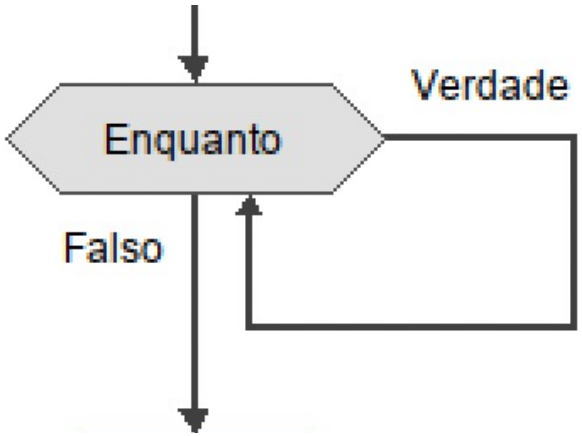
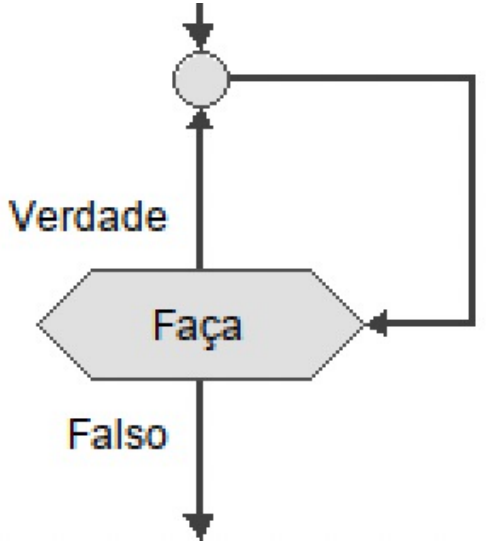
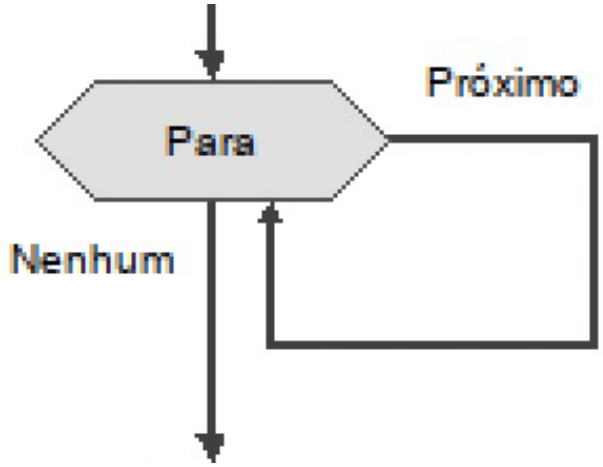
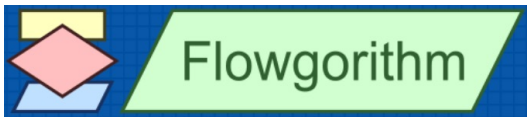
Instruções Cíclicas









Instruções Cíclicas

Instruções Cíclicas (ciclos; laços de repetição) – permitem executar repetidamente (em ciclo) um conjunto de ações (instruções) sob o controlo de uma condição (teste de condição).

Tipos de instruções cíclicas:

- `enquanto` *(while)*
- `faca...enquanto` *(do...while)*

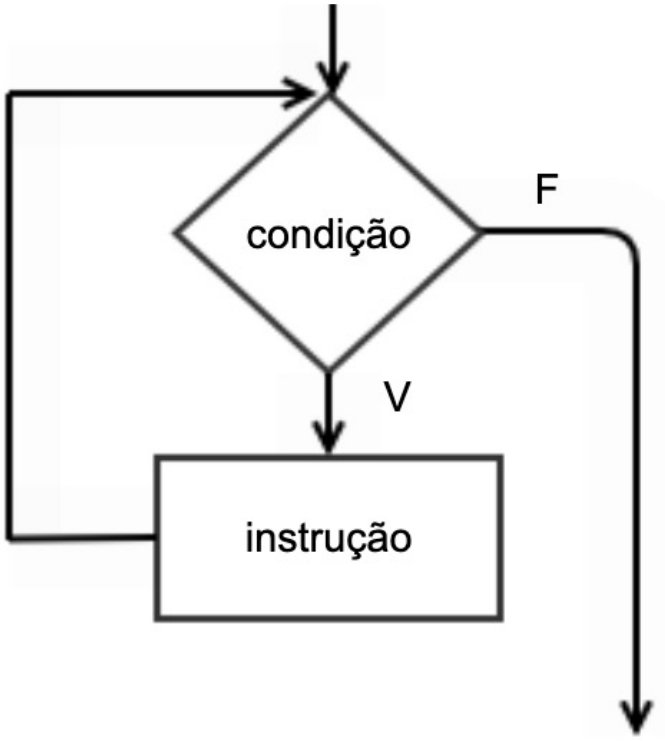
→ ciclos executados um número **indeterminado** de vezes
(ex.: *ler números até ser lido o valor 0*)

