

Instituto Federal de Educação, Ciência e Tecnologia do Paraná - IFPR

disciplina de

ALGORITMOS E LINGUAGENS DE PROGRAMAÇÃO

ESTRUTURAS DE REPETIÇÃO, LAÇOS DE REPETIÇÃO OU LOOPS

Professor Jefferson de Oliveira Chaves
jefferson.chaves@ifpr.edu.br

AGENDA

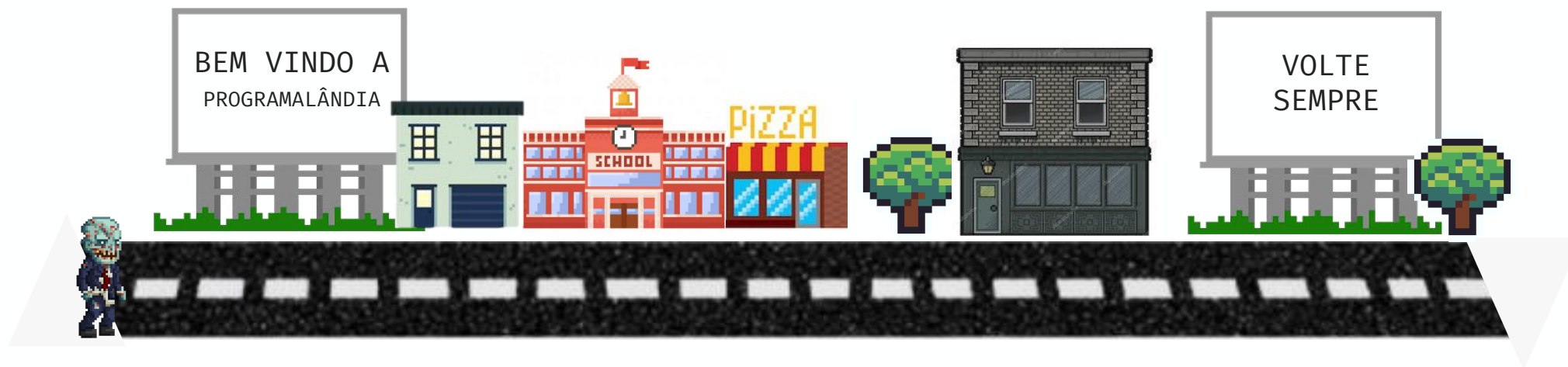
- ∴ Conhecer e compreender as estruturas de repetição;
- ∴ Compreender a utilização das estruturas:
 - ∴ Enquanto;
 - ∴ Para;
 - ∴ Faça Enquanto;



