INSTITUTO DE INFORMÁTICA
UFRGS

INF01040 – Introdução à Programação

# Algoritmos com Vetores

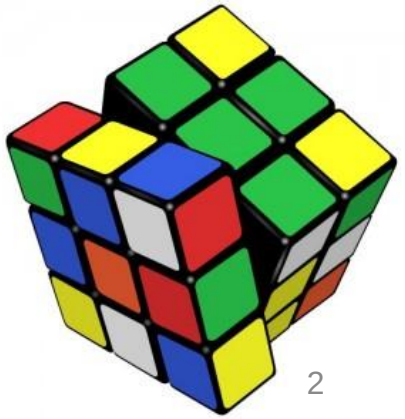
## Pesquisa e Classficação

## Algoritmos de pesquisa e de classificação (ordenação) em vetores

# Considere o problema

- Preencher um arranjo de 5 elementos inteiros por leitura
- Imprimir os valores do arranjo em uma linha
- **Ordenar os valores do arranjo em ordem crescente**
- Imprimir o arranjo ordenado em uma linha

Como ordenar um arranjo?

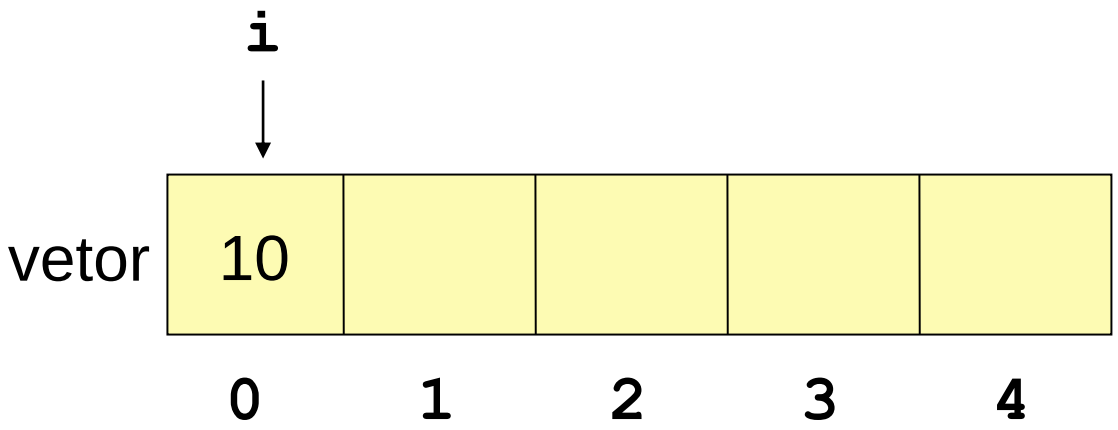


# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 1**

i = 0

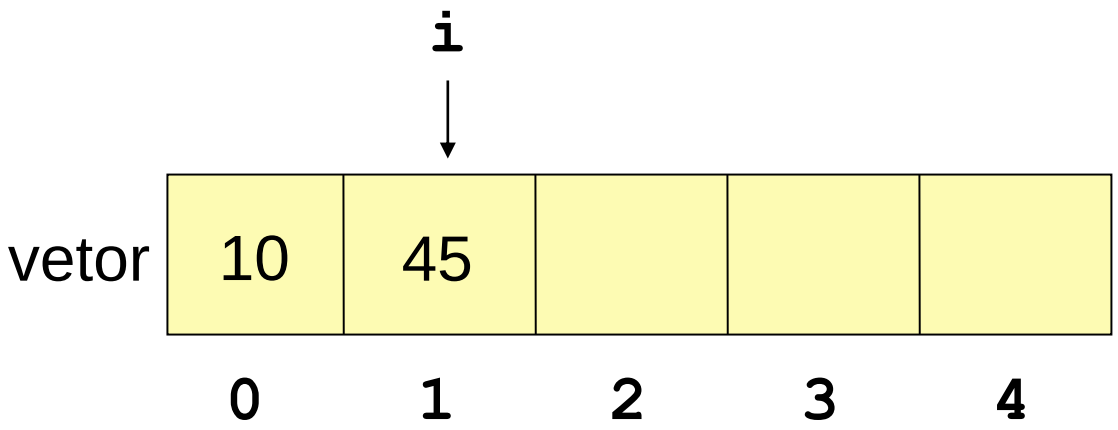


# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 2**

i = 1

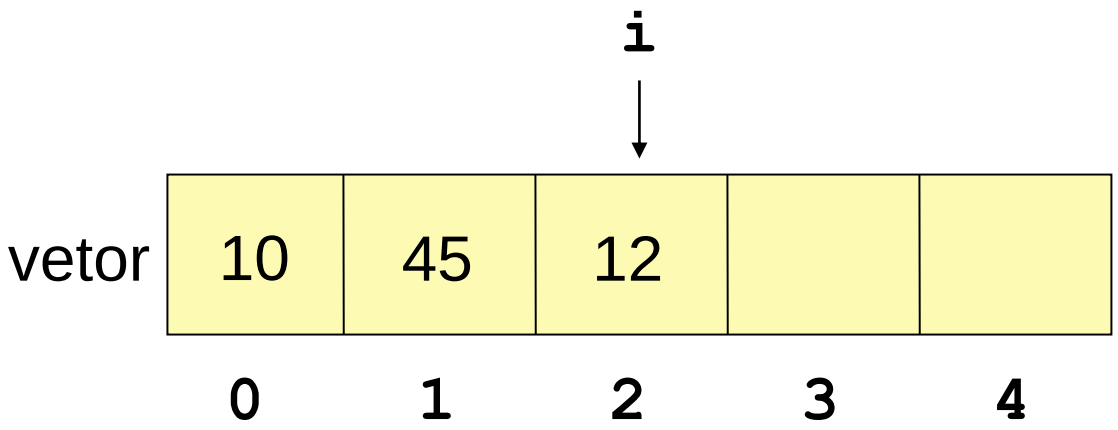


# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 3**

i = 2

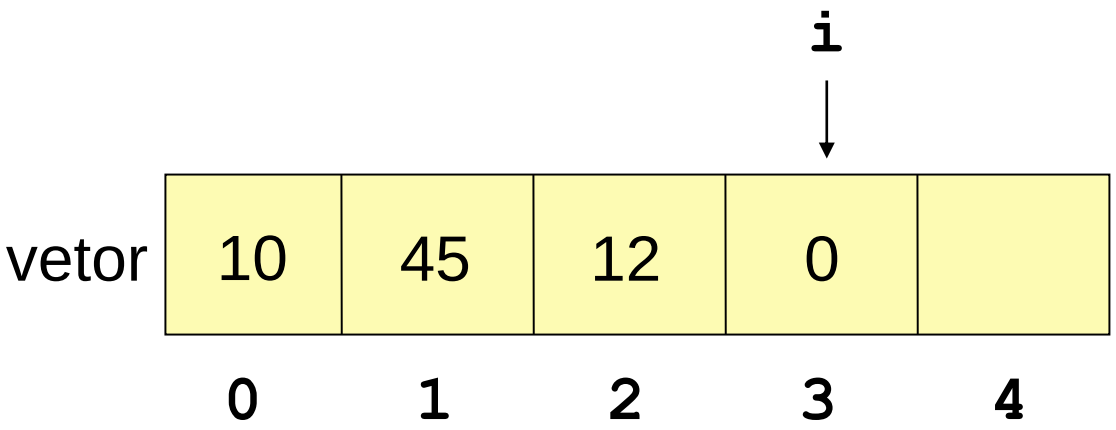


# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 4**

i = 3

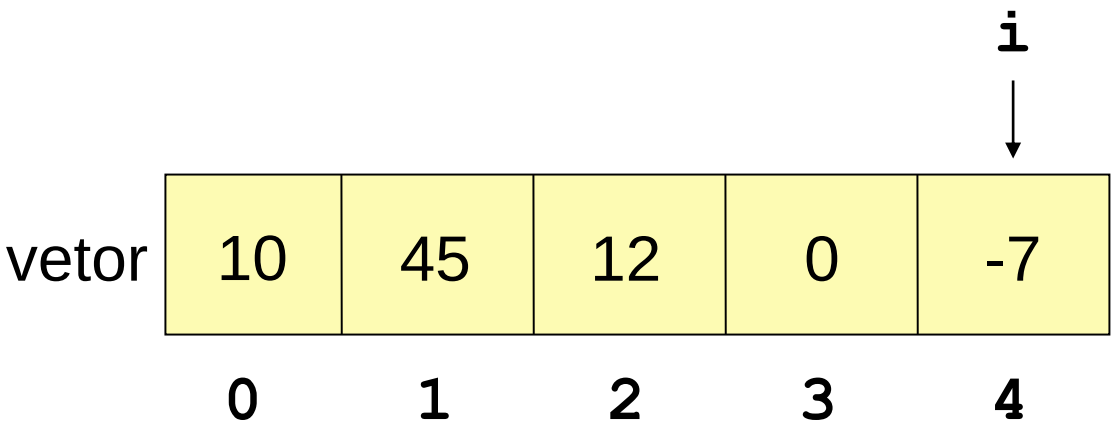


# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 5**

i = 4



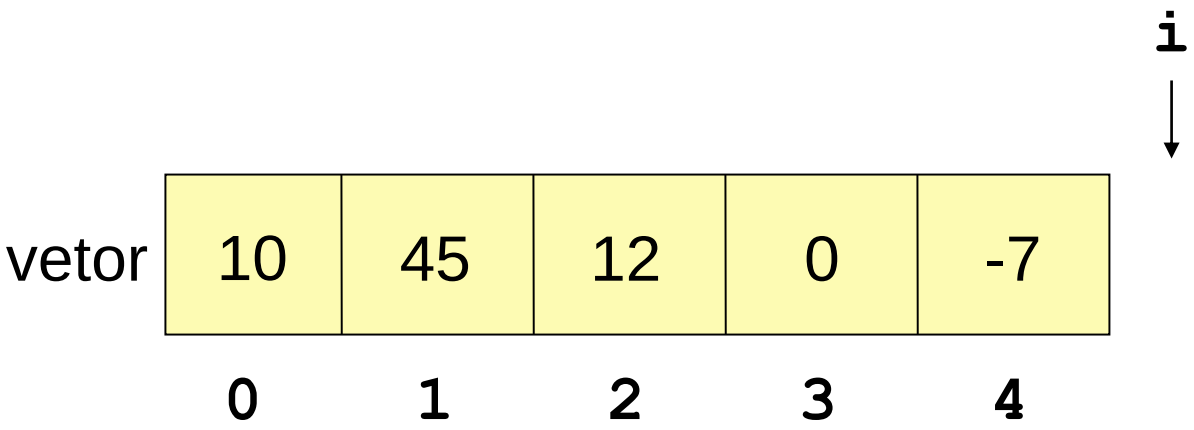
# (Relembrando) Inicialização, por leitura, do vetor

```
Constante: MAX 5
Variáveis:
  Inteiro vetor[MAX], i;
...
Para (i = 0; i<MAX; i++)
Faça
  Ler vetor[i];
Fim
...
```

**Iteração 6**

i = 5

Fora do laço...



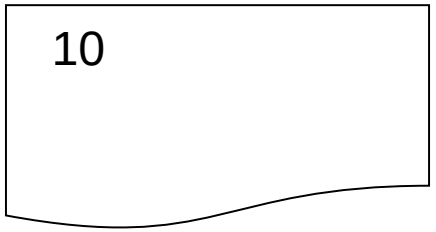
# (Relembrando) Imprimir os valores do arranjo em uma linha

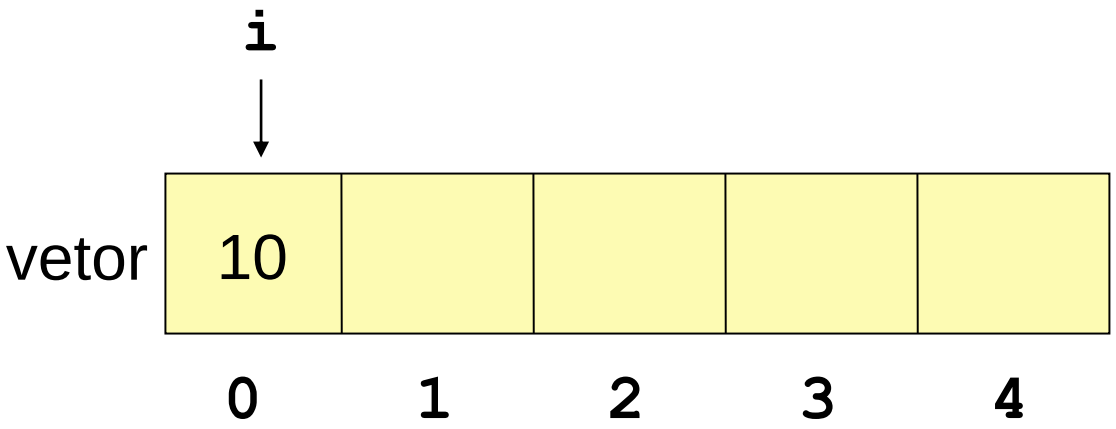
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

**Iteração 1**

i = 0

tela



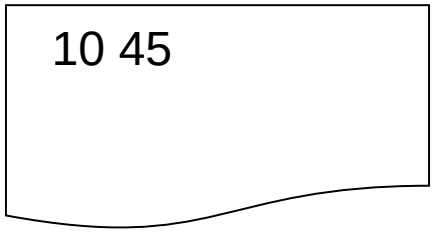
# (Relembrando) Imprimir os valores do arranjo em uma linha

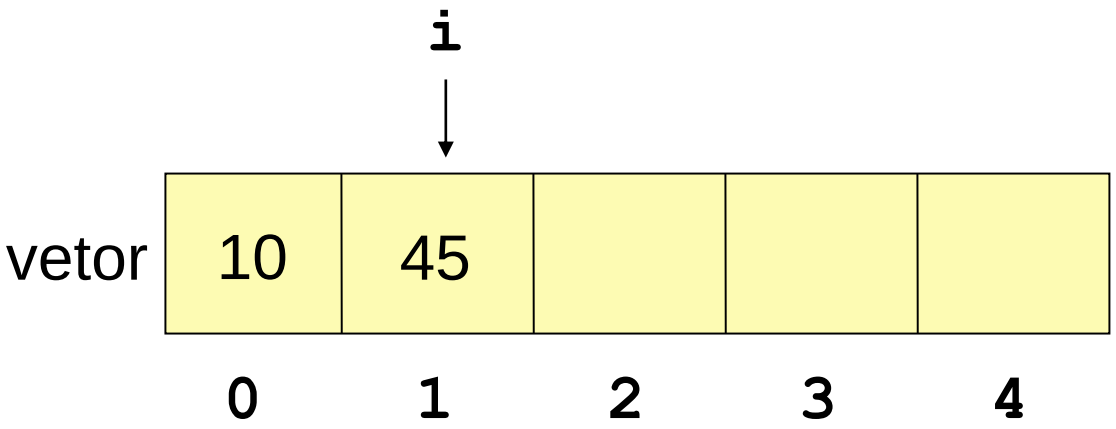
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

**Iteração 2**

i = 1

tela



# (Relembrando) Imprimir os valores do arranjo em uma linha

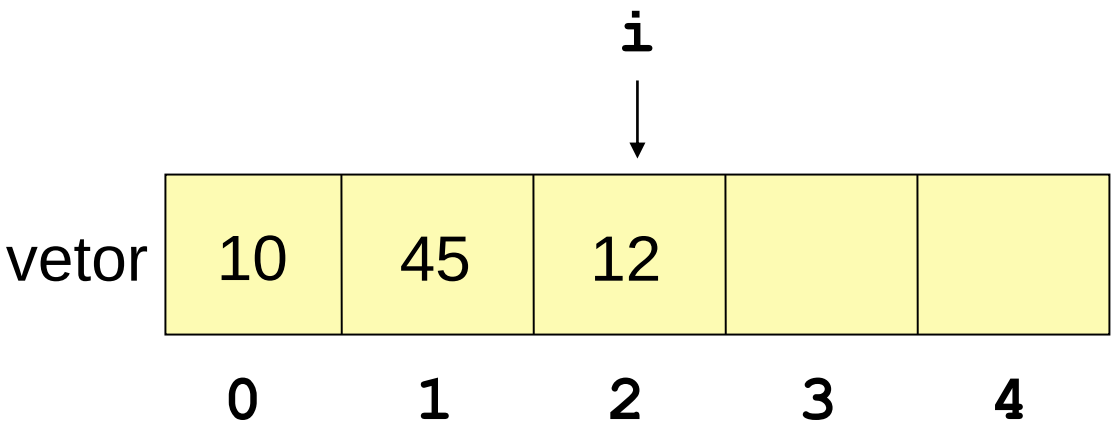
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