- `para` *(for)*

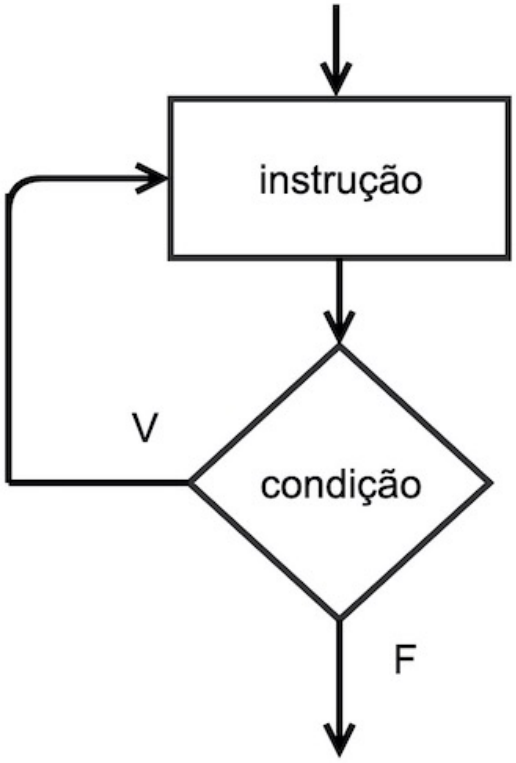
→ ciclos executados um número **determinado** de vezes
(ex.: *ler 10 números*)

