

LÓGICA DE PROGRAMAÇÃO COM SCRATCH

Algoritmos: informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. Um algoritmo é então uma sequência de passos computacionais que transformam a entrada em uma saída.

Em suma, trata-se de uma sequência de passos que realizam cálculos sobre algum dado informado para se atingir um objetivo, representado pelo resultado desses cálculos.

Ações do dia a dia podem ser consideradas como algoritmos. Exemplo:

Algoritmo — Trocar a lâmpada

- Passo 1 — Pegar a escada;
- Passo 2 — Posicionar a escada embaixo da lâmpada queimada;
- Passo 3 — Desligar o disjuntor;
- Passo 4 — Subir na escada;
- Passo 5 — Retirar a lâmpada queimada;
- Passo 6 — Descer da escada com a lâmpada queimada;
- Passo 7 — Pegar uma lâmpada nova;
- Passo 8 — Subir na escada;
- Passo 9 — Colocar a lâmpada nova;
- Passo 10 — Descer da escada;
- Passo 11 — Ligar o disjuntor;
- Passo 12 — Ligar o interruptor;
- Passo 13 — Se a lâmpada não acender então volte ao Passo 3;
- Passo 14 — Guardar a escada;
- Passo 15 — Jogar a(s) lâmpada(s) queimada(s) no lixo.

(adaptado de CASTELLANI(2020))

Dois aspectos importantes sobre algoritmos podem ser notados no exemplo anterior:

1. Se alguns passos forem eliminados, teremos uma tarefa incompleta ou sem qualidade (ex.: poderíamos emitir os passos 3, 6, 7 e 8, mas teríamos a tarefa de trocar lâmpada incompleta e insegura)
2. Se os passos tiverem sua ordem alterada poderão “quebrar” a tarefa (ex.: não poderemos trocar o passo 4 pelo passo 2, pois estaríamos substituindo a lâmpada antes mesmo de retirar a antiga).

Os algoritmos podem expressos utilizando diagramas, linguagem natural (próxima da linguagem escrita) ou linguagem específica. O algoritmo será “traduzido” posteriormente em programa ao ser escrito com linguagem de programação (como o C#).

Com o Scratch podemos treinar os conceitos de algoritmo e programação de maneira visual, lúdica e com resultados observáveis na hora.

Sequências de instruções

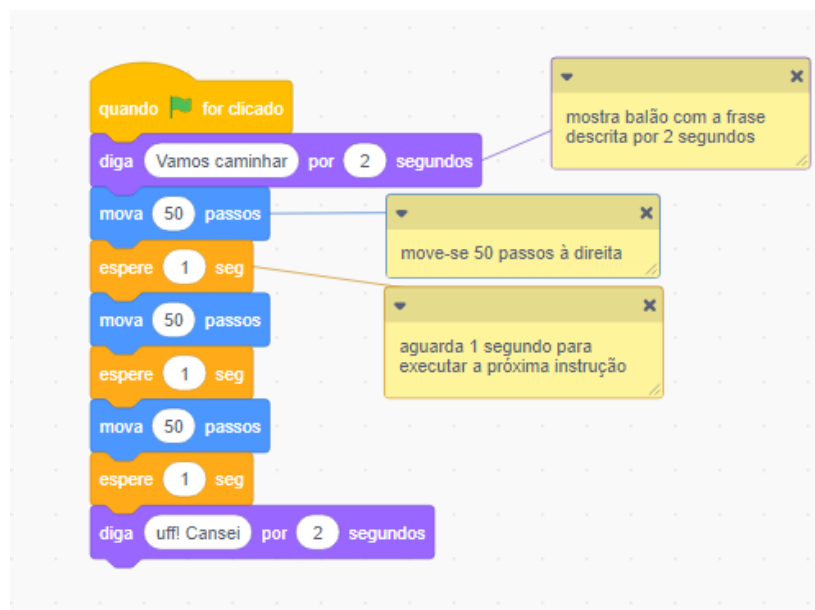
Como definido anteriormente, algoritmo é uma sequência de passos computacionais que transformam dados de entrada em uma informação de saída. Esses passos podem ser definidos como “declarações” (declaração de variável, declaração de laços, declaração de condicionais entre outras).