**Iteração 3**

i = 2

tela

10 45 12



# (Relembrando) Imprimir os valores do arranjo em uma linha

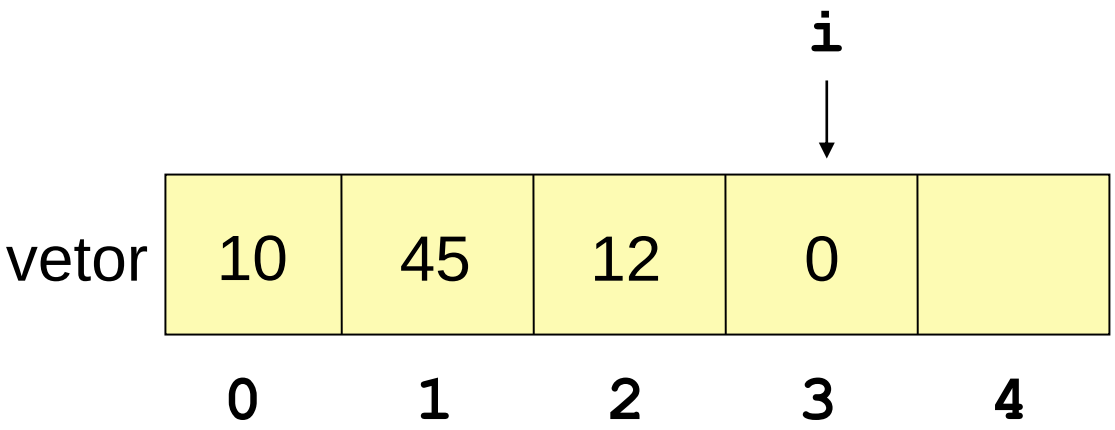
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

**Iteração 4**

i = 3

tela

10 45 12 0



# (Relembrando) Imprimir os valores do arranjo em uma linha

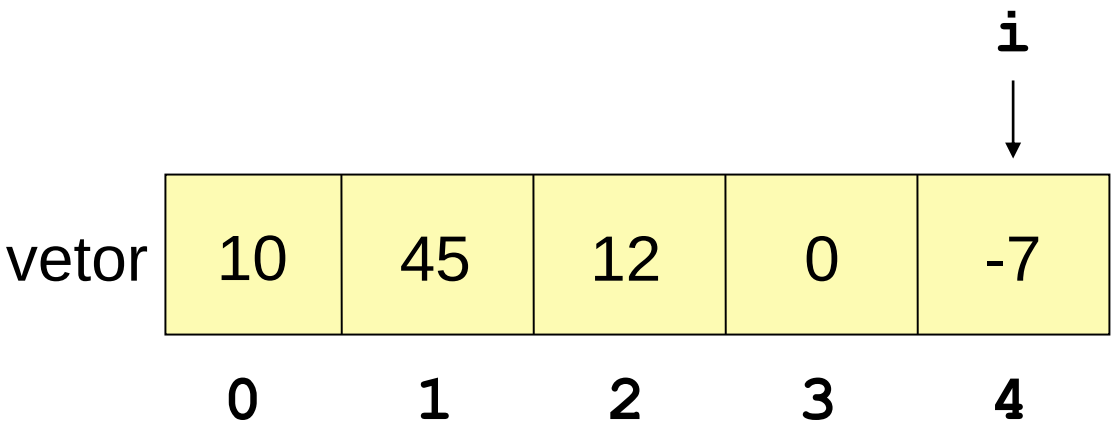
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

**Iteração 5**

i = 4

tela

10 45 12 0 -7



# (Relembrando) Imprimir os valores do arranjo em uma linha

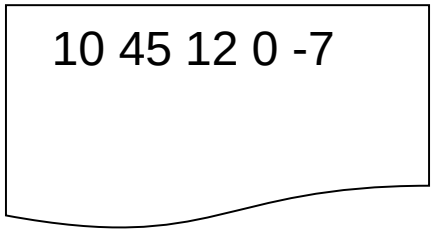
```
...
Para (i = 0; i<MAX; i++)
Faça
   Imprimir vetor[i];
Fim
...
```

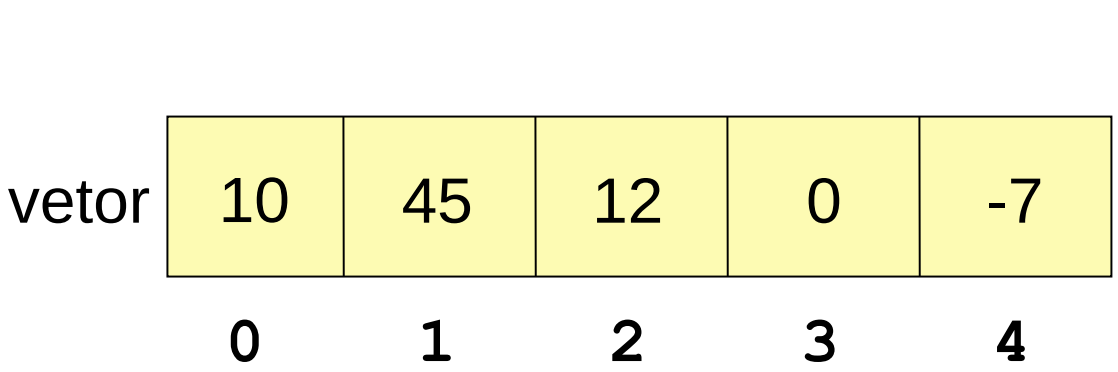
**Iteração 6**

i = 5

Fora do laço...

tela



# Algoritmos de busca e classificação

- Encontram e classificam (**ordenam**) os valores contidos em arranjos segundo algum **critério**
- Existem diversos algoritmos já propostos
- Cada um é mais ou menos eficiente e complexo de implementar que outro
  - Insertion sort
  - Selection sort
  - Bubble sort
  - Quick sort
  - ...

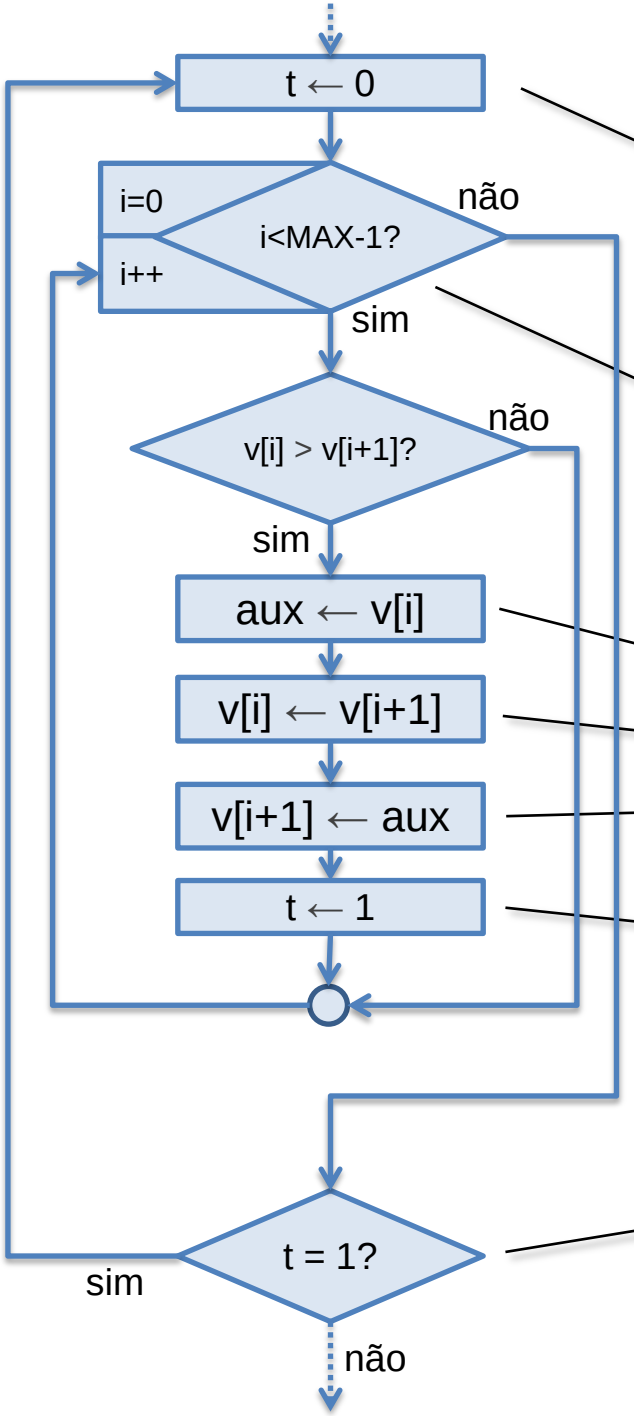
Ordenação por flutuação ou por "bolha"

Nesta aula ►

# Bubble Sort

- A ideia geral é fazer os “maiores” elementos flutuarem para o final do arranjo, como bolhas na água
- Etapas
  - Percorrer várias vezes o vetor
  - Encontrar pares de valores fora de ordem
  - Trocar os valores de posição par a par
  - Repetir enquanto houverem trocas
  - **Otimização:** controlar o tamanho a parte desordenada do vetor evita percorrer sempre até o final

# Bubble Sort (simplificado)



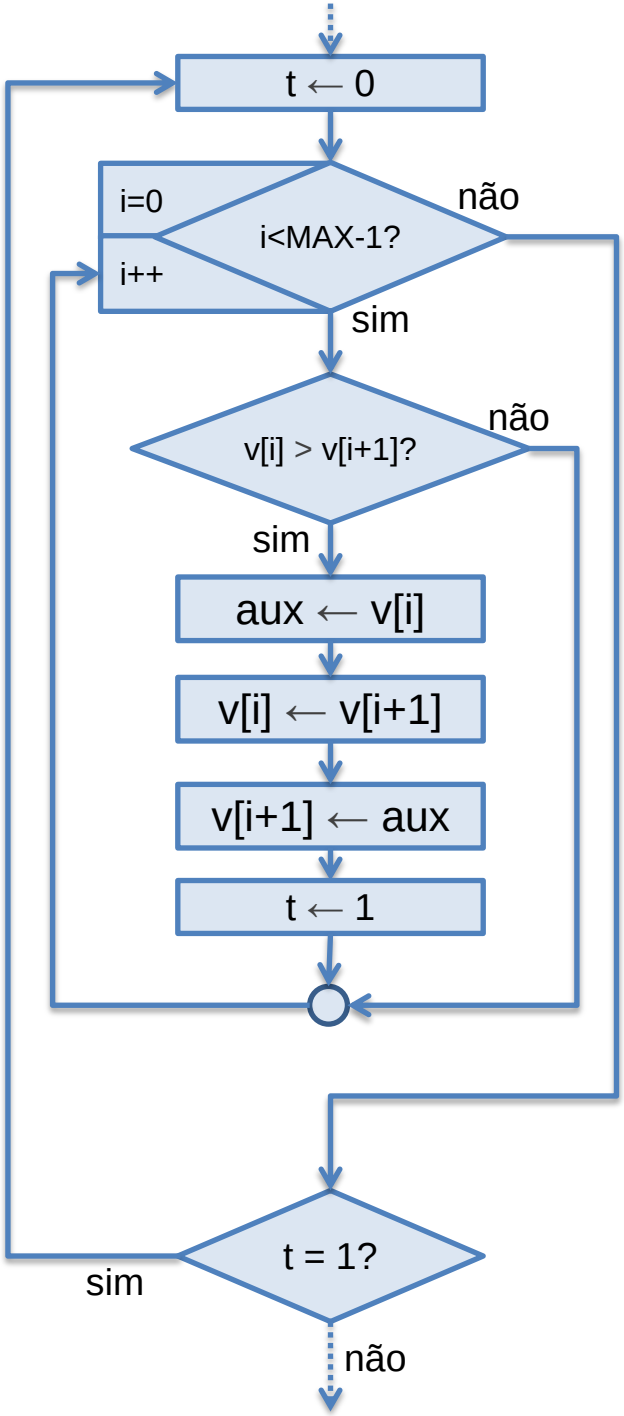

Sinalizador, indica se houve troca

Cuidar que o i não pode ir até o final do vetor

Troca os valores desordenados

Houve troca, então muda o sinalizador

Houve alguma troca? Caso positivo, é necessário outra passada

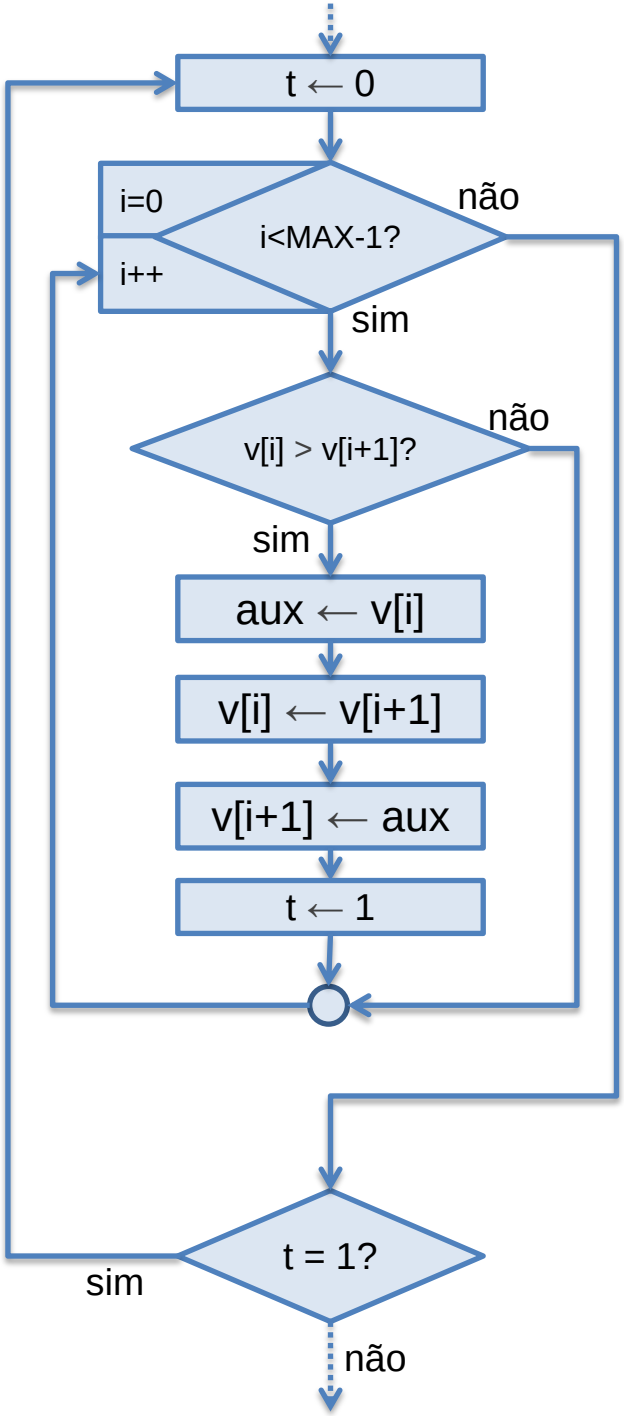



# Bubble Sort (simplificado)

**Iteração do-while 1**

t = 0

| vetor | 10 | 45 | 12 | 0 | -7 |
|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** |