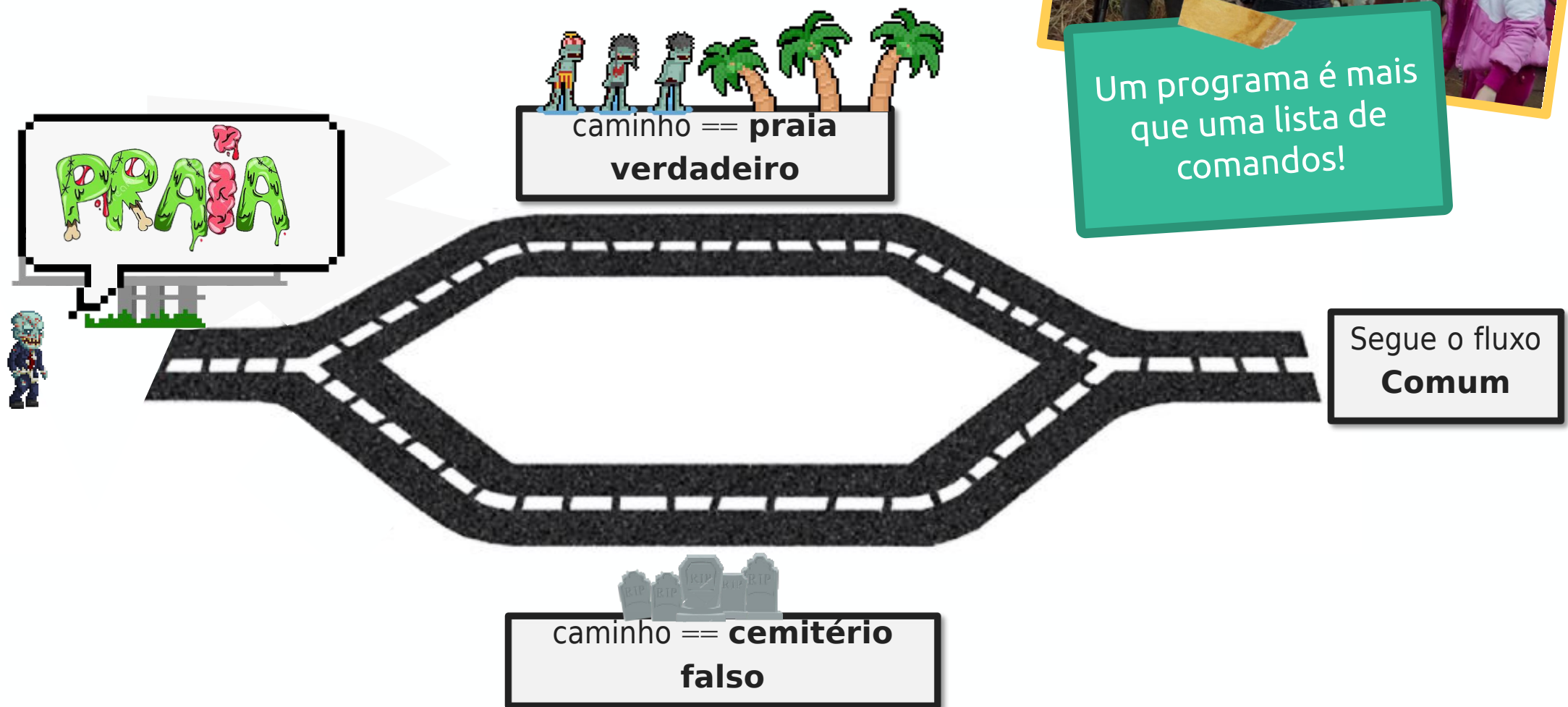
EXECUÇÃO DE UM ALGORITMO



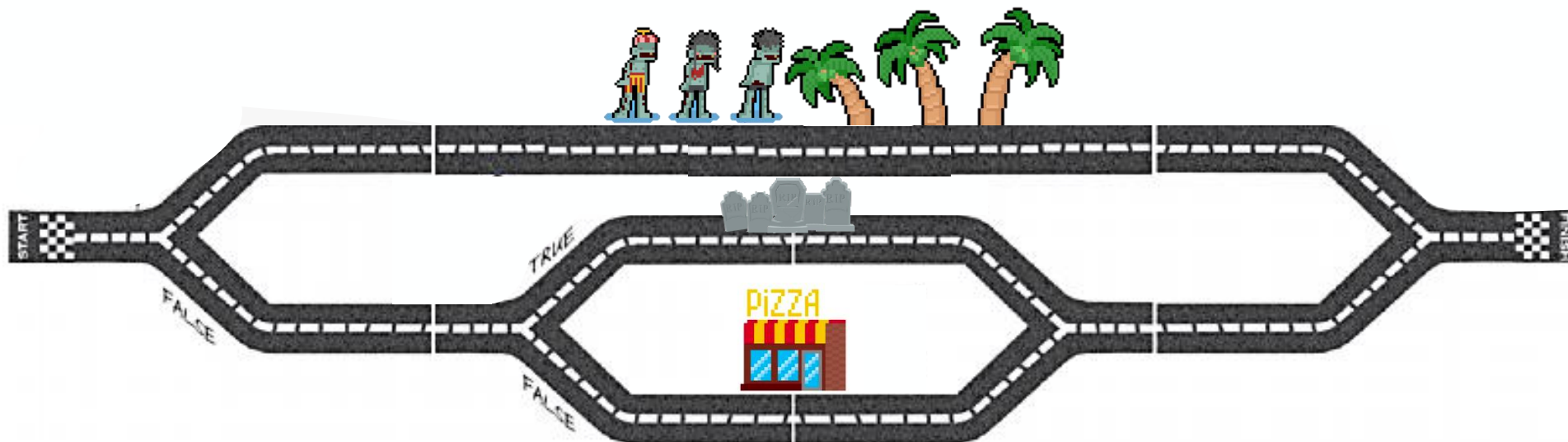
Um programa é mais
que uma lista de
comandos!