Para computar as saídas a partir de uma entrada, várias declarações são necessárias e elas serão executadas sequencialmente. Podem ser usadas para realizar cálculos, armazenar informações em variáveis, repetir uma sequência de instruções ou executar uma instrução ou outra de acordo com uma condição.

Separamos assim em dois tipos de declarações:

- Declarações simples: uma instrução que realizará uma ação apenas (um bloco comum no Scratch).
- Declarações compostas: essas trazem dentro de si uma sequência de outras instruções que serão executadas, geralmente, de acordo com uma instrução (blocos laranja do Scratch, como “se então” e “sempre”).

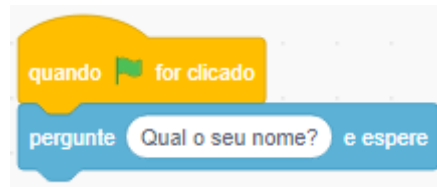
Exemplo: no Scratch crie um novo projeto, e no sprite do gato inclua instruções para que o gato ande 50 passos e pare por 1 segundo três vezes. Ao final ele diz que se sente cansado. Programe como na figura a seguir.



Desafio 1: no exercício anterior, note que o gato para em um canto da tela e quando rodamos o programa de novo (clicando na bandeira verde) ele inicia nessa posição, ao invés de voltar ao centro da tela. Vamos incluir uma instrução para que, antes de andar, ele volte à posição $x=0$ e $y=0$ (bloco “vá para”). Posicione esse bloco corretamente entre as instruções para alcançar o efeito desejado.

Entradas e saídas

Parte essencial em um algoritmo é informar dados para o algoritmo processar. Com o Scratch isso pode ser alcançado com o bloco “pergunte”. Ele é configurado com uma frase que é a pergunta que o personagem vai mostrar na tela; em seguida, vai abrir uma caixa onde o usuário pode digitar sua resposta.

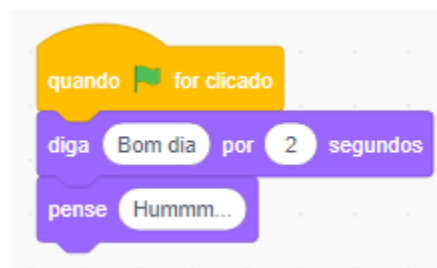


A resposta de um bloco “pergunte” está armazenado na variável “resposta” nos blocos do tipo “sensores”.

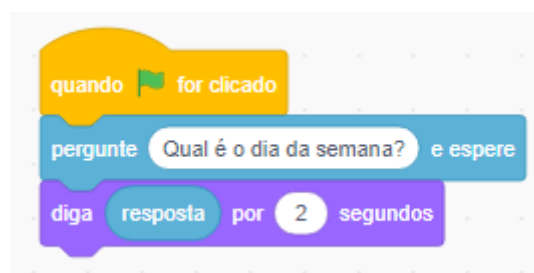


A saída de dados pode ser expressa de algumas maneiras:

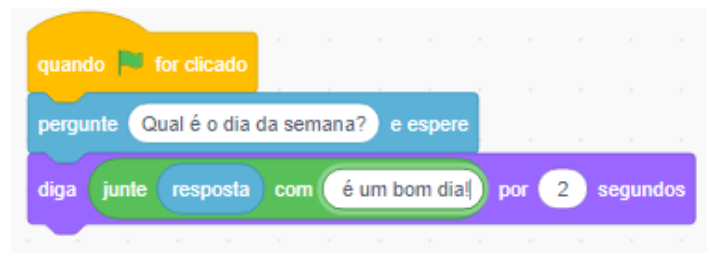
- Blocos “diga” que mostram balão de diálogo com um texto (há opção de mostrar rapidamente ou por um número definido de segundos)
- Blocos “pense” que mostram balão de pensamento com um texto (há opção de mostrar rapidamente ou por um número definido de segundos)