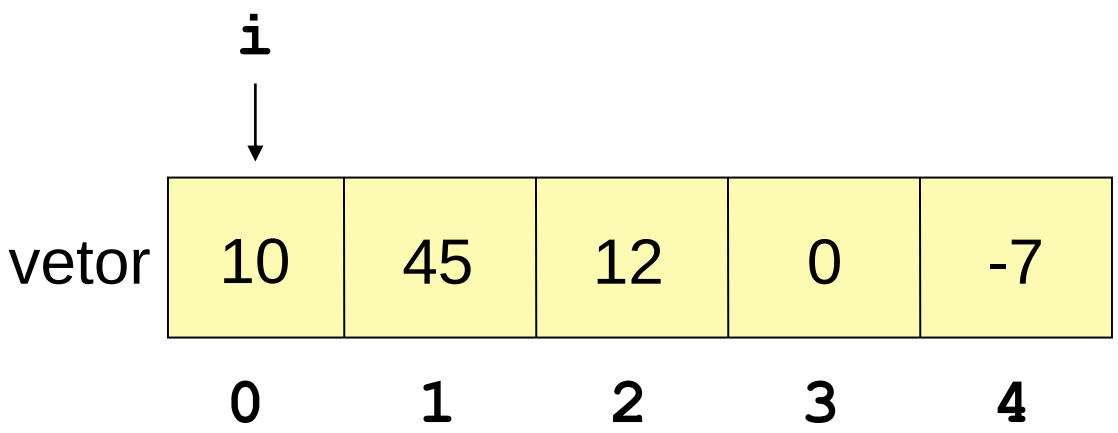

# Bubble Sort (simplificado)

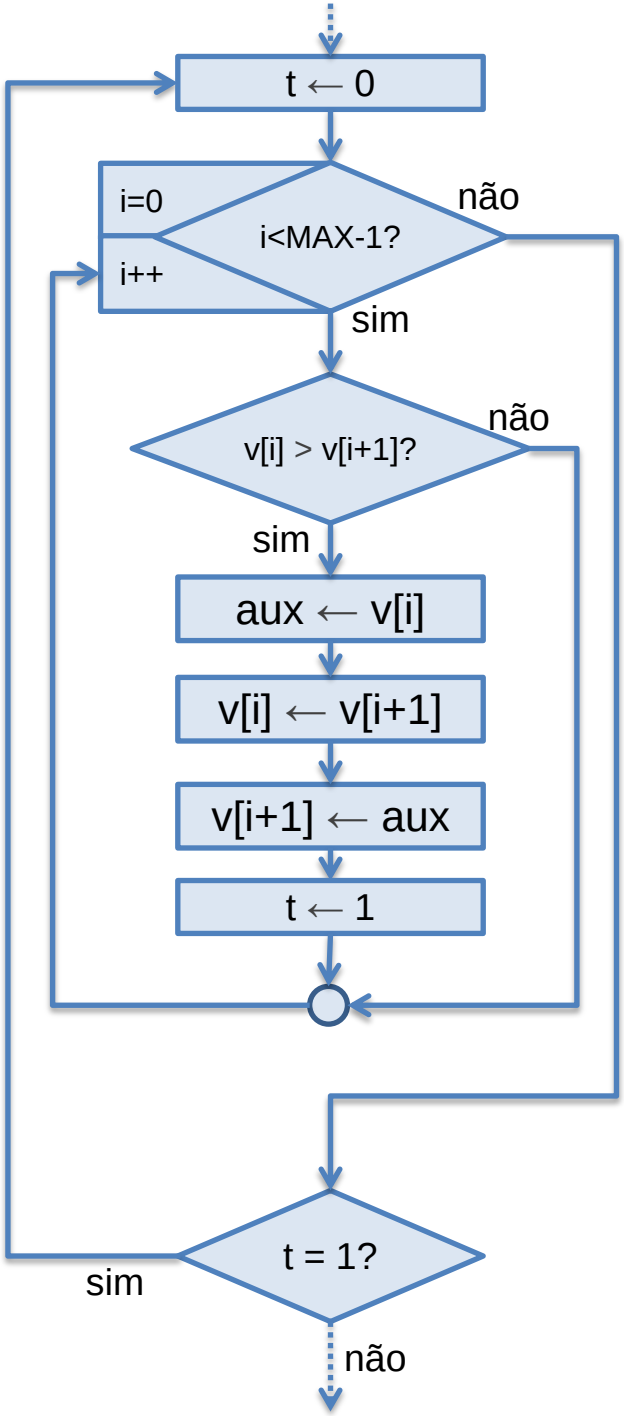
**Iteração do-while 1**

t = 0

**Iteração for 1**

i = 0





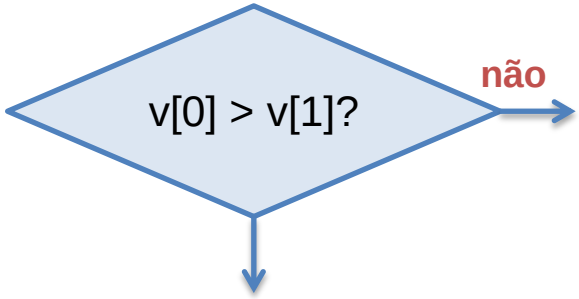

# Bubble Sort (simplificado)

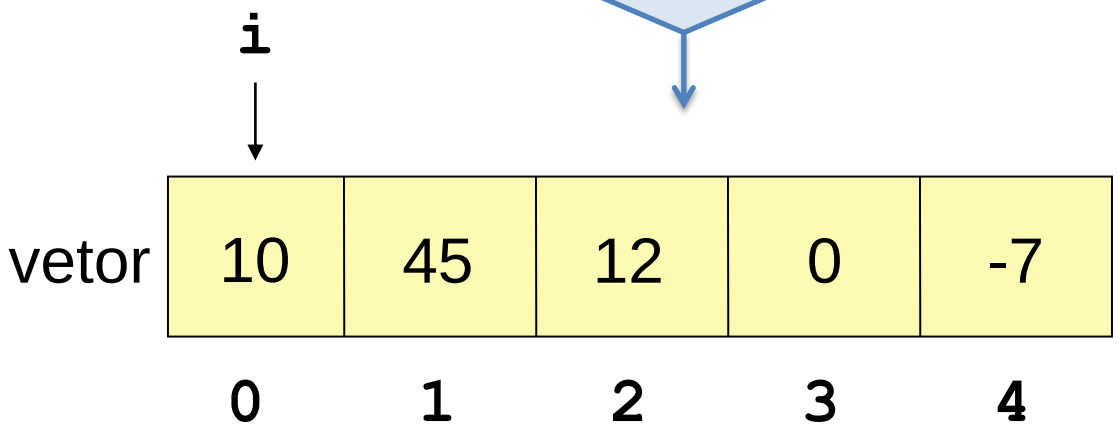
**Iteração do-while 1**

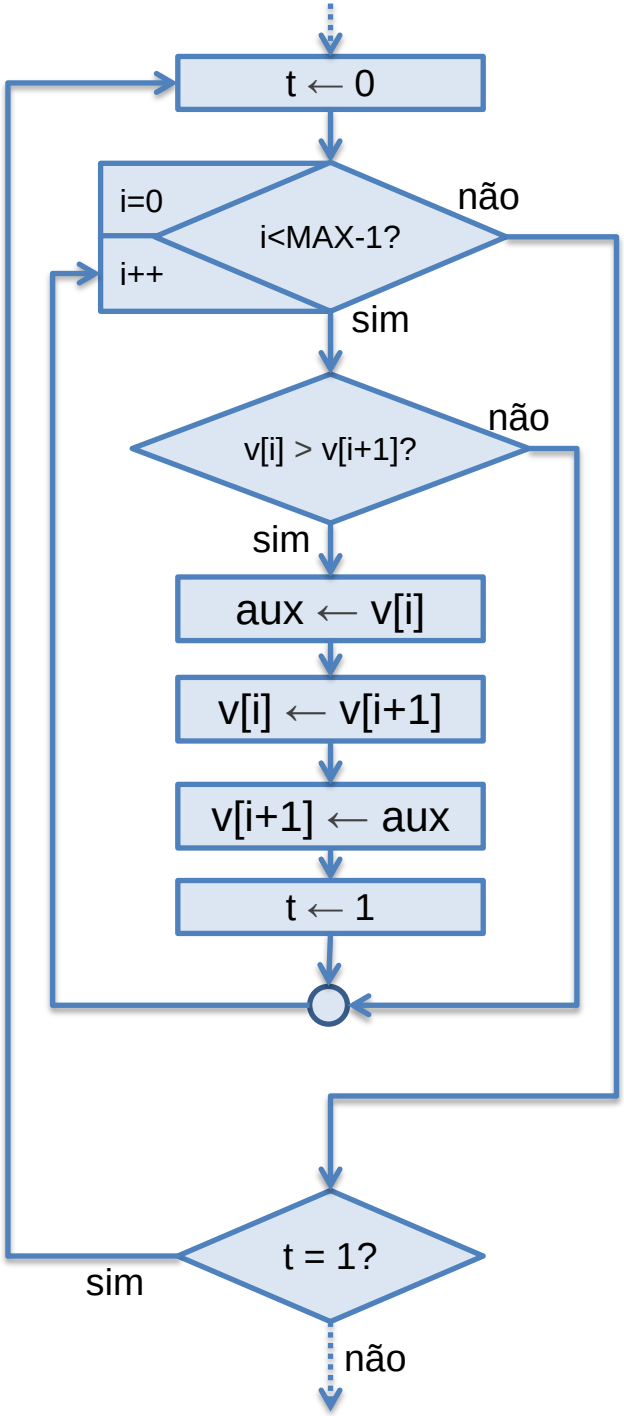
t = 0

**Iteração for 1**

i = 0





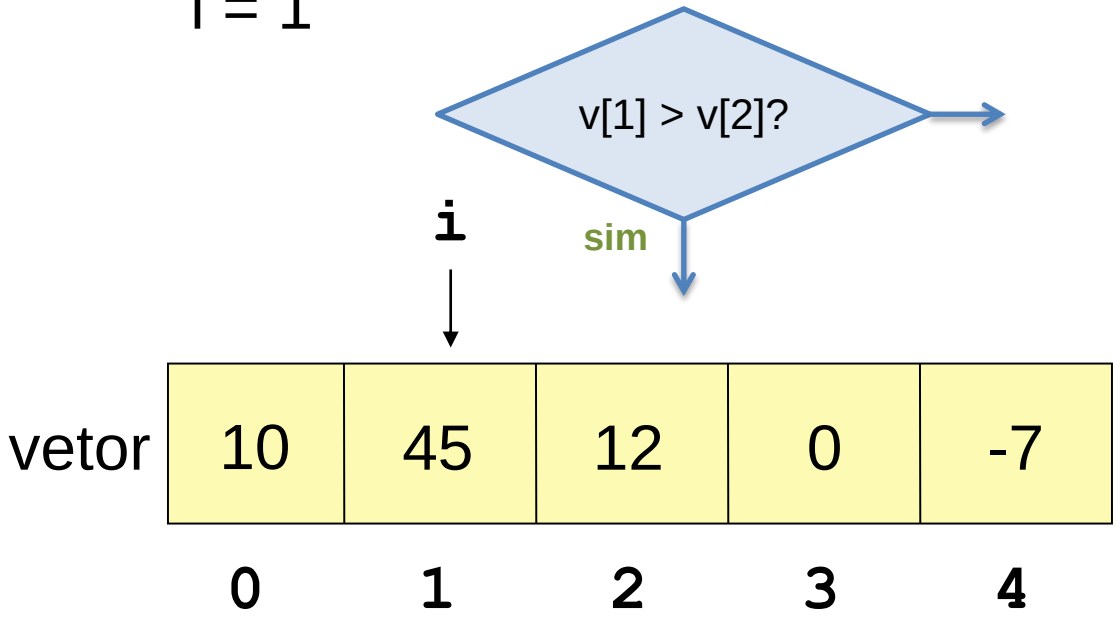

# Bubble Sort (simplificado)

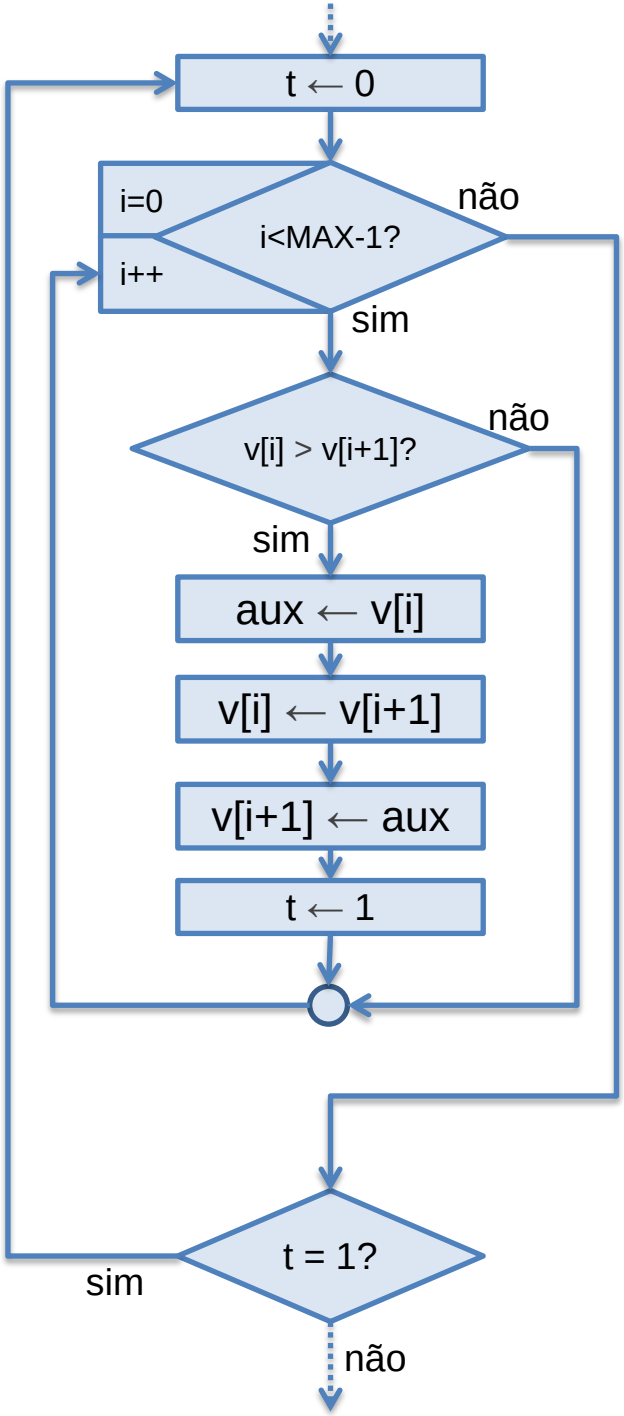
**Iteração do-while 1**

t = 0

**Iteração for 2**

i = 1






# Bubble Sort (simplificado)

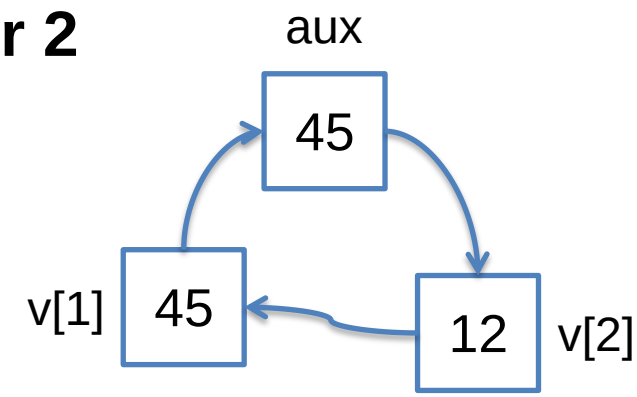
**Iteração do-while 1**

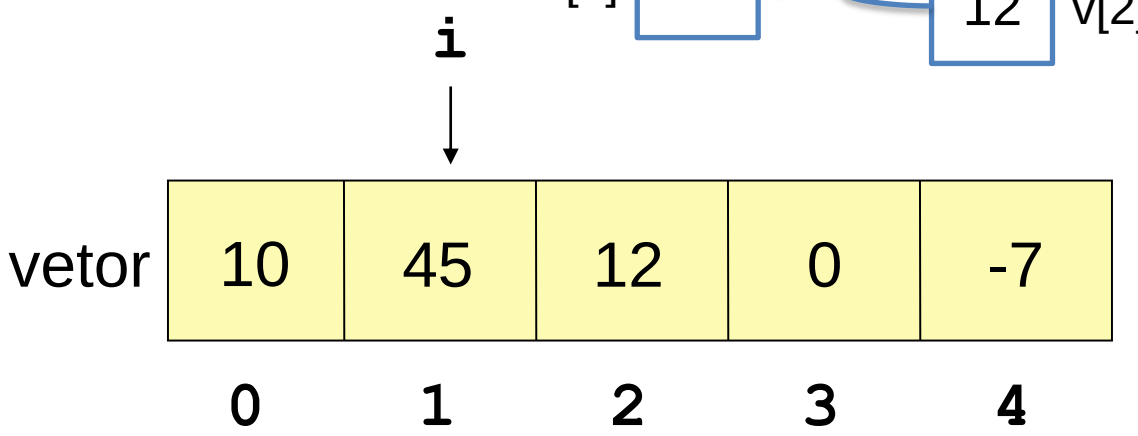