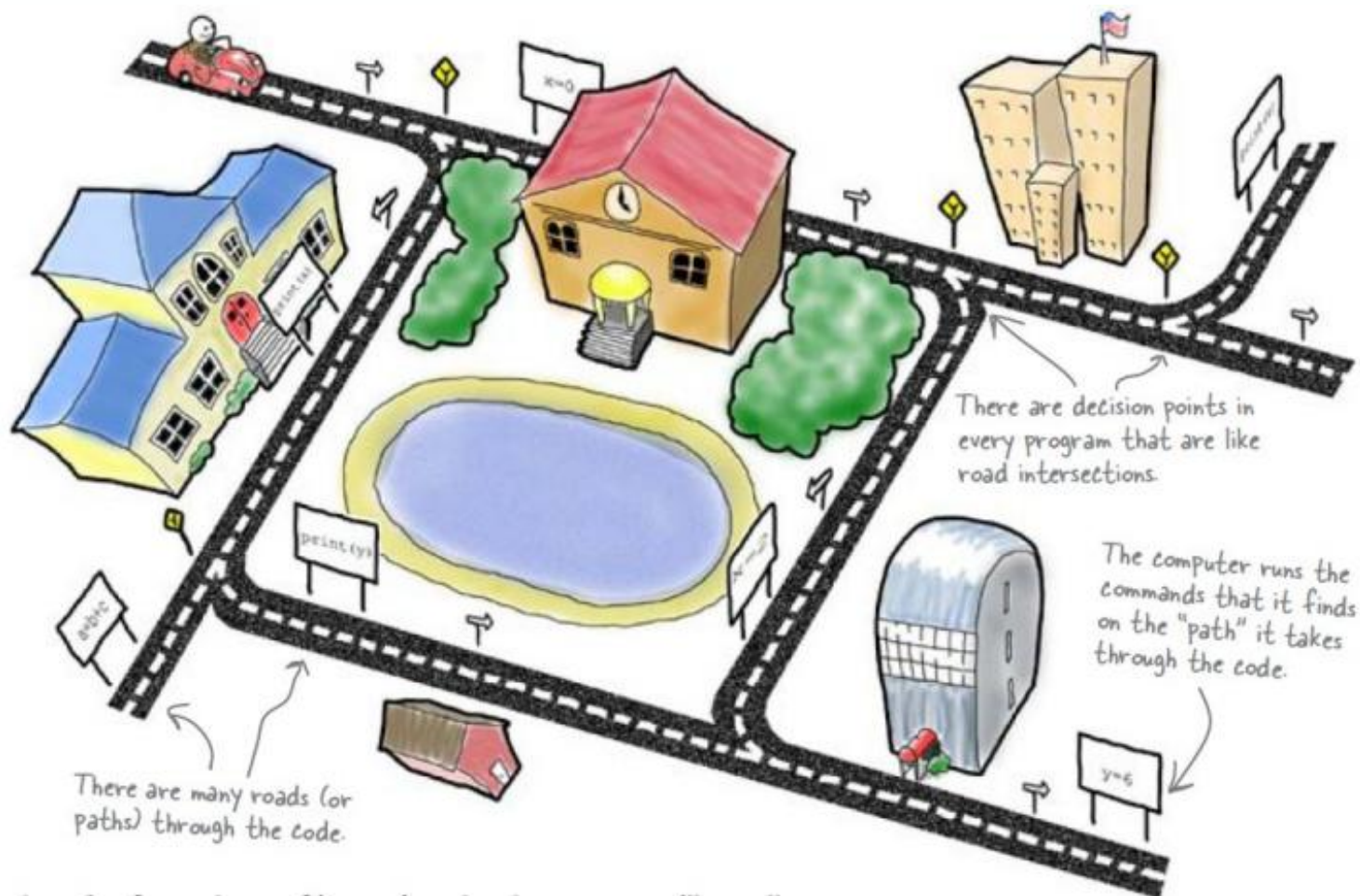
EXECUÇÃO DE UM ALGORITMO



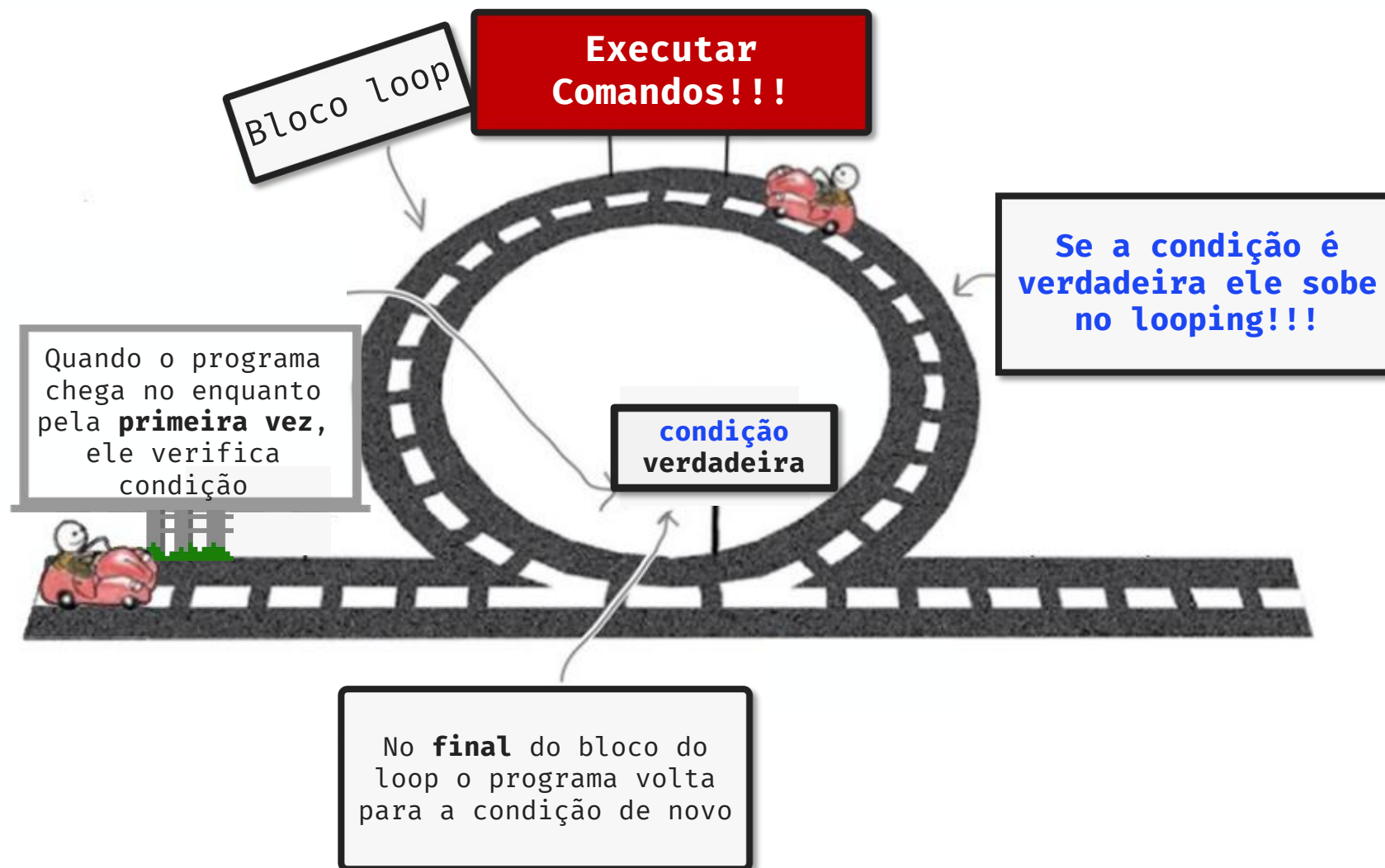
EXECUÇÃO DE UM ALGORITMO



EXECUÇÃO



EXECUÇÃO DE UM ALGORITMO



Quando usar estruturas de repetição?

Como
encontrar
Ariel na fila?



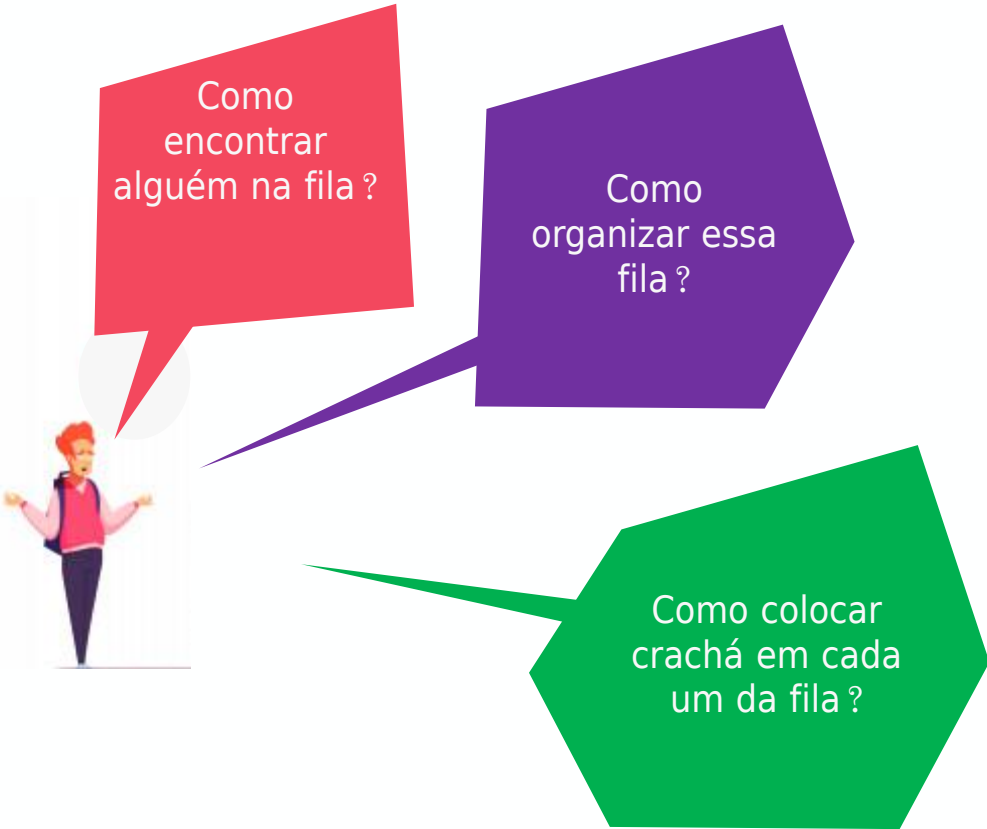
Quando usar estruturas de repetição?

Como
encontrar
Ariel na fila?



Você é Ariel?
Você é Ariel?
Você é Ariel?
Você é Ariel?

Quando usar estruturas de repetição?