Instruções Cíclicas

Ciclos executados um número **indeterminado** de vezes



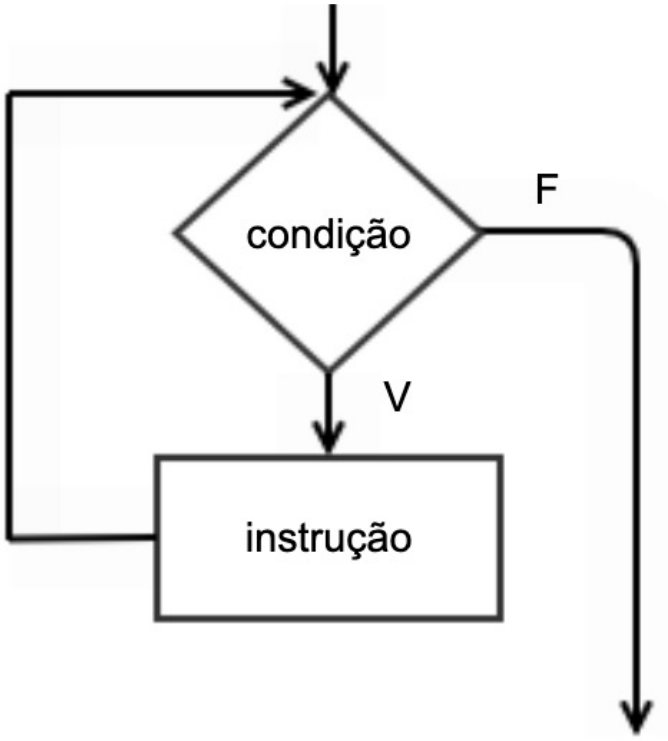
`enquanto`

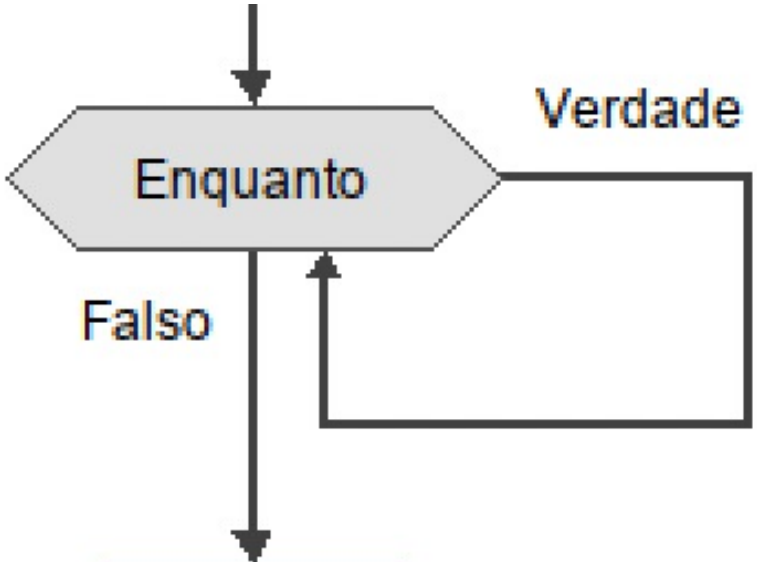


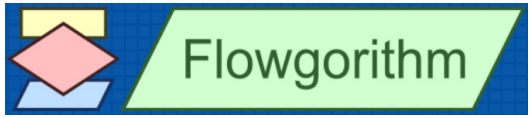
`faca ... enquanto`

Instruções Cíclicas

Ciclos executados um número **indeterminado** de vezes



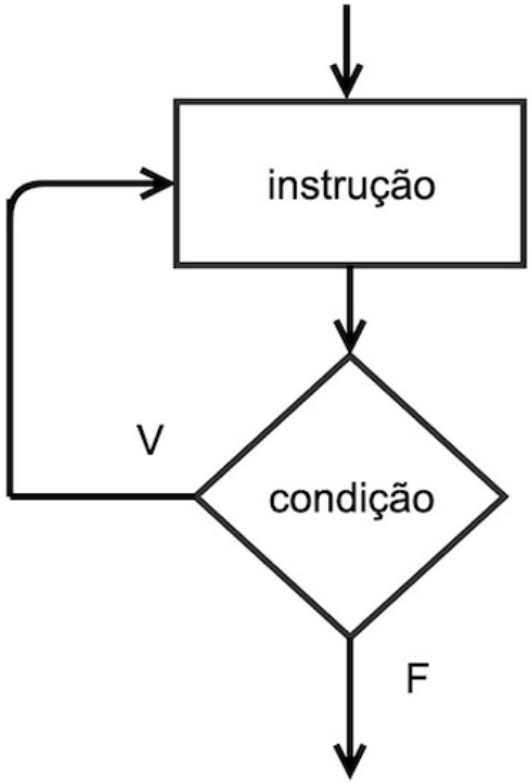


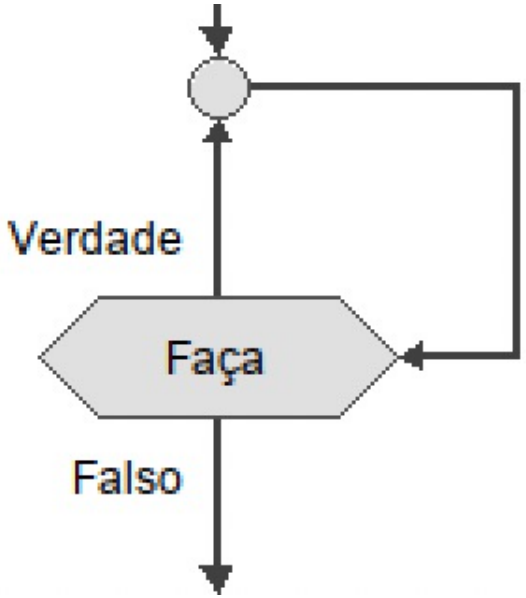


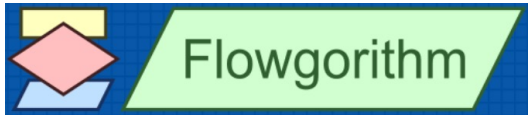
`enquanto`

Instruções Cíclicas

Ciclos executados um número **indeterminado** de vezes







`faca ... enquanto`

Instruções Cíclicas

- Ciclo `enquanto`

```
enquanto( <condição> )
    <instrução>
```

- Enquanto a condição se verificar (enquanto for verdadeira), a instrução (que pode ser simples ou composta) é executada repetidamente.
- Quando a condição passar a ser falsa, o ciclo termina.
- O teste da condição é feito no início do ciclo: **implica que a instrução pode não ser executada nenhuma vez**.

Instruções Cíclicas

Exemplo

Ler uma sequência de números inteiros enquanto não for lido o valor 0. Contar o número de valores diferentes de 0.

```
programa(){
    funcao inicio(){
        inteiro n, contador = 1
        escreva ( "Insira um número" )
        leia(n)
        enquanto( n != 0 ){
            escreva ( "Insira um número" )
            leia( n )
            contador = contador + 1
         }
         escreva( contador )
    }
}
```

É necessário ler o primeiro valor fora do ciclo

Se o primeiro valor for 0, o ciclo não executa nenhuma vez (mas o contador fica em 1).

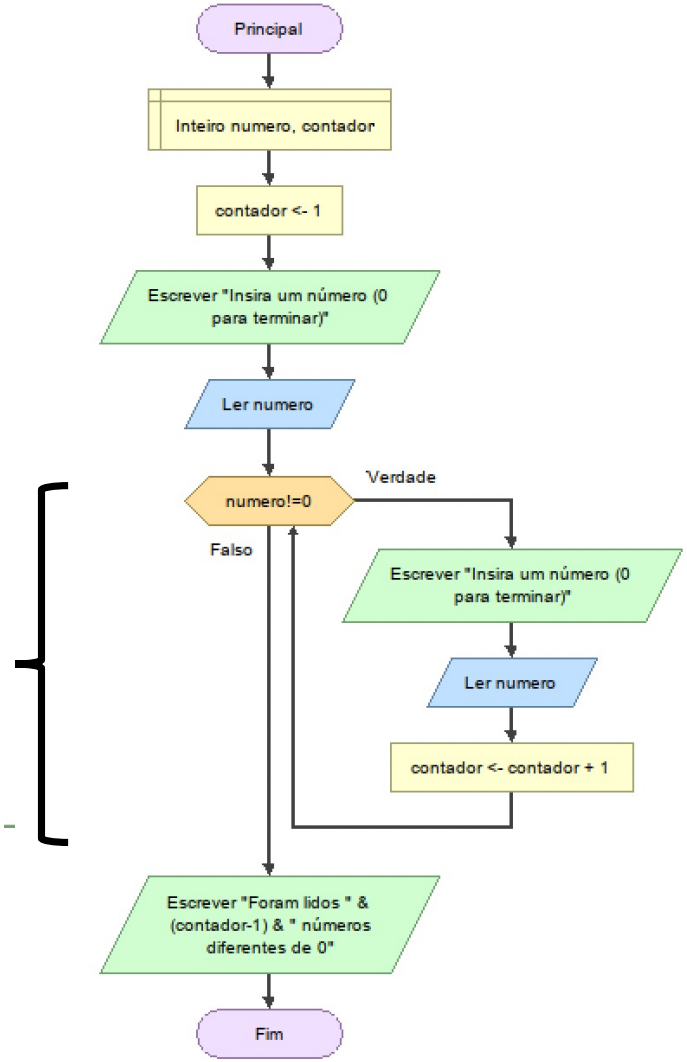
Instruções Cíclicas

Exemplo

ciclo `enquanto`

Python

```
while numero != 0:
    print("Insira um número (0 para terminar)")
    numero = int(input())
    contador = contador + 1
```



Instruções Cíclicas

- Ciclo `faca ... enquanto`

```
faca
    <instrução>
enquanto( <condição> )
```

- Executa a instrução (simples ou composta) enquanto a condição for verdadeira.
- Quando a condição passar a ser falsa, o ciclo termina.
- O teste da condição é feito no final do ciclo: **implica que a instrução é executada, pelo menos, uma vez**.

