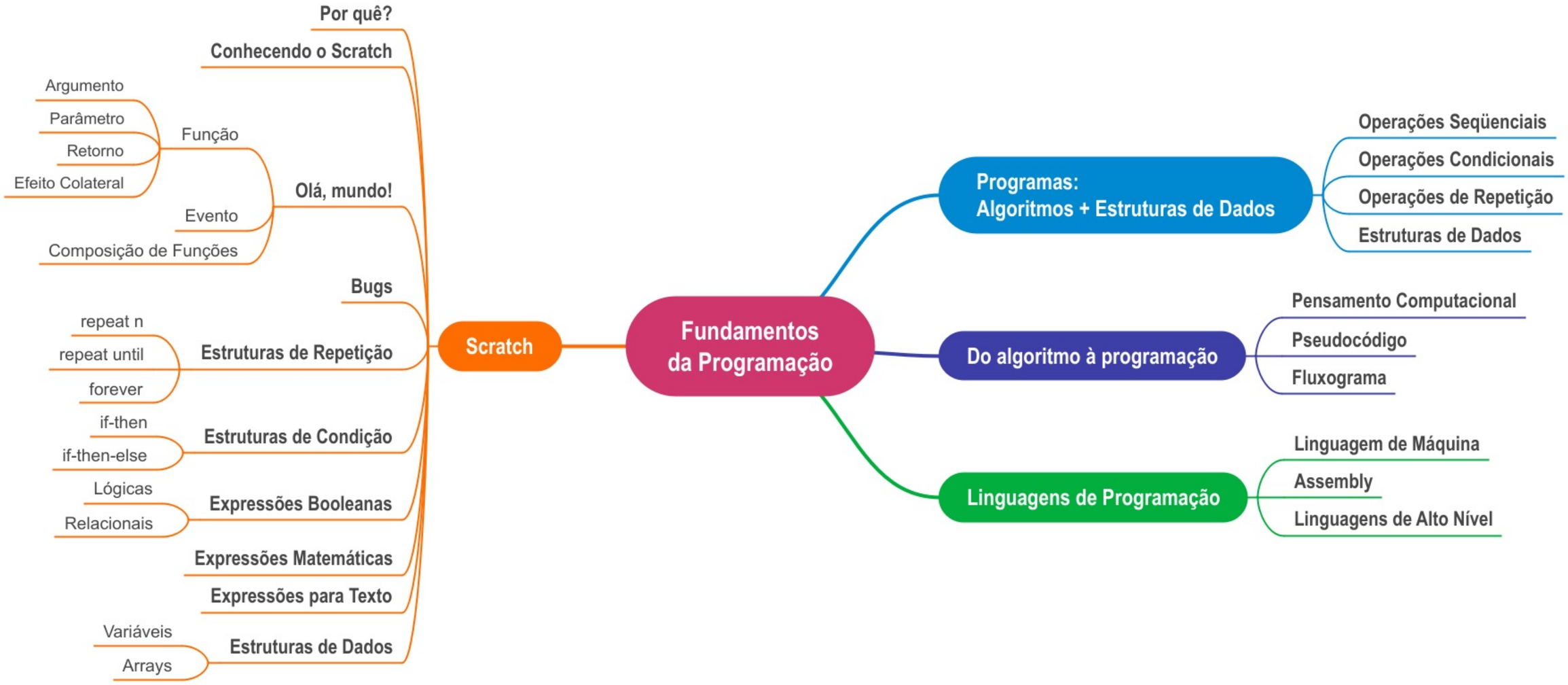
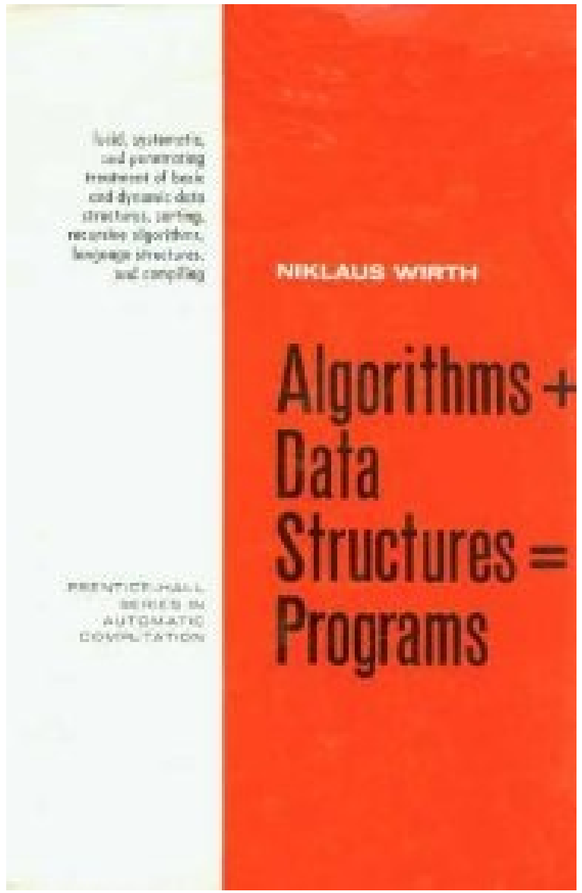
FUNDAMENTOS DA PROGRAMAÇÃO

Fundamentos da programção



Programas = Algoritmos + Estruturas de Dados

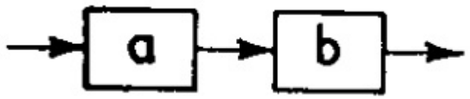
Programas = Algoritmos + Estruturas de Dados



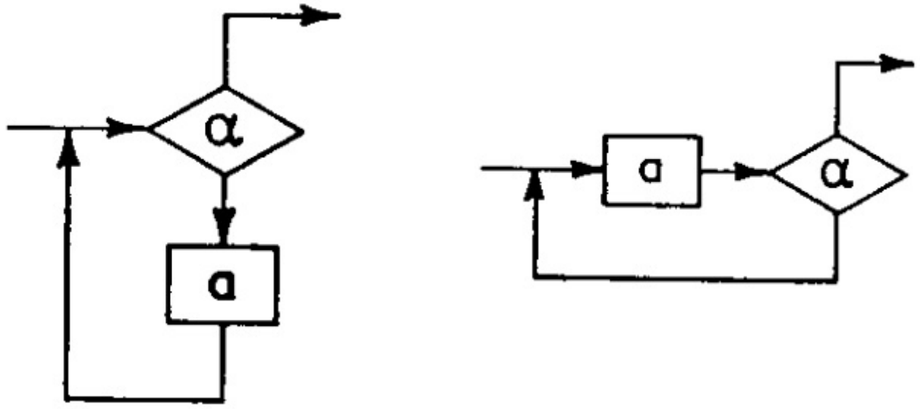
Adaptado de: Tyomitch, na Wikipedia
(https://en.wikipedia.org/wiki/File:Niklaus_Wirth,_UrGU.jpg)

Algoritmos: três operações fundamentais

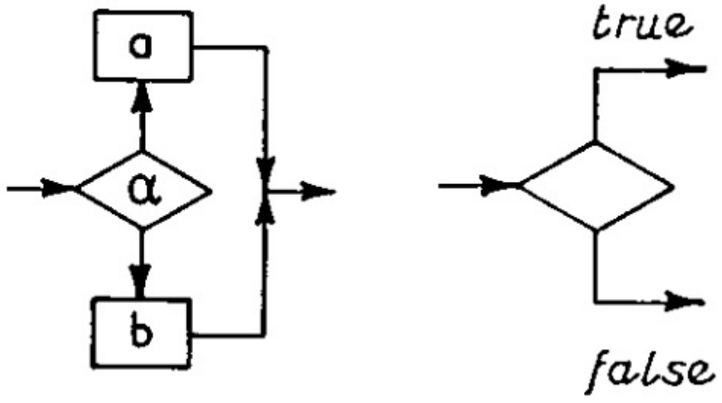
Operações seqüenciais



Operações de repetição

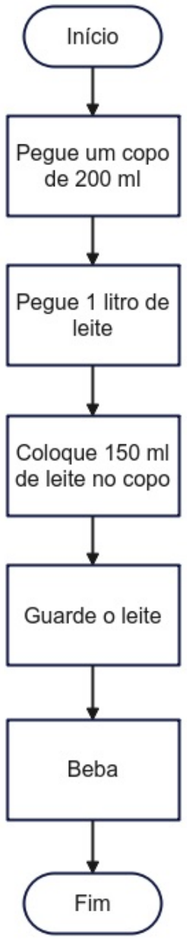


Operações condicionais

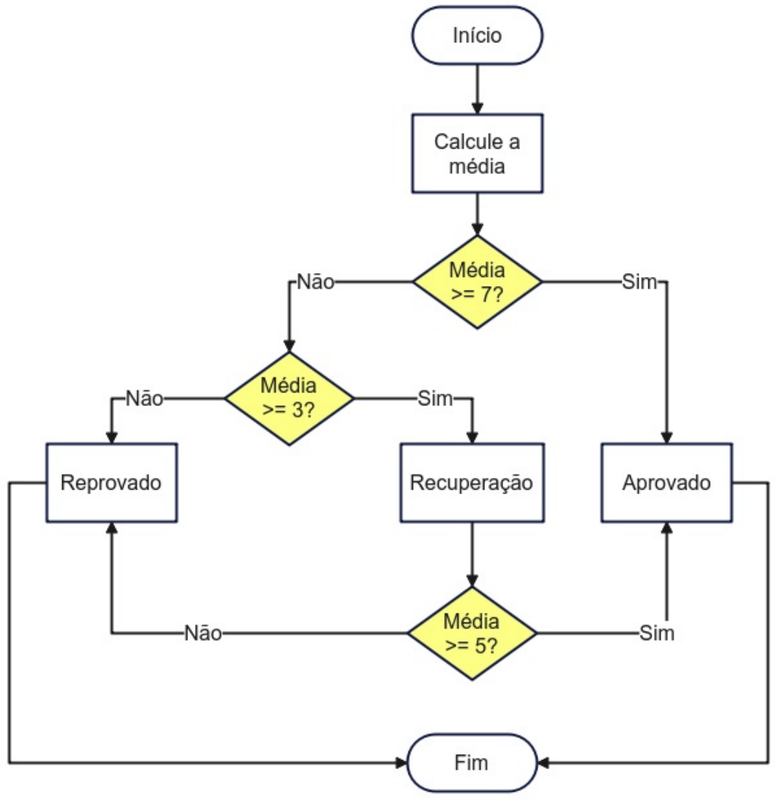


Algoritmos: três operações fundamentais

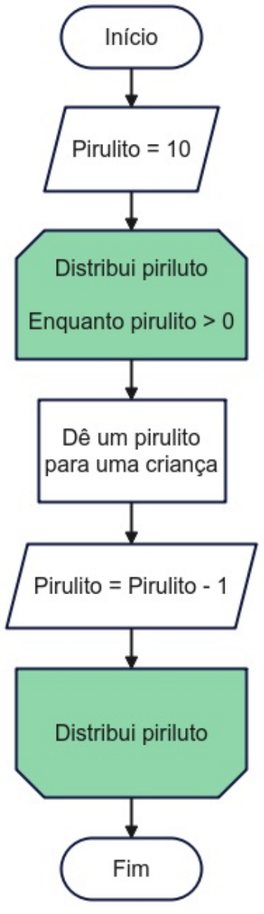
Operações seqüenciais



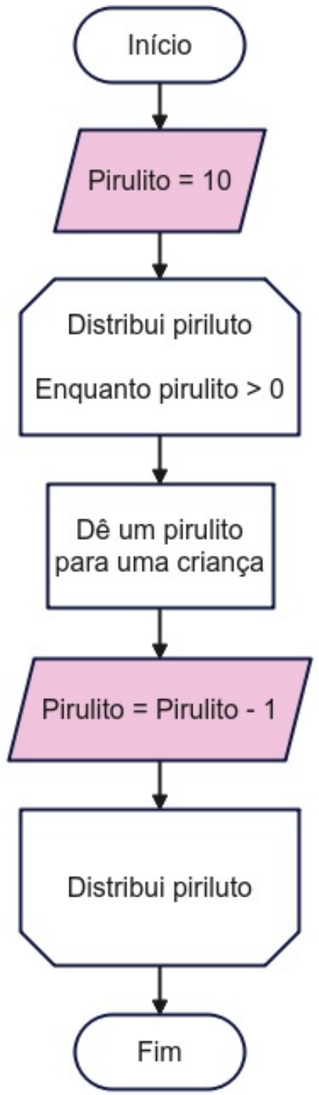
Operações condicionais



Operações de repetição



Estruturas de dados: armazenam dados em processamento



Do algoritmo à programação

Do algoritmo à programação

Codifique o algoritmo em um programa, utilizando qualquer linguagem, através dos fundamentos da programação:

- Ações
- Condições (expressões booleanas)
- Repetições
- Estruturas de dados
    (variáveis, arrays, outras)
- Abstrações diversas:
    Funções
        (parâmetros e argumentos)
        (retorno e efeito colateral)
- Avançados
    (eventos, ouvintes)
    (threads, outros)
- Boas práticas

Problema

Use o pensamento computacional para criar um algoritmo que solucione seu problema:

- pseudocódigo
- fluxograma
- qualquer coisa

TIRE A MÃO DO TECLADO!

Adaptado de Charlie Harris, no Unsplash (https://unsplash.com/photos/a-person-typing-on-a-keyboard-lXGoRVlV7u4)

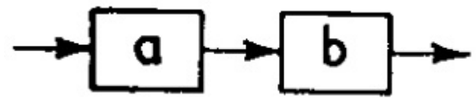
Do algoritmo à programação

Algoritmo de busca binária de um nome em uma lista telefônica, expresso em pseudocódigo:

```
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
```

Do algoritmo à programação

AÇÕES: operações seqüenciais bem definidas a serem executadas pelo computador.
Podem ser abstraídas em FUNÇÕES.

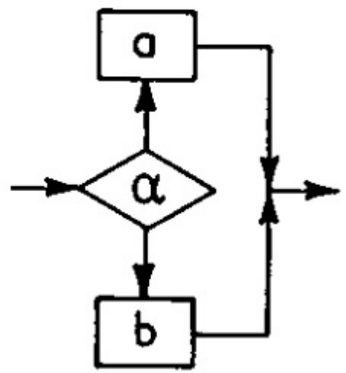



```
1    Pegue a lista telefônica
2    Abra na página do meio
3    Procure o nome na página
4    Se a pessoa procurada está na página
5       Ligue para a pessoa e vá para a linha 13
6    Ou se a pessoa está na parte anterior da lista:
7       Abra na página do meio da metade esquerda da lista
8       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

Do algoritmo à programação

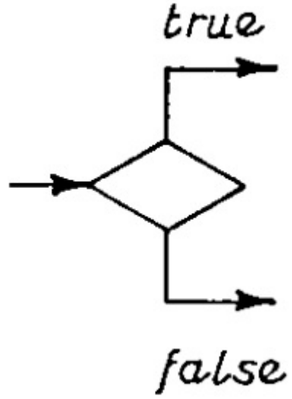
CONDIÇÕES: permitem decidir entre dois ou mais caminhos de ações.

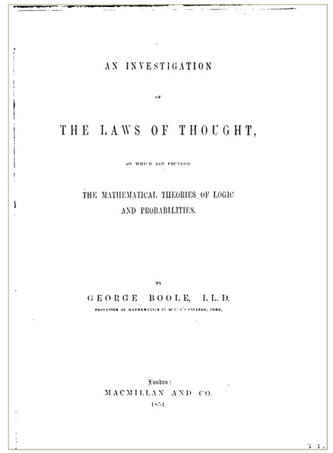


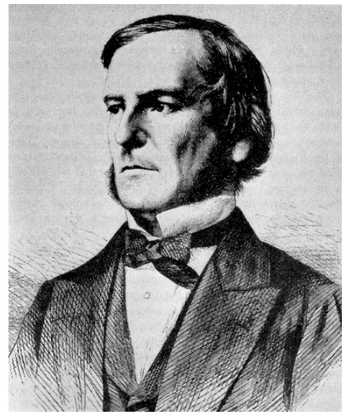

```
1    Pegue a lista telefônica
2    Abra na página do meio
3    Procure o nome na página
4    Se a pessoa procurada está na página
5       Ligue para a pessoa e vá para a linha 13
6    Ou se a pessoa está na parte anterior da lista:
7       Abra na página do meio da metade esquerda da lista
8       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
```

Do algoritmo à programação

CONDIÇÕES: permitem decidir entre dois ou mais caminhos de ações.
Os caminhos dependem do resultado de EXPRESSÕES BOOLEANAS.




AN INVESTIGATION
OF
THE LAWS OF THOUGHT,
ON WHICH ARE FOUNDED
THE MATHEMATICAL THEORIES OF LOGIC
AND PROBABILITIES.
BY
GEORGE BOOLE, LL.D.
PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK.
London:
MACMILLAN AND CO.
1854.

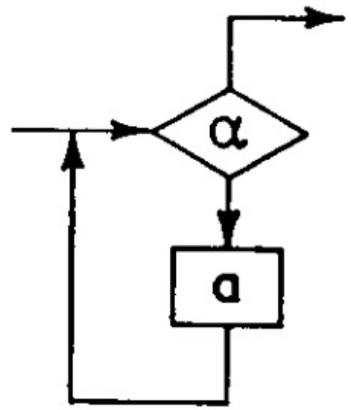

Haks, na Wikimedia
(https://commons.wikimedia.org/wiki/File:George_Boole.jpg)

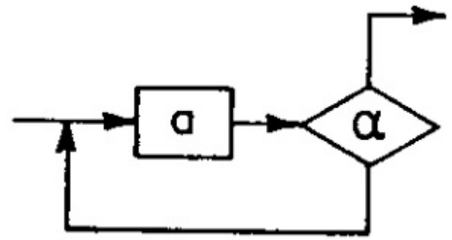
```
1    Pegue a lista telefônica
2    Abra na página do meio
3    Procure o nome na página
4    Se a pessoa procurada está na página
5       Ligue para a pessoa e vá para a linha 13
6    Ou se a pessoa está na parte anterior da lista:
7       Abra na página do meio da metade esquerda da lista
8       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

Do algoritmo à programação

REPETIÇÕES: permitem repetir trechos de código zero, uma ou mais vezes.



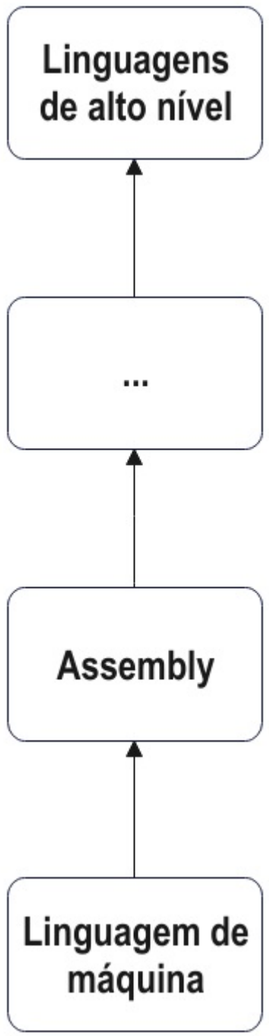



```
1    Pegue a lista telefônica
2    Abra na página do meio
3    Procure o nome na página
4    Se a pessoa procurada está na página
5       Ligue para a pessoa e vá para a linha 13
6    Ou se a pessoa está na parte anterior da lista:
7       Abra na página do meio da metade esquerda da lista
8       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

Linguagens de programação codificam o algoritmo: baixo e alto nível

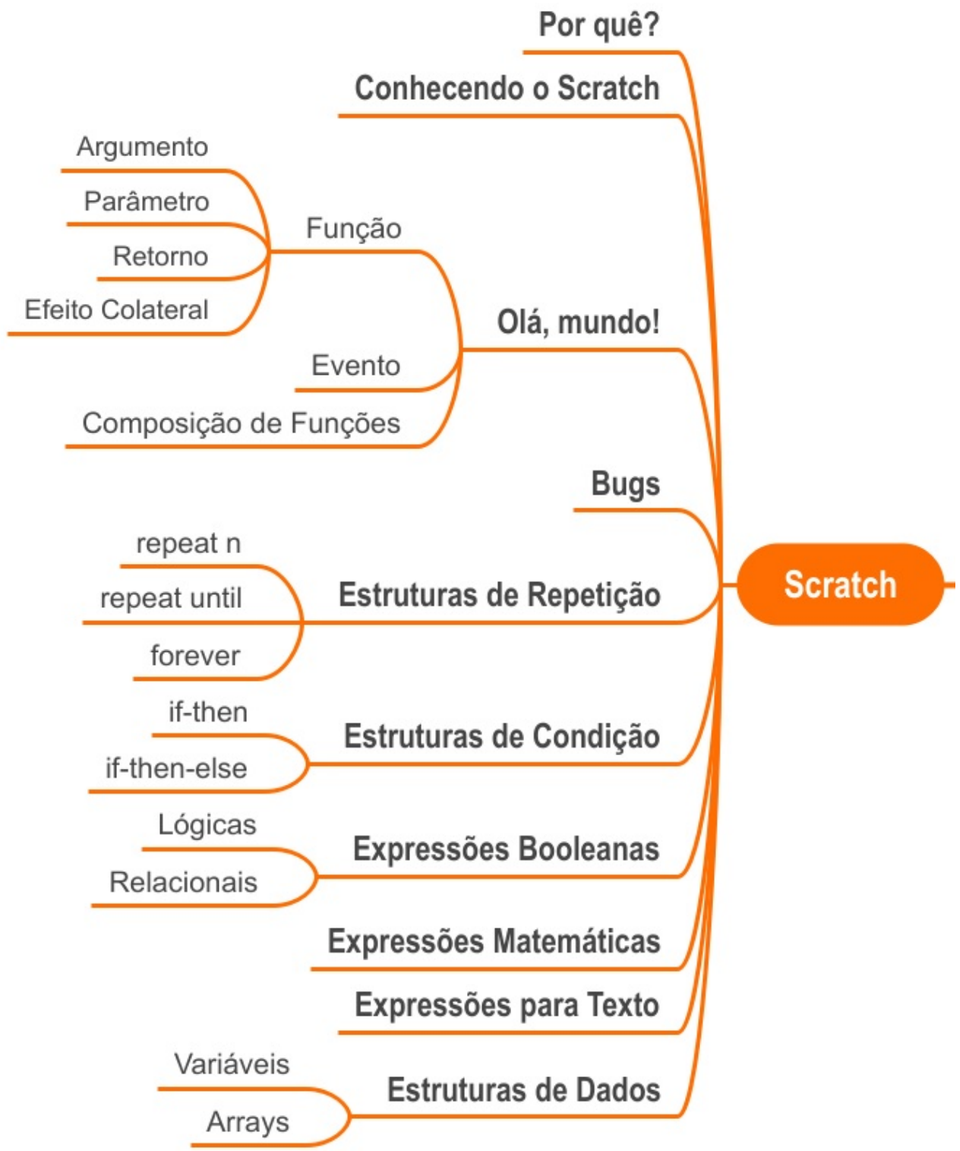
Linguagens de programação codificam o algoritmo: baixo e alto nível



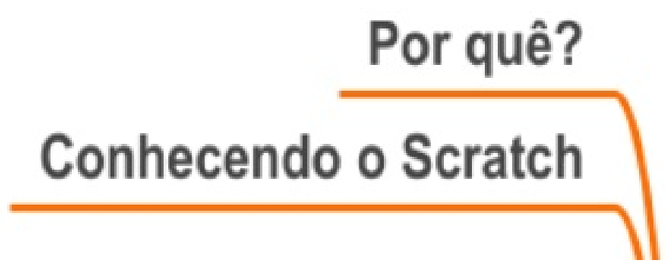
```
        .file   "ola.c"
        .text
        .section        .rodata
.LC0:
        .string "Ol\303\241, mundo!"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        call    puts@PLT
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0"
        .section        .note.GNU-stack,"",@progbits
        .section        .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
```

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("%s\n", "Olá, mundo!");
    return 0;
}
```

Scratch


Scratch
Por quê?
Conhecendo o Scratch
Olá, mundo!
Função
Argumento
Parâmetro
Retorno
Efeito Colateral
Evento
Composição de Funções
Bugs
Estruturas de Repetição
repeat n
repeat until
forever
Estruturas de Condição
if-then
if-then-else
Expressões Booleanas
Lógicas
Relacionais
Expressões Matemáticas
Expressões para Texto
Estruturas de Dados
Variáveis
Arrays

Scratch



Scratch: por quê?