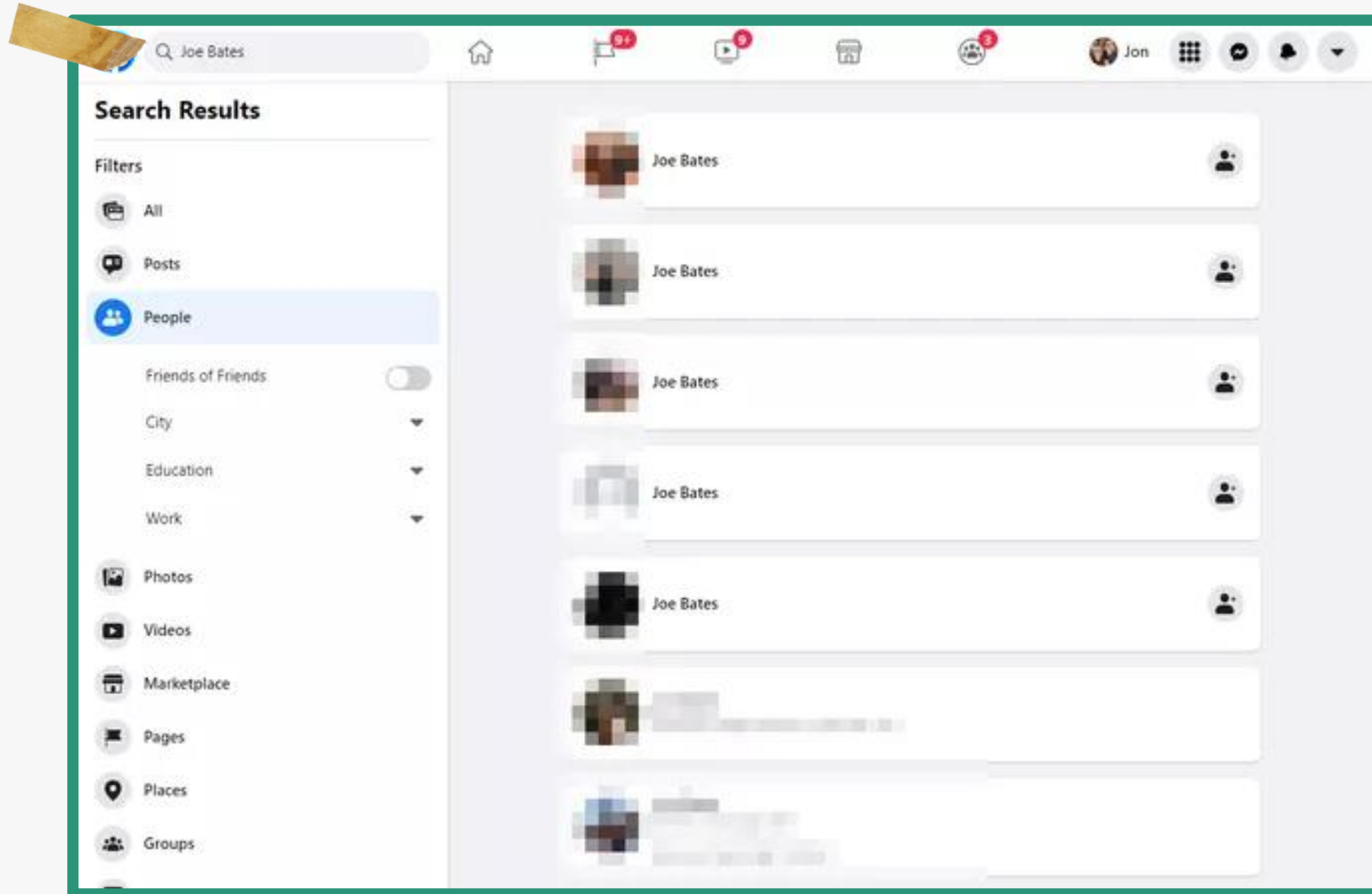


Como encontrar alguém na fila?

Como organizar essa fila?

Como colocar crachá em cada um da fila?

Quando usar estruturas de repetição?



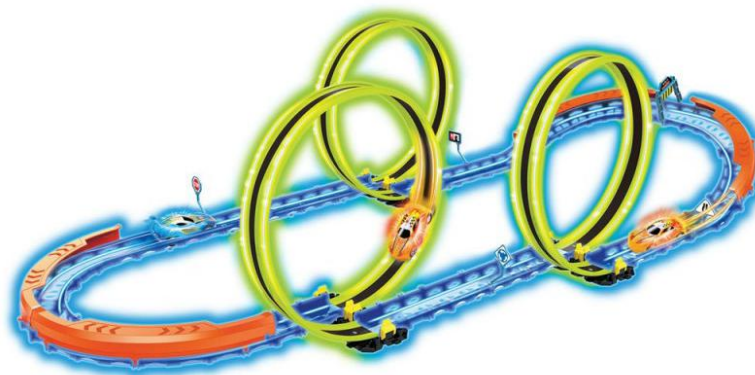
REPETINDO COISAS

- ∴ Por vezes precisaremos repetir uma instrução de acordo com uma condição lógica;
- ∴ Nesses casos usamos estruturas de repetição(loop);

REPETINDO COISAS



Enquanto



Faça/Enquanto



Para

Como fazer para
imprimir na tela os
números na sequência
de 0 a 5?





```
escreva("1")
```

```
escreva("2")
```

```
escreva("3")
```


```
escreva("4")
```

```
escreva("5")
```

```
//manteiga no milho!
```

**E agora? Como
fazer para imprimir
na tela os números
na sequência de 0 a
100? Estou frito!**





```
escreva("1")  
escreva("2")  
escreva("3")  
escreva("4")  
escreva("5")  
escreva("6")  
escreva("7")  
escreva("8")  
escreva("...")
```

```
//deve haver um meio mais inteligente...
```

Estrutura de repetição

ENQUANTO

Estrutura ENQUANTO

O que é?

É uma estrutura de controle que **repete** um bloco de comandos enquanto uma condição for verdadeira.

Quando usar?

Quando não se sabe previamente a quantidade de repetições a serem realizadas

Problema exemplo:

Faça um programa que leia números inteiros até que um zero seja lido. Ao final, mostre a soma dos número lidos.