Instruções Cíclicas

Exemplo

Ler uma sequência de números inteiros enquanto não for lido o valor 0. Contar o número de valores diferentes de 0.

```
programa(){
    funcao inicio(){
        inteiro n, contador = 0
        faca {
            escreva ( "Insira um número" )
            leia( n )
            contador = contador + 1
         } enquanto( n != 0 ){
         escreva( contador )
    }
}
```

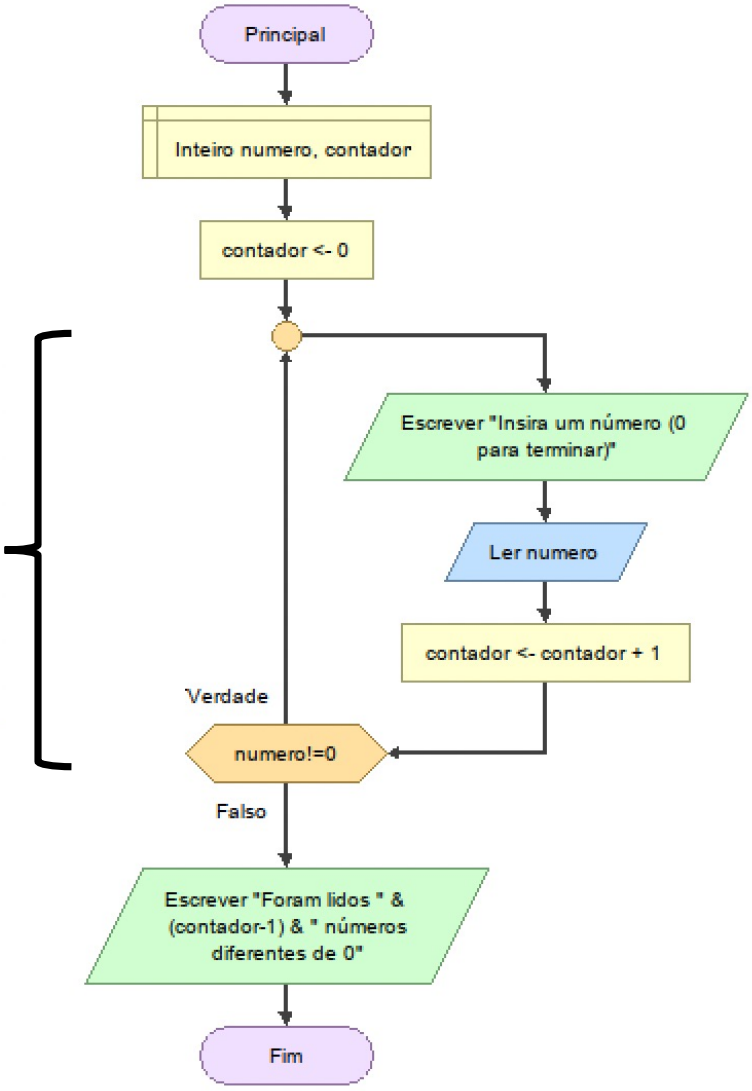
Se o primeiro valor for 0, o ciclo executa uma vez e termina (mas o contador fica em 1).

Instruções Cíclicas

Exemplo

Python

```
while True:    #This simulates a Do Loop
    print("Insira um número (0 para terminar)")
    numero = int(input())
    contador = contador + 1
    if not(numero != 0): break    #Exit loop
```



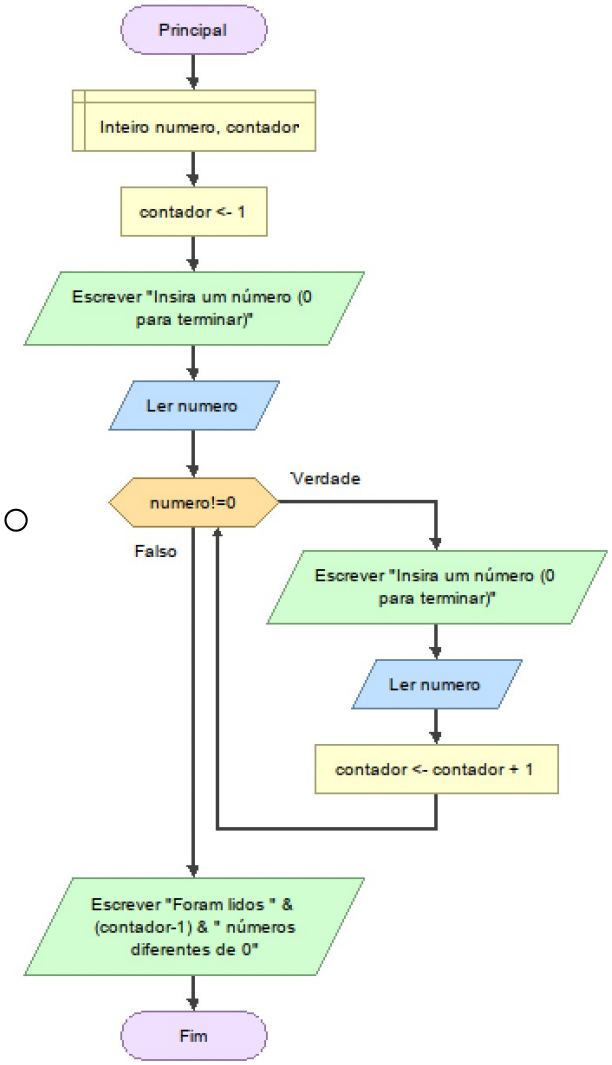
ciclo `faca ... enquanto`

Em Python não existe nenhum ciclo equivalente a `faca ... enquanto` mas este ciclo é simulado através do ciclo `while`