- Linguagem visual de programação.

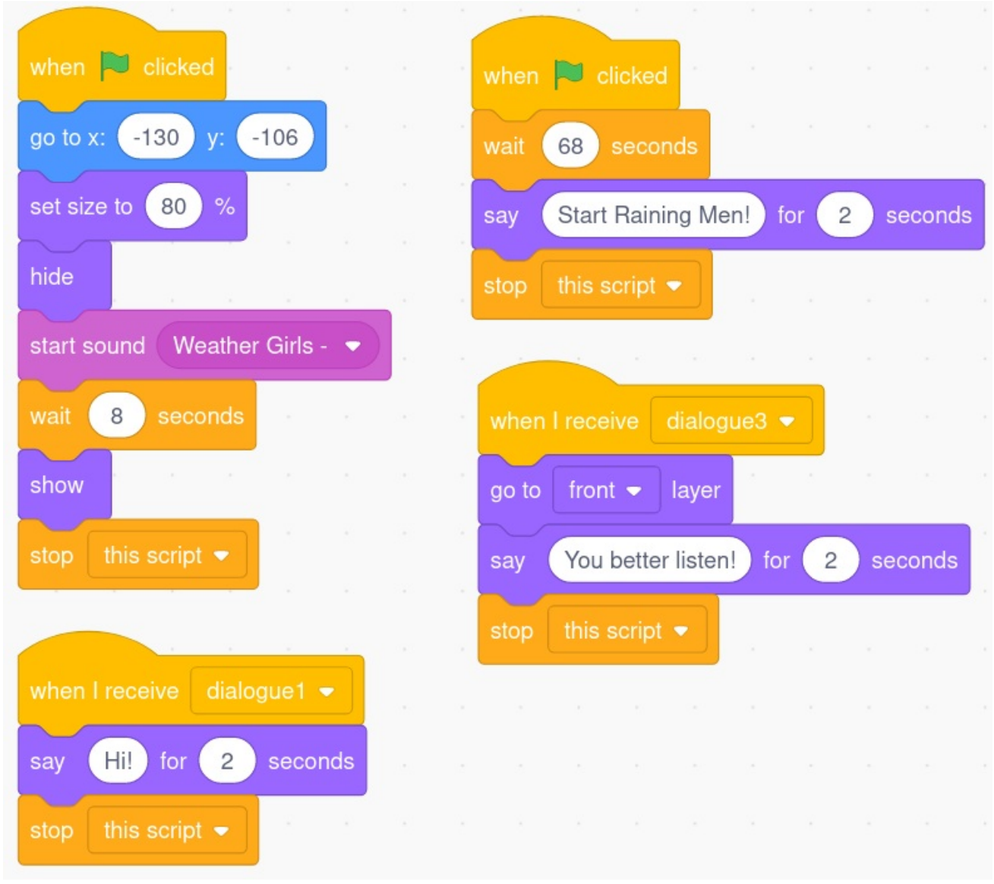
- Aprender os fundamentos da programação sem ficar preocupado com sintaxes difíceis.

- Criar animações, jogos, arte, software, etc., simplesmente arrastando peças (blocos).

- Não se engane: Scratch é uma verdadeira linguagem de programação.

- Alternativa:

  Snap*!* (mais avançada, também usaremos)

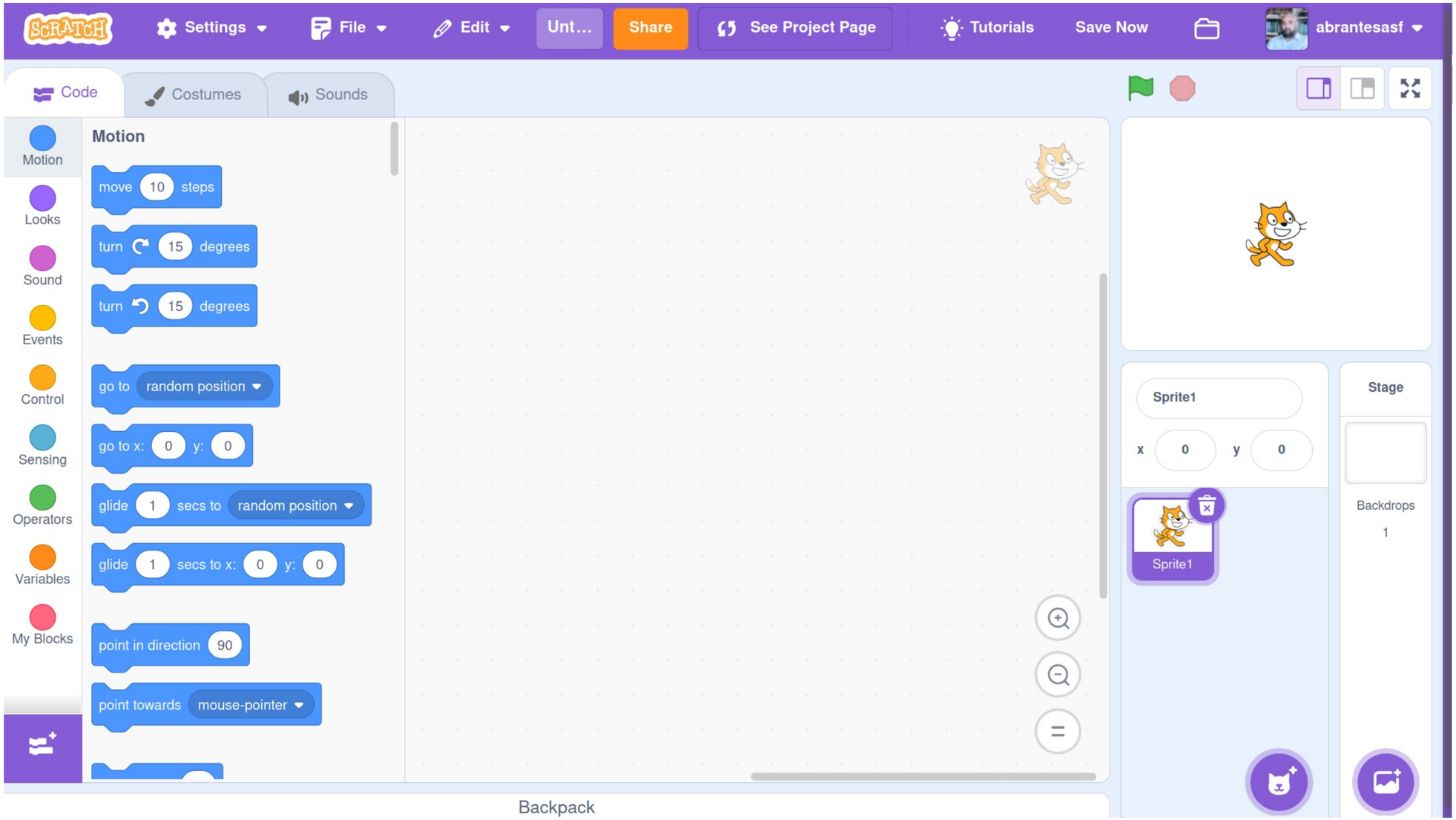


Scratch: por quê?

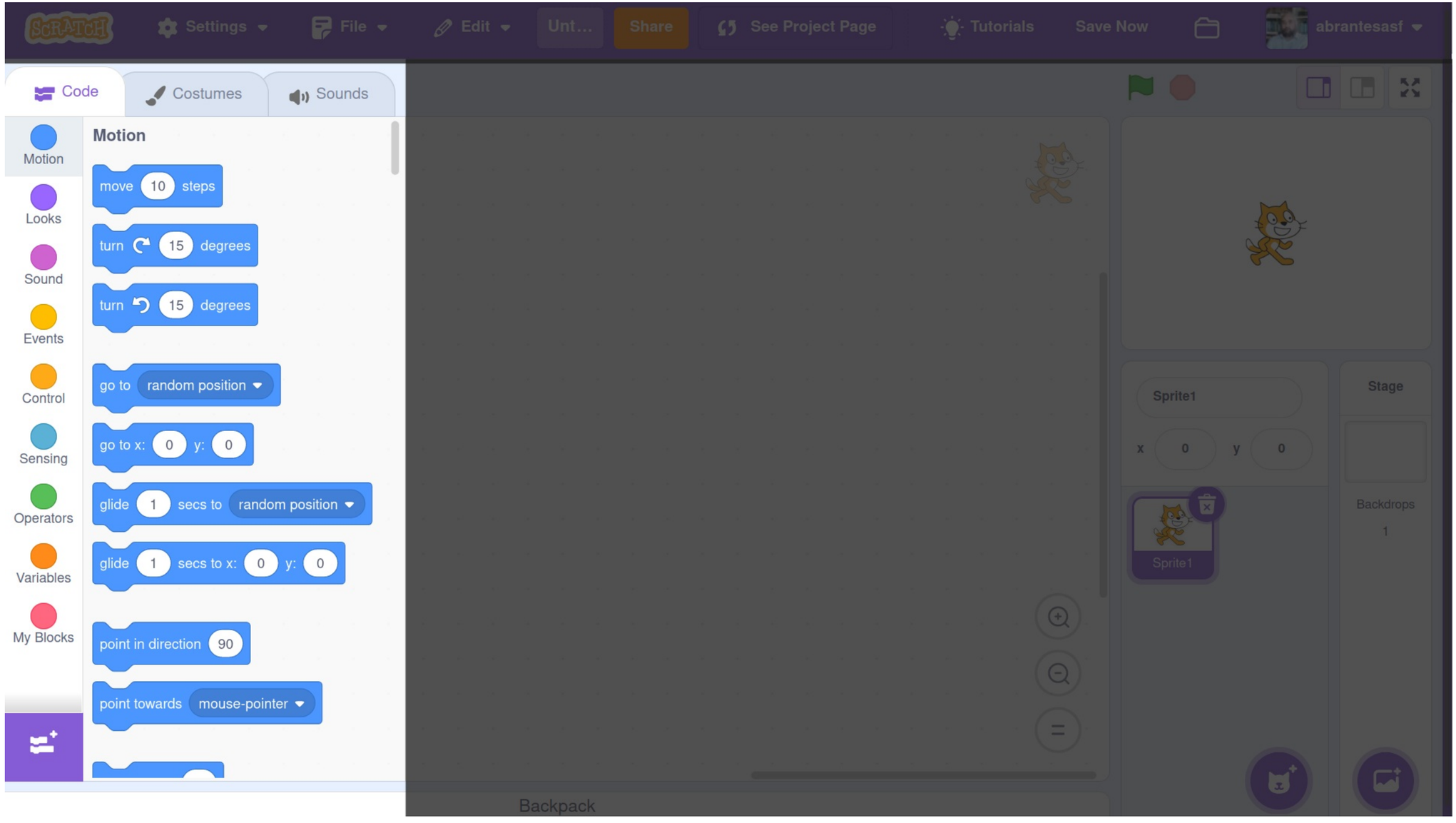
```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    printf("%s\n", "Olá, mundo!");
    return 0;
}
```

Scratch: conhecendo

https://scratch.mit.edu



Scratch: conhecendo

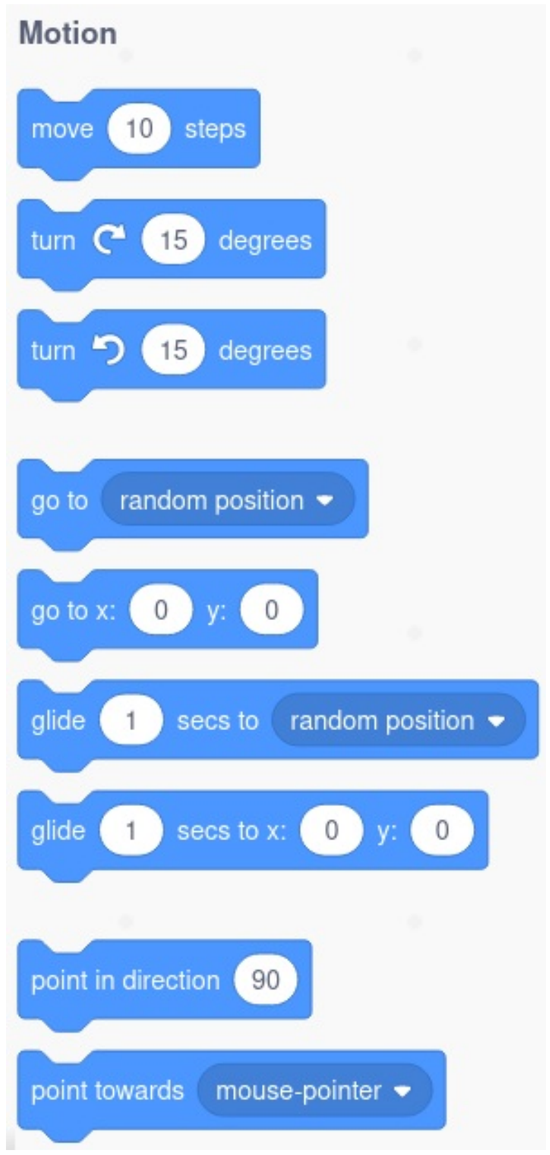


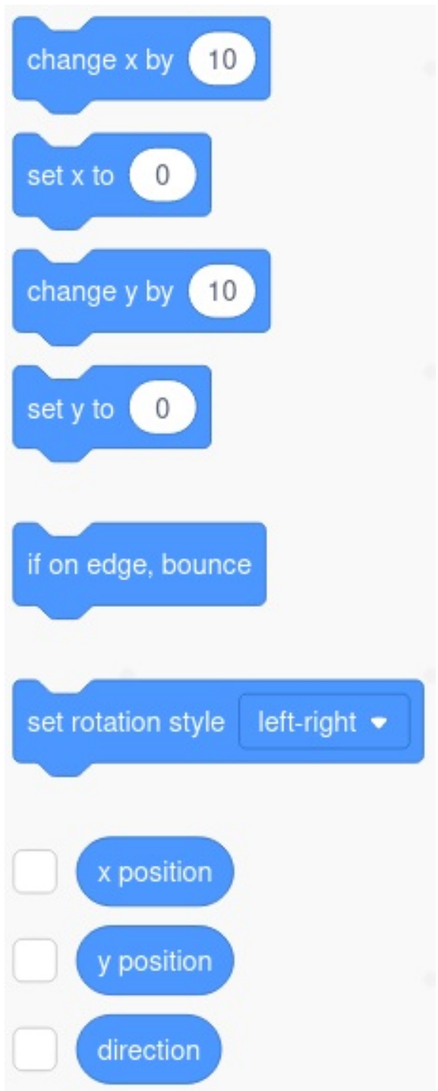
Guias de:

- Blocos (code)
- Fantasias (costumes)
- Sons (sounds)

Scratch: conhecendo

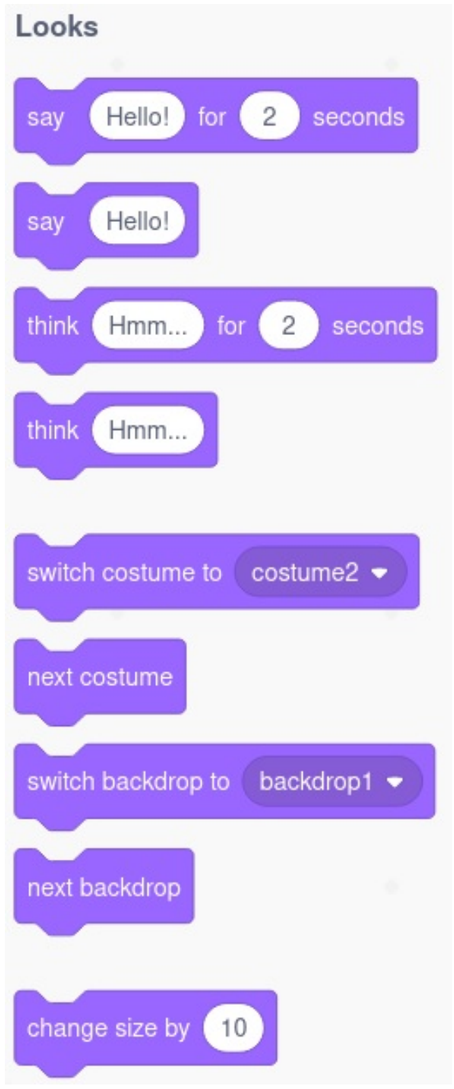
Guias de Blocos (code):
- Motion (movimento)

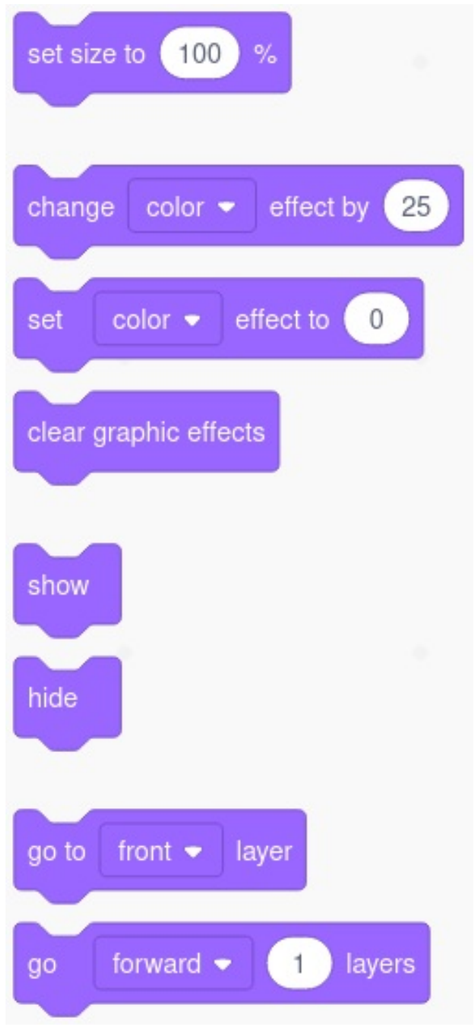


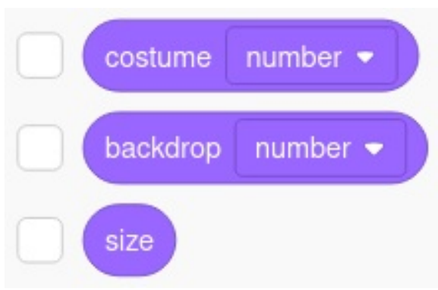


Scratch: conhecendo

Guias de Blocos (code):
- Looks (aparência)

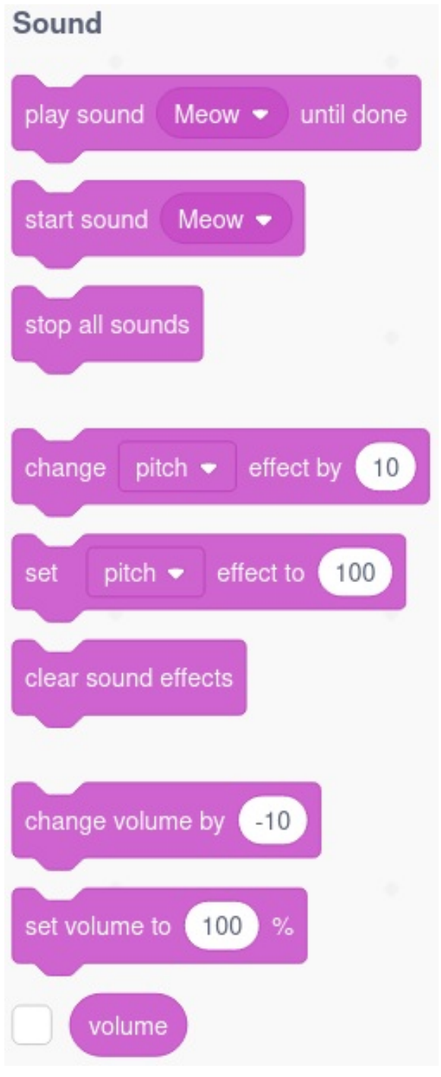




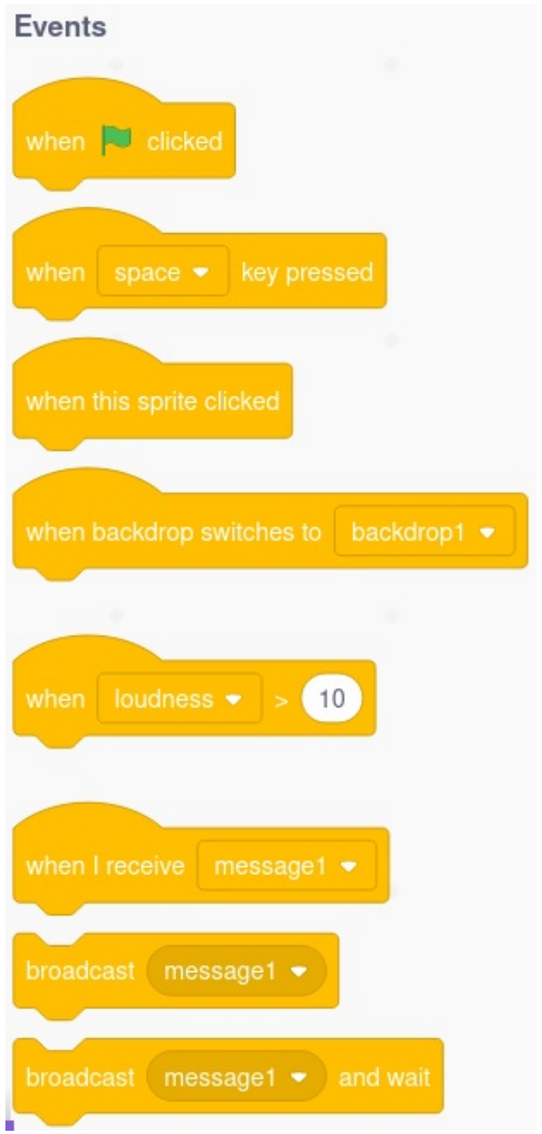


Scratch: conhecendo

Guias de Blocos (code):
- Sound (som)