Digite o primeiro número: 5

Digite outro número: 2

Digite outro número: 4

Digite outro número: 0

A soma é 11

Estrutura ENQUANTO

Sintaxe

```
enquanto(<condição>) {  
    //instruções  
}
```

Regra

Quando a condição for verdadeira:

Executa os comandos do bloco e
volta

Quando a condição for falsa:

Pula fora!


```
enquanto(<condição>)  
{  
    //instruções  
}
```



ENQUANTO

/*o que acontece a seguir o
algoritmo abaixo

→ inteiro x = 0
inteiro y = 4

enquanto (x < 2

y = y + 2
x = x + 1

escreva(x, " - ", y, "\n")
}

escreva(" fim do programa! ")

x < 2

2 < 2

falso

x = x + 1

x = 1 + 1

x = 2

0 1 2

var x

4 6 8

var y

1 - 6

2 - 8

fim do programa!

TÃO FÁCIL QUANTO VOAR...



condição
lógica

enquanto (<condição>)
{

bloco de
instruções
que serão
repetidas **enquanto**
a condição for
verdadeira

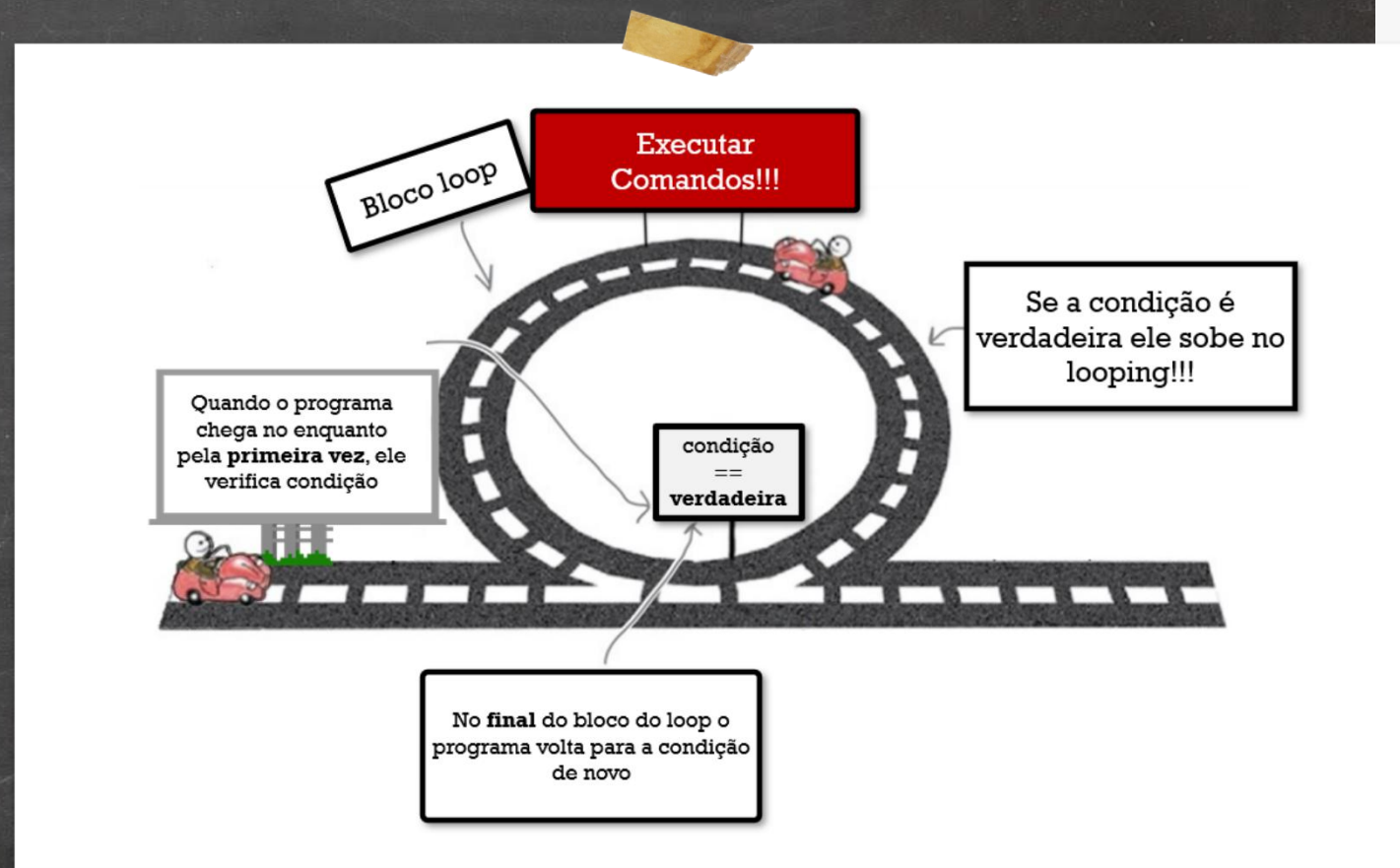
-----> instrucao 1;
instrucao 2;
:
instrucao n;

}

//o que acontece ao executarmos
o algoritmo abaixo?

```
inteiro contador = 0;
```

```
enquanto(contador < 3)  
{  
    escreva(contador)  
}
```



//o que acontece ao executarmos
o algoritmo abaixo?

```
inteiro contador = 0;
```