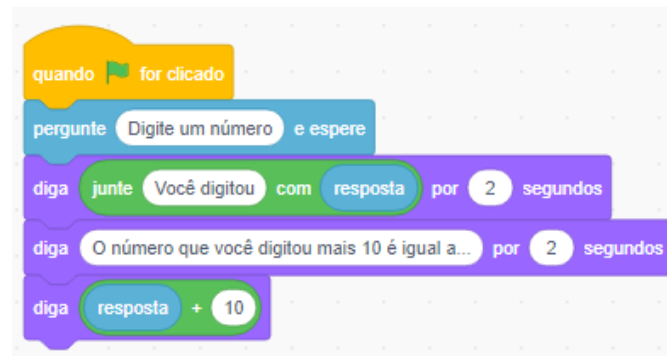
Podemos combinar entrada de dados e saída de dados como a seguir.



Pode ser útil utilizar o bloco “Junte” (categoria “Operadores”) para formar frases mais interessantes:



Também podemos usar a resposta de “pergunte” para armazenar números e até operar com eles.



Exemplo: programa Scratch que faz a tabuada de um número informado pelo usuário.



Nota importante: após realizar uma pergunta, caso faça uma segunda pergunta a variável “resposta” ficará apenas com a última resposta digitada pelo usuário; a anterior será perdida.

Desafio 2: monte um programa Scratch que peça que o usuário digite sua idade. Depois faça o gato pensar a idade digitada e, muito educadamente, faça-o dizer “você ainda é muito jovem”.

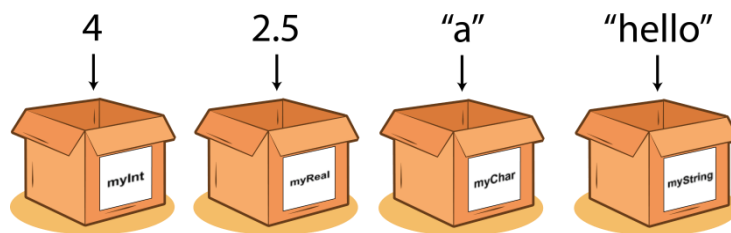
Desafio 3: monte um programa Scratch para calcular metade de um número. O usuário irá digitar um número e o gato em seguida fará o cálculo e dirá qual é a metade dele.

Desafio 4: monte um programa Scratch para fazer o gato andar para a direita e para cima. Primeiro posicione-o no centro com o bloco “vá para”. Depois pergunte ao usuário quantos passos ele deve andar para a direita; use o número digitado para mover-se com o bloco “mova”; em seguida pergunte ao usuário quantos passos ele deve andar para cima; use o número digitado para mover-se com o bloco “adicione...a y”. Ao final faça-o dizer “missão cumprida”.

Variáveis

Em algoritmos, geralmente é útil e necessário guardar os resultados de um cálculo intermediário e ser capaz de reutilizar essa informação em algum momento posterior do código. Isso é possível através de variáveis. Em uma variável nós podemos guardar dados numéricos (inteiro, decimal...), valores textuais ou lógicos (verdadeiro/falso). Uma variável pode ainda ser uma lista de elementos.

Pense em uma variável como uma caixa localizada na memória do computador onde você poderá guardar uma informação. Essa caixa possui um nome para facilitar a sua localização. Depois, quando você precisar da informação guardada, você usará o nome da caixa para obter o dado que necessita, podendo fazer novos cálculos e até substituir, posteriormente, o valor da variável.