t = 0

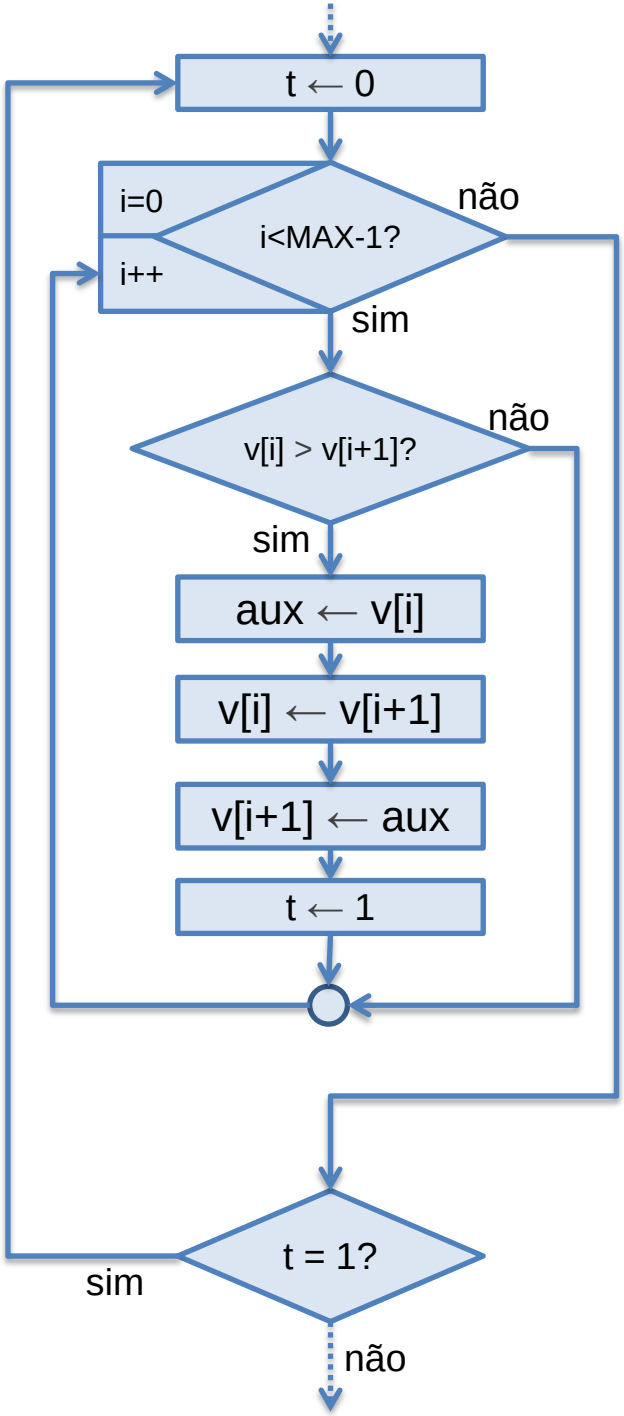
**Iteração for 2**

i = 1

aux = 45






# Bubble Sort (simplificado)

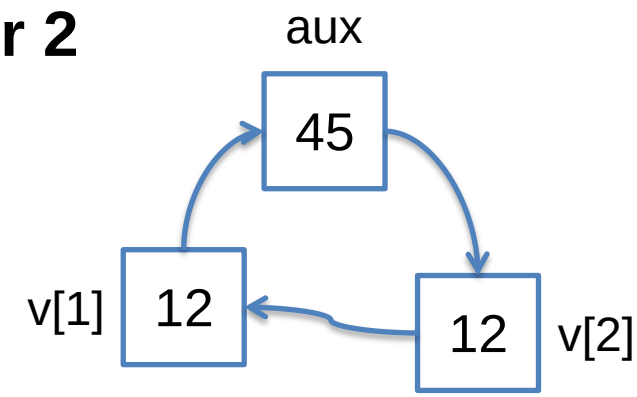
**Iteração do-while 1**

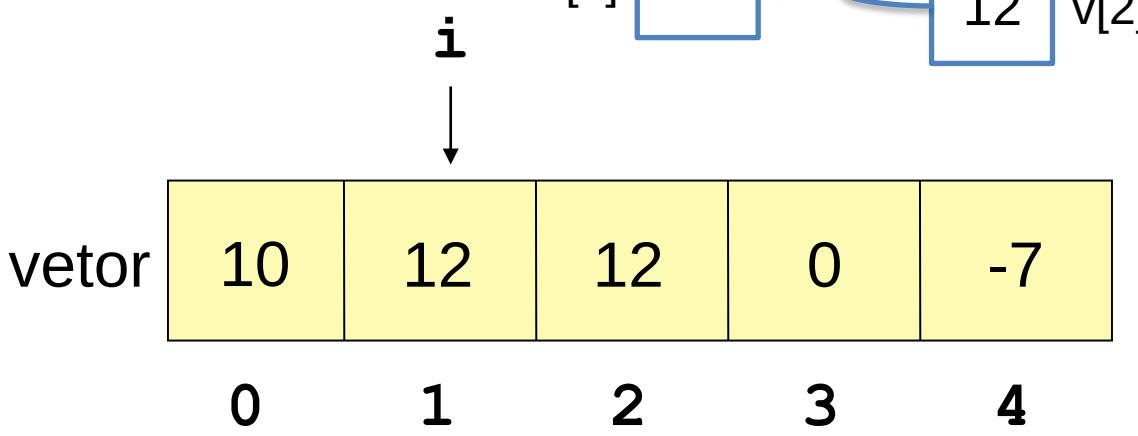
t = 0

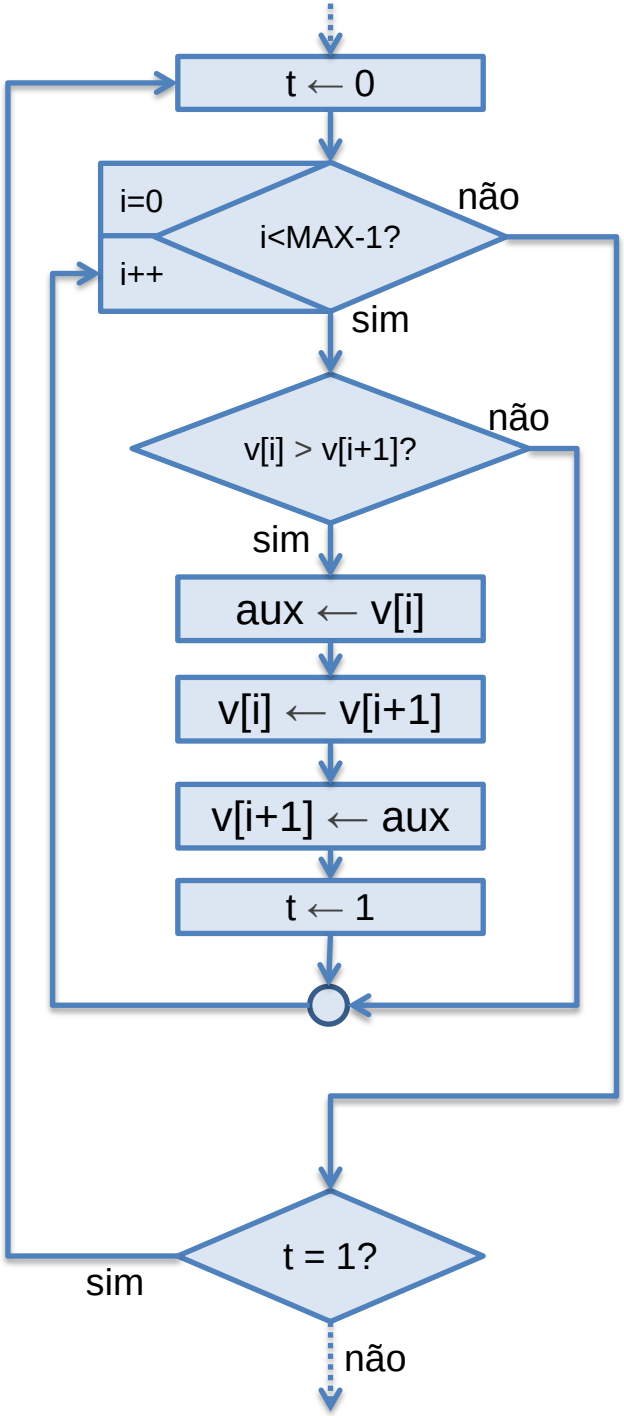
**Iteração for 2**

i = 1

aux = 45






# Bubble Sort (simplificado)

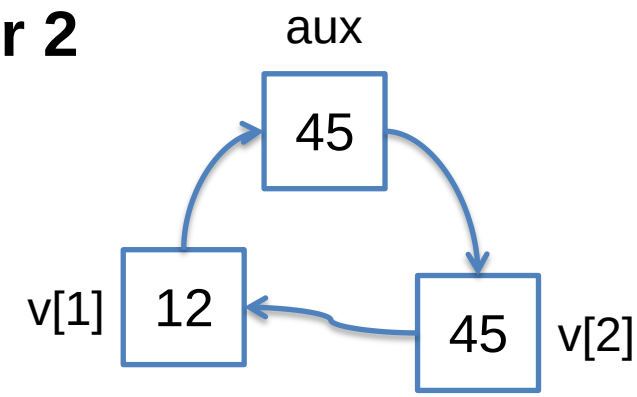
**Iteração do-while 1**

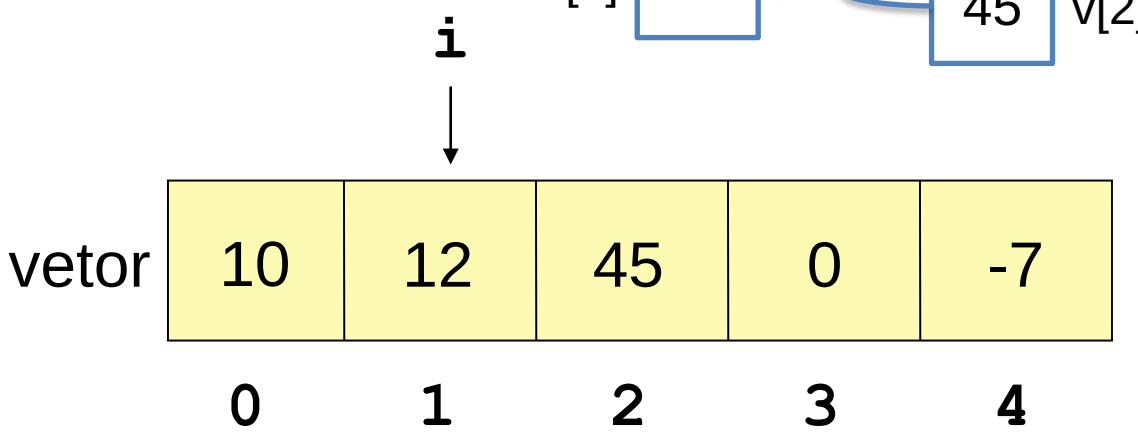
t = 0

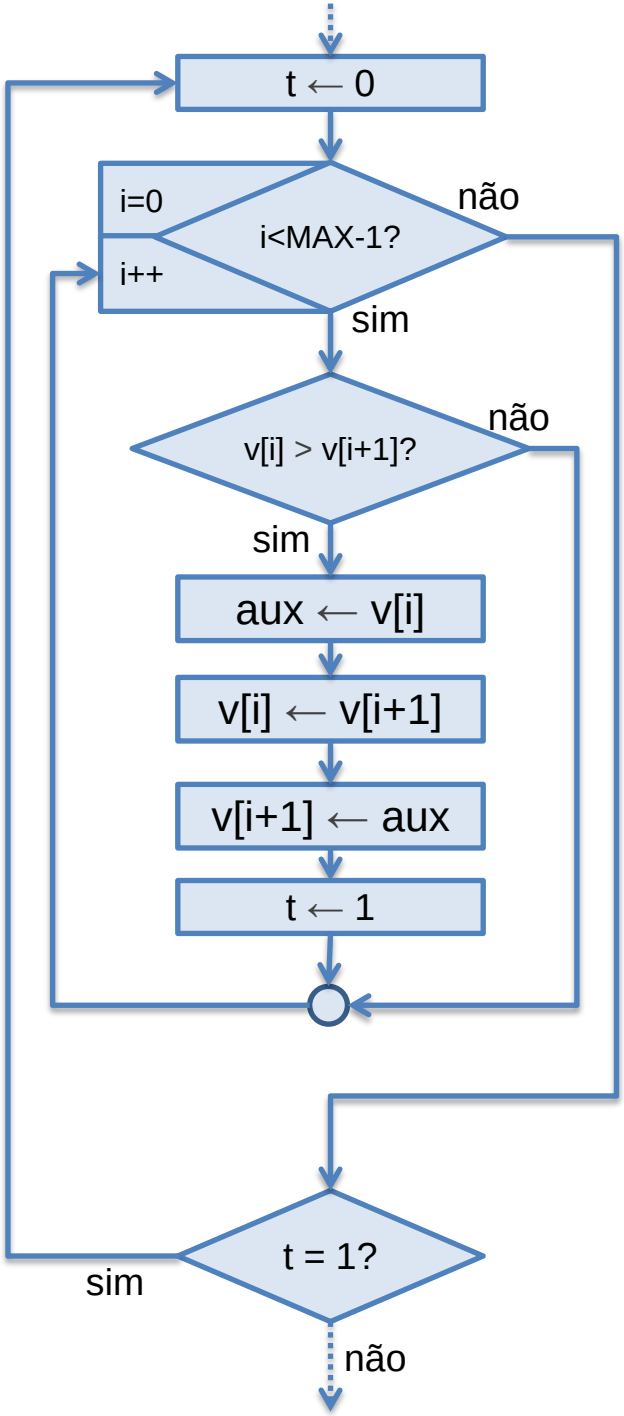
**Iteração for 2**

i = 1

aux = 45






# Bubble Sort (simplificado)

**Iteração do-while 1**

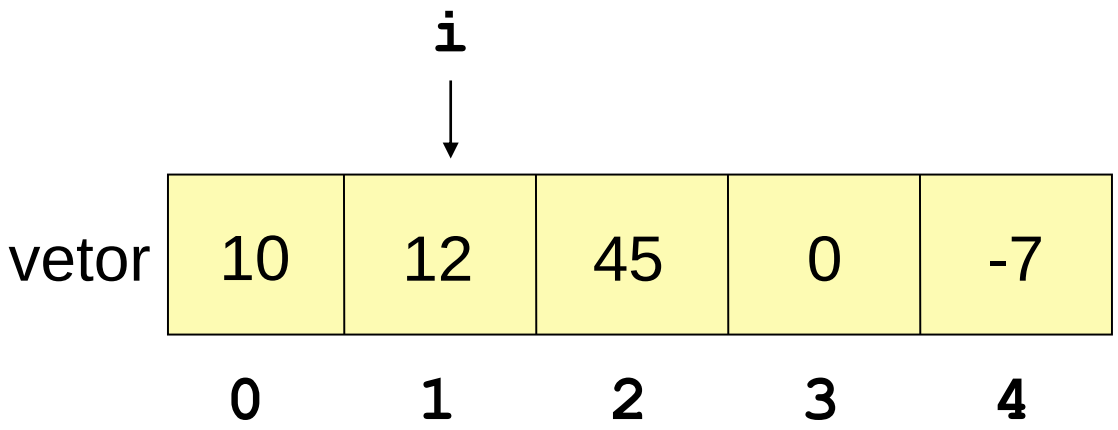
t = 1

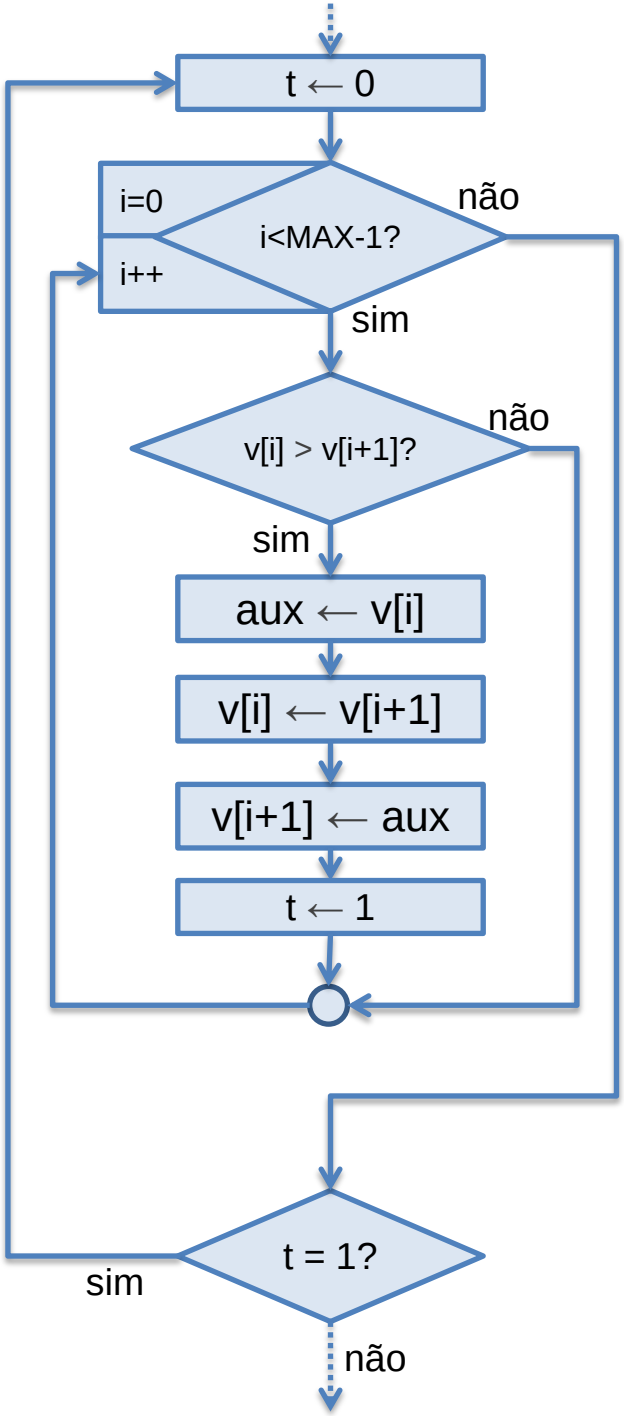