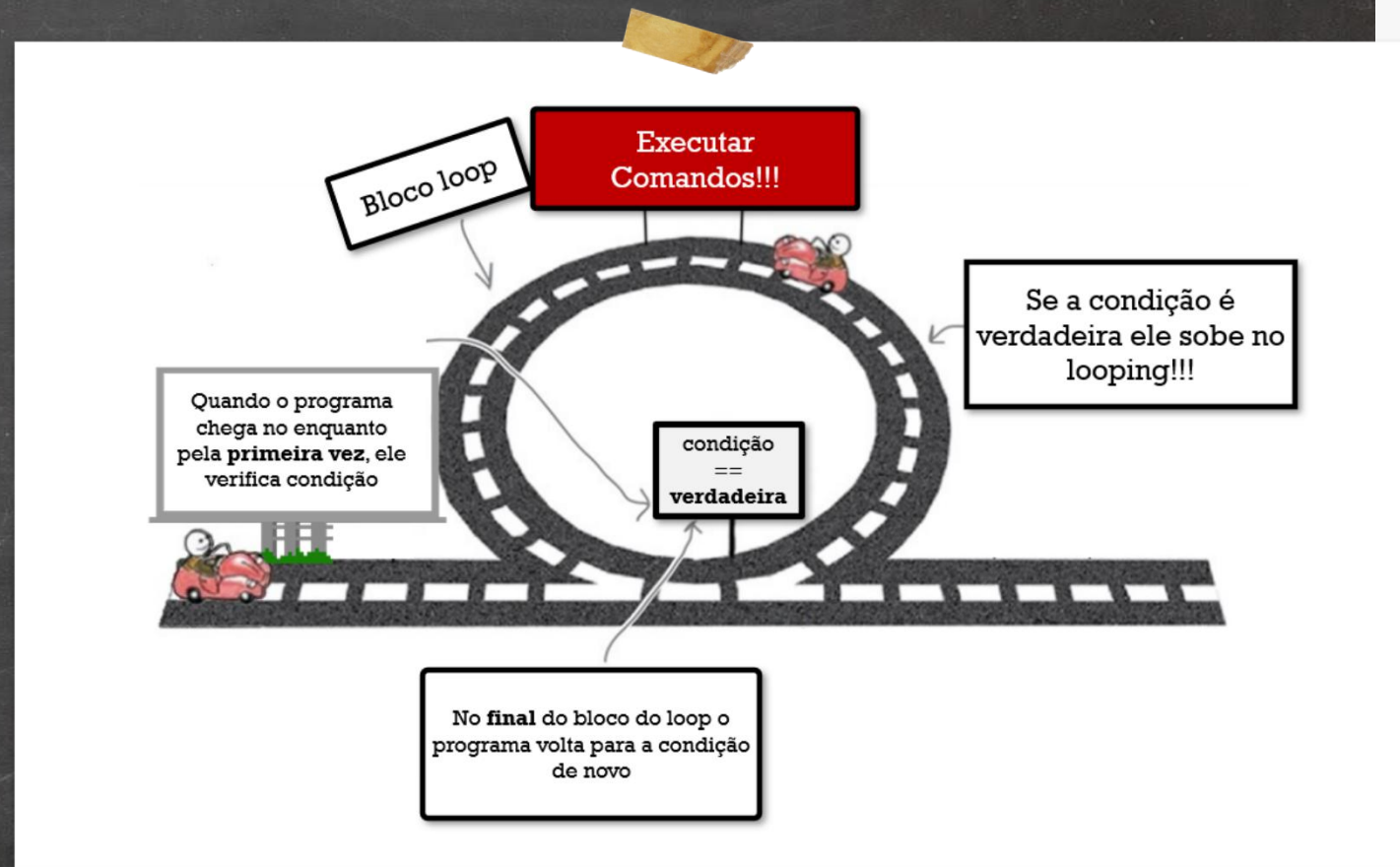
```
enquanto(contador < 3)
```

```
{
```

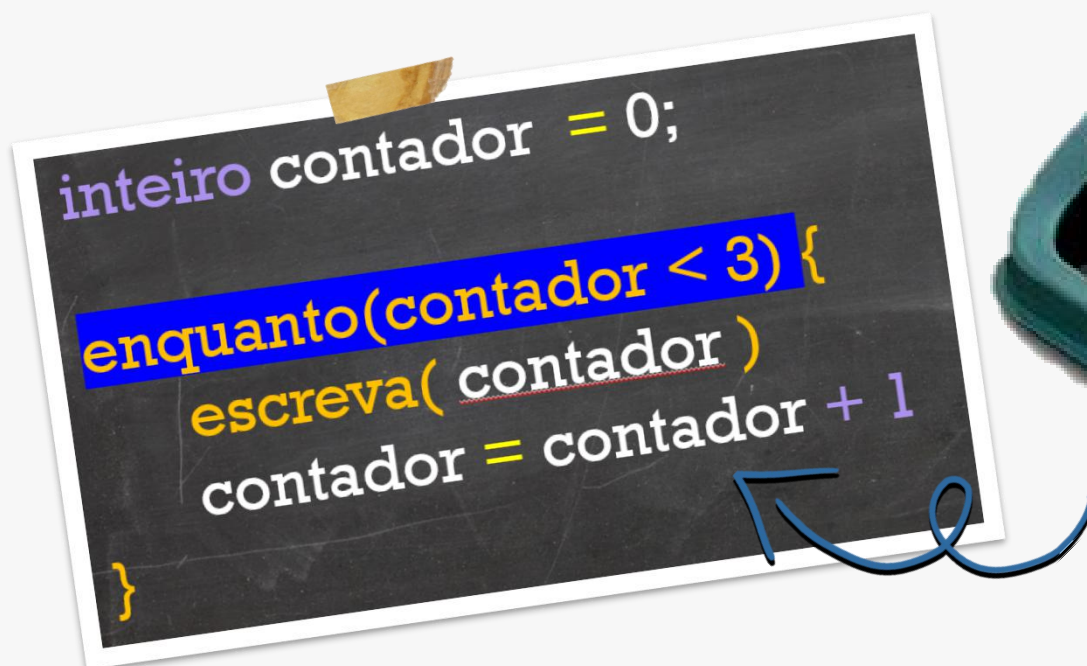
```
  escreva( contador )
```

```
  contador = contador + 1
```

```
}
```



No estudo de repetições,
um conceito fundamental
são os **contadores**



CONTADORES

- ∴ Contadores são importantes para garantir que o loop, em algum momento, **será falso**.
- ∴ A ideia é garantir que o loop não seja infinito;
- ∴ Somar mais um, ou **incrementar** é tão comum, que existe um operadores especial para isso:
- ∴ **incremento unário:** contador ++;
- ∴ **decremento unário:** contador --;



//o que acontece ao executarmos
o algoritmo abaixo?