Fonte: <https://medium.com/@stevenpcurtis.sc/what-is-a-variable-3447ac1331b9>

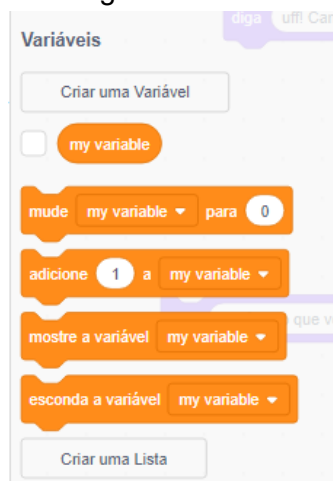
Na figura acima, temos as variáveis “myInt” com valor 4 (número inteiro), “myReal” com valor 2.5 (número decimal), “myChar” com valor “a” (caractere) e “myString” com o valor “hello” (texto).

Já utilizamos uma variável pronta no Scratch: a variável “resposta” que sempre carrega o valor da última resposta digitada pelo usuário ao usar o bloco “pergunte”.

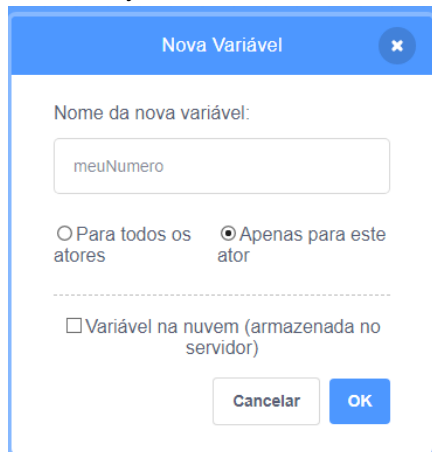
Em qualquer algoritmo, variáveis devem ser primeiro criadas (declaradas) e inicializadas (receber um valor inicial), dando-lhe um nome. É importante que o nome da variável seja claro, para ajudar a leitura do código.

O Scratch permite a criação de variáveis:

- Selecione o ator ao qual quer incluir blocos de programação
- Vá à região “Variáveis” no menu lateral. Clique em “Criar uma variável”

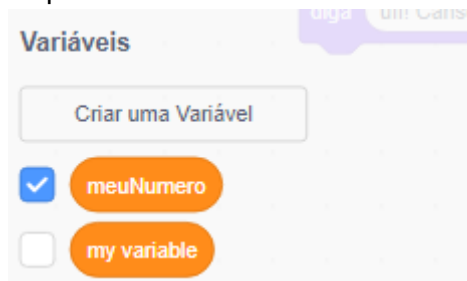


- Dê um nome à variável. Você pode escolher se a variável será usada por outros objetos na cena do Scratch ou apenas pelo ator atual.



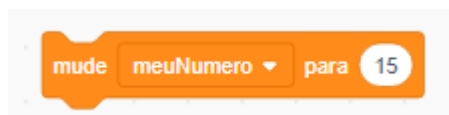
A dialog box titled "Nova Variável" with a close button (X) in the top right corner. It contains a text input field labeled "Nome da nova variável:" with the text "meuNumero" entered. Below the input field are two radio buttons: "Para todos os atores" (unselected) and "Apenas para este ator" (selected). At the bottom, there is a checkbox labeled "Variável na nuvem (armazenada no servidor)" which is unchecked. At the very bottom are two buttons: "Cancelar" and "OK".

- A nova variável aparece na lista e pode ser usada. O a caixa selecionável à esquerda indica se a variável ficará visível ou não na tela da aplicação Scratch.