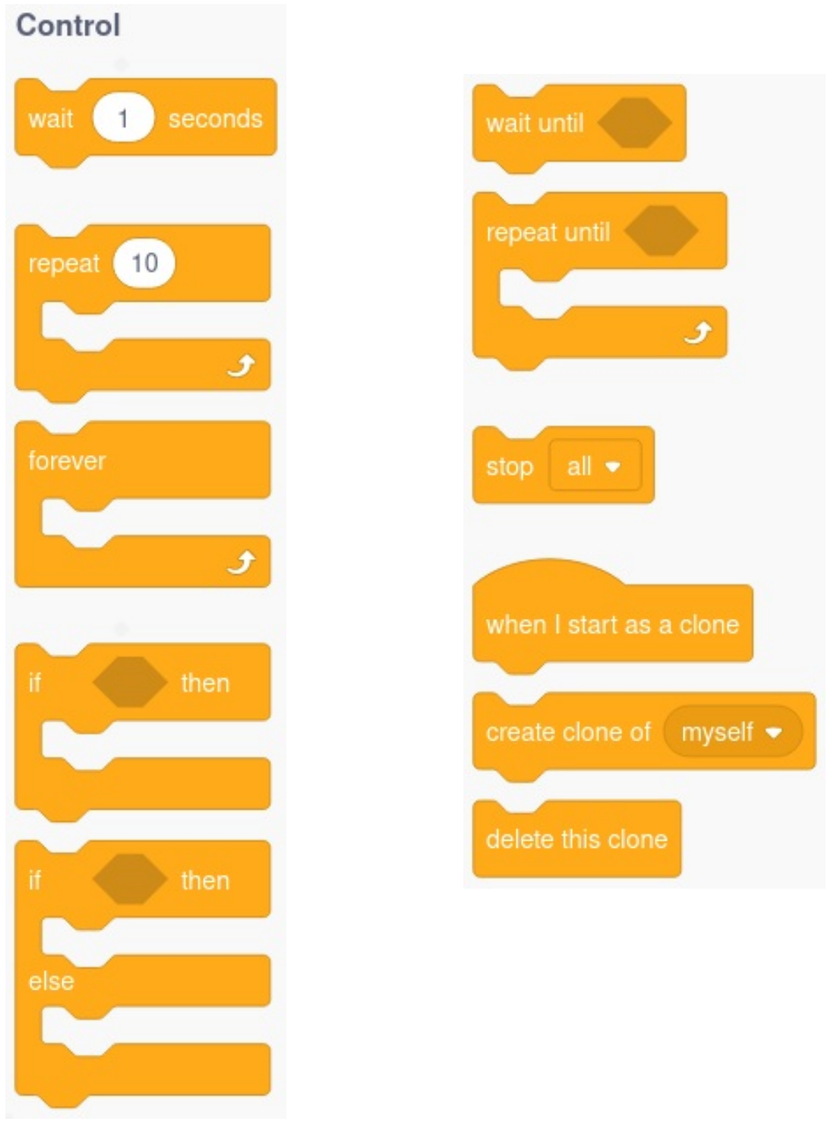


Scratch: conhecendo



Guias de Blocos (code):
- Events (eventos)

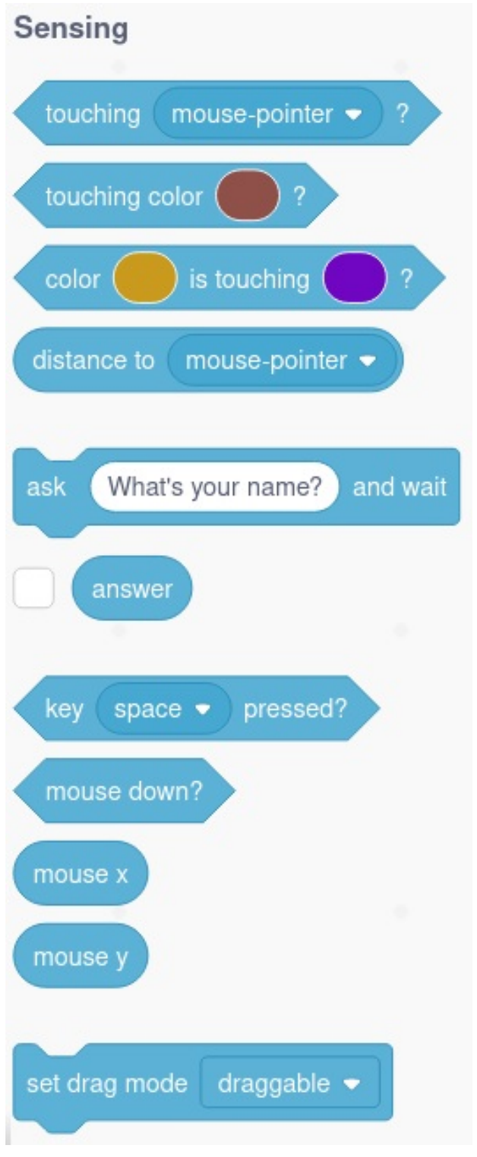
Scratch: conhecendo

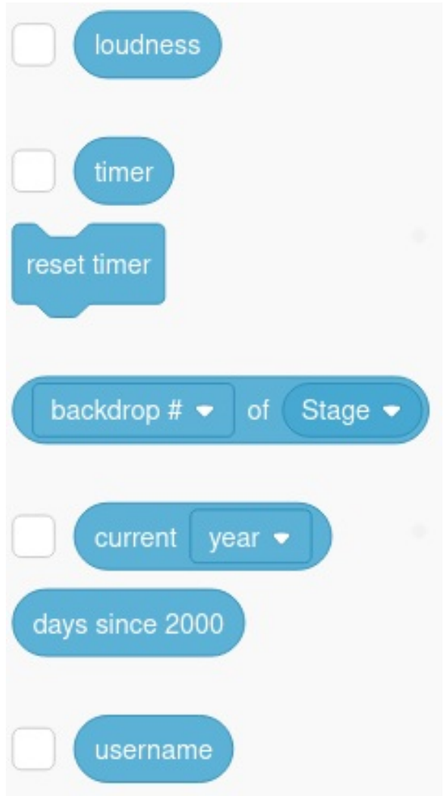


Guias de Blocos (code):
- Control (controles)

Scratch: conhecendo

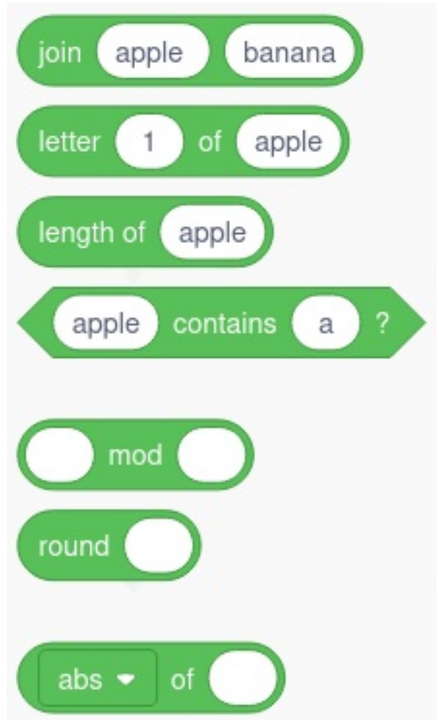
Guias de Blocos (code):
- Sensing (detecção)

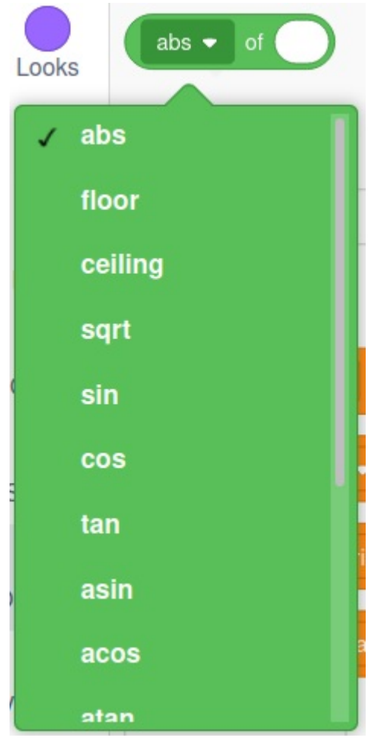




Scratch: conhecendo

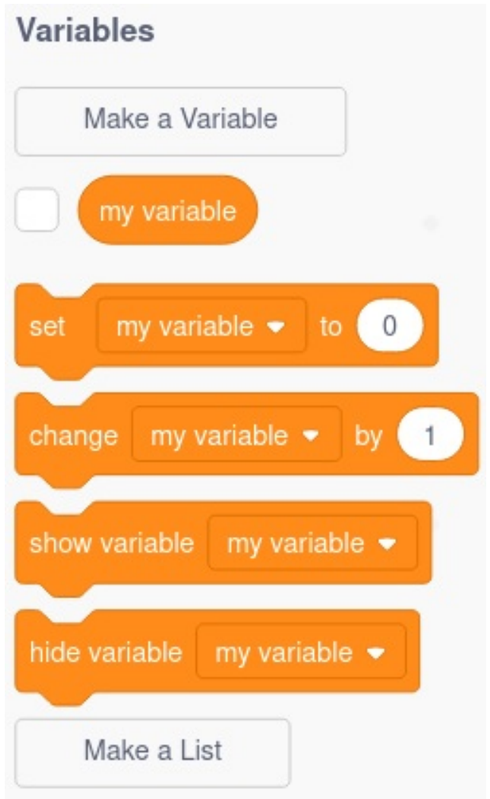
Guias de Blocos (code):
- Operators (operadores)

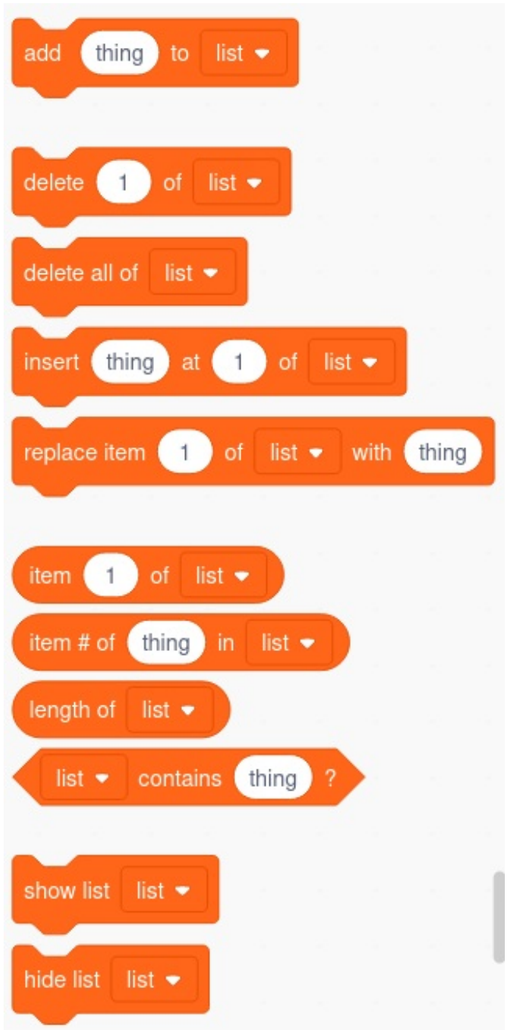




Scratch: conhecendo

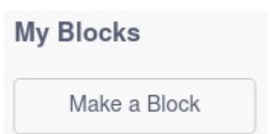
Guias de Blocos (code):
- Variables (variáveis)
- Lists (listas)





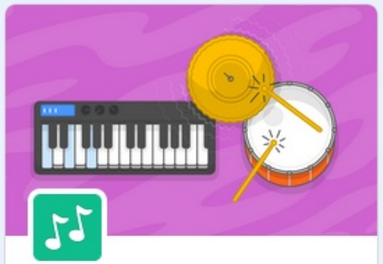
Scratch: conhecendo

Guias de Blocos (code):
- My Blocks (meus blocos)

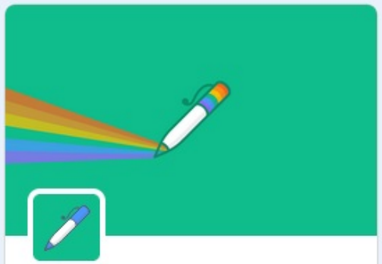


Scratch: conhecendo

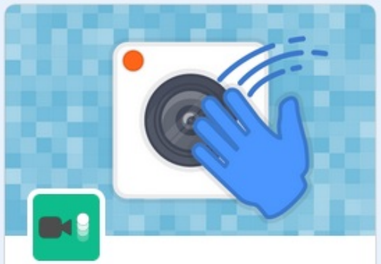
Guias de Blocos (code):
- Extension (blocos extras)



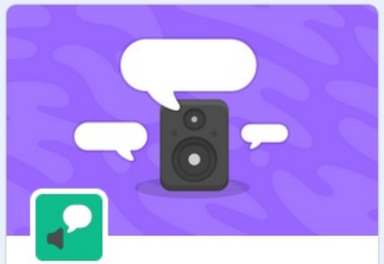
Music
Play instruments and drums.



Pen
Draw with your sprites.



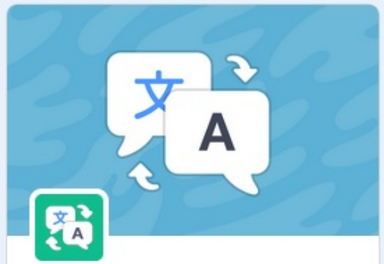
Video Sensing
Sense motion with the camera.



Text to Speech
Make your projects talk.

Requires

Collaboration with
Amazon Web Services



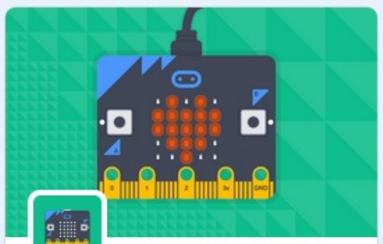
Translate
Translate text into many languages.

Requires

Collaboration with
Google

Makey Makey
Make anything into a key.

Collaboration with
JoyLabz



micro:bit
Connect your projects with the world.

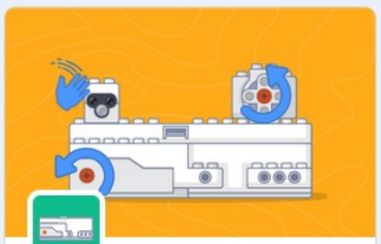
Requires

Collaboration with
micro:bit

LEGO MINDSTORMS EV3
Build interactive robots and more.

Requires

Collaboration with
LEGO



LEGO BOOST
Bring robotic creations to life.

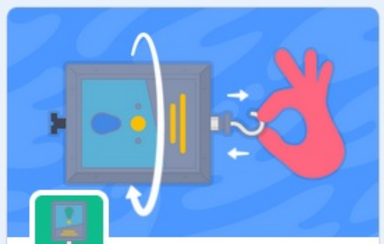
Requires

Collaboration with
LEGO

LEGO Education WeDo 2.0
Build with motors and sensors.

Requires

Collaboration with
LEGO



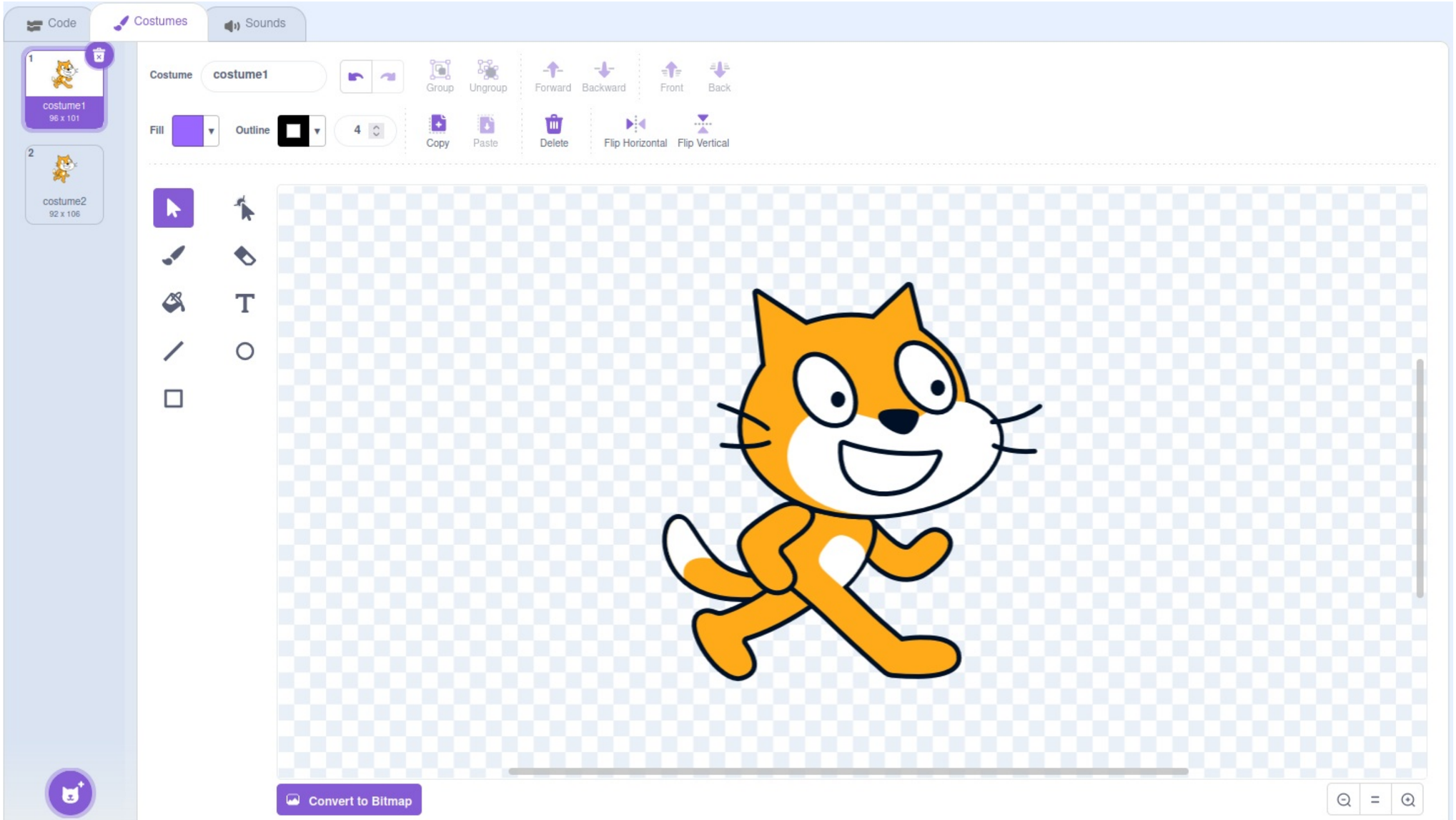
Go Direct Force & Acceleration
Sense push, pull, motion, and spin.