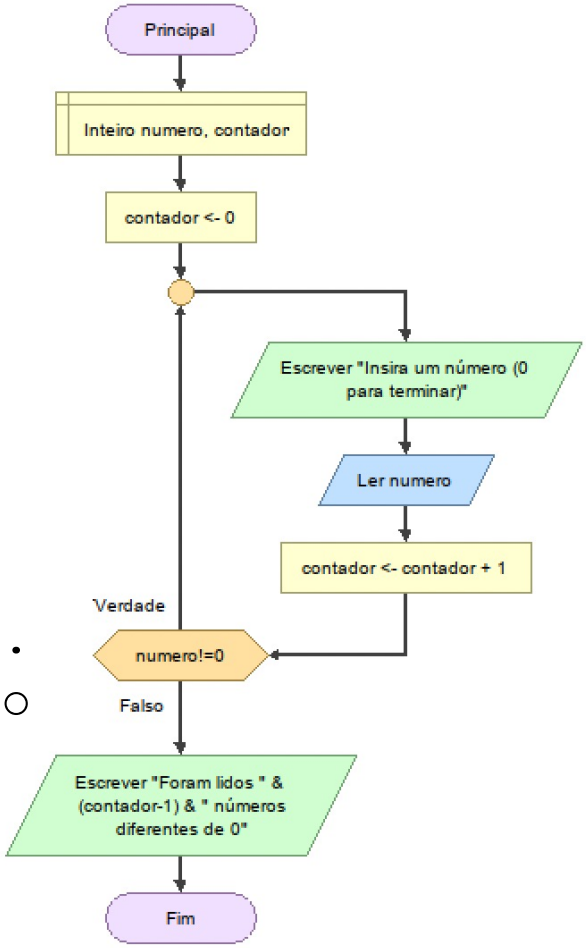
Instruções Cíclicas

Exemplo

enquanto



faca ... enquanto



Instruções Cíclicas

Usar `enquanto` ou `faca ... enquanto`?

- No caso do exemplo anterior seria mais vantajoso usar `faca ... enquanto`.
- Em geral, deve-se escolher o ciclo que for mais adequado a cada situação, tendo em conta as características de cada um deles e o contexto em que vai ser usada.
- Qualquer um destes ciclos deve ser usado em situações em que se desconhece quantas vezes o ciclo será executado (ou seja, vai ser executado um número indeterminado de vezes).
- E nas outras situações, quando se sabe quantas vezes o ciclo vai ser executado? Usar o ciclo `para`.

Instruções Cíclicas

- Ciclo `para`

```
para( <início> ; <fim> ; <passo> )
    <instrução>
```

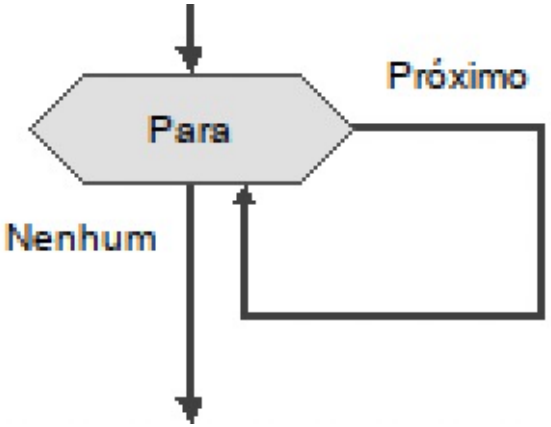
- Executa a instrução (simples ou composta) desde a condição inicial (`início`) e enquanto for verdadeira uma condição final (`fim`).
- Quando a condição final passar a ser falsa, o ciclo termina.
- O `passo` (incremento ou decremento) é uma instrução que permite ir da condição inicial até à condição final.