Houve troca, então muda o sinalizador

**Iteração for 2**

i = 1

aux = 45






# Bubble Sort (simplificado)

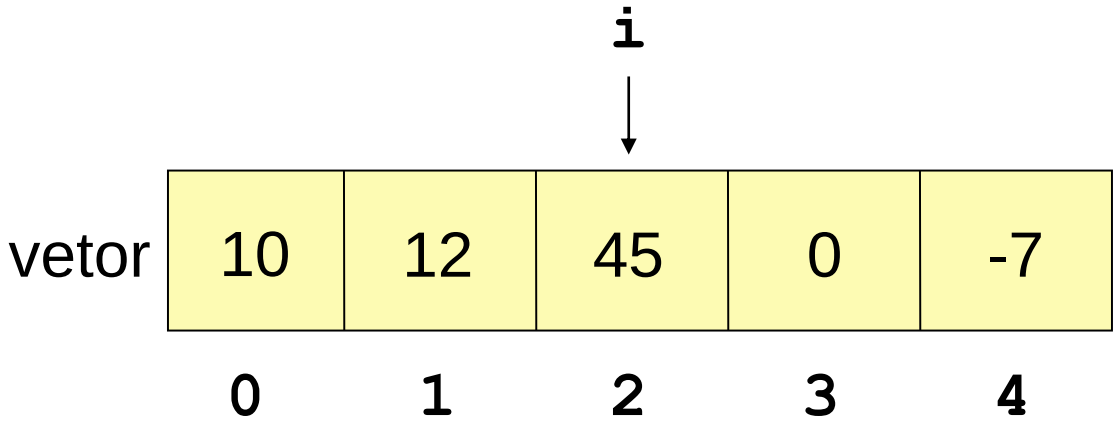
**Iteração do-while 1**

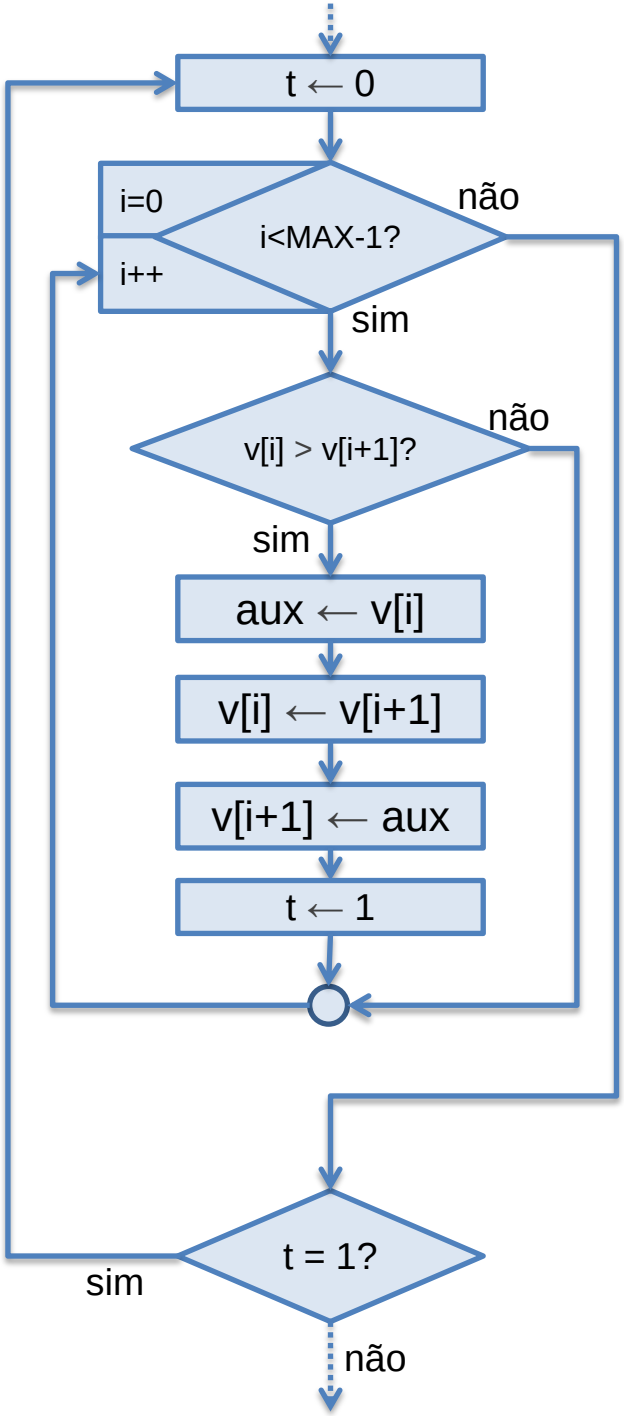
t = 1

**Iteração for 3**

i = 2

aux = 45






# Bubble Sort (simplificado)

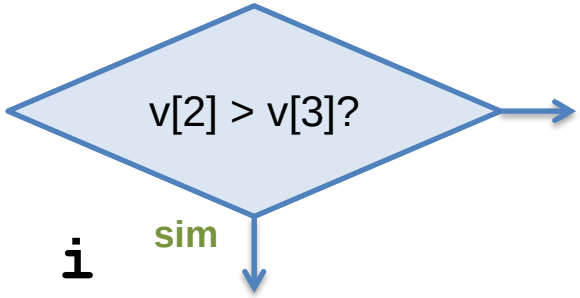
**Iteração do-while 1**

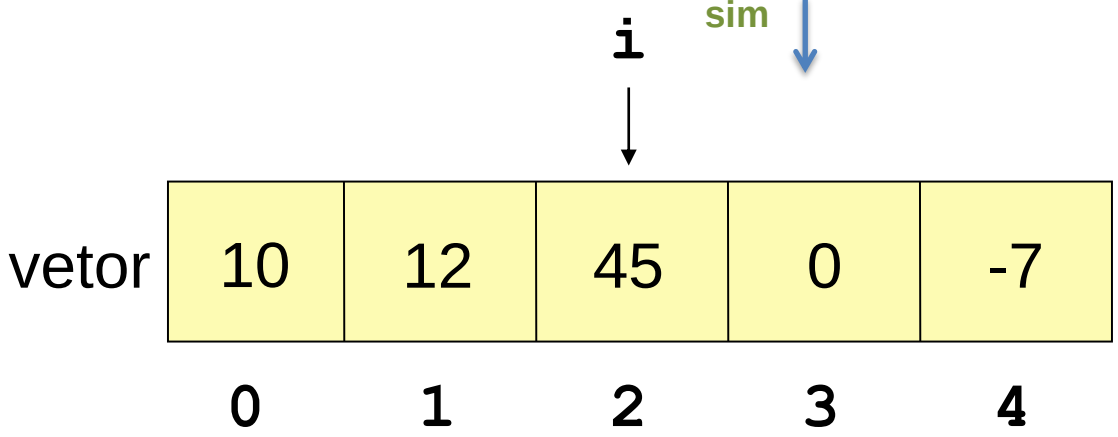
t = 1

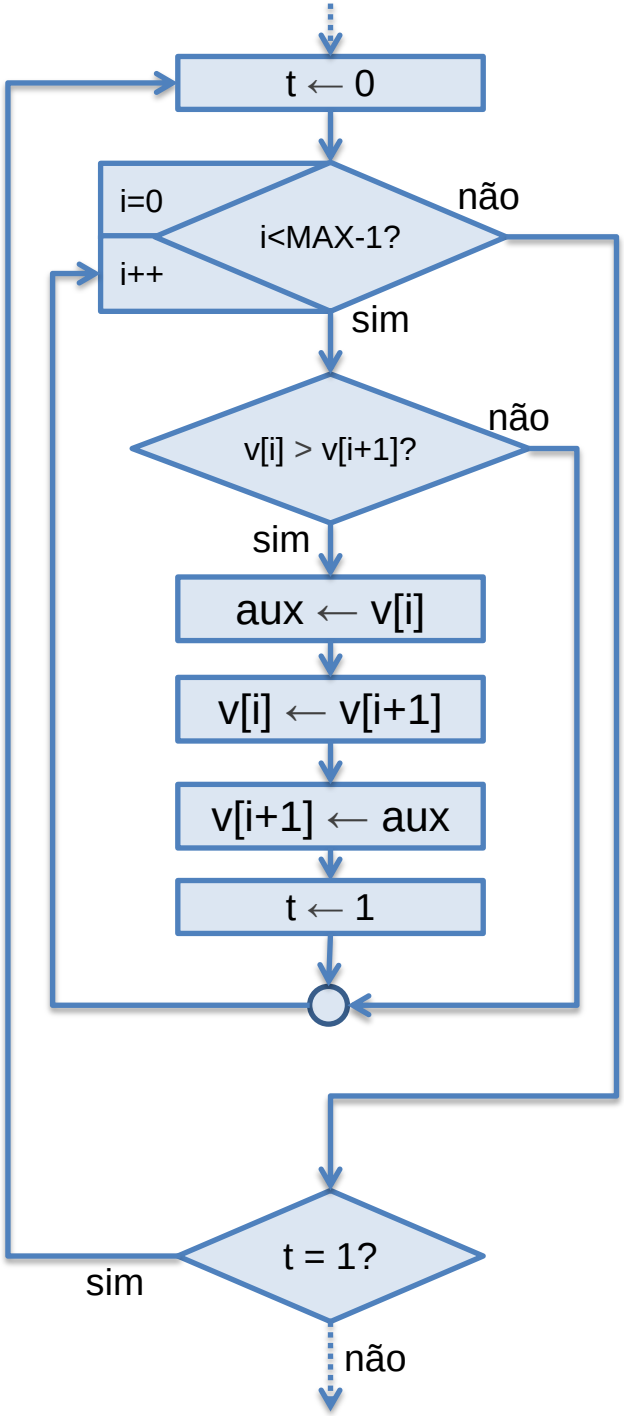
**Iteração for 3**

i = 2

aux = 45






# Bubble Sort (simplificado)

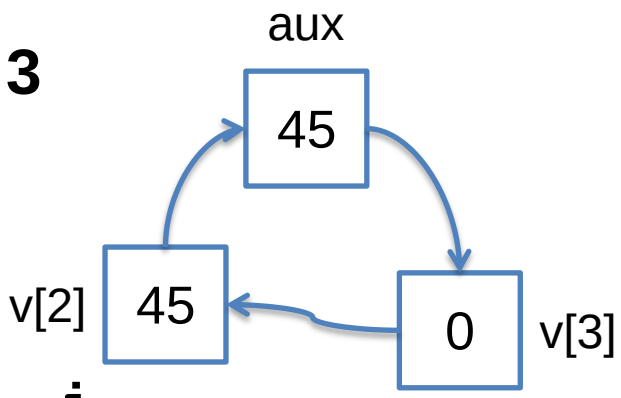
**Iteração do-while 1**

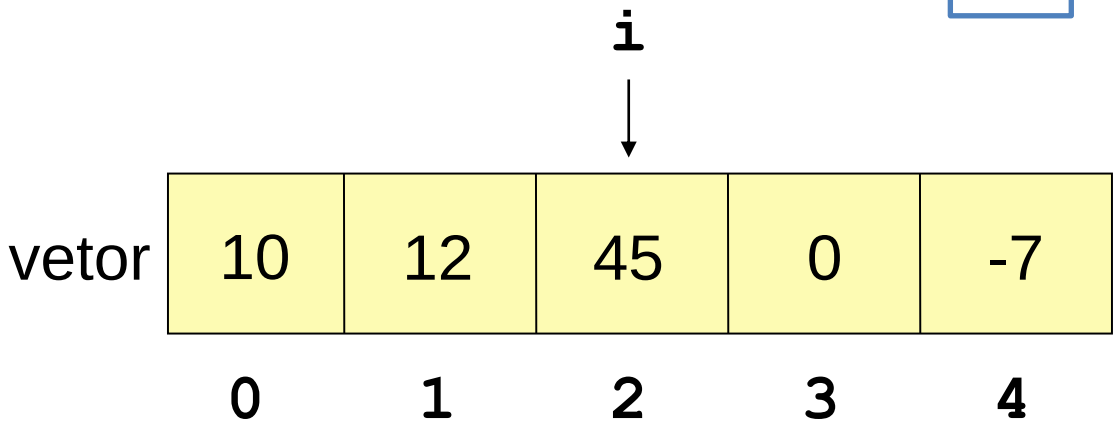
t = 1

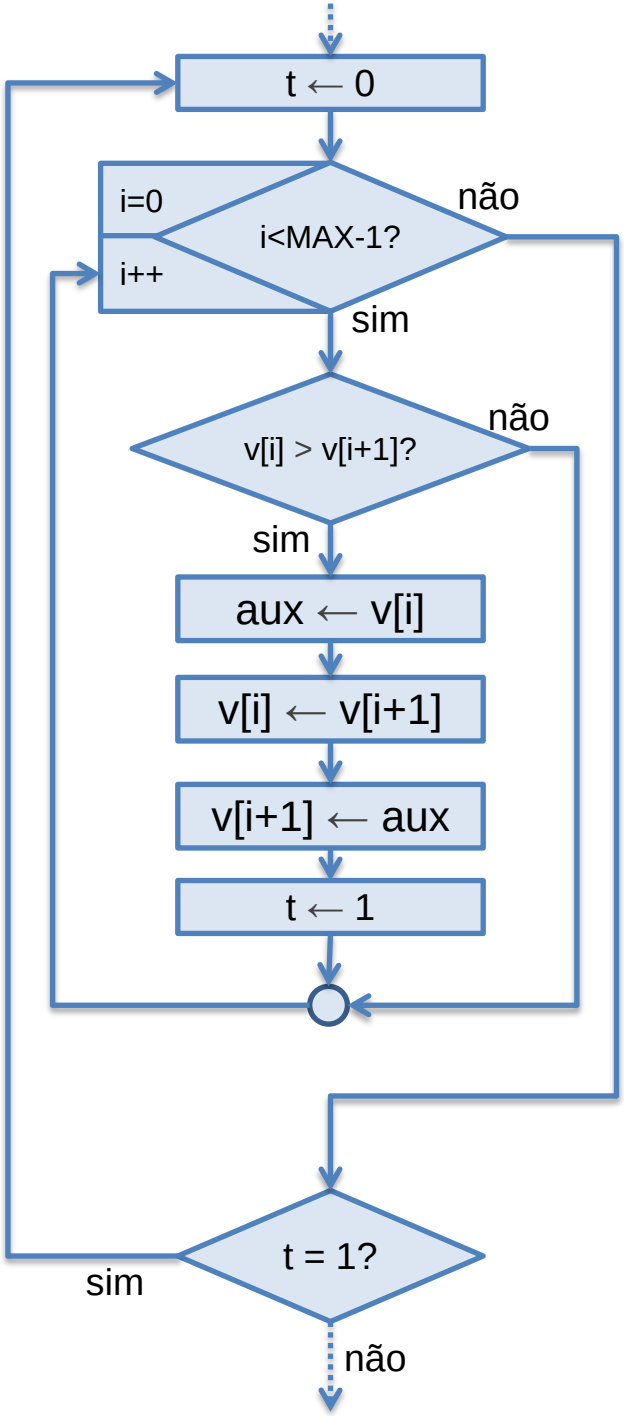