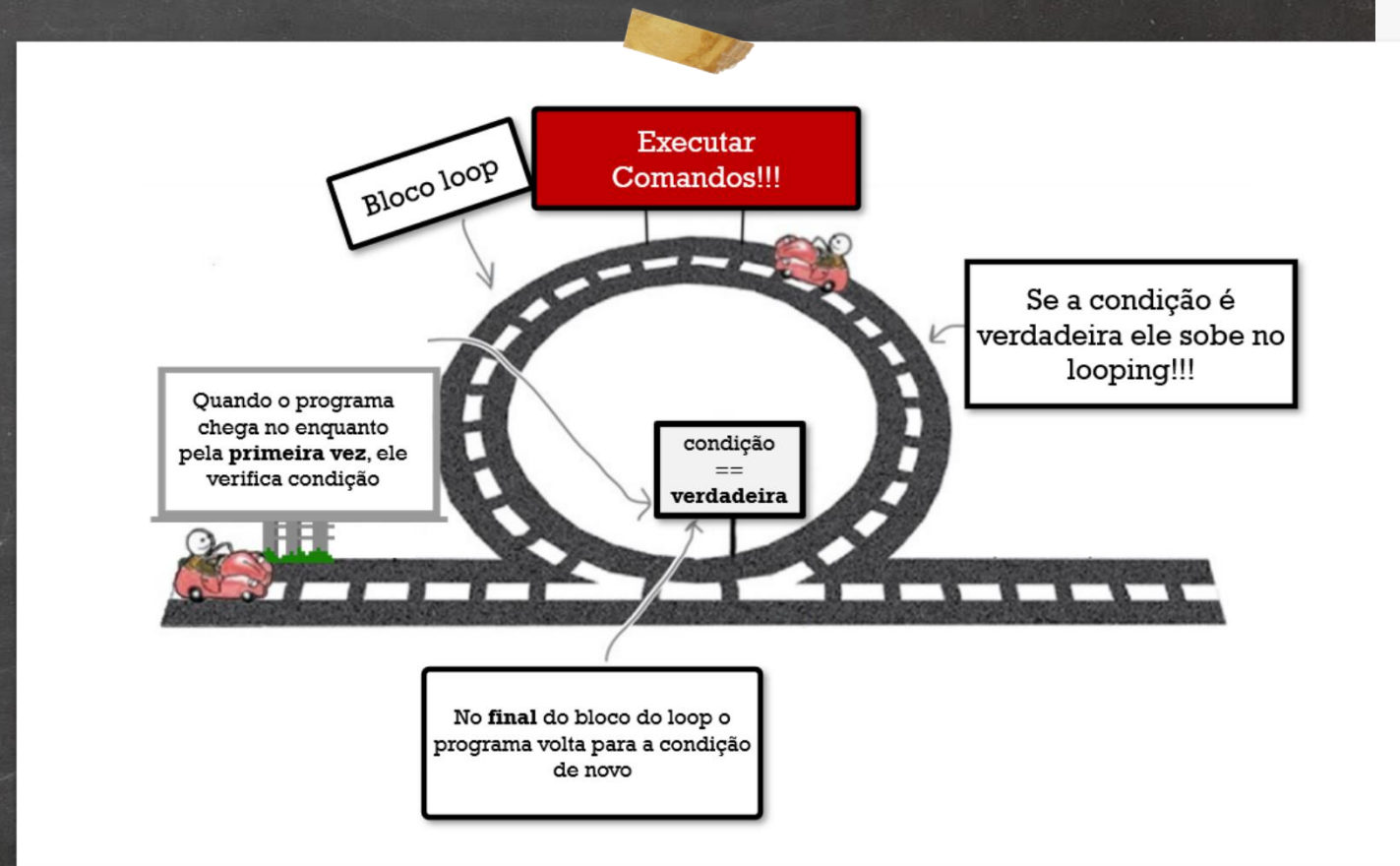
```
inteiro contador = 0;
```

```
enquanto(contador < 3)
```

```
{
```

```
    escreva( contador )  
    contador++
```

```
}
```

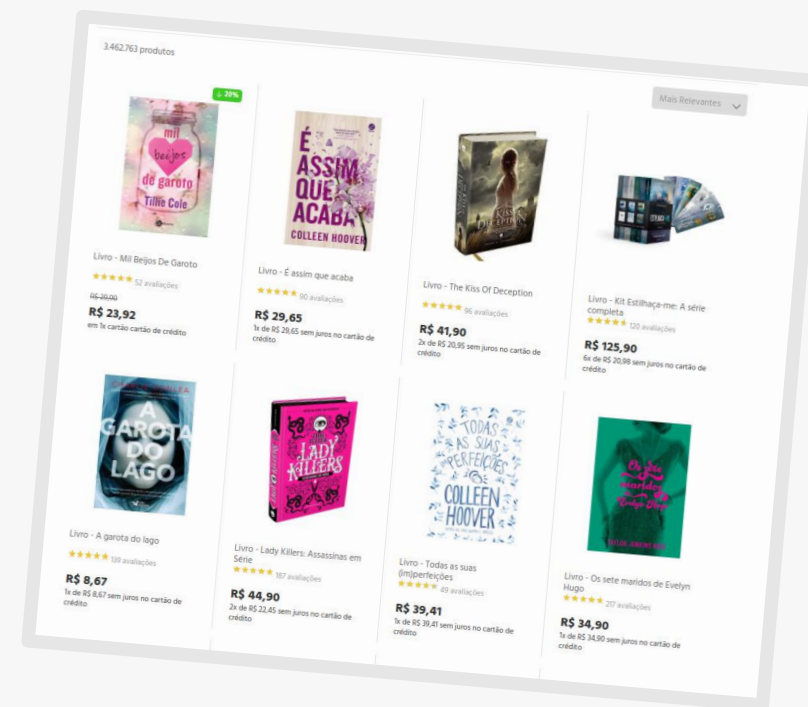


//o que acontece ao se executar o algoritmo abaixo?
inteiro **numero** = 99;

enquanto(numero != 0)
{
 leia(numero)
 escreva("você digitou", **numero**);
}

QUANTAS VEZES REPETIR?

- ∴ Podemos repetir um bloco **determinado (0 ou N)** número de vezes;
- ∴ Exibir os onze jogadores do Corinthians;
- ∴ Exibir uma lista de amigos no Facebook;
- ∴ Os livros da loja Submarino;



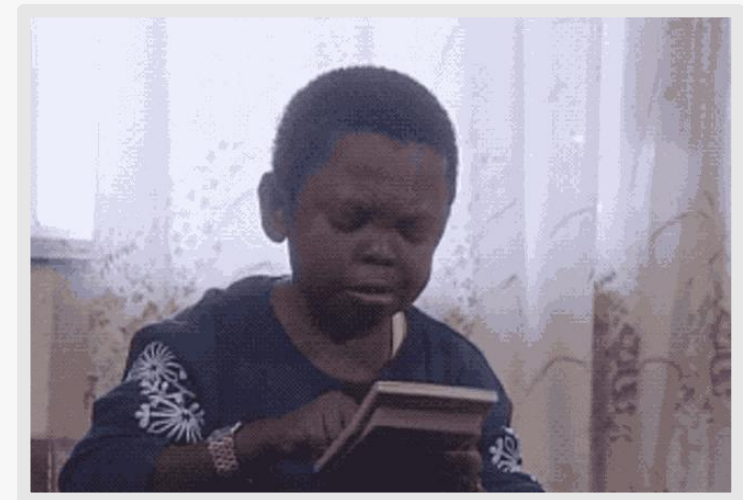
Jogadores do Corinthians

- ∴ Faça um programa que liste os 11 jogadores titulares do Corinthians;
- ∴ O programa deve exibir cada jogador ENQUANTOS os 11 jogadores titulares não forem mostrados na tela.