As operações sobre a variável são de:

- Escrita com o bloco "mude":

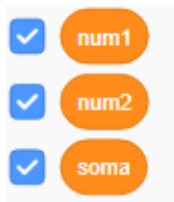


- Leitura, com diversos blocos:

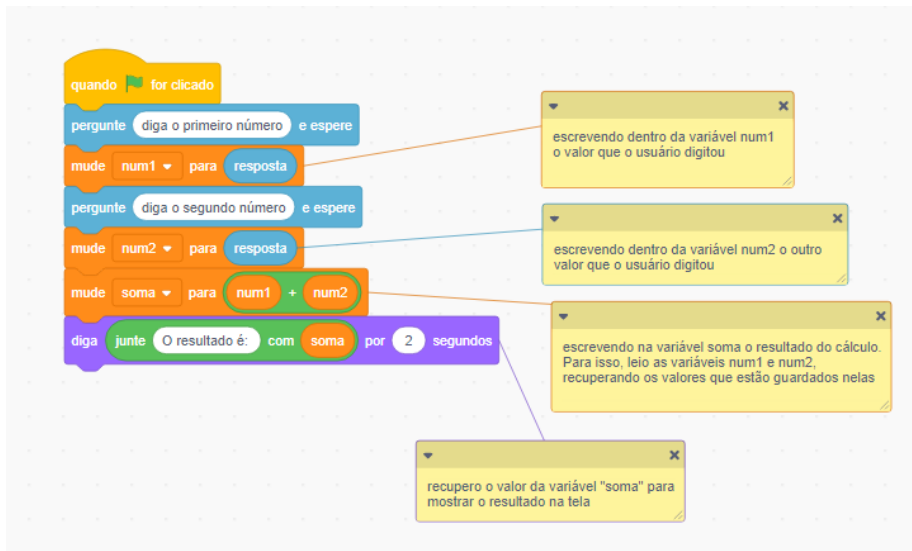


Vamos agora analisar um exemplo em que a variável se torne indispensável. Pense em uma situação em que você precisa programar um código para somar dois números informados pelo usuário. Antes de continuar o texto, tente fazer isso sem nenhuma variável.

É bem provável que você tenha tido dificuldade em achar uma solução. Agora vejamos como fica simples com uma variável. Primeiro declaramos algumas variáveis:



Em seguida podemos montar o código usando-as:



Desafio 5: modifique o programa Scratch do desafio 2 para que, além de perguntar a idade, pergunte o nome do usuário. No final, faça com que o gato diga o nome da pessoa seguido da frase “você ainda é muito jovem”.

Desafio 6: crie um programa Scratch para simular o rolar de 2 dados. Crie as variáveis adequadas e inicialize cada uma usando o bloco “número aleatório entre [1] e [6]”. Dessa maneira, as variáveis receberão um número sorteado. Após isso, faça o gato dizer o número de cada dado e, em seguida, dizer a soma de pontos obtidos com os dados.

Desafio 7: programe no Scratch uma aplicação que receba a idade de uma pessoa. A partir disso, calcule quantos dias a pessoa viveu (considere 1 ano = 365 dias, ignorando bissextos). Faça o gato dizer essa informação na tela. Depois, a partir dos dias, calcule quantas horas a pessoa viveu e faça o gato dizer essa resposta na tela.

Estruturas Condicionais

É bastante comum em algumas situações, durante a execução de uma tarefa termos que tomar decisões. Por exemplo, imagine a tarefa de abrir uma porta:

1. Andar até a porta
2. Se a porta está trancada então
 - a. Destrancar a porta com a chave
3. Girar a maçaneta
4. Se a porta abre empurrando então
 - a. Empurrar a porta
 - Senão
 - b. Puxar a porta

No exemplo, somos apresentados a duas decisões: uma para destrancar a porta apenas se a porta está trancada (a afirmação “porta trancada” é uma verdade). A outra para decidir se a porta deve ser empurrada ou puxada para abri-la (se a afirmação “porta abre empurrando” é verdadeira, então empurro a porta; se a afirmação for falsa, então puxo a porta).

Decisões são importantes em um algoritmo, pois podem alterar o fluxo das instruções, incluindo condições para que um conjunto de instruções seja executado. Para isso usamos estruturas condicionais, que podem executar um conjunto de instruções se e apenas se a condição apresentada for verdadeira ou pode apresentar um conjunto de instruções para a afirmação verdadeira e outro para quando for falsa.