**Iteração for 3**

i = 2

aux = 45






# Bubble Sort (simplificado)

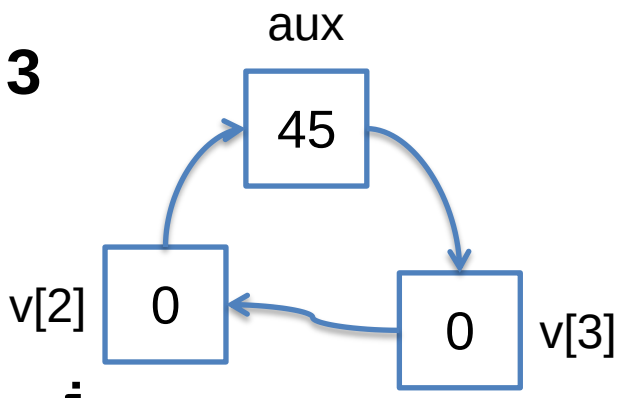
**Iteração do-while 1**

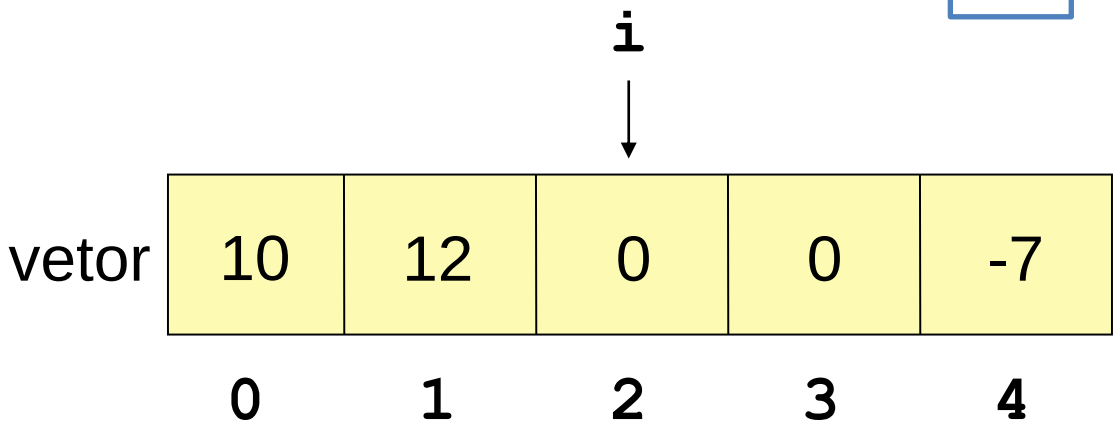
t = 1

**Iteração for 3**

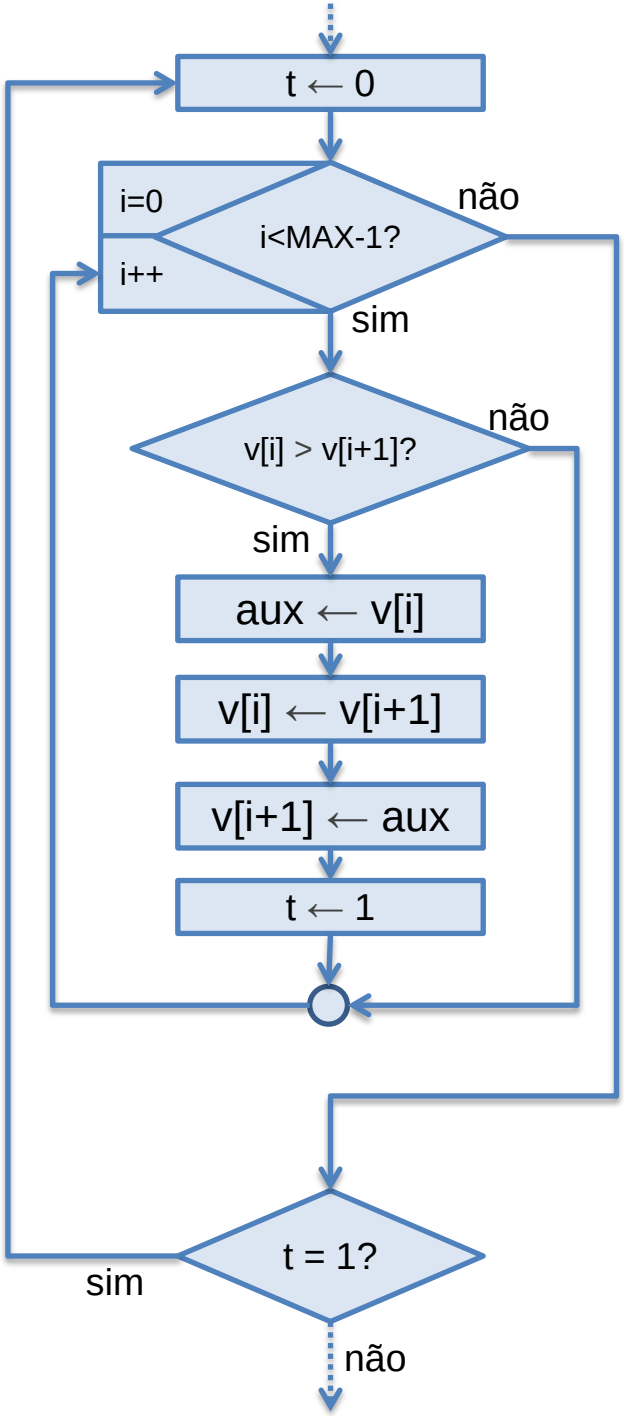
i = 2

aux = 45



# Bubble Sort (simplificado)



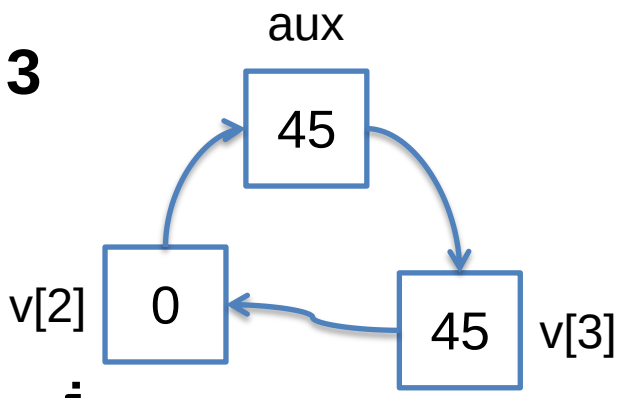

**Iteração do-while 1**

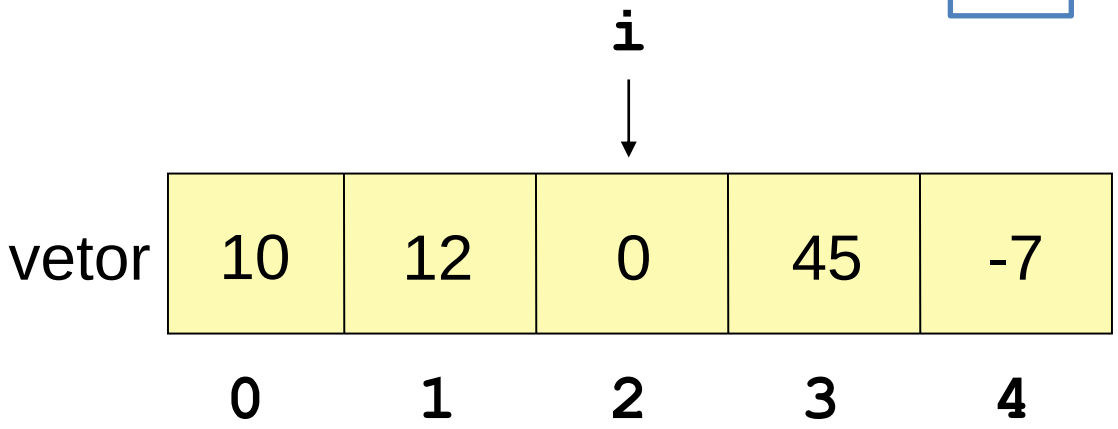
t = 1

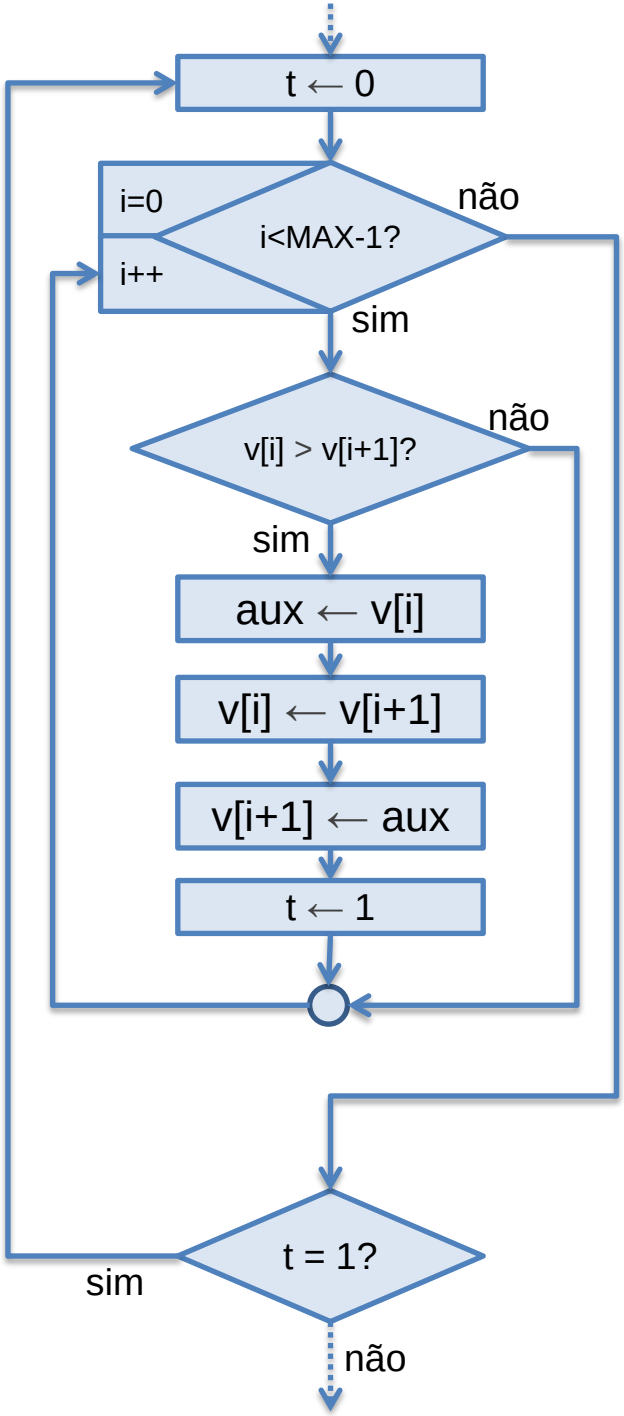
**Iteração for 3**

i = 2

aux = 45






# Bubble Sort (simplificado)

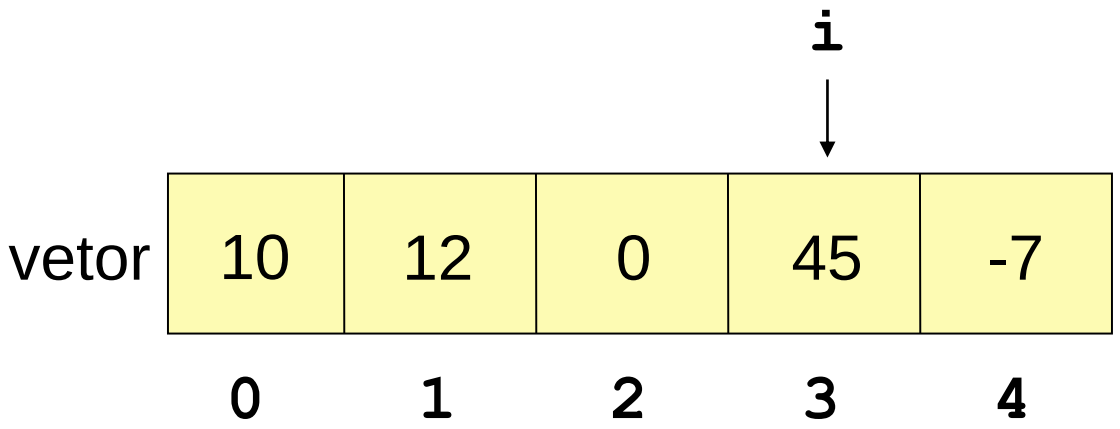
**Iteração do-while 1**

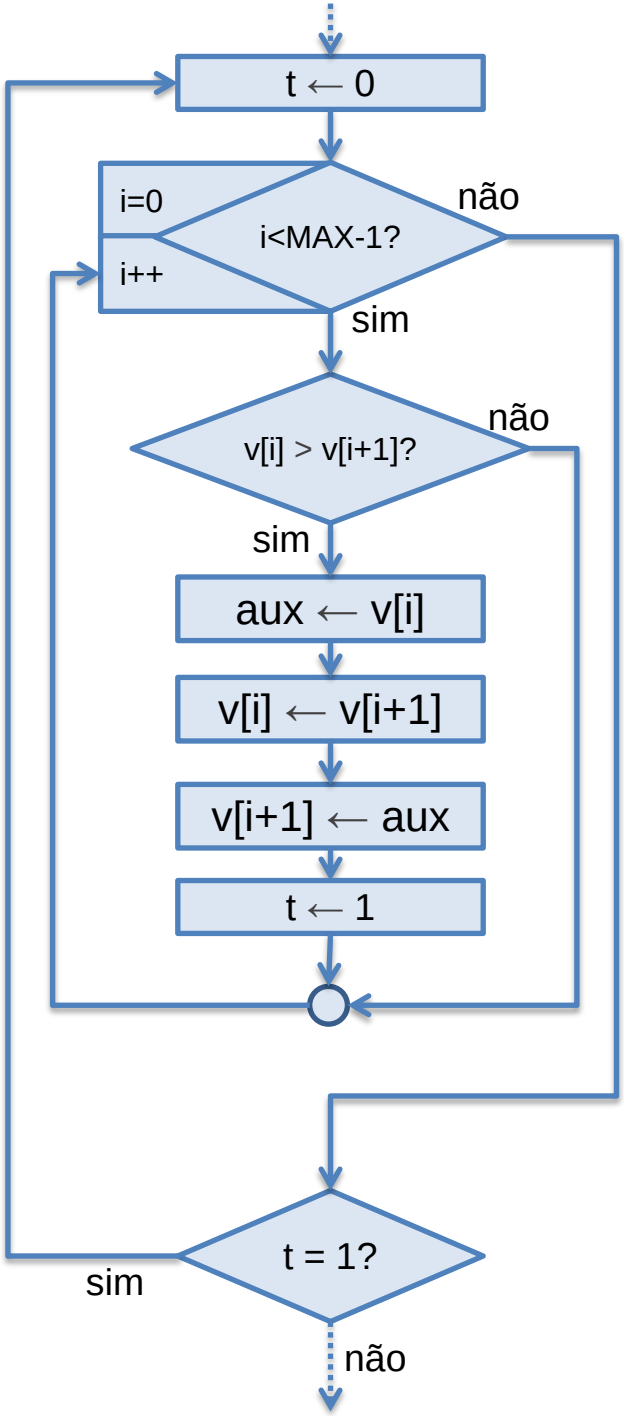
t = 1

**Iteração for 4**

i = 3

aux = 45