QUANTAS VEZES REPETIR?

- ∴ Podemos repetir um bloco **indeterminado** número de vezes;
- ∴ Escrever venceu quando a face 6 de um dado for sorteada;
- ∴ Buscar uma pessoa em um estádio de futebol quando não se sabe o número de torcedores;
- ∴ Dividir um número por 2 até que esse número seja menor que 2;



Jogando dados

- ∴ Faça um programa que sorteie um número aleatório entre 1 e 6 representando as faces de um dado;
- ∴ o programa deve permitir que o usuário “chute” o valor sorteado.
- ∴ **Enquanto** o usuário errar, um novo chute deve ser solicitado.



TREINE SEU CÉREBRO



Um elefante incomoda muita gente?

Faça um programa que imprima na tela o “lenga lenga” do elefante, até que existam 100 elefantes.

estrutura de repetição

FAÇA - ENQUANTO

```
faça {  
    //instruções  
} enquanto(<condição>)
```



faca{

bloco de
instruções

é realizado antes
da primeira
iteração.



instrução 1
instrução 2
:
instrução n

} enquanto (<condição>)

condição lógica

define a condição
para repetição do
laço.





//o que acontece ao se executar o algoritmo abaixo?

inteiro numero

faca {

 leia(numero)

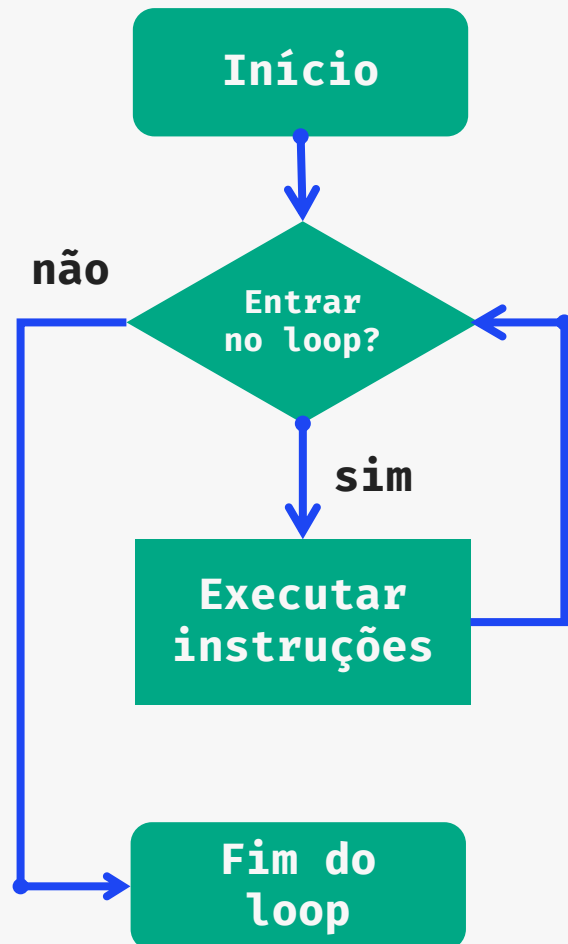
 escreva("você digitou", numero);

} enquanto(numero != 0)

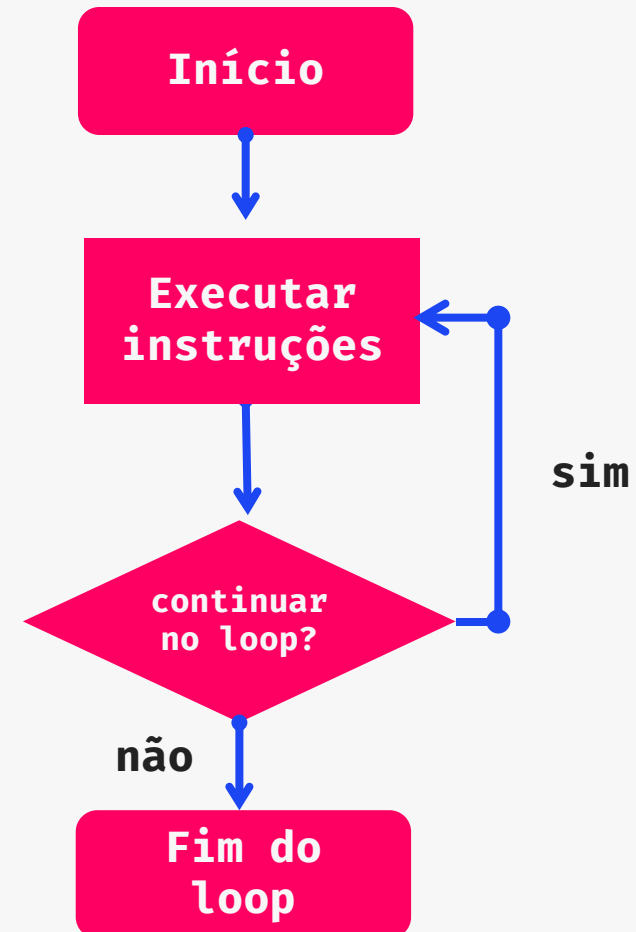
ESTRUTURAS DE REPETIÇÃO

A principal diferença entre a estrutura de repetição **enquanto** e **faça enquanto** é que este é **executado pelo menos uma vez**;

enquanto



faca/enquanto



A PRÁTICA LEVA A PERFEIÇÃO

1. Faça um algoritmo que exiba os números de 1 a 1000 um abaixo do outro;
2. Faça um algoritmo que exiba os números de 1 a 1000 um ao lado do outro separados por um traço;

A PRÁTICA LEVA A PERFEIÇÃO

3. Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.
4. Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.

A PRÁTICA LEVA A PERFEIÇÃO



5. Faça um jogo que o algoritmo sorteie um número e o usuário informa outro. Compare os valores e ao ser igual mostre a mensagem “venceu o jogo”.
Exiba as mensagens perdeu ao perder e ganhou ao ganhar.
Repita enquanto não ganhar.

6. Aprimore seu jogo: Informe o usuário ao errar, se o número informado foi mais alto ou mais baixo que o sorteado.
Informe também o número de tentativas.



OBRIGADO

perguntas?

Jefferson.chaves@ifpr.edu.br

45 998508359

github.com/jeffersonchaves