Requires

Collaboration with
Vernier

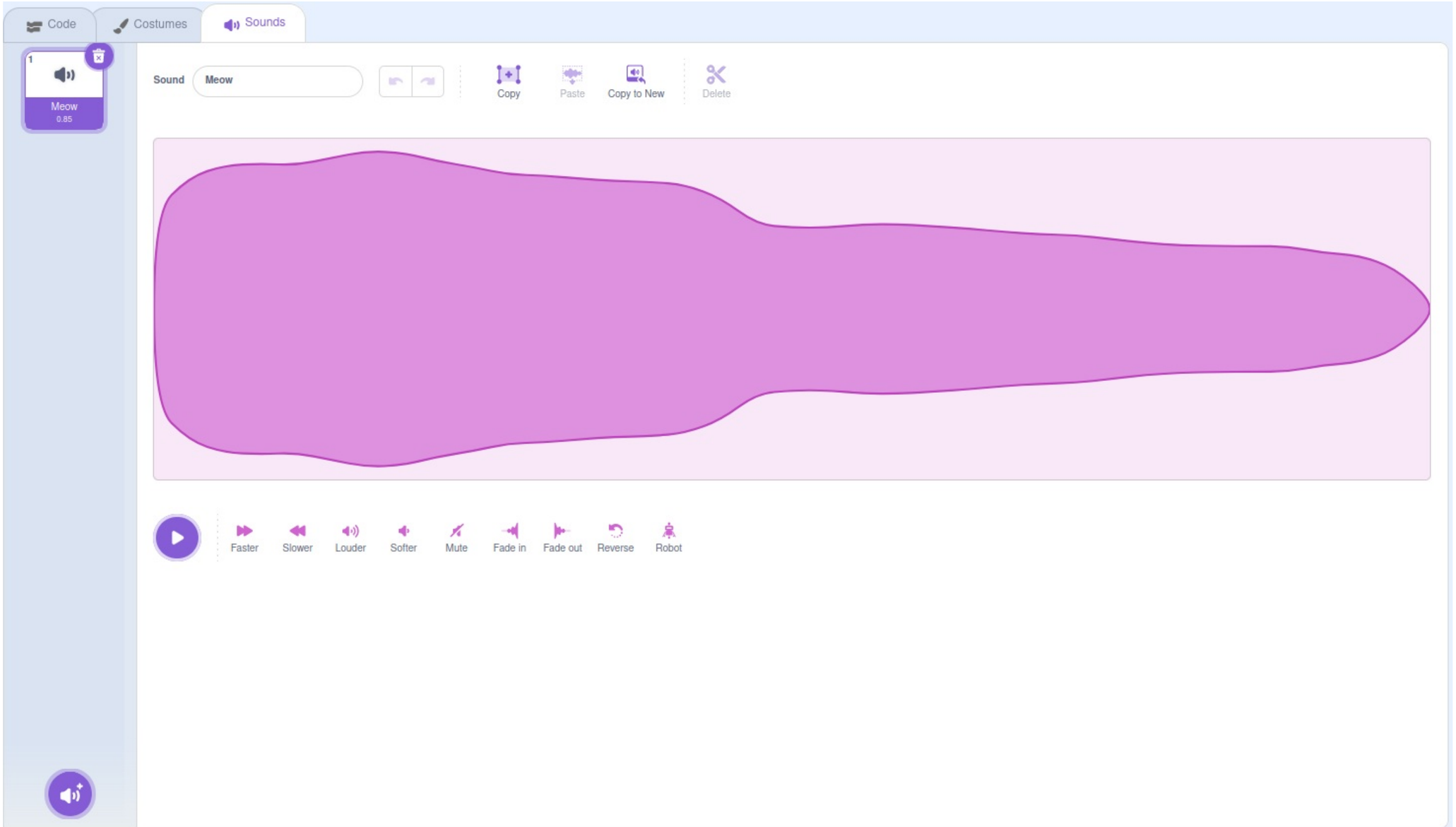
Scratch: conhecendo

Guia Costumes (fantasias)

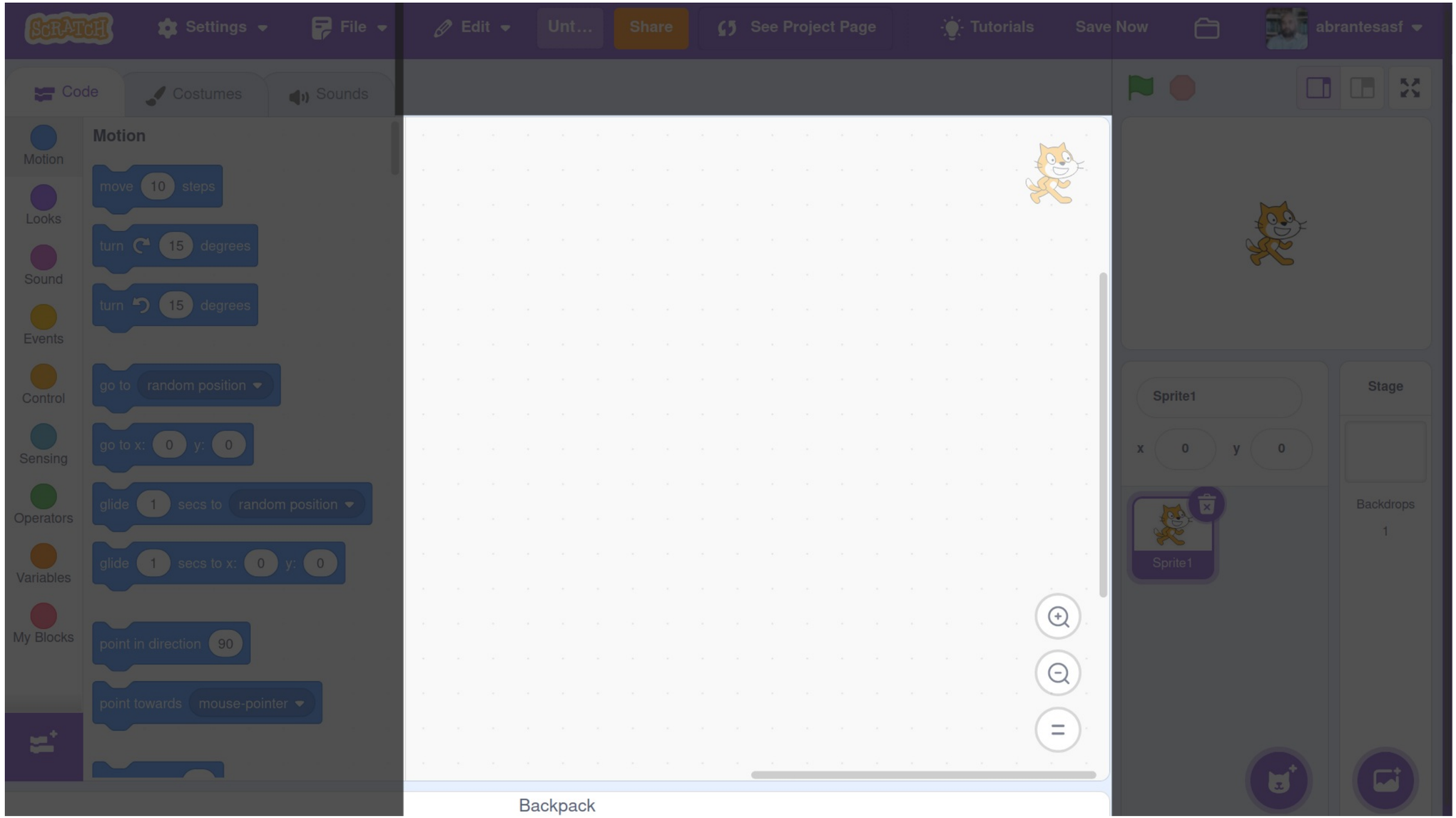


Scratch: conhecendo

Guia Sounds (sons)



Scratch: conhecendo

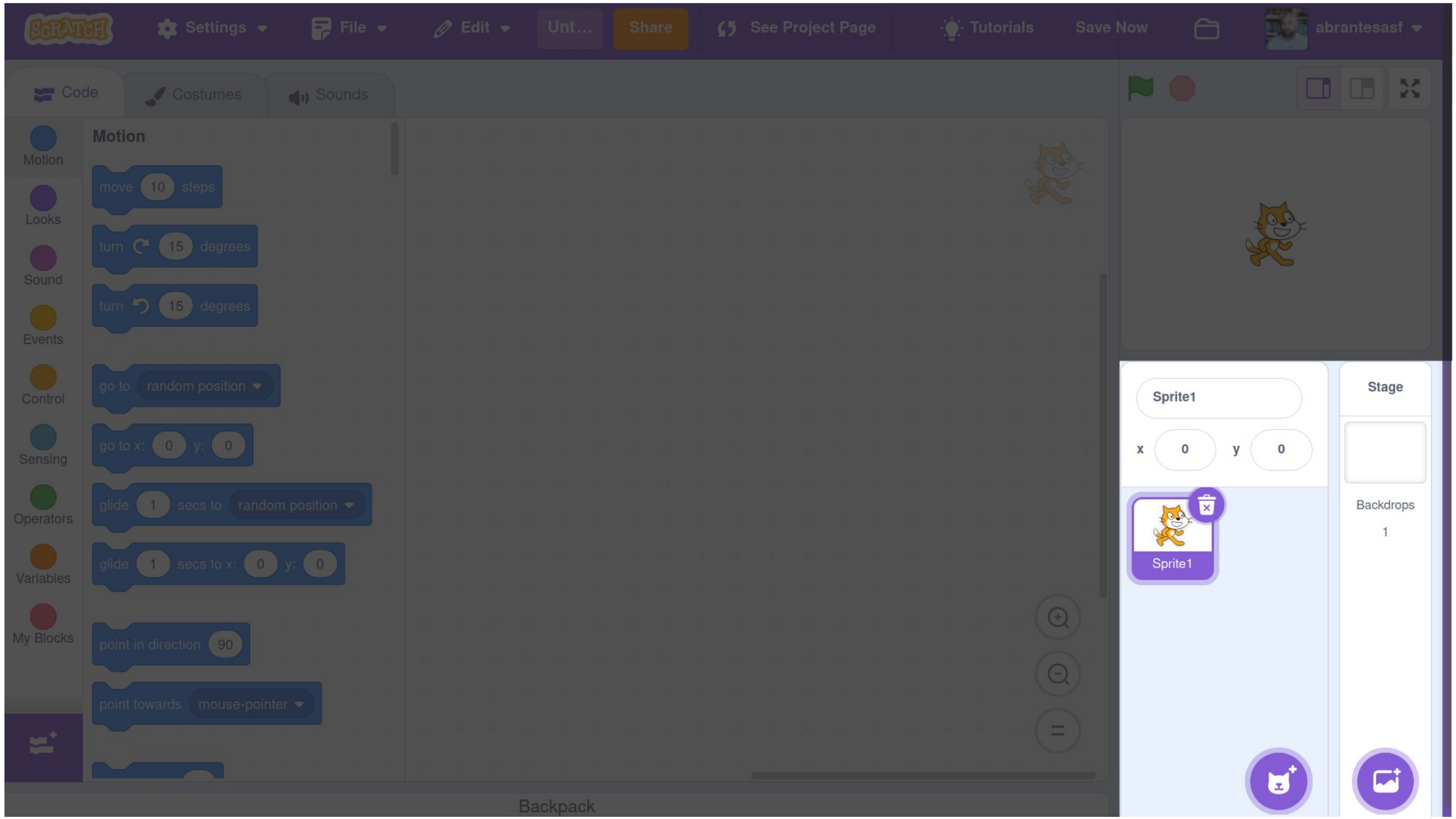


Área de código.

Cada personagem (sprite) tem sua própria área de código.

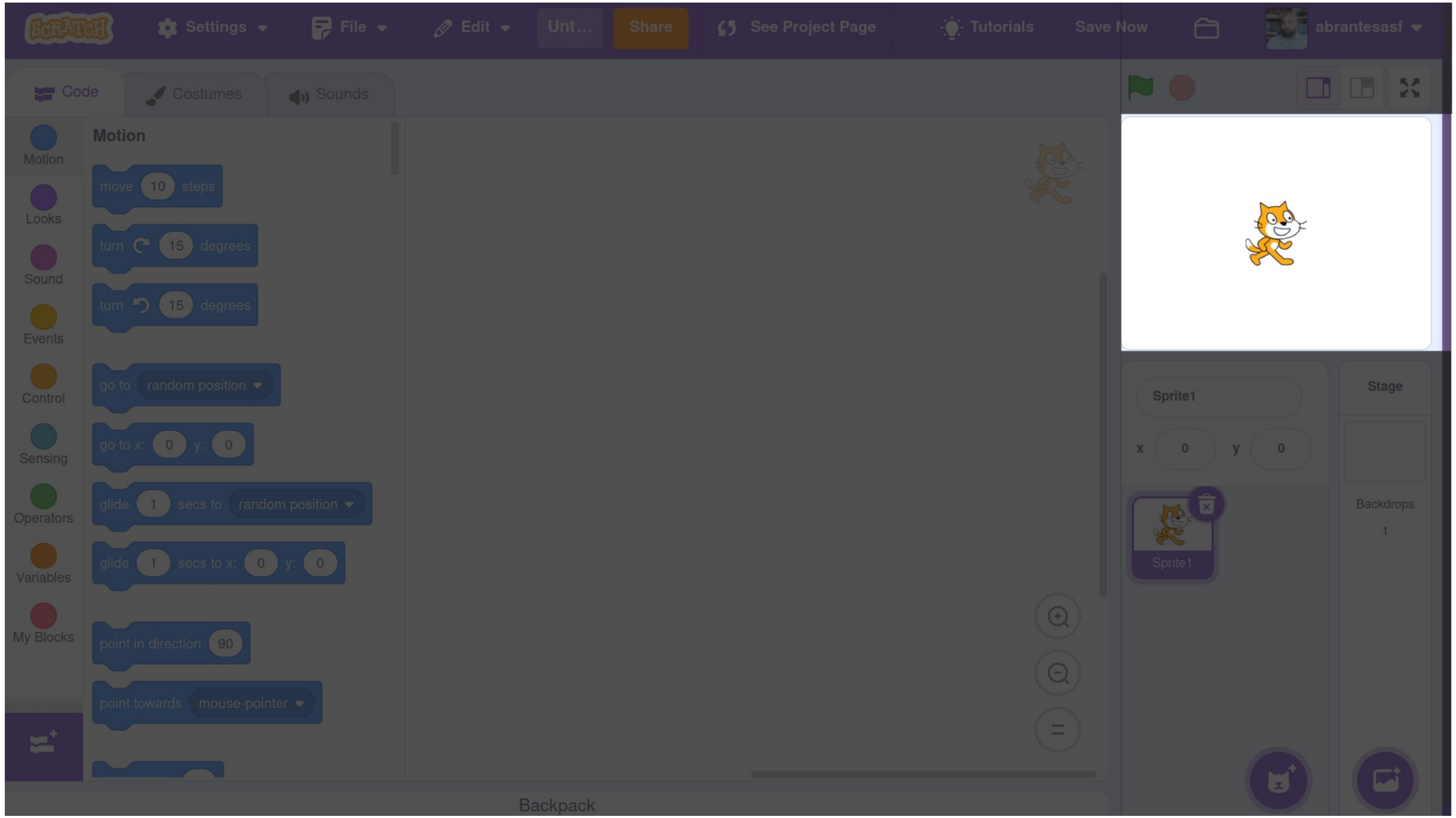
O personagem padrão é o gatinho "Scratch".

Scratch: conhecendo



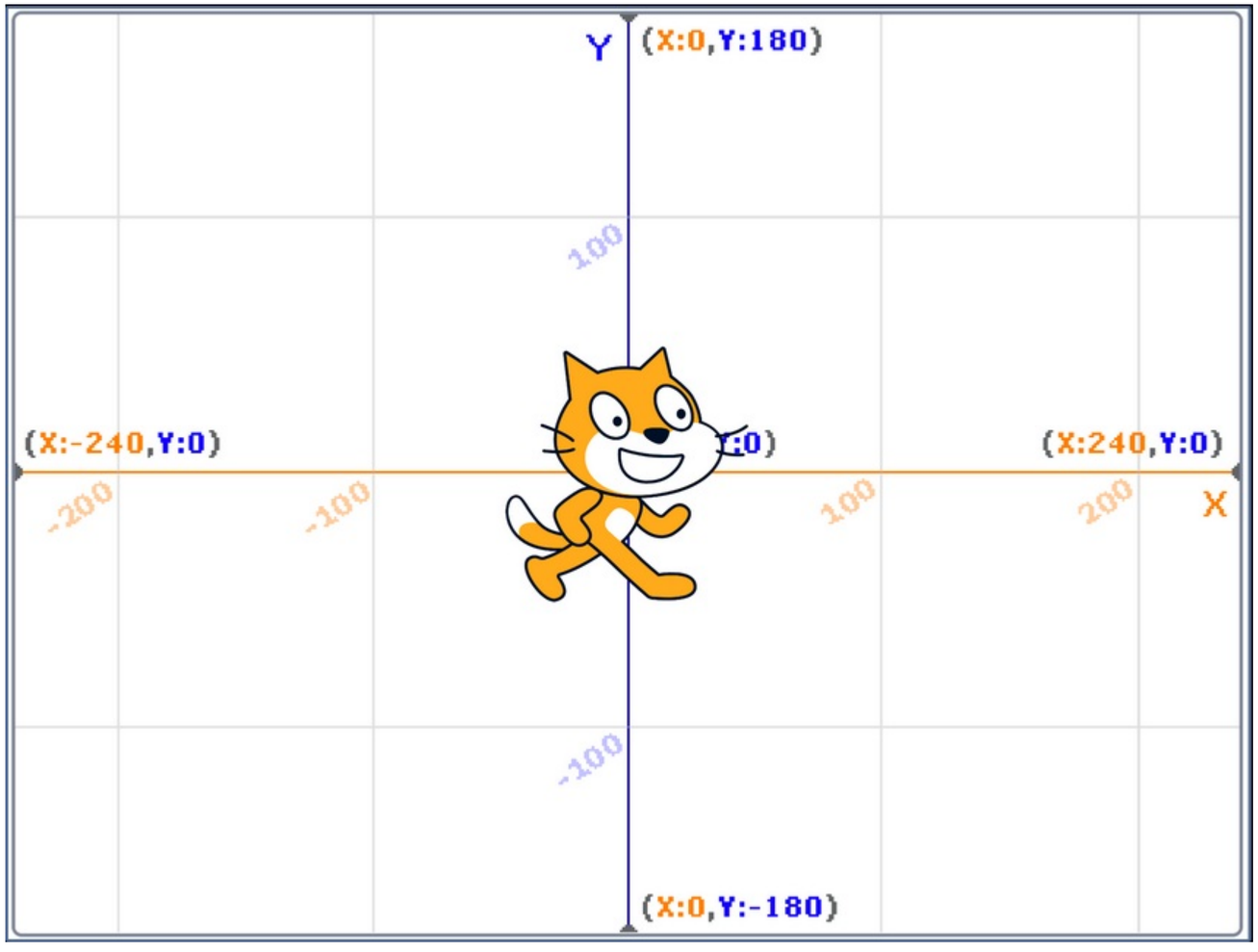
Área dos personagens (sprites) e do palco (stage).

Scratch: conhecendo



Palco (stage), é o "mundo" onde o programa é executado.

Scratch: conhecendo



O palco (stage), o "mundo" onde o programa é executado, tem um tamanho padrão de 480 x 360 pixels, com um sistema de coordenadas (não visível) onde o ponto (0, 0) está localizado extamente no centro no palco. Assim:

- largura: vai de -240 até 240 pixels (no eixo x)
- altura: vai de -180 até 180 pixels (no eixo y)

Você precisará se lembrar desse sistema de coordenadas.

Scratch


Argumento
Parâmetro
Retorno
Efeito Colateral
Função
Evento
Composição de Funções
Olá, mundo!

Primeiro programa: Olá, mundo!

- **Função**: "say"

  Um bloco que executa uma ou mais ações, escondendo os detalhes. Isso é um exemplo de **abstração**. A função pode ter:

  Argumento: é o input da função, aquilo que o usuário passa para "dentro" da função. Nesse programa o argumento da função say é "Olá, mundo!".

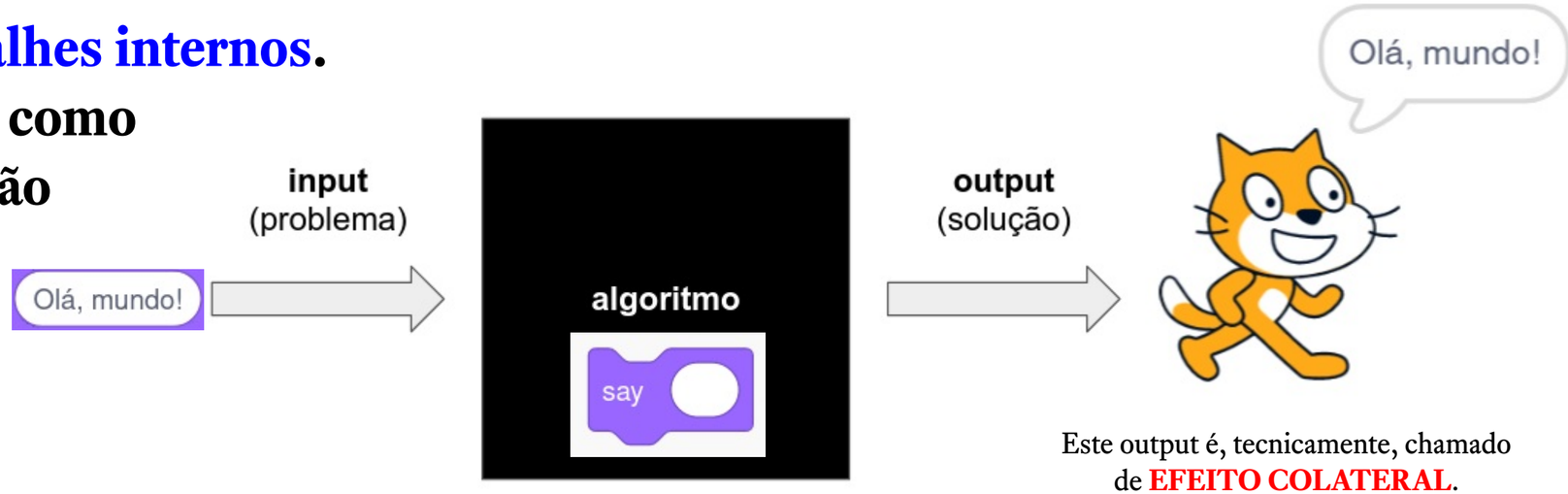
- **Evento**: "When green flag clicked"

  Alguma coisa gráfica ou interativa que **ocorre** em um programa. Os eventos são disparados por inúmeras ações, ex.: clicar na bandeira inicia o programa.

  Seu algoritmo pode ter **ouvintes** para **escutar** esses eventos, e pode **responder** ou não a eles.

Entendendo seu primeiro programa

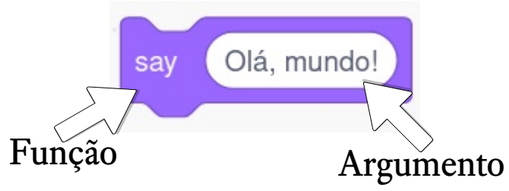
Função: um bloco que executa uma ou mais ações, escondendo os detalhes internos. Só preciso saber usar a função: como ela funciona internamente eu não preciso me preocupar. Isso é abstração! Só me preocupo com a interface de uso da função.



Este output é, tecnicamente, chamado de **EFEITO COLATERAL**.

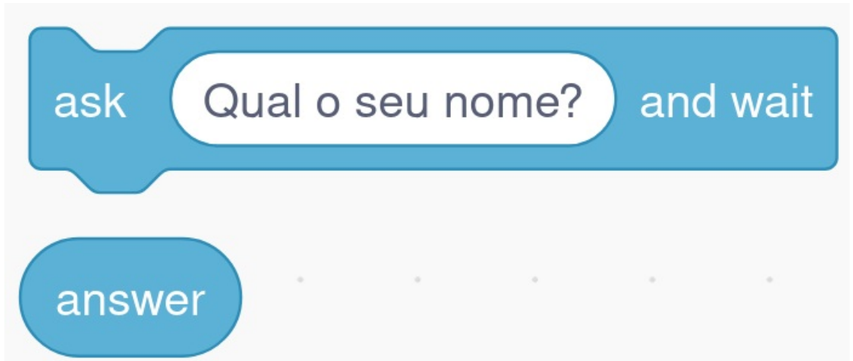
Uma função pode receber zero ou mais argumentos: os argumentos são o input da função, aquilo que nós passamos para "dentro" da função para que ela realize o processamento necessário. Outras coisas importantes:

- **parâmetro**
- **retorno**
- **efeito colateral**



Seu programa pode ser interativo!