# Bubble Sort (simplificado)

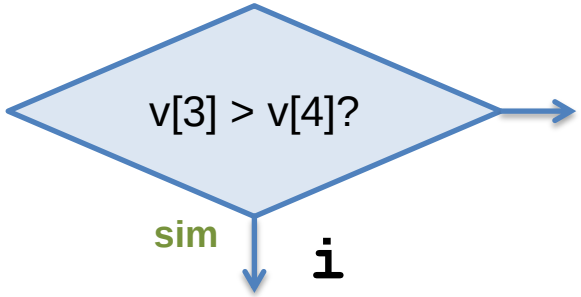
**Iteração do-while 1**

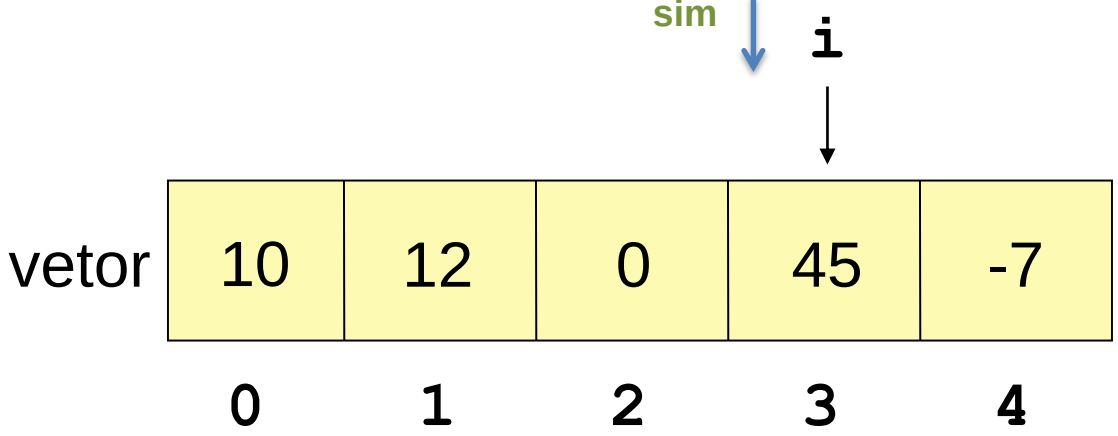
t = 1

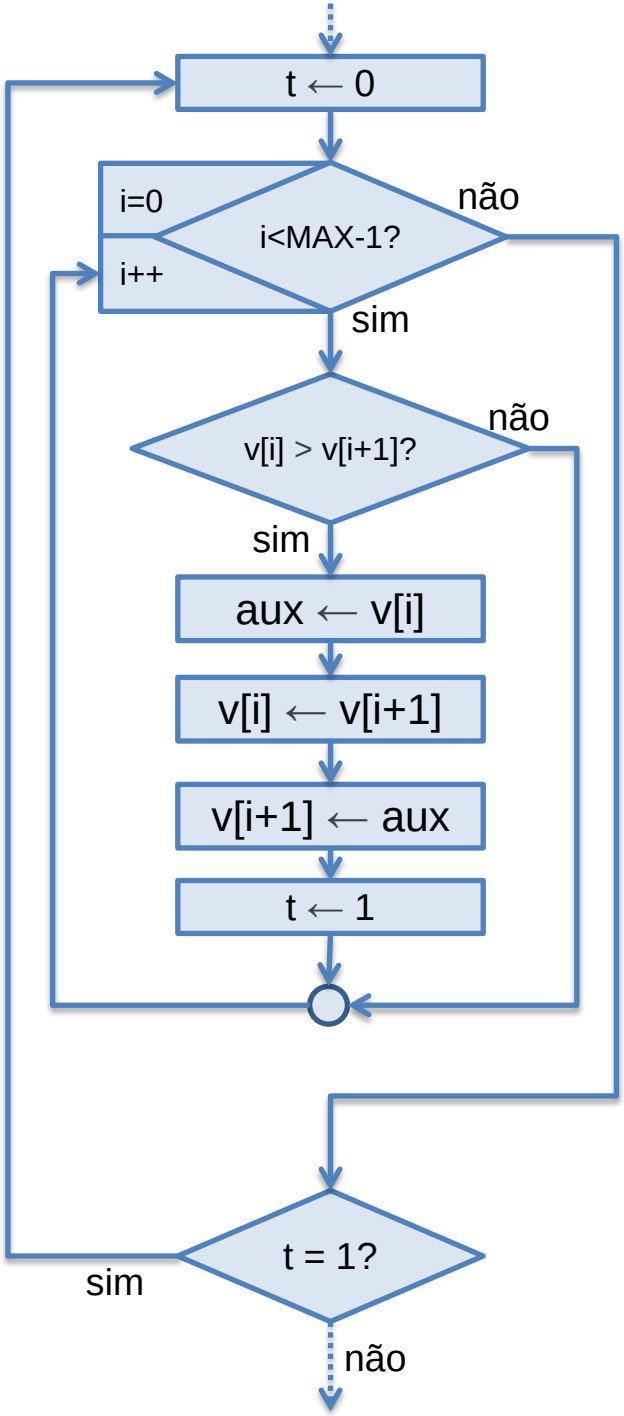
**Iteração for 4**

i = 3

aux = 45






# Bubble Sort (simplificado)

**Iteração do-while 1**

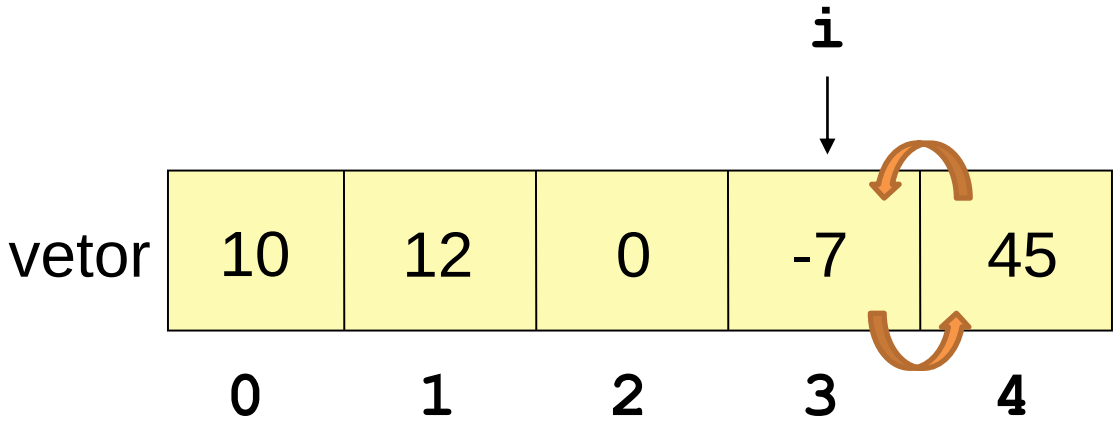
t = 1

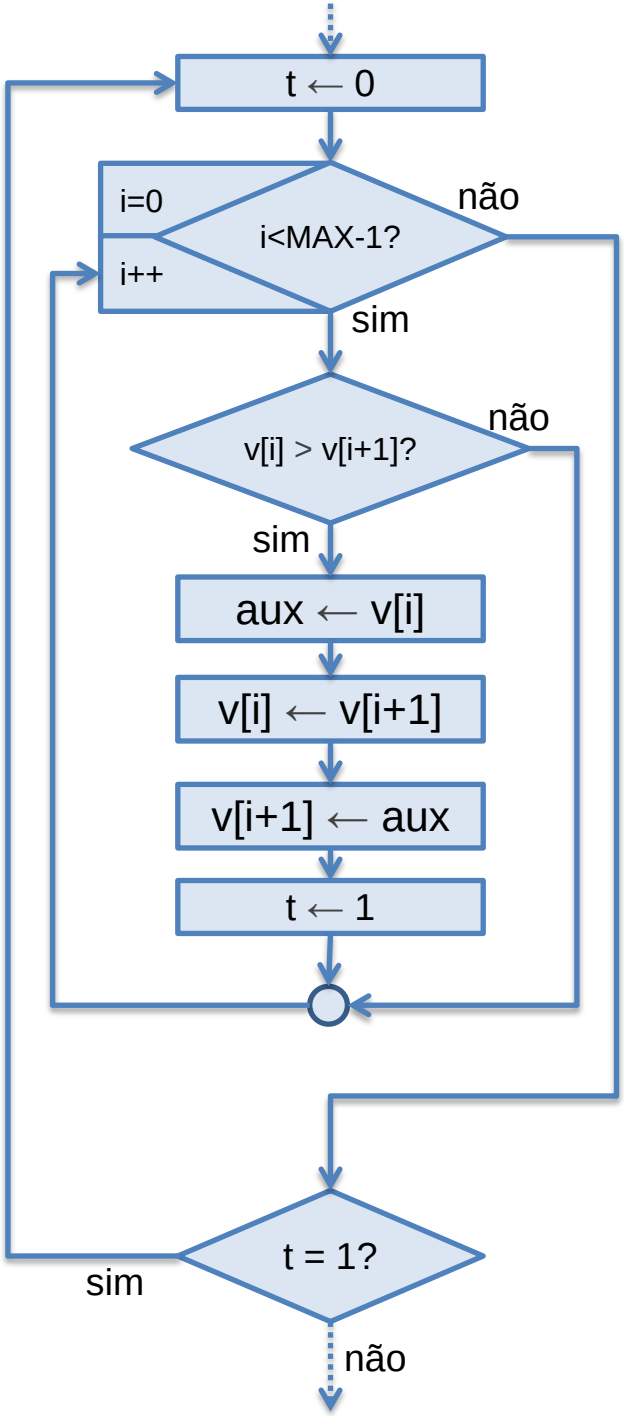
**Iteração for 4**

i = 3

aux = 45

*Realiza a Troca...*






# Bubble Sort (simplificado)

**Iteração do-while 1**

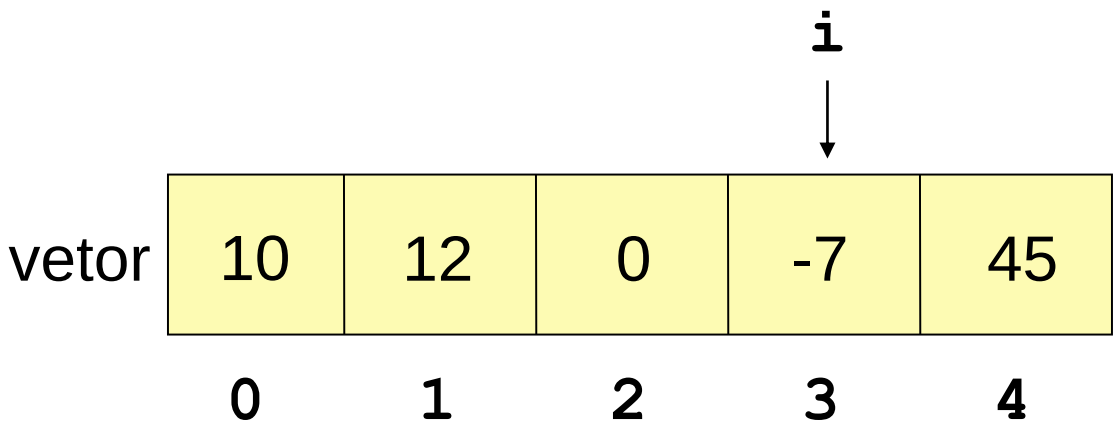
t = 1

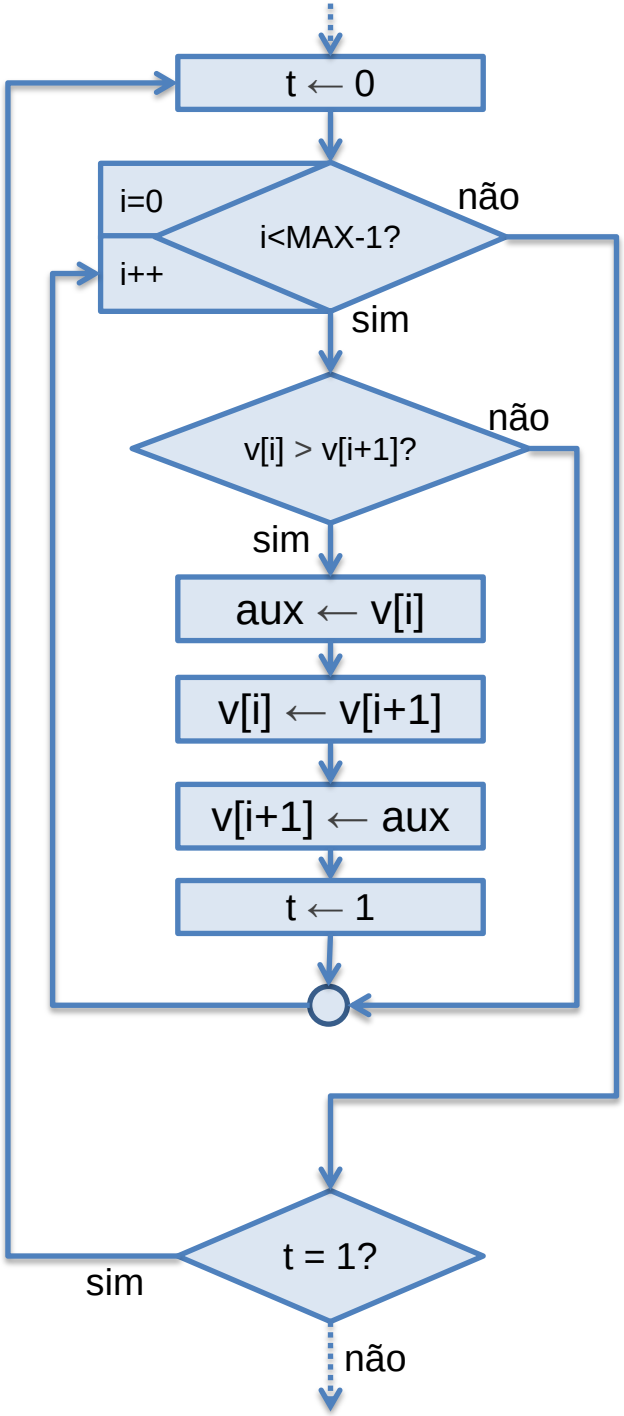
**Iteração for 4**

i = 3

aux = 45

Esta é a última posição que pode ser acessada nesse vetor devido ao teste com i+1