De maneira geral:

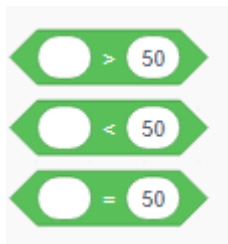
```
SE (condição é verdade) ENTÃO
    Executa conjunto de instruções
FIM SE

Ou

SE (condição é verdade) ENTÃO
    Executa conjunto 1 de instruções
SENÃO
    Executa conjunto 2 de instruções
FIM SE
```

Nos algoritmos as afirmações certamente são muito mais matemáticas e menos vagas do que “porta está trancada”, por exemplo. Geralmente envolvem comparações entre valores, como, por exemplo, $10 > 5$, ou por variáveis que contenham valores, como por exemplo $x < y$ ou $num1 = num2$.

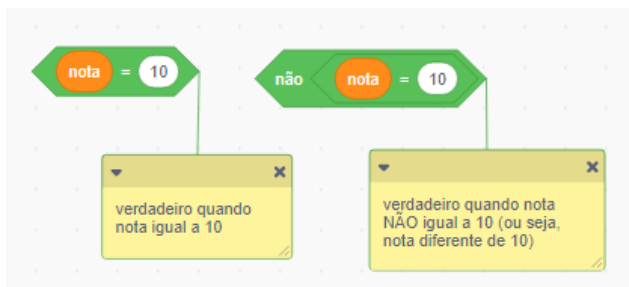
Para condições o Scratch provê alguns operadores como os seguintes:



Além disso, podemos combiná-los com operadores E (usado quando queremos que duas afirmações sejam verdadeiras) e OU (usado quando queremos que ao menos uma das afirmações seja verdadeira). Veja os exemplos.



Há ainda a possibilidade de inverter uma condição com o operador lógico “não”.



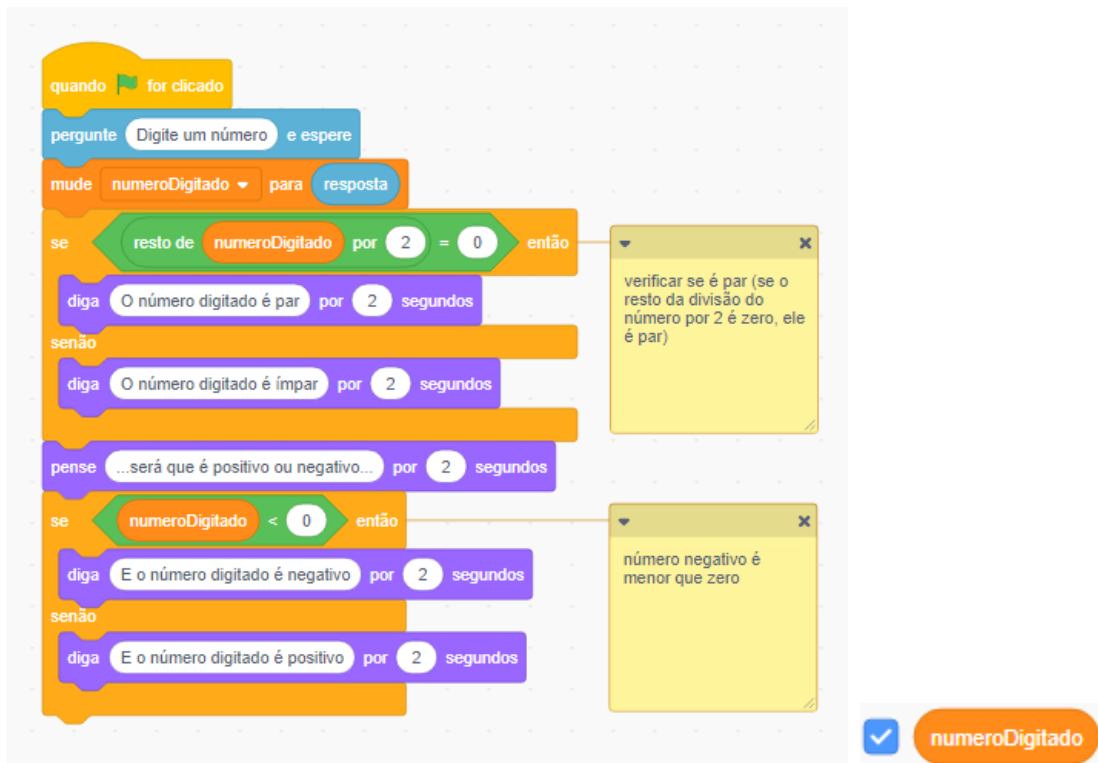
Combinamos então essas condições com as estruturas condicionais disponíveis no Scratch. Na imagem abaixo à esquerda estão os blocos correspondentes ao “se-então” e à direita ao “se-então/senão” explicados acima.