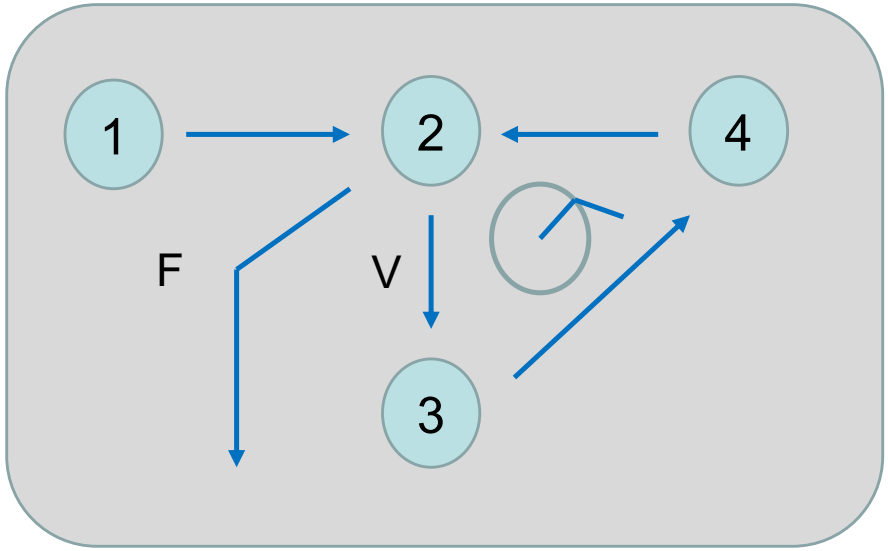
Instruções Cíclicas

1 2 4

```
para( <início> ; <fim> ; <passo> )
    <instrução>
```

3



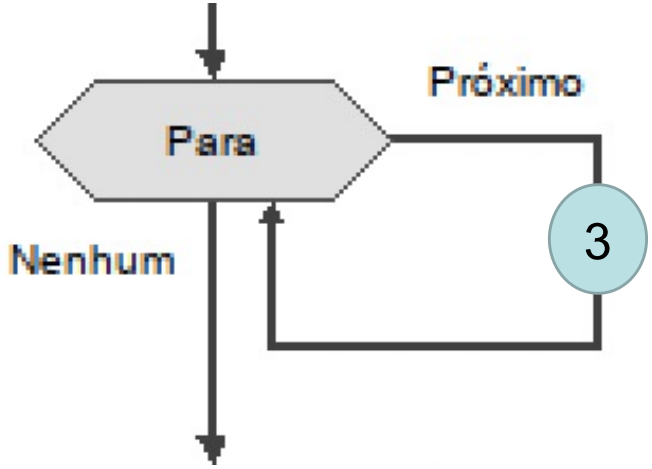


Flowgorithm

Instruções Cíclicas

```
para( <início> ; <fim> ; <passo> )
    <instrução>
```

1 2 4 3



Flowgorithm

Para

O ciclo PARA (For) incrementa ou decrementa uma variável (contador) dentro do intervalo e passo especificados. Substitui o ciclo FAÇA em muitas situações e é usado frequentemente na manipulação de matrizes.

Variável:

Valor Inicial: 1

Valor Final: 2

Sentido:

Crescente

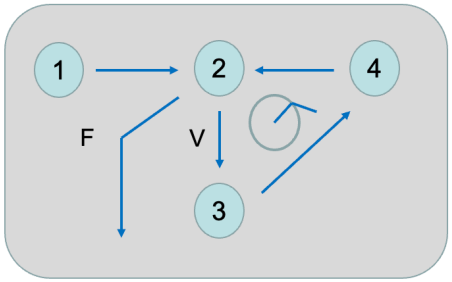
Decrescente

Incremento/decremento: 1 4

Instruções Cíclicas

Exemplo

Escrever a tabuada da multiplicação de um número inteiro.




```
programa
{
    funcao inicio()
    {
        inteiro n, i
        escreva( "Insira o número que pretende para a tabuada: " )
        leia( n )
        para( i=1; i<=10; i=i+1 ){
                escreva( n, " x ", i, " = ", n*i , "\n" )
        }
    }
}
```

1 2 4 3

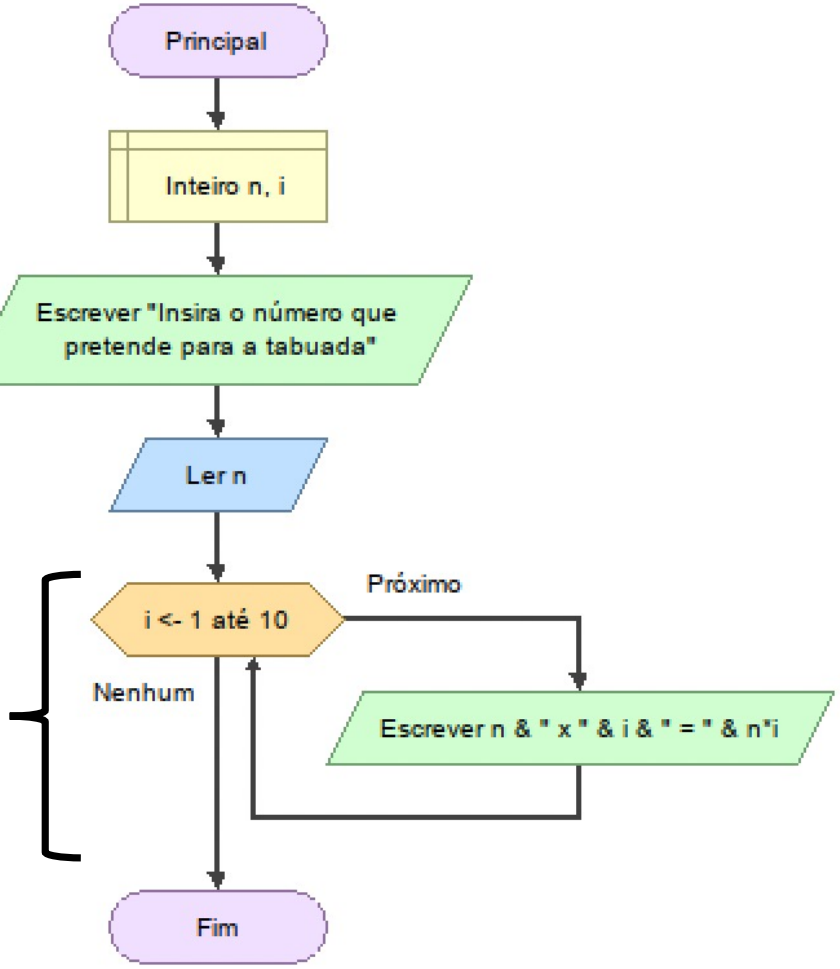
Instruções Cíclicas

Exemplo

ciclo `para`

Python

```
for i in range(1, 10 + 1, 1):
    print(str(n) + " x " + str(i) + " = " + str(n * i))
```