Função: bloco que executa uma ou mais ações, escondendo os detalhes internos de funcionamento. É uma abstração.

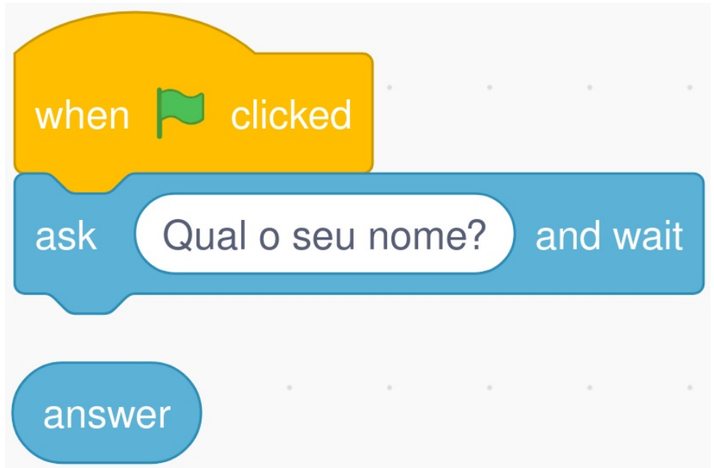


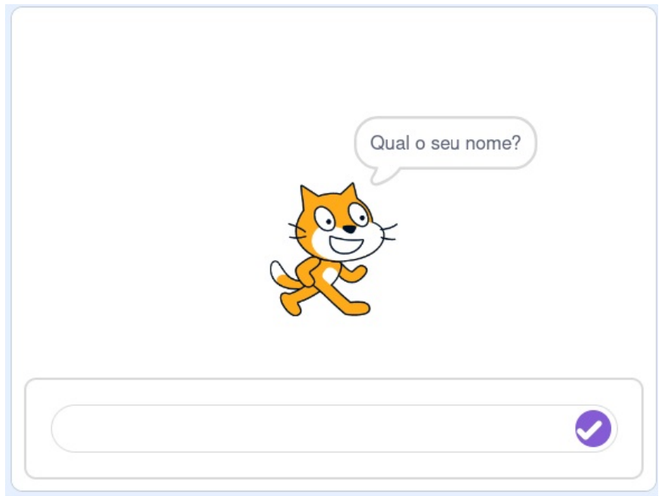
Argumento: é o input, aquilo que passamos para dentro da função. Qual é o argumento da função ask?

Retorno: é o output, aquilo que a função te devolve para que você possa utilizar em outras partes de seu programa. Qual é o retorno da função ask?

Obs.: o retorno fica armazenado em uma variável chamada de answer (resposta). Isso é um padrão do Scratch.

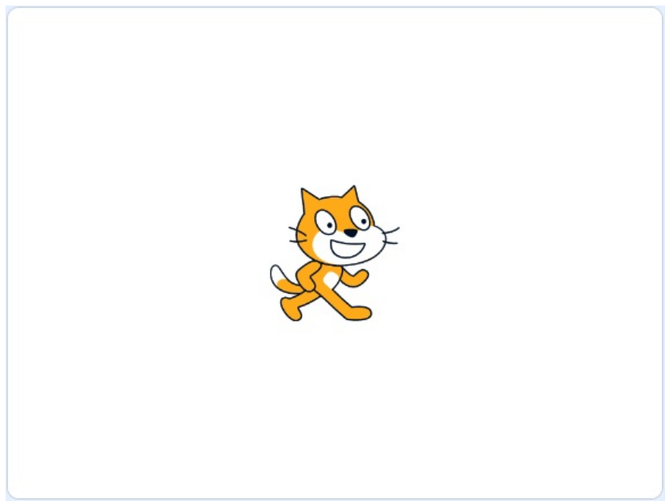
Seu programa pode ser interativo!





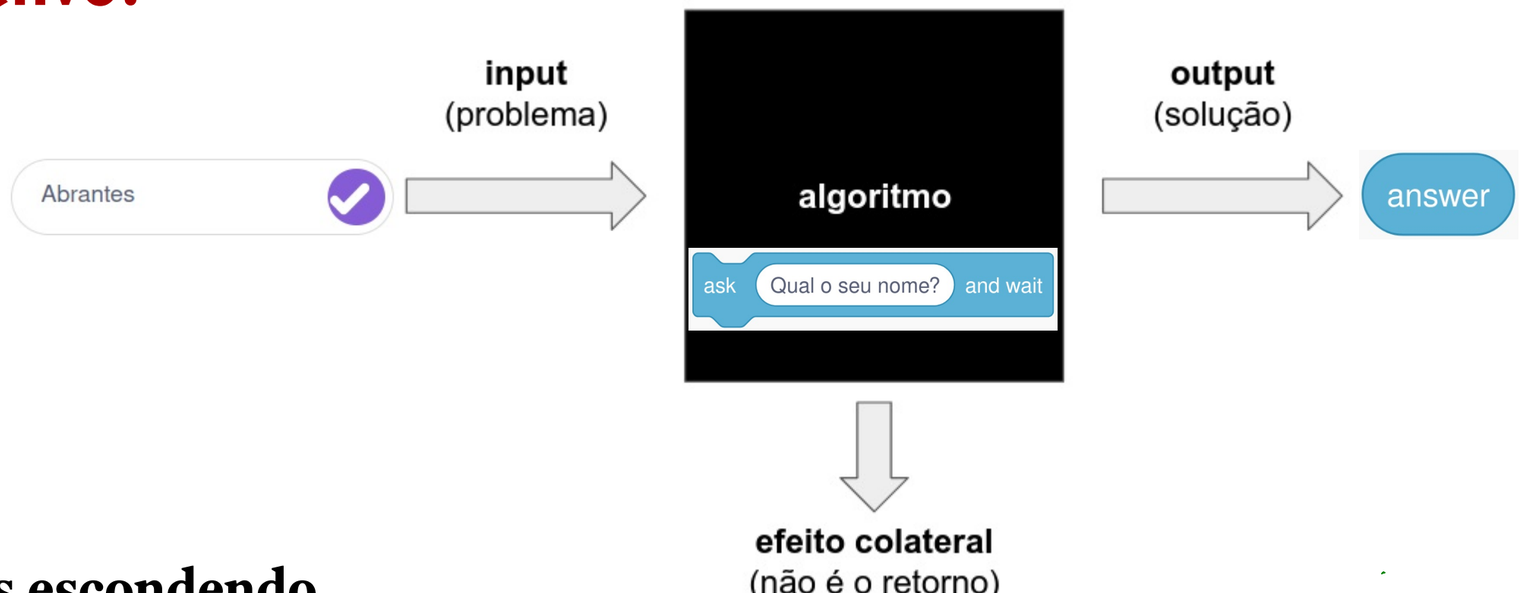
Função: executa uma ou mais ações escondendo os detalhes internos.

Retorno: é o output, aquilo que a função te devolve para que você possa utilizar em outras partes de seu programa.



Efeito colateral: é algo que a função realiza que não está diretamente relacionado ao retorno.

Seu programa pode ser interativo!



Função: executa uma ou mais ações escondendo os detalhes internos.

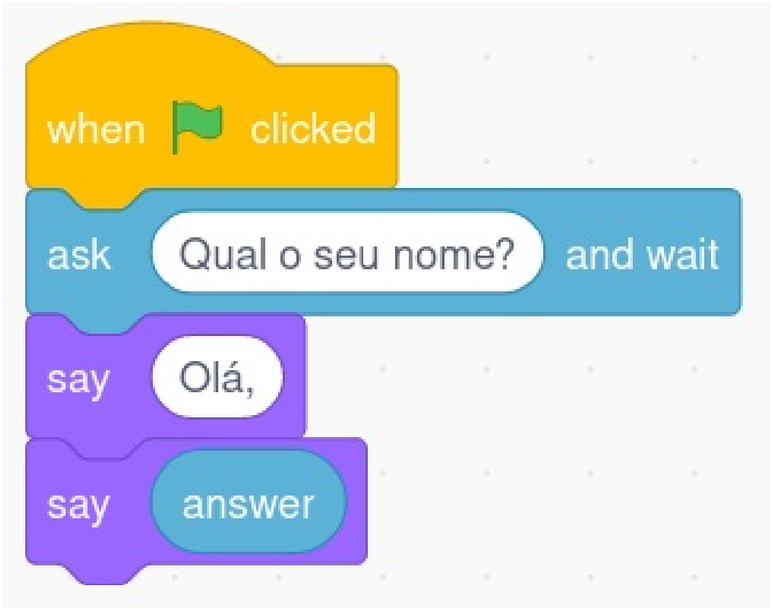
Retorno: é o output, aquilo que a função te devolve para que você possa utilizar em outras partes de seu programa.

Efeito colateral: é algo que a função realiza que não está diretamente relacionado ao retorno.

Scratch

Bugs

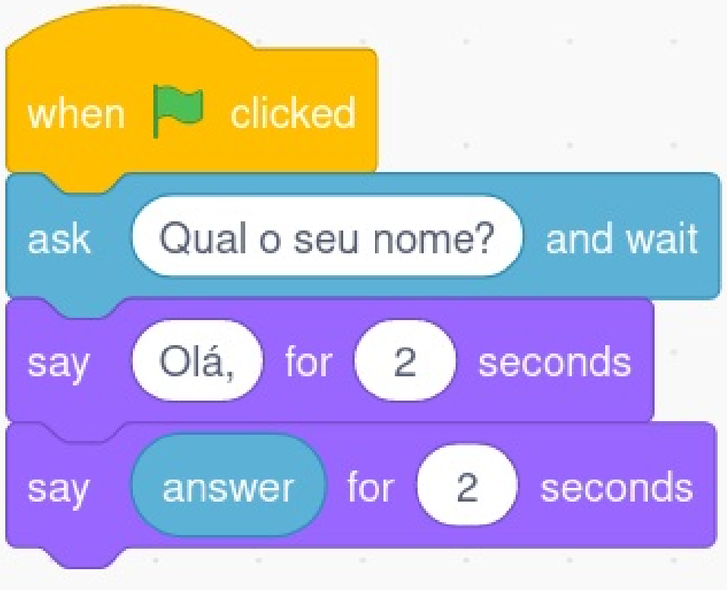
Bugs ocorrem!



Nosso primeiro bug: um erro que faz nosso programa não funcionar do modo correto, do modo que esperamos.

Por que o bug ocorreu?

Bugs ocorrem!



Primeira tentativa de debugar o erro de nosso programa.

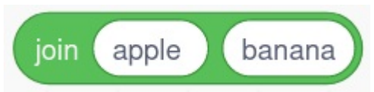
Composição de funções!

Segunda tentativa de debugar o erro de nosso programa. Agora utilizaremos composição de funções!

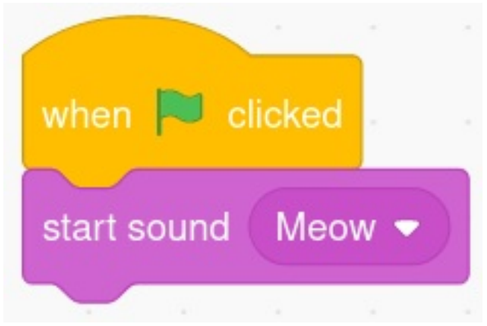
Composição de funções:

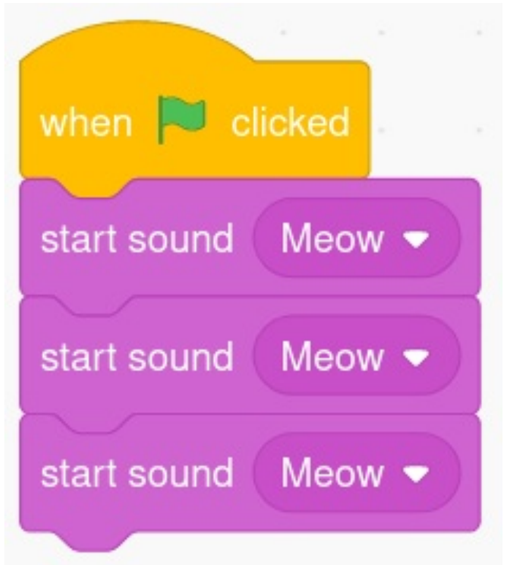
O output de uma função é passado como input (argumento) para outra função! É um conceito muito poderoso e muito importante na programação!

Como na matemática, as funções são avaliadas da mais interna para a mais externa.



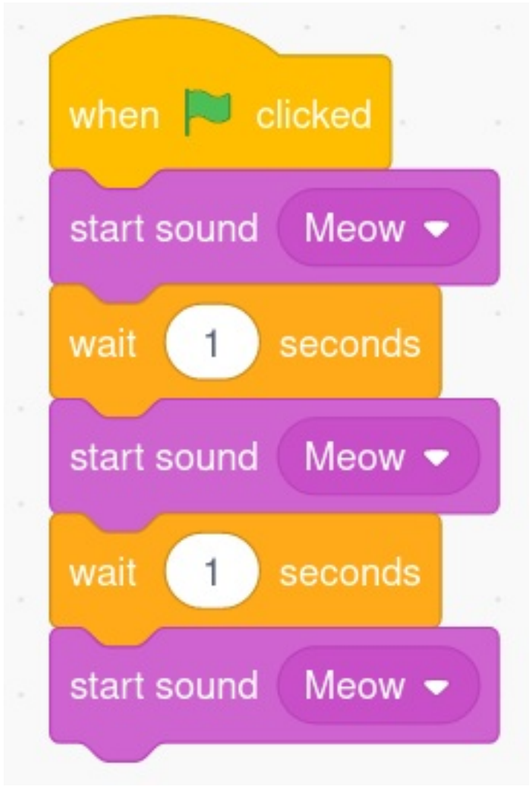
Fazendo o gato miar!





Quantas vezes o gato miará?

Fazendo o gato miar!



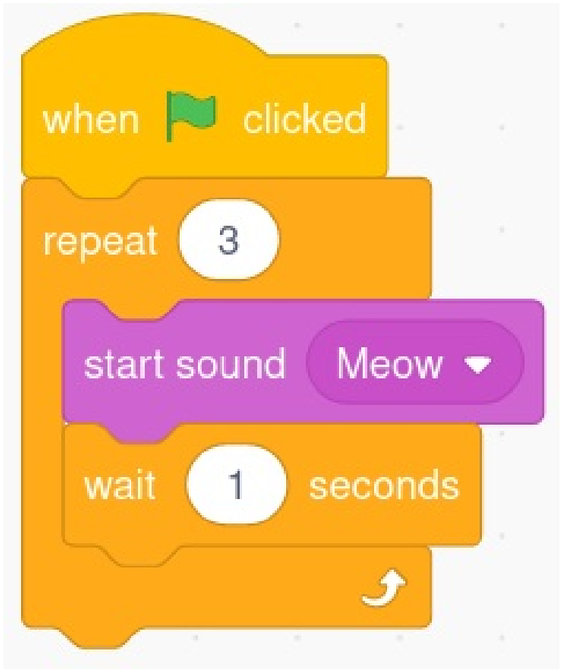
Para fazer o gato miar 3 vezes:

Este algoritmo está correto?
Sim, o algoritmo faz o que deveria fazer.

Este algoritmo está bem projetado?
Não, há muito código repetido, e isso dificulta a leitura, a claridade e o entendimento. Além disso, dificulta modificações e ajuste no código: Como fazer o gato miar 10 vezes? E para miar 1000 vezes? E para mudar o tempo de espera entre miados?

Em geral: não repita a si mesmo!

Fazendo o gato miar!

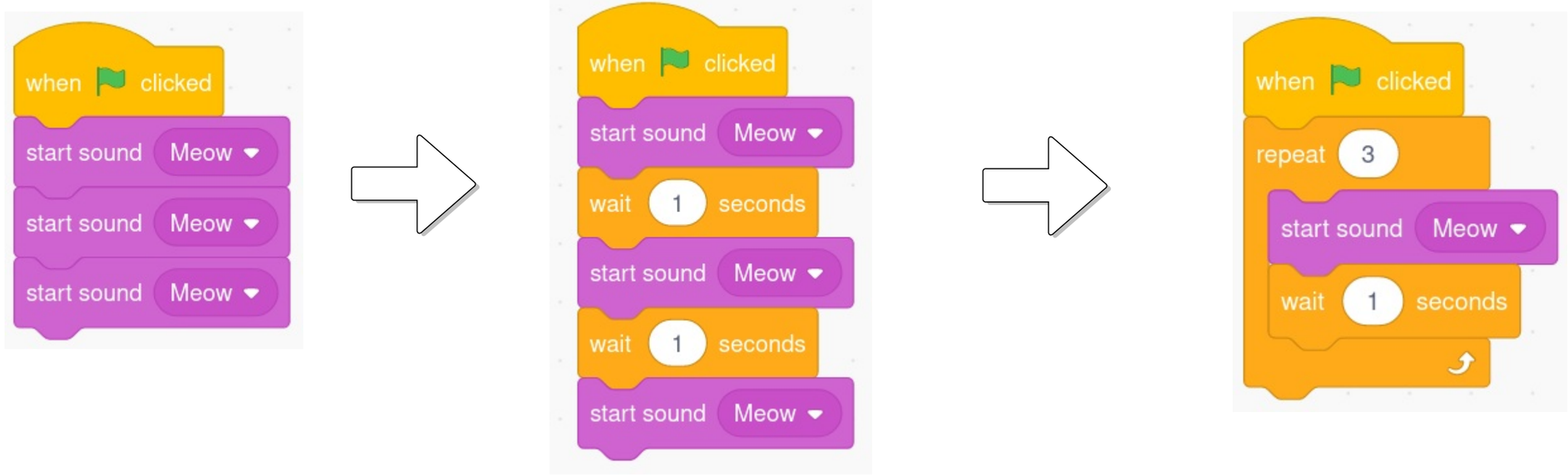


Este algoritmo está melhor projetado!

- **Mais fácil de entender**
- **Fácil de aumentar ou diminiuir os miados**
- **Fácil de alterar o tempo de espera entre miados**

Sempre que você precisar repetir ações, considere utilizar uma estrutura de repetição!

Fazendo o gato miar!

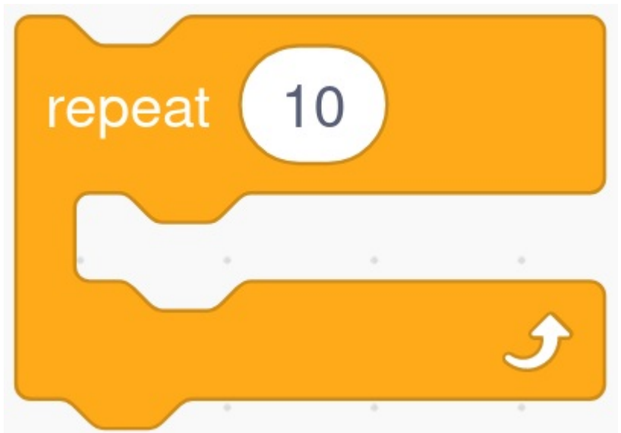


Bug → Correção → Melhoria

É possível melhorar ainda mais? Sim, criando abstrações (funções) para esconder detalhes não importantes. Veremos como criar abstrações em outra aula.

Scratch

Estruturas de repetição



repeat n:

Repete um trecho de código n vezes.

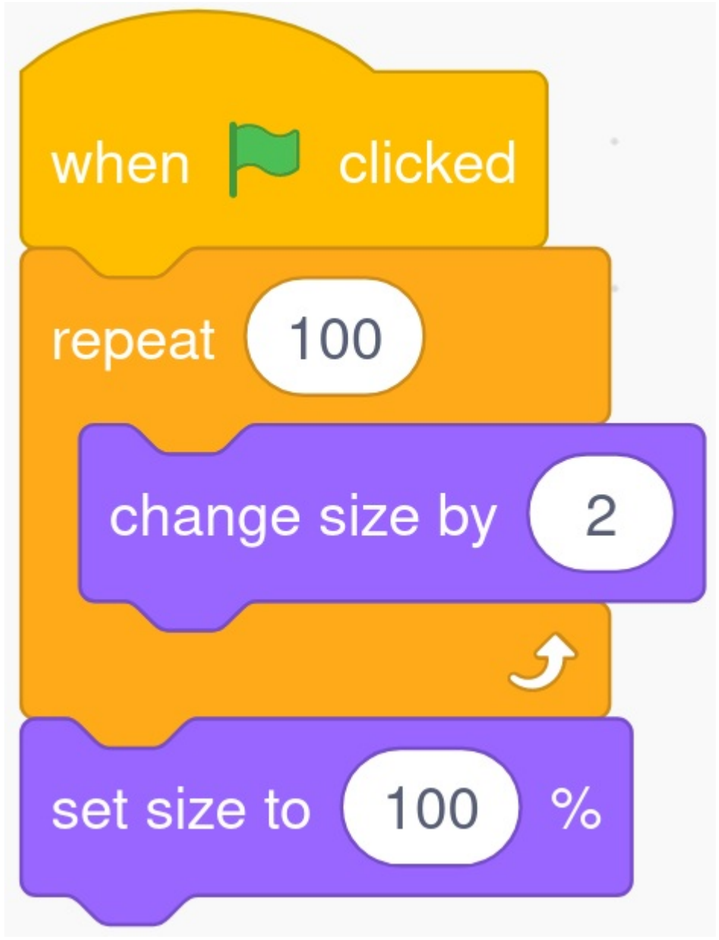
repeat until:

Repete um trecho de codigo dependendo do resultado de uma expressão booleana.