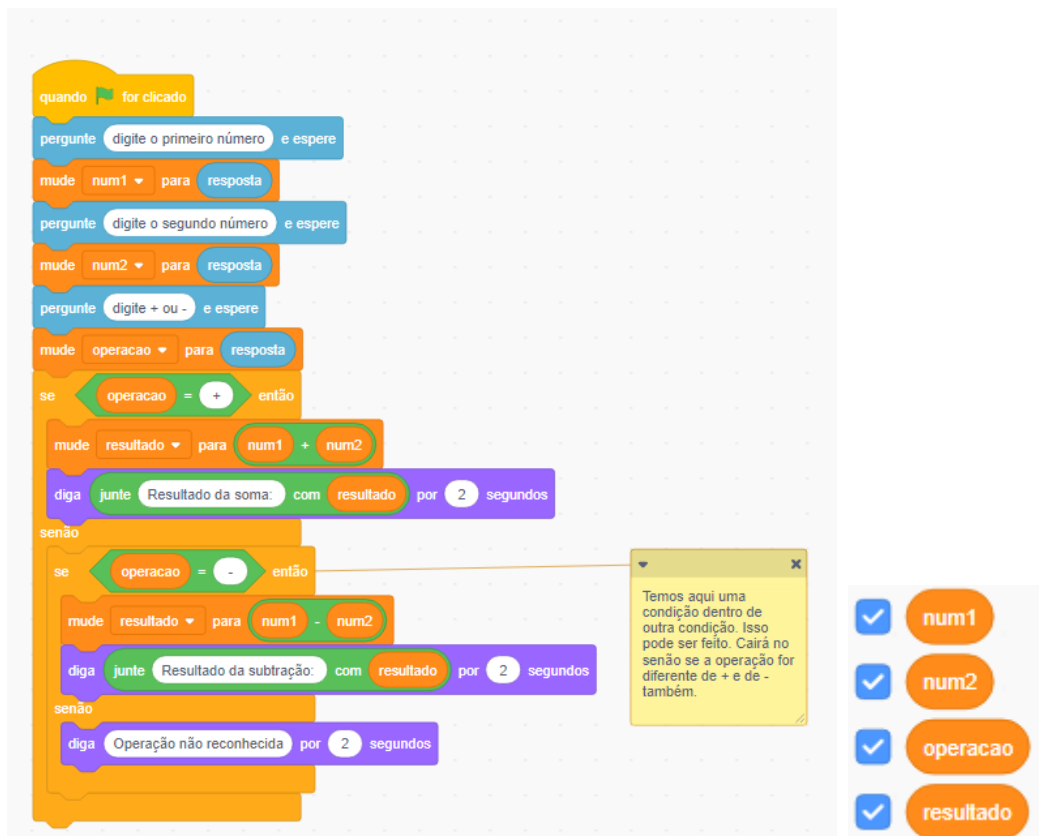
Agora vamos usar as condições nos blocos “se-então”.



Exemplo: verificando se um número digitado pelo usuário é par ou ímpar e se é negativo ou positivo.