Instruções Cíclicas

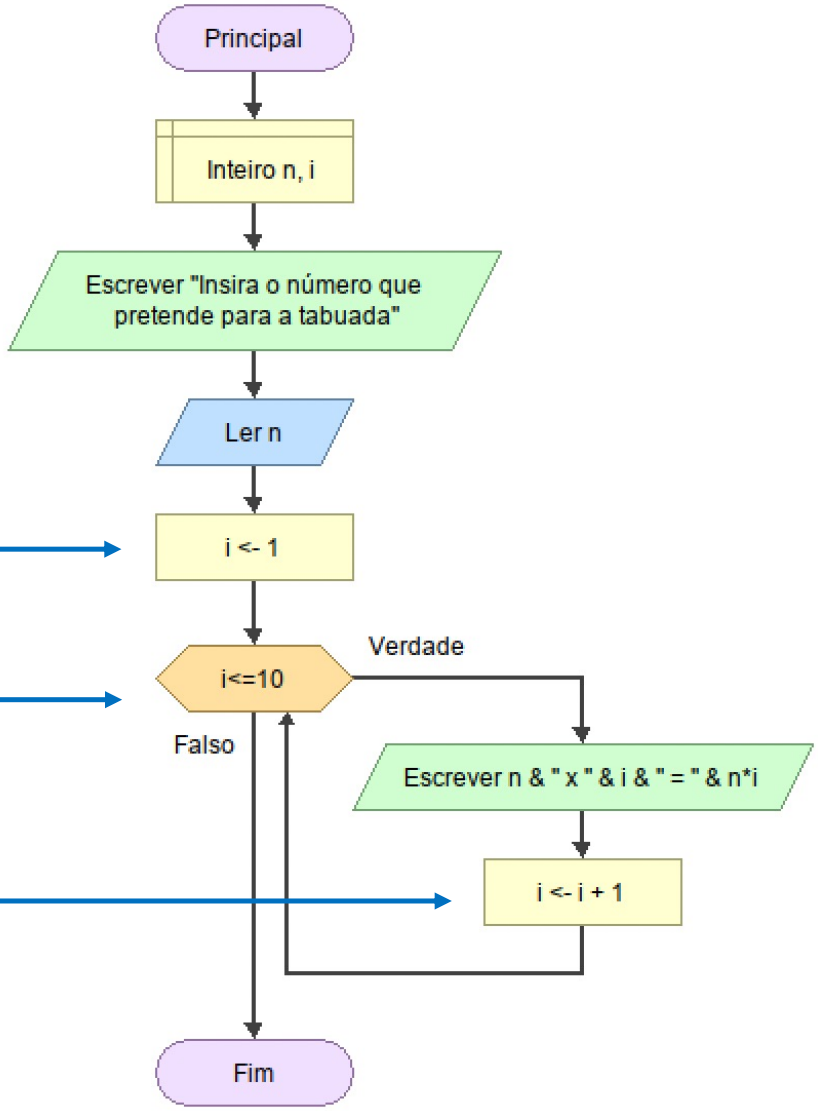
Escrever a tabuada da multiplicação de um número inteiro

Pode-se resolver também com um ciclo `enquanto`

início

fim

passo (incremento de 1)



Instruções Cíclicas

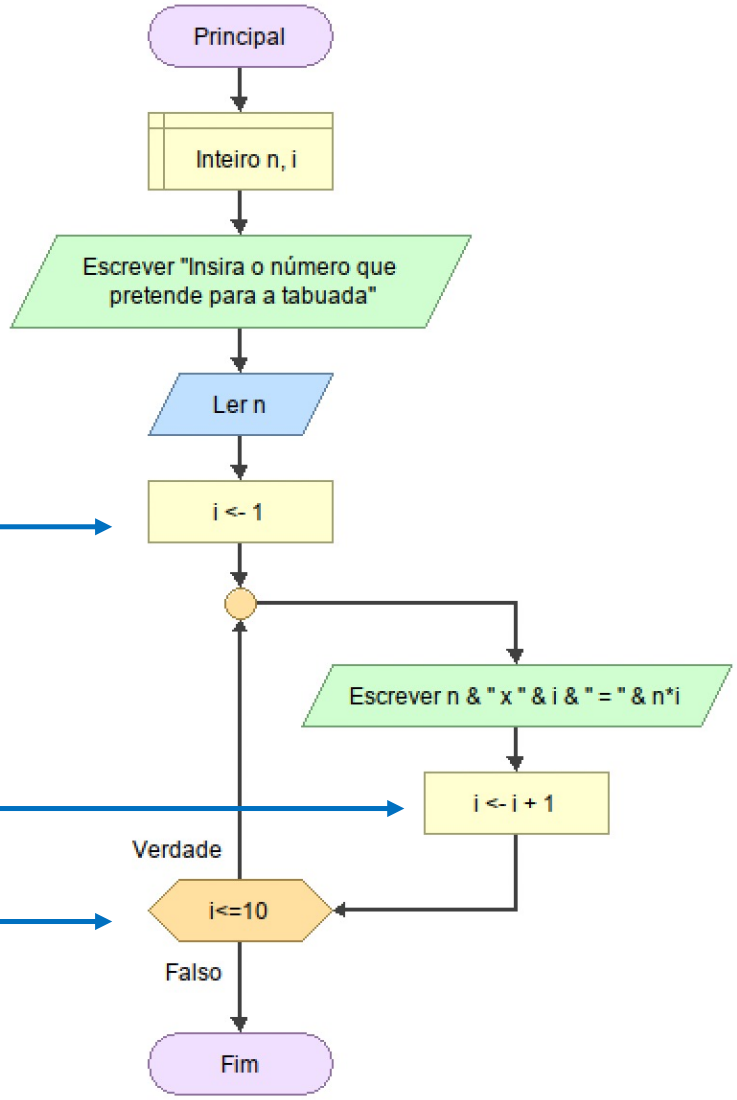
Escrever a tabuada da multiplicação de um número inteiro