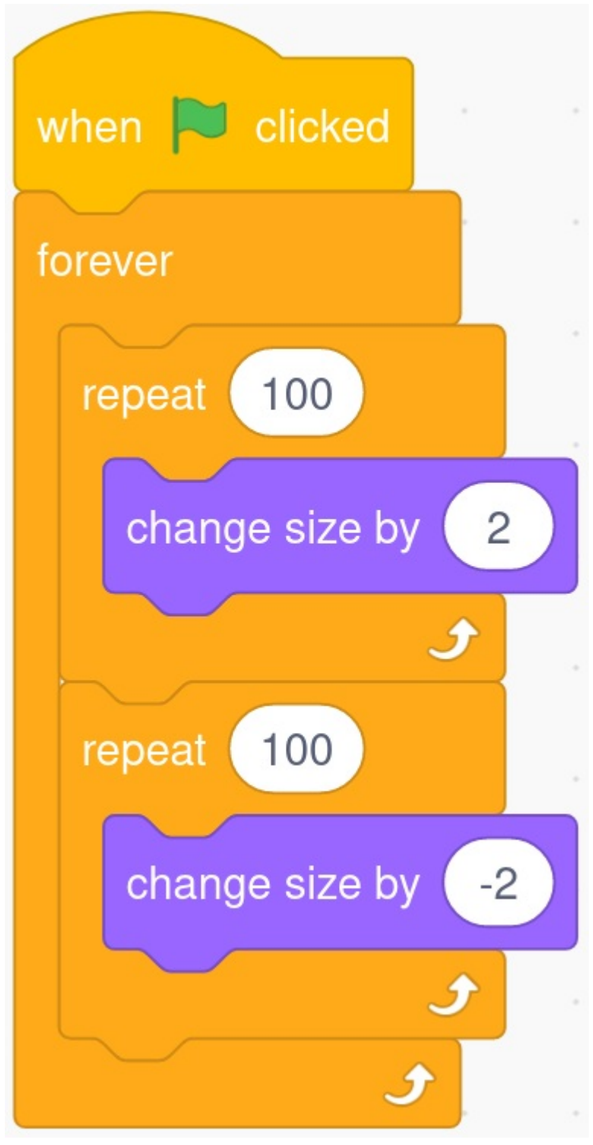
forever:

Repete infinitamente um trecho de código.

Estruturas de repetição



Estruturas de repetição



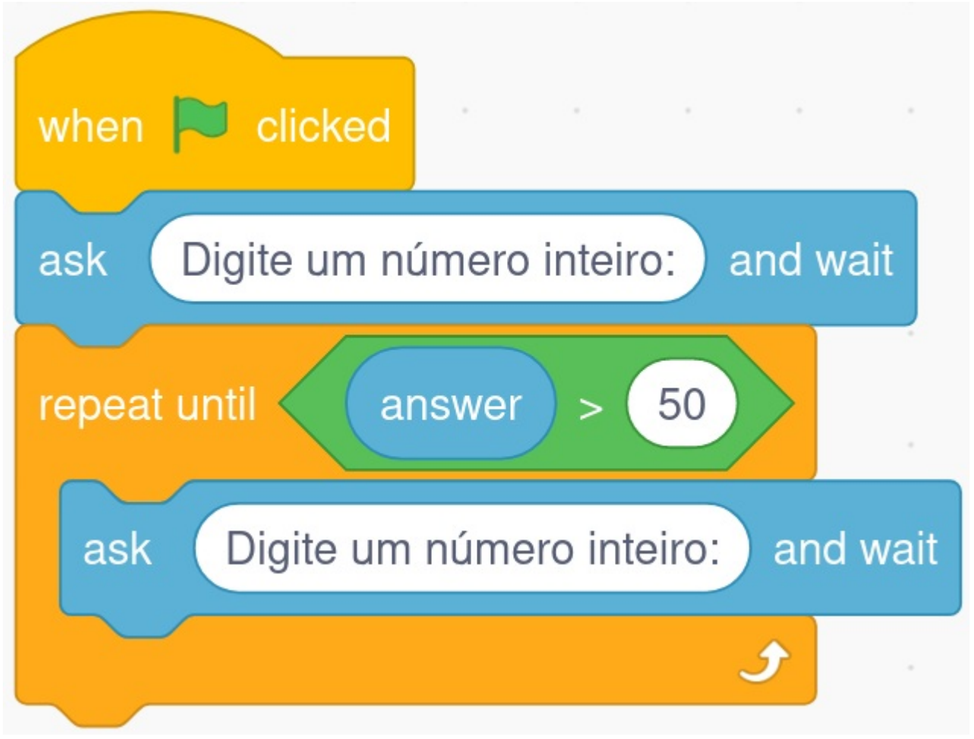
Note que podemos utilizar estruturas de repetição de forma aninhada, ou seja, uma dentro da outra.

As estruturas de repetição também são chamadas de loops (laços).

Este código exibe loops aninhados! Temos 2 loops finitos dentro de um loop infinito!

O loop infinito só termina se o usuário parar o programa por algum meio.

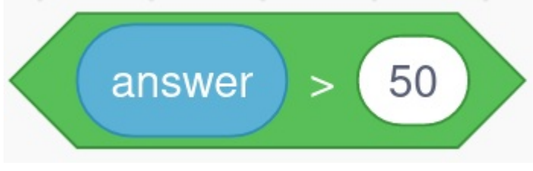
Estruturas de repetição



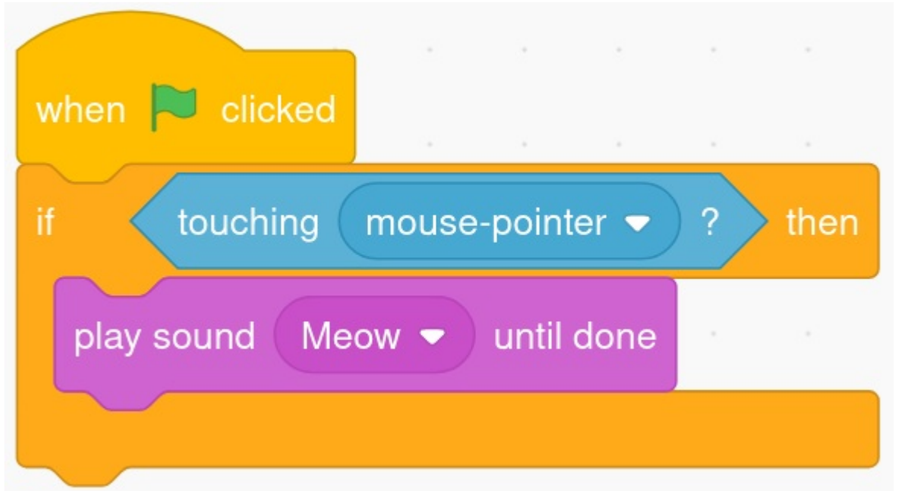
O loop repeat until é controlado por uma expressão booleana, uma pergunta cuja resposta só pode ser verdadeira ou falsa.

Neste exemplo a pergunta será repetida até que o usuário informe um número maior do que 50.

O que ocorre se, logo na primeira vez, o usuário já informar um número maior do que 50?



Estruturas de repetição: loop infinitos são úteis?

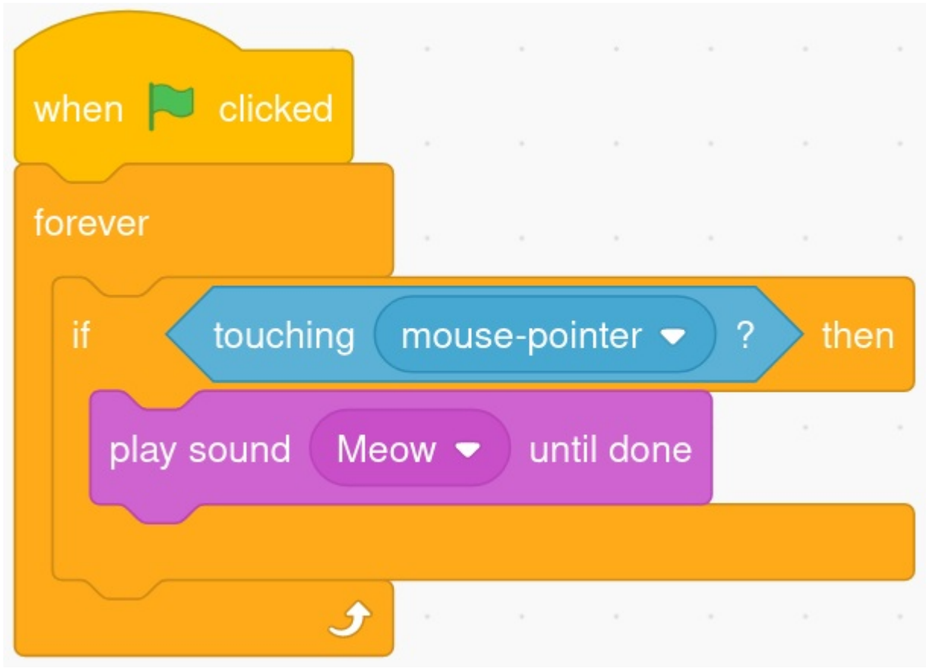


Problema: fazer o gato miar toda vez que o mouse passar sobre ele.

Este algoritmo está correto?
O algoritmo faz o que deveria fazer?

Este algoritmo está bem projetado?
O agoritmo está bem escrito, sem repetições desnecessárias, sem ambigüidades, é claro e direto, é de fácil entendimento?

Estruturas de repetição: loop infinitos são úteis?



Problema: fazer o gato miar toda vez que o mouse passar sobre ele.

Este algoritmo está correto?

O algoritmo faz o que deveria fazer?

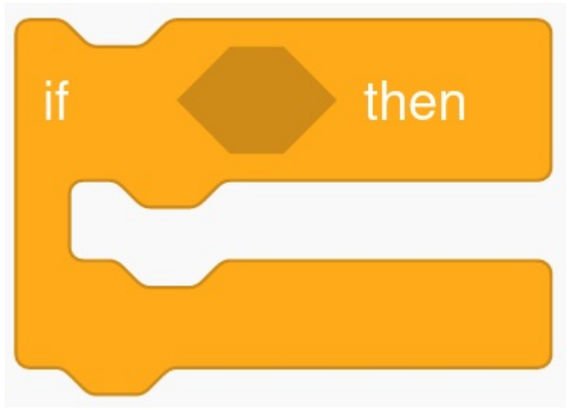
Este algoritmo está bem projetado?

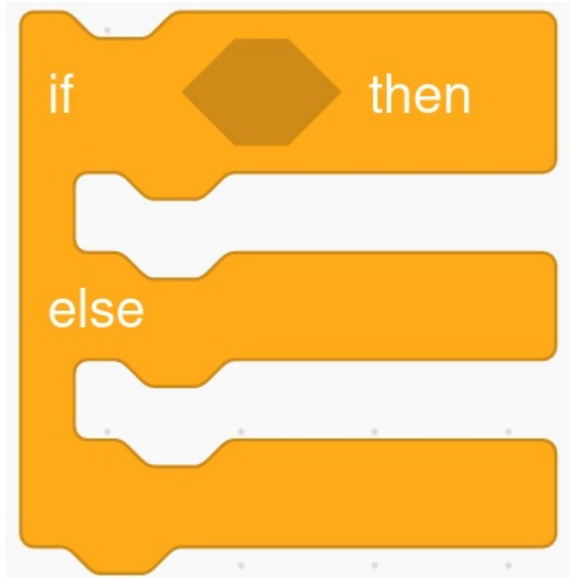
O agoritmo está bem escrito, sem repetições desnecessárias, sem ambigüidades, é claro e direto, é de fácil entendimento?

Em geral loops infinitos são erros no programa, mas podem ser utilizados em diversas situações nas quais o programa precisa verificar condições por tempo indeterminado!

Scratch

Estruturas condicionais





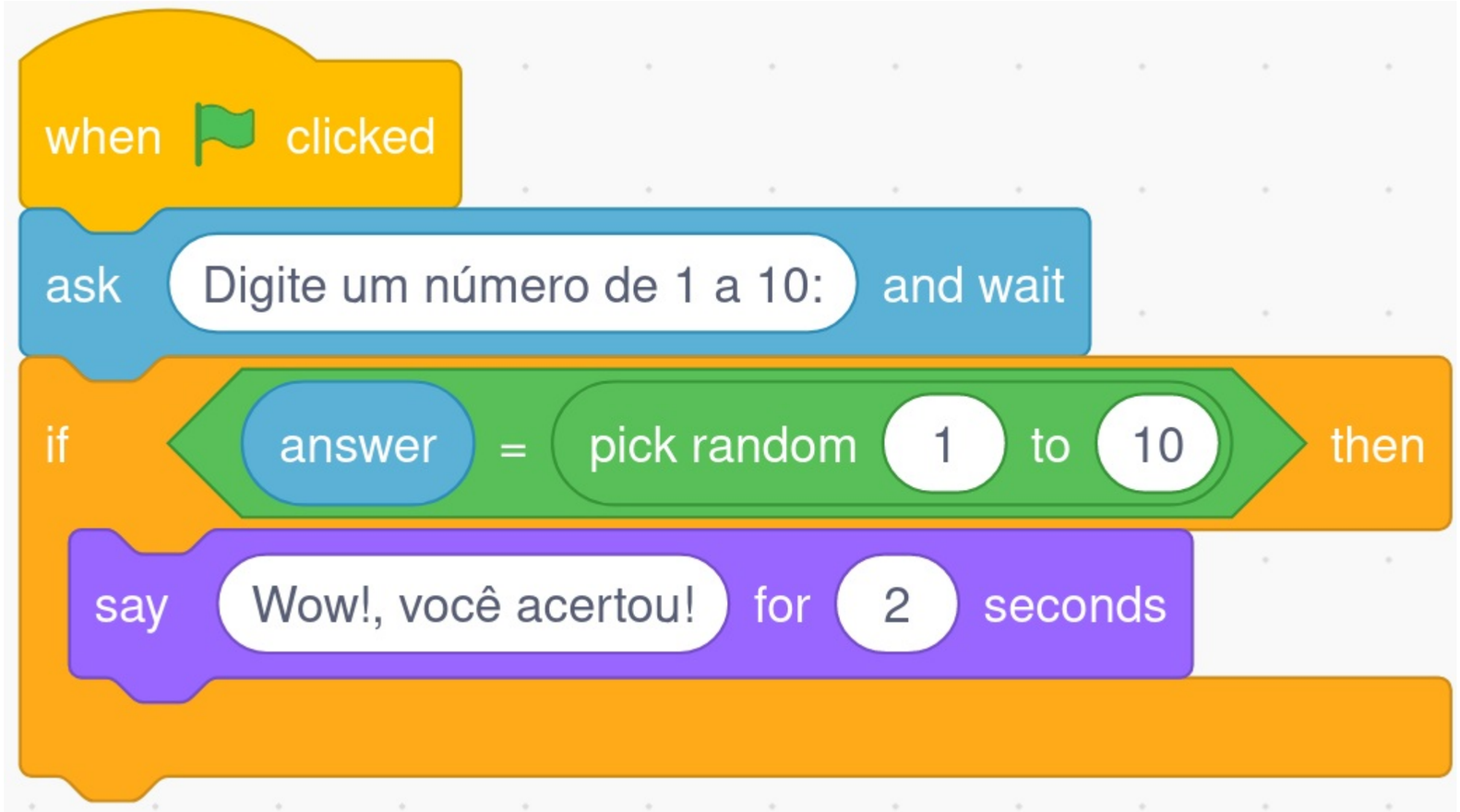
if - then:

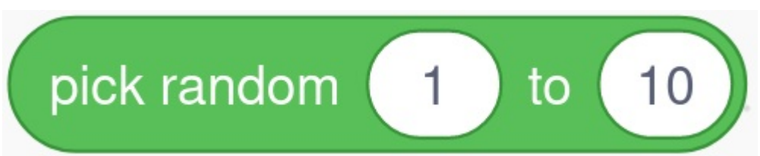
Executa uma ou mais ações se o resultado de uma expressão booleana for TRUE.

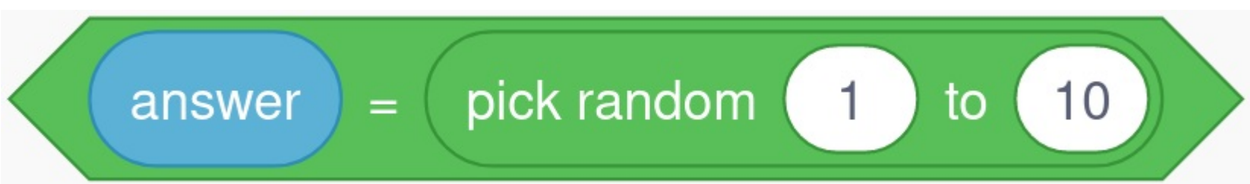
if - then - else:

Permite escolher entre 2 caminhos de ação, dependendo do resultado TRUE ou FALSE de uma expressão booleana.

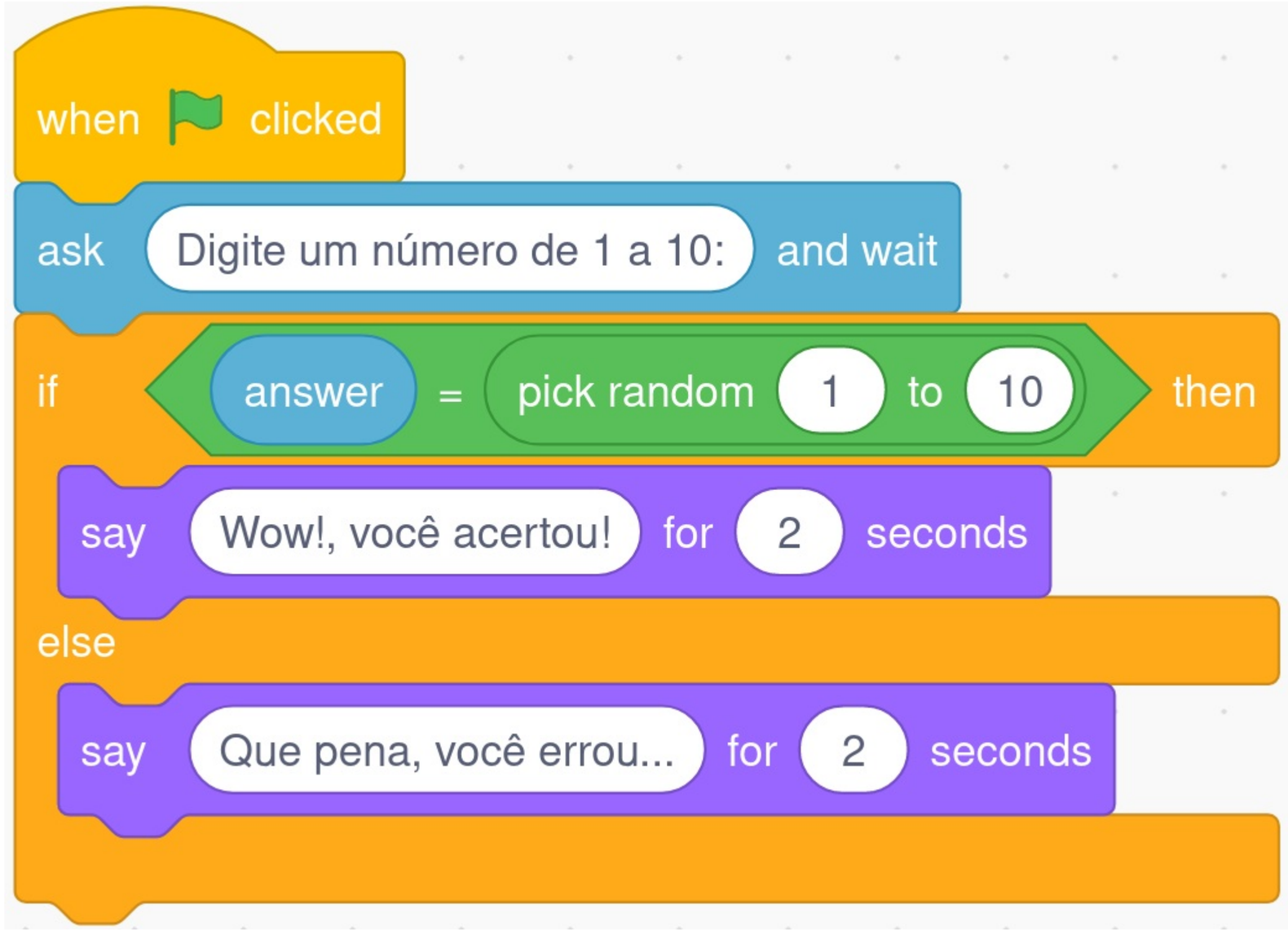
Estruturas condicionais







Estruturas condicionais



Estruturas condicionais também podem ser aninhadas!