Exemplo: calculadora de soma e subtração.



Desafio 8: faça um programa no Scratch para mover o personagem para a direita ou esquerda. Primeiro posicione o personagem no centro da tela; depois pergunte ao usuário para qual lado quer mover; se ele escrever “direita”, mova 100 passos à direita; caso contrário, mova 100 passos à esquerda.

Desafio 9: expanda o exemplo anterior (calculadora de soma e subtração) para incluir ainda a opção de multiplicação.

Desafio 10: faça um código Scratch para montar um “par ou ímpar” simples. O personagem armazenará em uma variável um valor aleatório entre 1 e 5 usando o bloco “número aleatório”. Depois o usuário digitará um número. Ao fim, somar os 2 números e verificar se deu par ou ímpar. Para escrever na tela quem venceu, considere que o personagem sempre escolherá ímpar e o usuário sempre escolherá par.

Estruturas de Repetição

Assim como a capacidade de realizar decisões, a possibilidade de repetir ações é algo muito útil em algoritmos. Isso evita uma repetição desnecessária de um bloco de código no algoritmo – ao invés de reescrever, ordenamos apenas que o código execute um certo número de vezes.

Leve-se ainda em consideração que um código pode ser repetido por um número indefinido de vezes – e nesse caso, depender de alguma condição que o faça parar.

De maneira geral, definimos assim os laços de repetição:

Repita (N vezes)

Executa conjunto de instruções

Fim Repita

Ou

Repita enquanto (uma condição é verdadeira)

Executa conjunto de instruções

Fim Repita

Ou

Repita até que (uma condição se torne verdadeira)

Executa conjunto de instruções

Fim repita

No Scratch temos laços de repetição (da esquerda para a direita na figura) para repetir um número definido de vezes; repetir indefinidamente; repetir até que uma condição seja verdadeira.



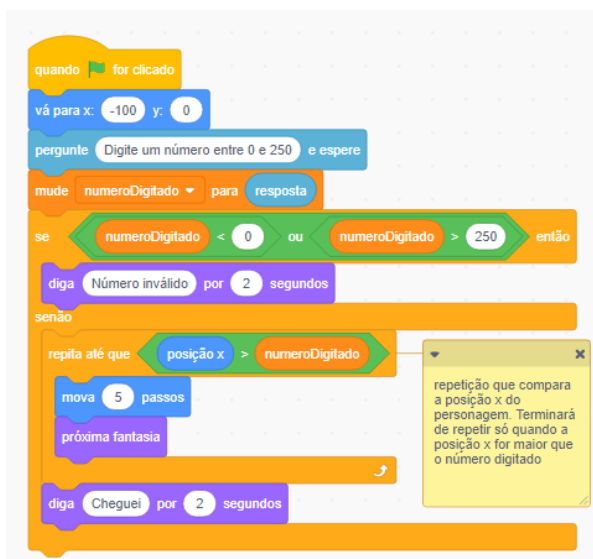
Exemplo: contando de 1 a 10



Exemplo: controlando o personagem para a direita ou para a esquerda



Exemplo: fazer o personagem caminhar até uma coordenada x do cenário



Desafio 11: Inspirado no exemplo da tabuada, refaça o programa Scratch desta vez usando laços de repetição para montar uma tabuada de um número informado pelo usuário. Serão necessárias variáveis.

Desafio 12: faça um programa que peça que o usuário digite uma palavra. Enquanto o usuário informar algo diferente de “Scratch”, peça que digite novamente uma palavra. Após digitar “Scratch”, encerre a repetição e finalize o programa com uma mensagem de “até logo”.

Referências:

CASTELLANI, Valk. Algoritmo, definição e exemplos simples do dia a dia. Disponível em <<https://medium.com/@valkcastellani/algoritmo-definição-e-exemplos-simples-do-dia-a-dia-343715536f65>> . Acesso em 07/2020.

GADOU, Benoit. Initiation to Algorithms with Scratch. 2019.