Pode-se resolver também com um ciclo `faca` ... `enquanto`

início

passo (incremento de 1)

fim



Instruções Cíclicas

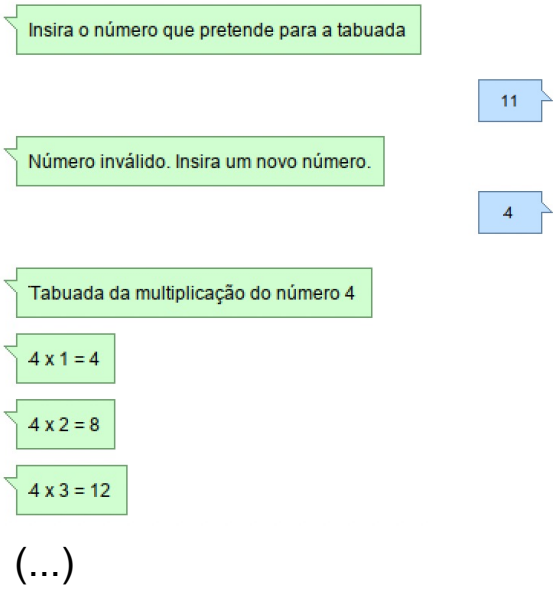
Usar `enquanto`, `faca ... enquanto` ou `para`?

- Neste caso, seria mais vantajoso usar `para`.
- Também aqui, em geral, deve-se escolher o ciclo que for mais adequado a cada situação, tendo em conta as características de cada um deles e o contexto em que vai ser usado.
- O ciclo `para` deve ser usado em situações em que se conhece à partida quantas vezes o ciclo será executado (ou seja, vai ser executado um número determinado de vezes).
- **Todos os ciclos `para` podem ser convertidos em ciclos `enquanto` ou `faca ... enquanto`; o inverso pode não ser verdadeiro!**

Exercício

Desenhar um fluxograma para escrever a tabuada da multiplicação de um número inteiro compreendido entre 1 e 10, inserido pelo utilizador. Caso seja inserido um número fora do intervalo, o fluxograma apresenta uma mensagem de erro e pede um novo número.

Exemplo:



(...)

Exercício (ficha 11, ex. 5)

Desenhar um fluxograma que permita calcular a área de triângulos (a partir da base e da altura), retângulos (a partir da largura e do comprimento) e círculos (a partir do raio).

A interface com o utilizador para um futuro programa deve ser baseada num menu de opções:

```
1 - Area do retângulo
2 - Area do triângulo
3 - Area do círculo
4 - Sair
```

Instruções Cíclicas

```
inteiro opcao

faca{
    escreva( "1 - Area do triângulo\n" )
    escreva( "2 - Area do retângulo\n" )
    escreva( "3 - Area do círculo\n" )
    escreva( "4 - Sair\n" )

    leia( opcao )

    escolha( opcao ){
        caso 1:
            pare
        caso 2:
            pare
        caso 3:
            pare
        caso 4:
            pare
        caso contrario:
            escreva( "Opção inválida!\n" )
    }
}enquanto( opcao != 4 )
```

Menu de Opções

Em linguagem algorítmica (Portugol), o menu de opções é feito através de uma instrução `escolha...caso` dentro de uma instrução `faca...enquanto`.

Neste exemplo falta acrescentar as instruções nos `casos` 1, 2, 3 e 4 correspondentes às opções 1,2, 3 e 4 do menu.

Se o utilizador não indicar uma opção válida (1, 2, 3 ou 4) é apresentada uma mensagem de erro (é executado o que estiver em `caso contrario`).

Instruções Cíclicas

```
while True:
    print( "1 - Area do triangulo" )
    print( "2 - Area do retangulo" )
    print( "3 - Area do circulo" )
    print( "4 - Sair" )

    opcao = int( input())

    match opcao:
        case 1:
            print()
        case 2:
            print()
        case 3:
            print()
        case 4:
            print()
        case _:
            print( "Opcao invalida!" )

    if opcao == 4:
            break
```

Menu de Opções - Python

Como nesta linguagem não existe o equivalente ao ciclo `faca ... enquanto`, usa-se o ciclo `while`.

Neste exemplo falta acrescentar as instruções nos `cases` 1, 2, 3 e 4 correspondentes às opções 1,2, 3 e 4 do menu.

Se o utilizador não indicar uma opção válida (1, 2, 3 ou 4) é apresentada uma mensagem de erro (é executado o que estiver em `case _`).