if-then-else principal

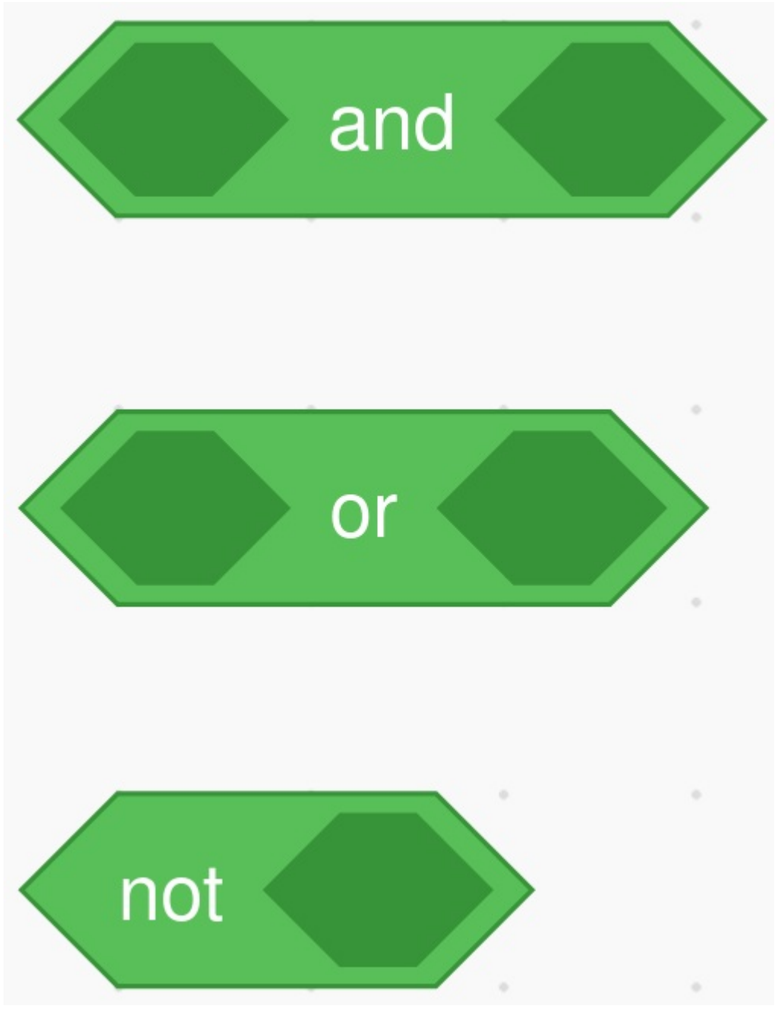
if-then-else aninhado no principal

if-then-else aninhado no principal

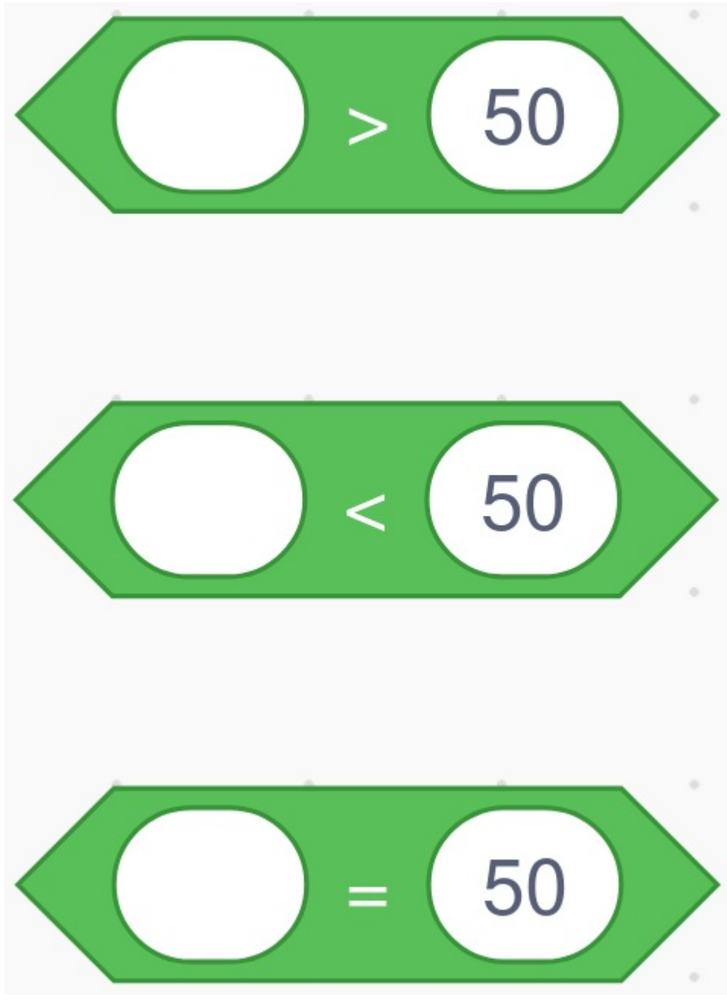

Lógicas
Relacionais
Expressões Booleanas

Expressões booleanas: dois grandes tipos

lógicas

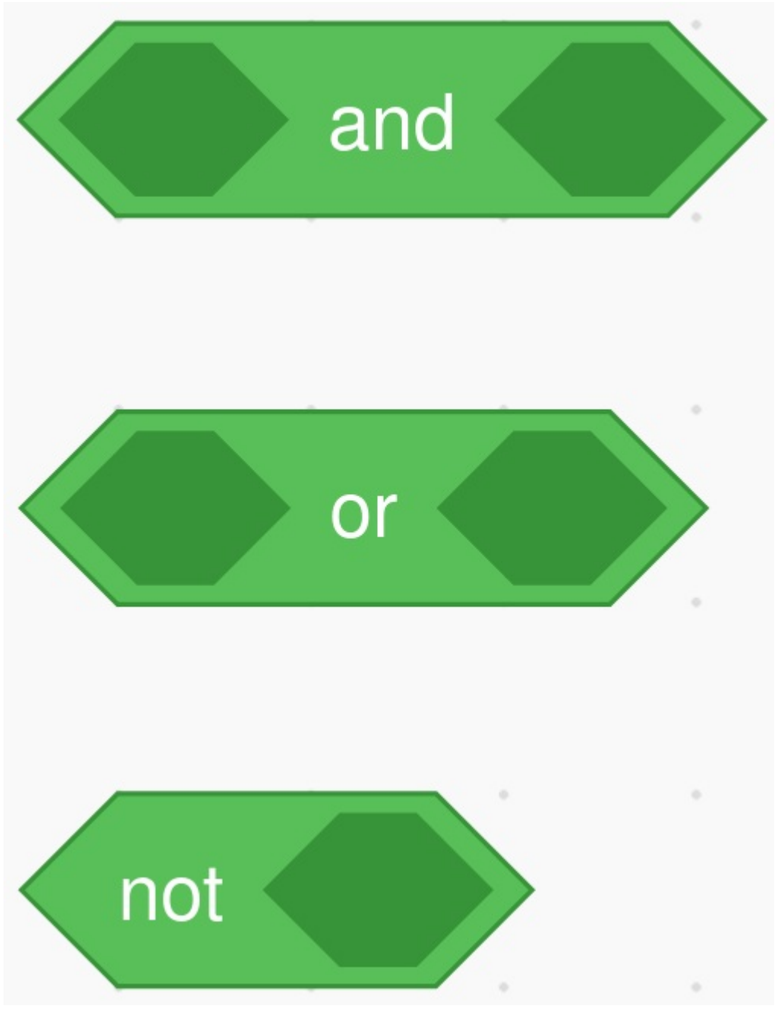


relacionais



Expressões booleanas lógicas:

lógicas



AND (E): retorna TRUE somente se ambas as condições forem verdadeiras

OR (OU): retorna TRUE se uma das condições for verdadeira.

NOT (NÃO): retorna TRUE se a condição for FALSE, e FALSE se a condição for TRUE.

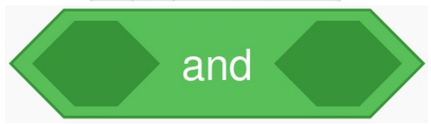
Algumas linguagens possuem:

- **XOR (OU exclusivo)**
- **NOR (NOT OR)**
- **NAND (NOT AND)**
- **XNOR (NOT XOR)**

Expressões booleanas lógicas: tabelas-verdade

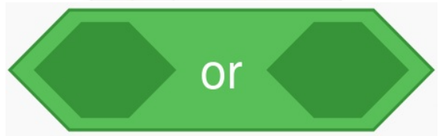
AND

A	B	A AND B
V	V	V
V	F	F
F	V	F
F	F	F



OR

A	B	A OR B
V	V	V
V	F	V
F	V	V
F	F	F



NOT

A	NOT A
V	F
F	V

⇦ **Diariamente**

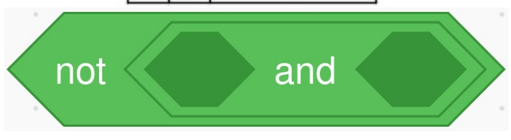
XOR

A	B	A XOR B
V	V	F
V	F	V
F	V	V
F	F	F

⇦ **Pouco**

NAND

A	B	A NAND B
V	V	F
V	F	V
F	V	V
F	F	V



NOR

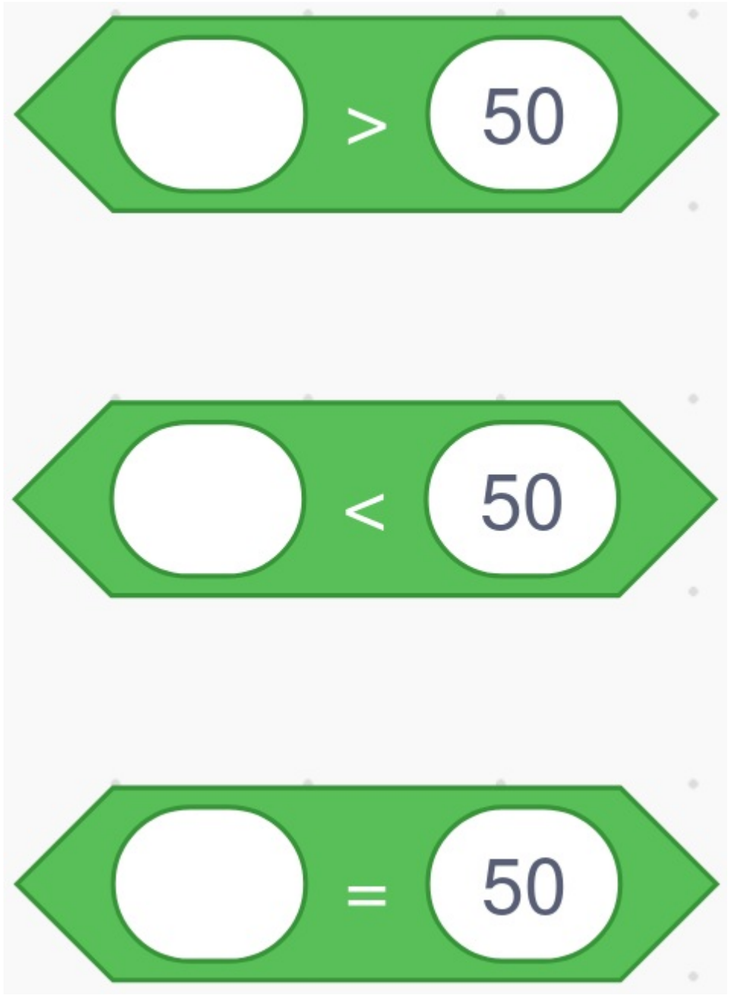
A	B	A NOR B
V	V	F
V	F	F
F	V	F
F	F	V

XNOR

A	B	A XNOR B
V	V	V
V	F	F
F	V	F
F	F	V

⇦ **Raro**

Expressões booleanas relacionais



> (maior do que): retorna TRUE somente se a condição for maior do que um valor

< (menor do que): retorna TRUE somente se a condição for menor do que um valor.

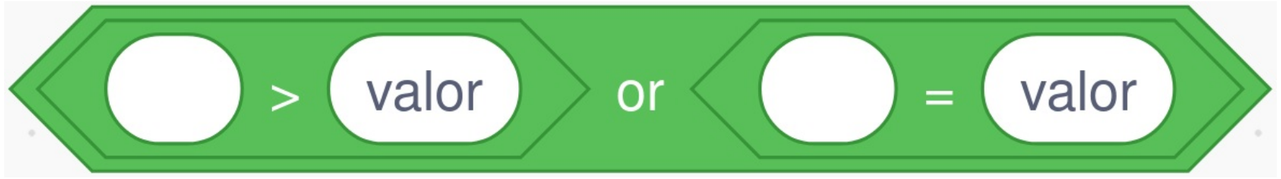
= (igual a): retorna TRUE se a condição for igual a um valor.

Algumas linguagens possuem:

- **>= (maior do que ou igual a)**
- **<= (menor do que ou igual a)**
- **<> ou != (diferente de)**

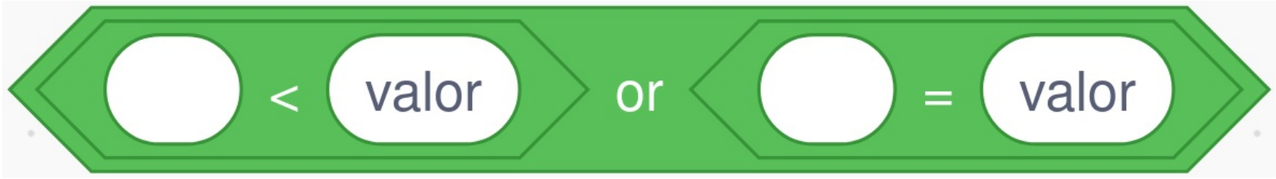
Expressões booleanas relacionais

> (maior do que)



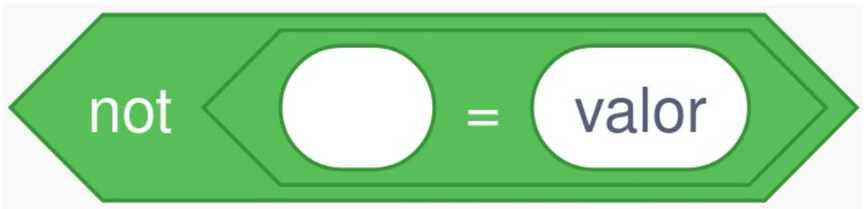
>= (maior do que ou igual a)

< (menor do que)



<= (menor do que ou igual a)

= (igual a)



!= ou <> (diferente de)

Expressões booleanas podem ser combinadas de maneiras complexas

A combinação de expressões booleanas nos permite criar controles sofisticados do fluxo do nosso programa. O domínio de lógica matemática é essencial.

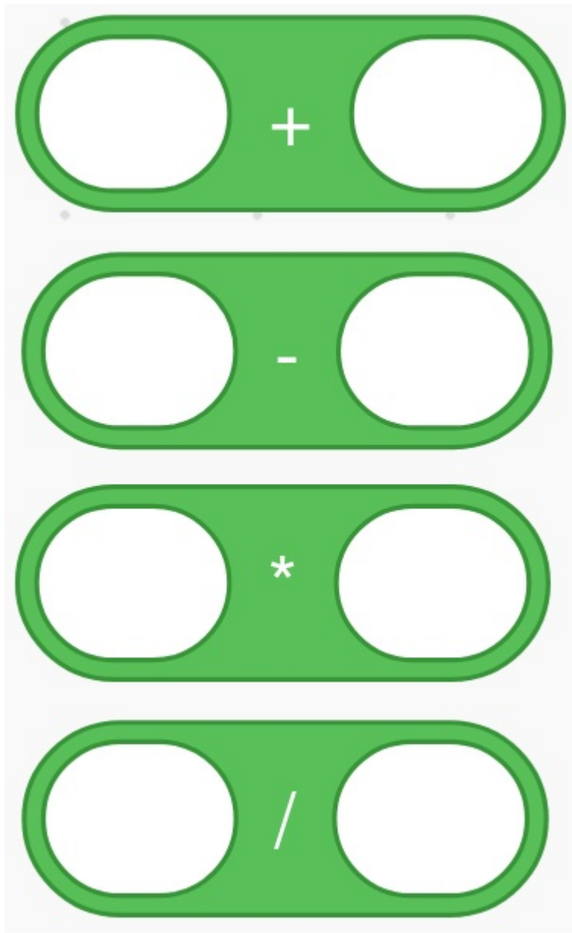
Scratch

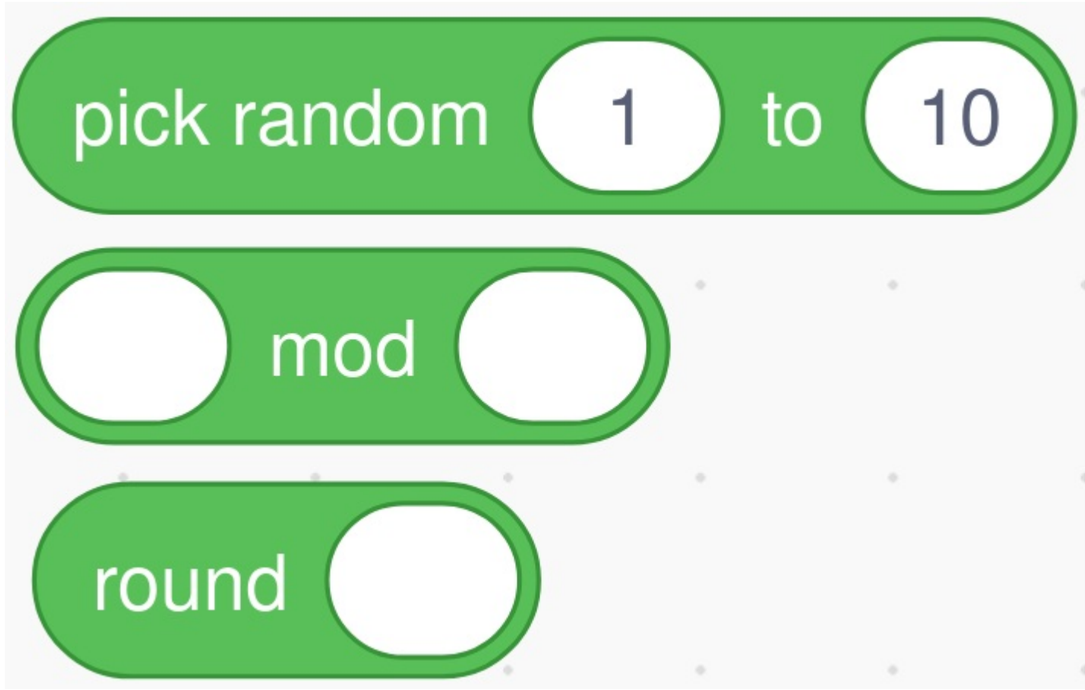
Expressões Matemáticas

Expressões para Texto

Expressões matemáticas

Scratch fornece blocos aritméticos e de outros tipos:





Expressões matemáticas

Scratch fornece blocos aritméticos e de outros tipos:

Funções matemáticas pré-definidas:

Básicas:
- abs
- floor
- ceiling

Trigonométricas:
- sin
- cos
- tan
- asin
- acos
- atan

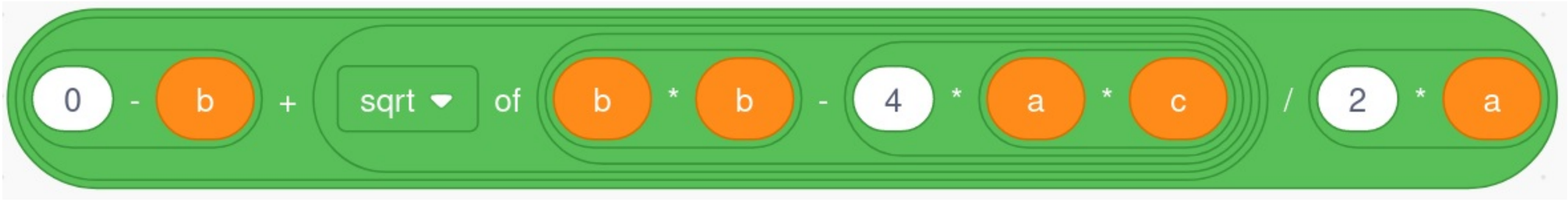
Potenciação e logarítmicas:
- sqrt
- 10^
- e^
- log
- ln

Expressões matemáticas

Ao escrever expressões matemáticas, devemos ter cuidado: a estrutura das fórmulas matemáticas deve ser transformada em uma expressão "plana", com a ordem apropriada das operações indicada por aninhamento e/ou parênteses. É causa comum de erros, principalmente por iniciantes.

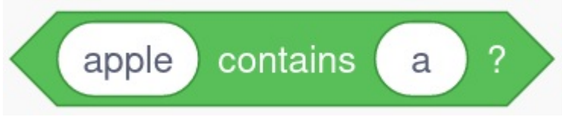
$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$(-b+(\sqrt{b^2-4ac}))/(2a)$$



Funções para textos (strings)

Scratch também fornece funcionalidade para trabalharmos com textos:



Scratch

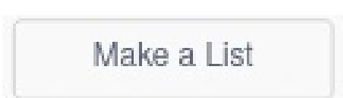
Estruturas de dados

Scratch fornece duas estruturas de dados:

- variável: **É um espaço na memória para o armazenamento de um único valor. Pode corresponder a uma ou mais células de memória.**

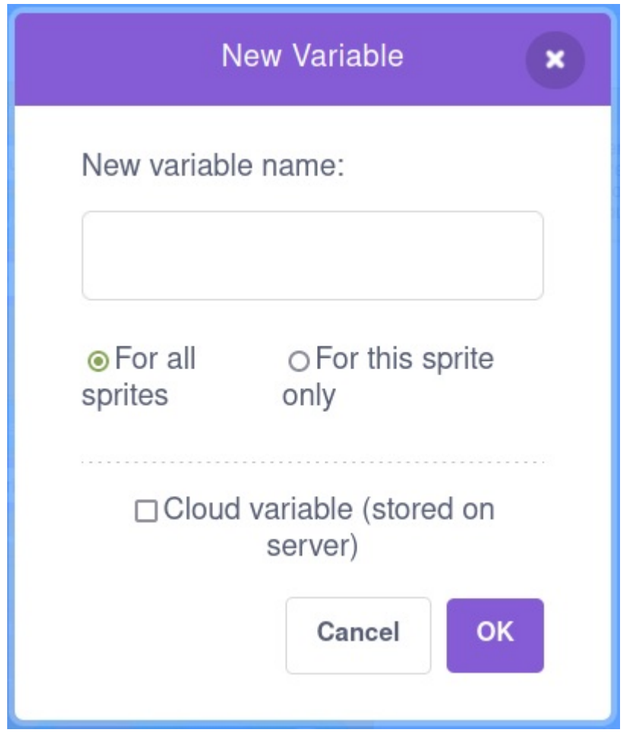
Algumas linguagens obrigam o usuário a escolher o tipo de dado que será armazenado (inteiro, real, caractere, etc). No Scratch não é necessário escolher o tipo de dado.

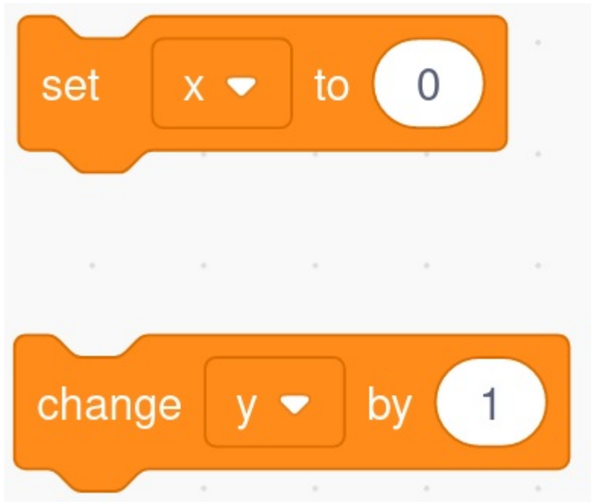
- array: **Também chamado de lista ou vetor, é uma estrutura de dados que armazena mais de um valor ao mesmo tempo. Utilizado em situações nas quais precisamos trabalhar com vários dados semelhantes, por exemplo: as notas dos alunos.**



Variáveis

Ao criar uma variável podemos utilizá-la em diversas partes de nosso programa. Devemos definir se ela será visível apenas no sprite atual ou em todos os sprites.

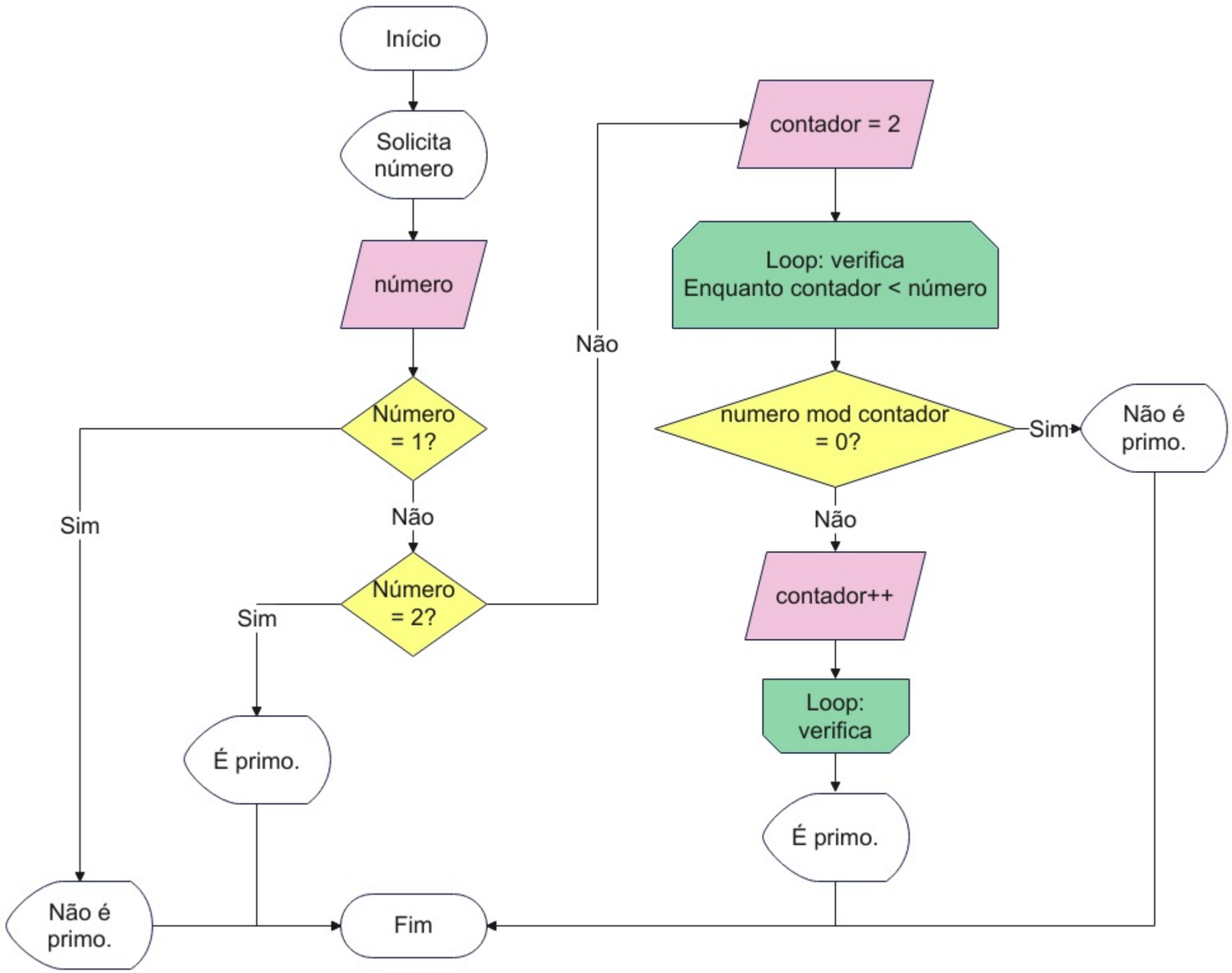


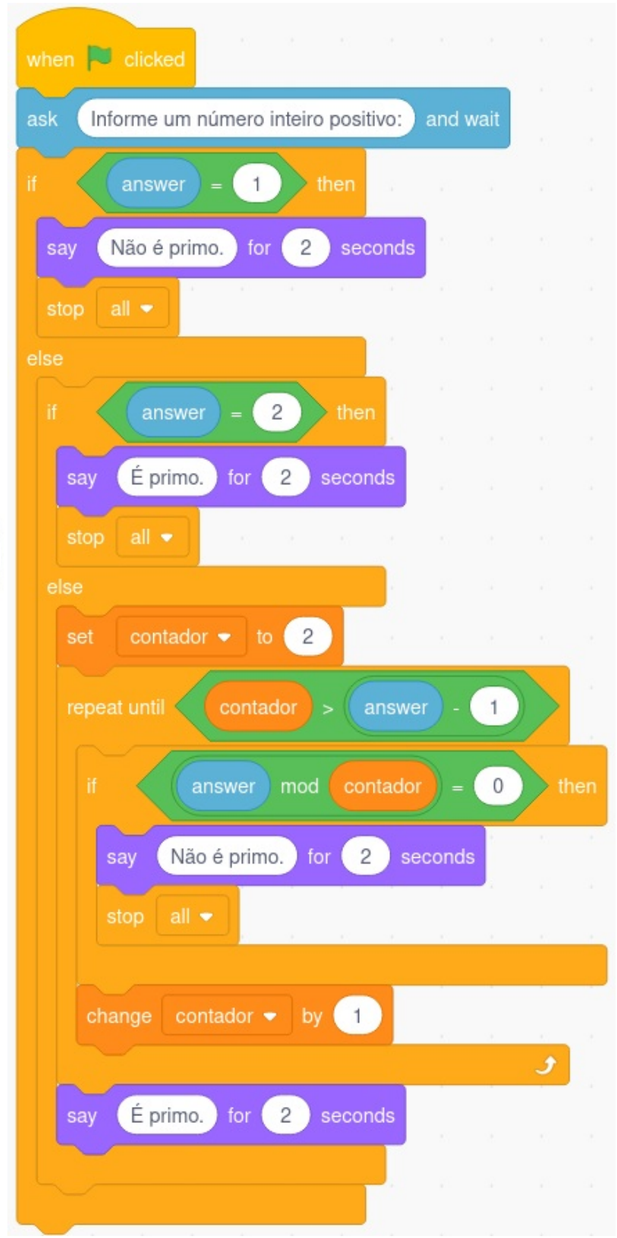


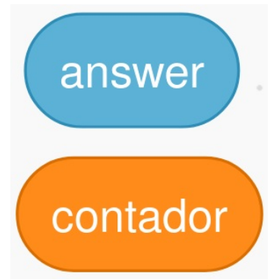
Exemplo do uso de variáveis: determinar se um número é primo

```
01. Solicite um número inteiro positivo ao usuário.
02. Se o número for "1":
03.     Imprima a mensagem "Não é primo."
04.     Termine o programa.
05. Se o número for "2":
06.     Imprima a mensagem "É primo."
07.     Termine o programa.
08. Caso contrário:
09.     Divida o número informado pelo usuário por todos
        os números, de 2 até o antecessor do número informado.
10.     Se em alguma divisão encontrarmos resto 0 (zero):
11.         Imprima a mensagem "Não é primo."
12.         Vá para a linha 16.
13.     Caso contrário:
14.         Imprima a mensagem "É primo."
15.         Vá para a linha 16.
16. Termine o programa.
```

Exemplo do uso de variáveis: determinar se um número é primo

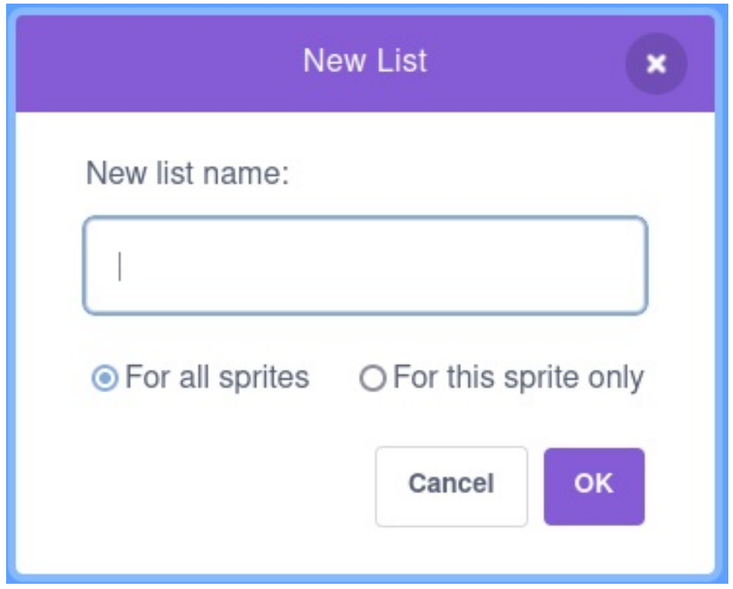


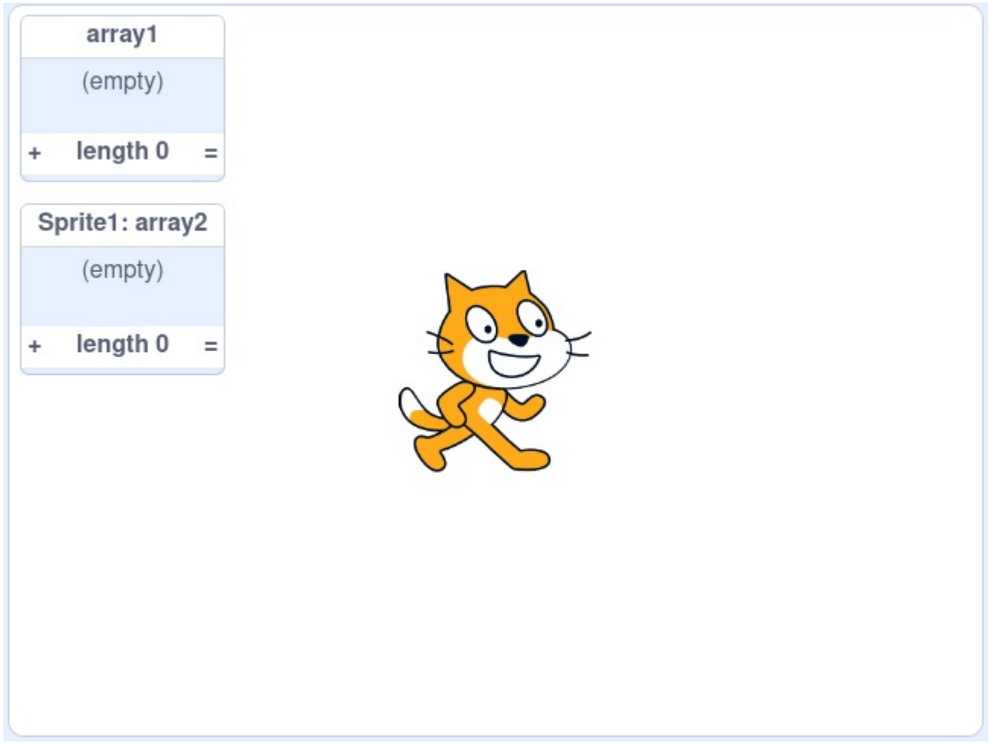




Arrays

Ao criar um array podemos utilizá-lo em diversas partes de nosso programa. Devemos definir se ele será visível apenas no sprite atual ou em todos os sprites.

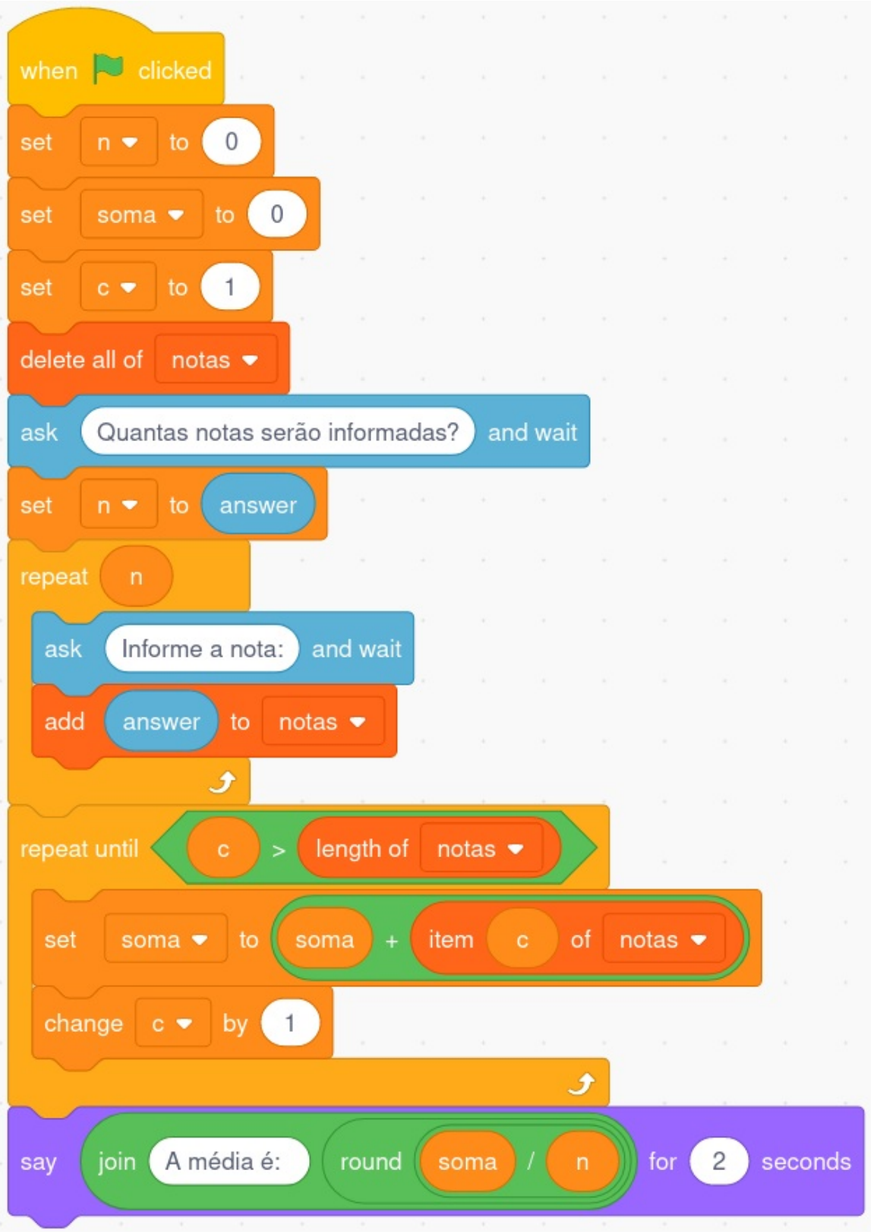


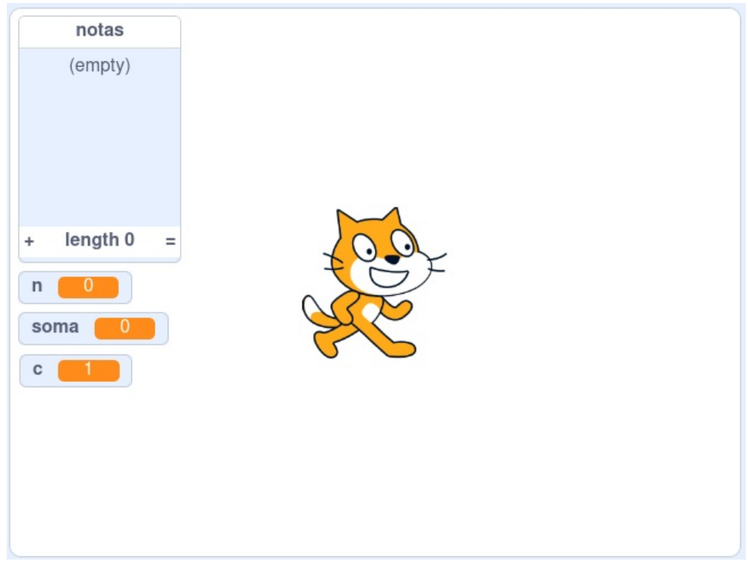


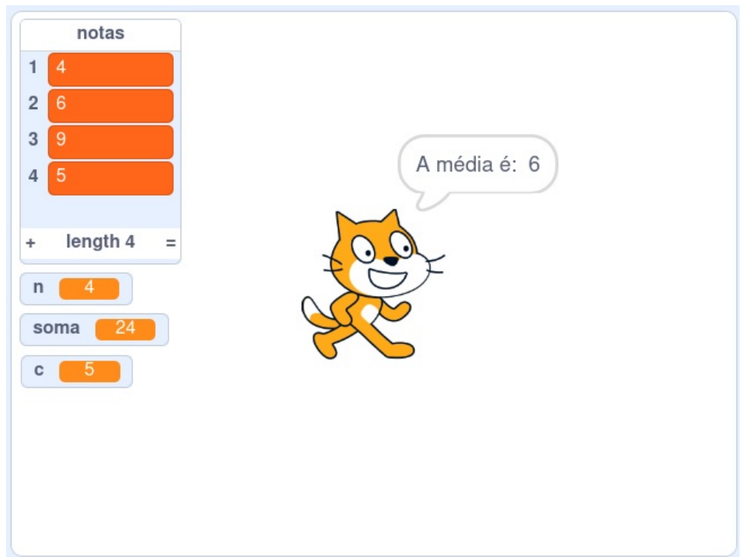
Exemplo de uso de arrays: calcular a média de notas

```
01. Perguntar ao usuário quantas notas serão informadas.
02. Repetir até que todas as notas tenham sido informadas:
03.     Perguntar a nota.
04.     Inserir a nota em uma lista.
05. Percorrer a lista, somando todas as notas.
06. Dividir a soma pelo número de notas, arredondando o resultado.
07. Exibir a média.
08. Terminar.
```

Exemplo de uso de arrays: calcular a média de notas





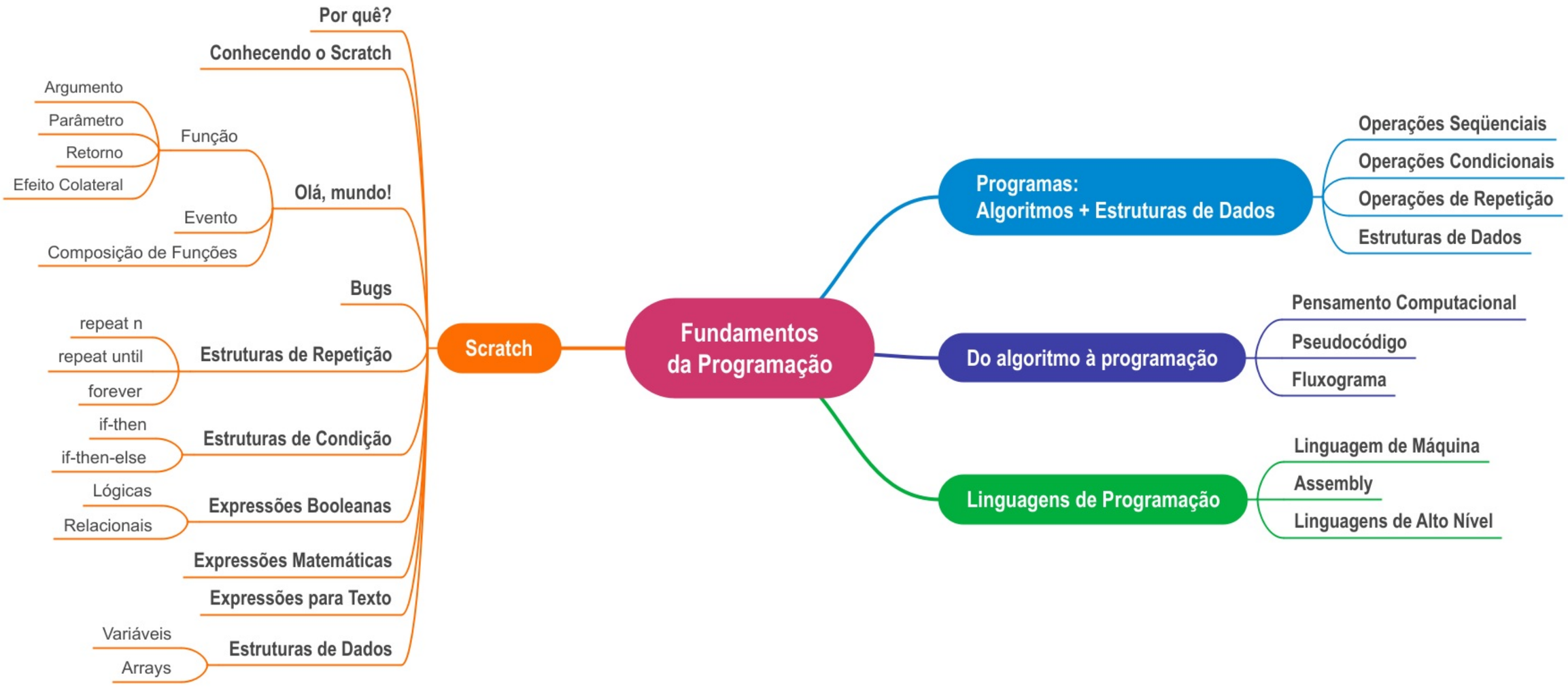


Como o Scratch é utilizado por crianças, a contagem de coisas começa no 1 e não no 0.

Por isso os elementos do array são numerados de 1 em diante.

Em linguagens de programação que não são para crianças, a contagem começa do 0.

Em resumo:


Fundamentos da Programação
Programas: Algoritmos + Estruturas de Dados
Operações Seqüenciais
Operações Condicionais
Operações de Repetição
Estruturas de Dados
Do algoritmo à programação
Pensamento Computacional
Pseudocódigo
Fluxograma
Linguagens de Programação
Linguagem de Máquina
Assembly
Linguagens de Alto Nível
Scratch
Por quê?
Conhecendo o Scratch
Olá, mundo!
Função
Argumento
Parâmetro
Retorno
Efeito Colateral
Evento
Composição de Funções
Bugs
Estruturas de Repetição
repeat n
repeat until
forever
Estruturas de Condição
if-then
if-then-else
Expressões Booleanas
Lógicas
Relacionais
Expressões Matemáticas
Expressões para Texto
Estruturas de Dados
Variáveis
Arrays