# Bubble Sort (simplificado)

**Iteração do-while 1**

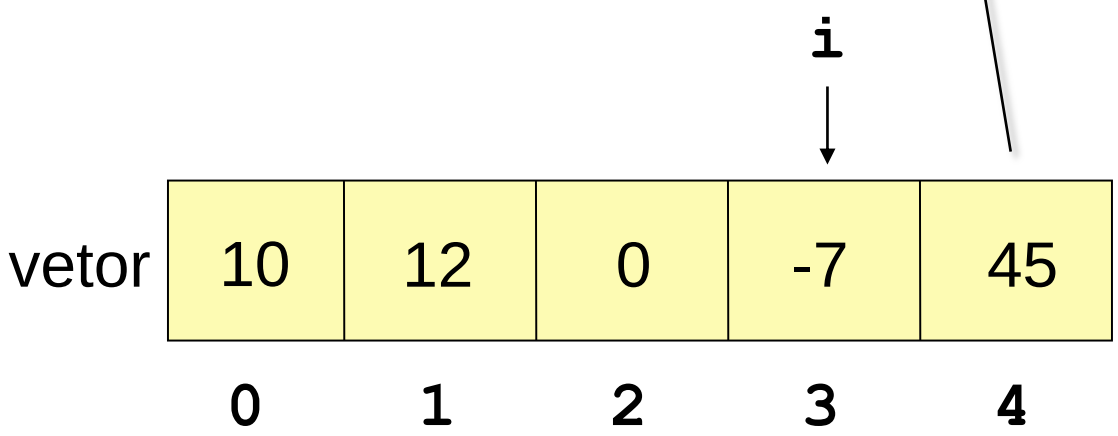
t = 1

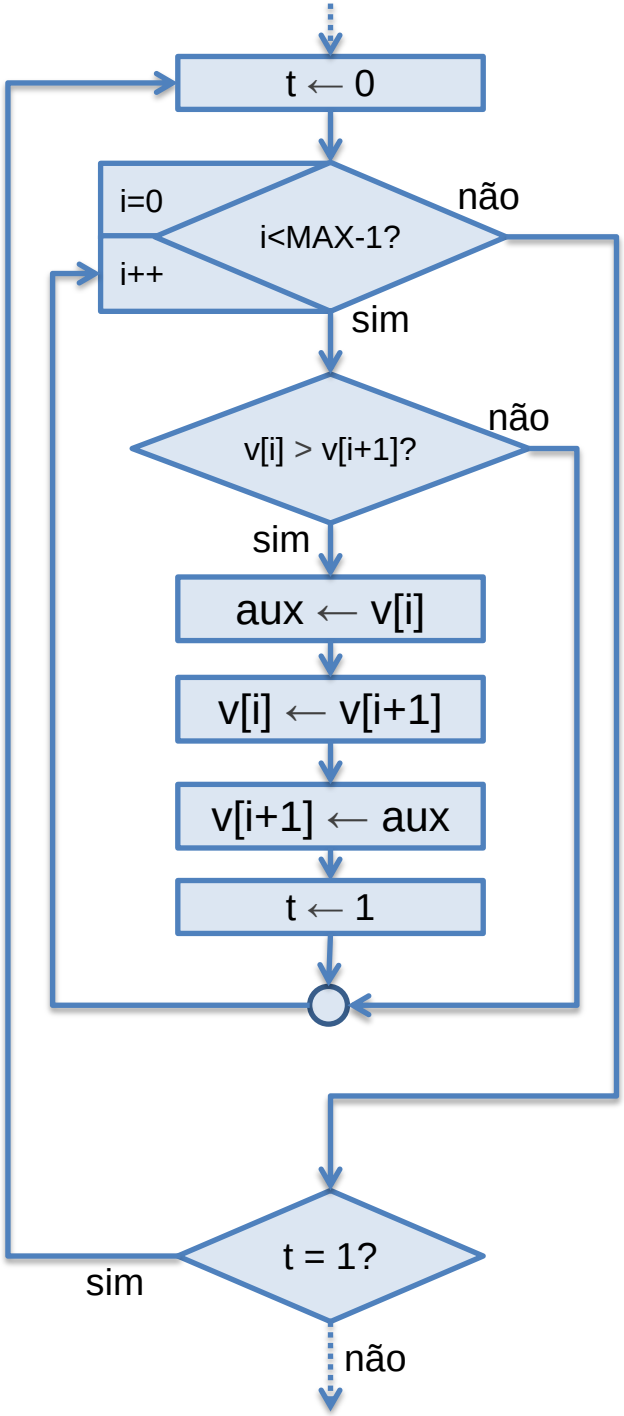
**Iteração for 4**

i = 3

aux = 45

Note que já conseguimos colocar o maior valor no final do arranjo

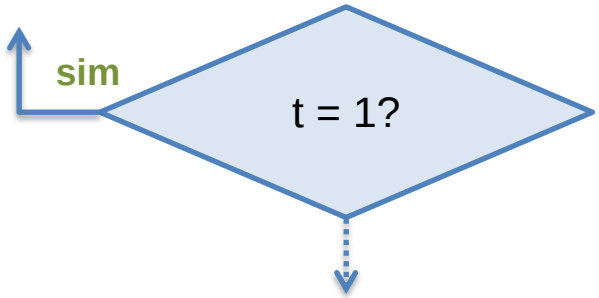




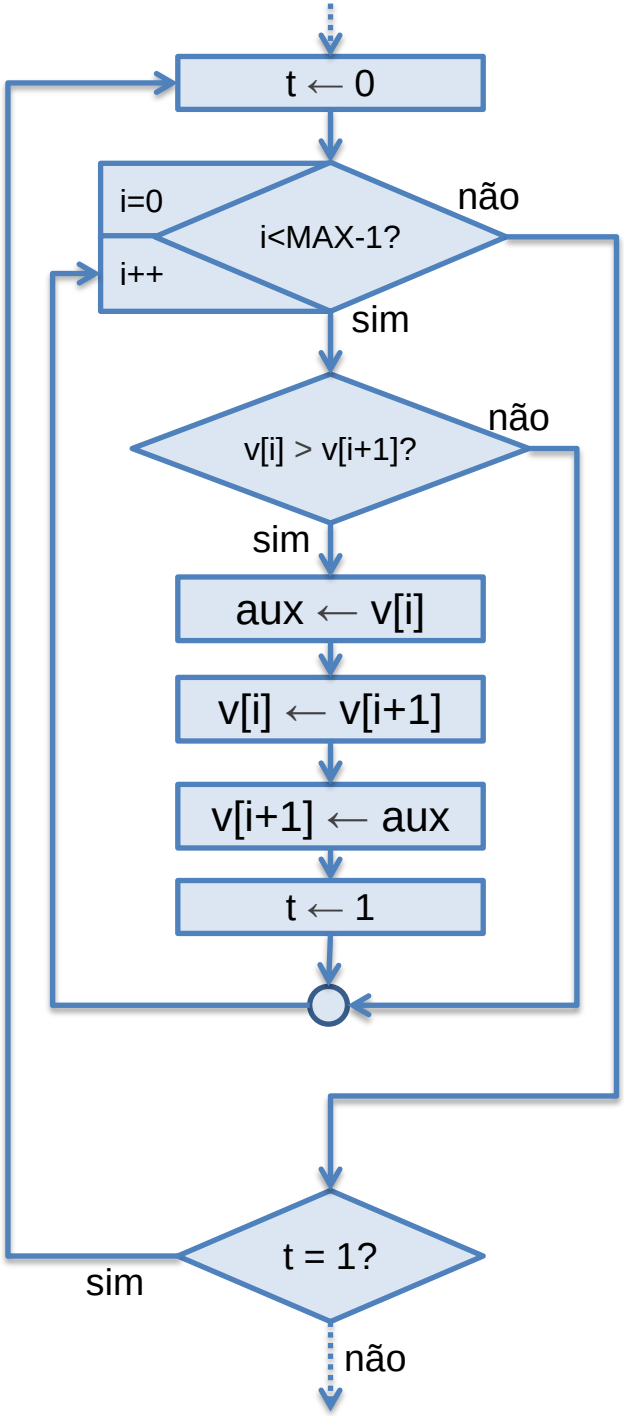

# Bubble Sort (simplificado)

## Iteração do-while 1

t = 1




| vetor | 10 | 12 | 0 | -7 | 45 |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |



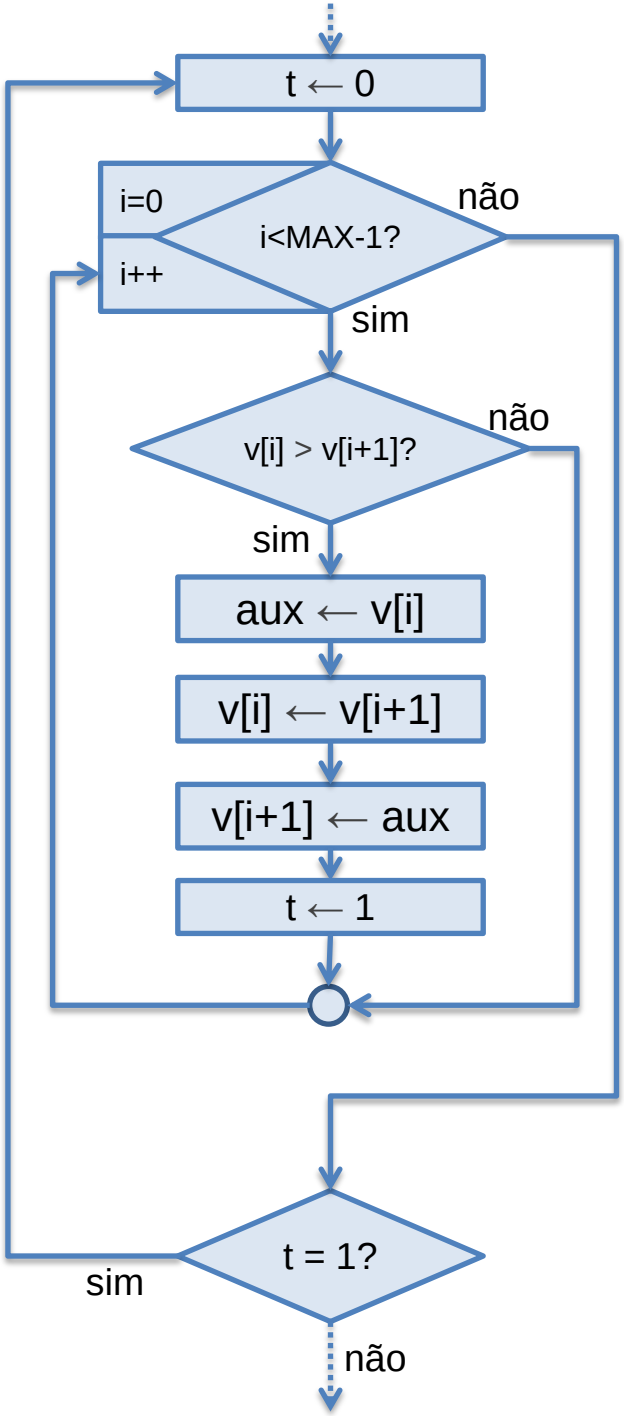

# Bubble Sort (simplificado)

## Iteração do-while 2

t = 0

Volta o sinalizador para zero indicando que ainda não houve troca nessa passada

| vetor | 10 | 12 | 0 | -7 | 45 |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |



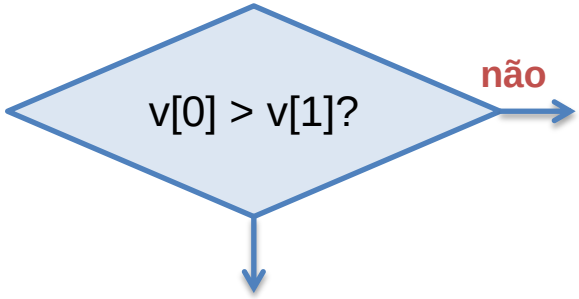

# Bubble Sort (simplificado)

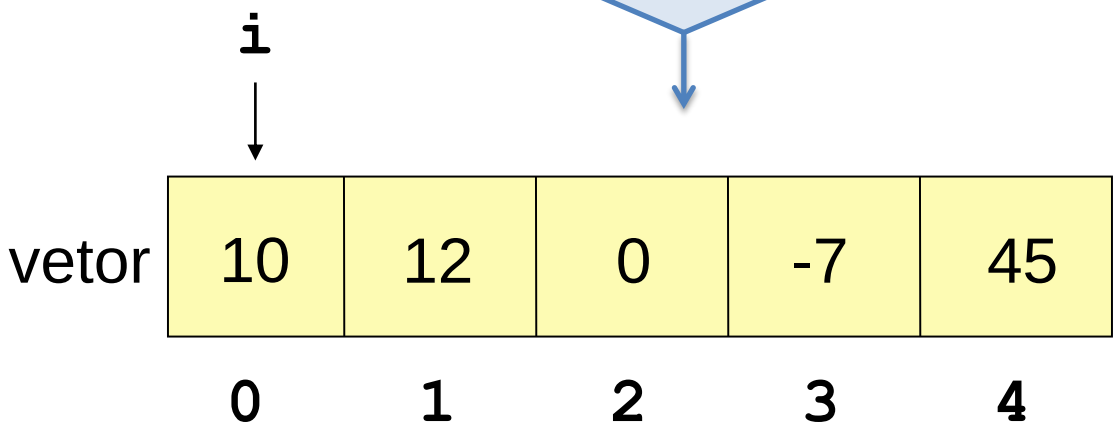
**Iteração do-while 2**

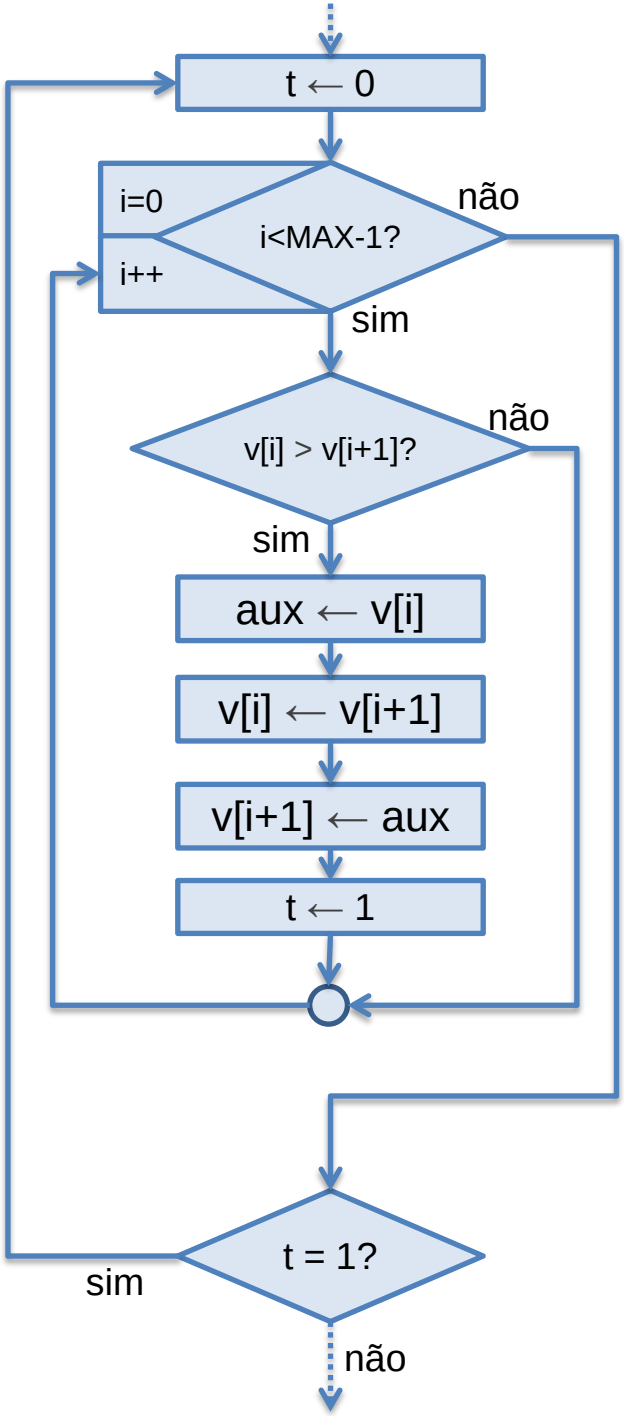
t = 0

**Iteração for 1**

i = 0





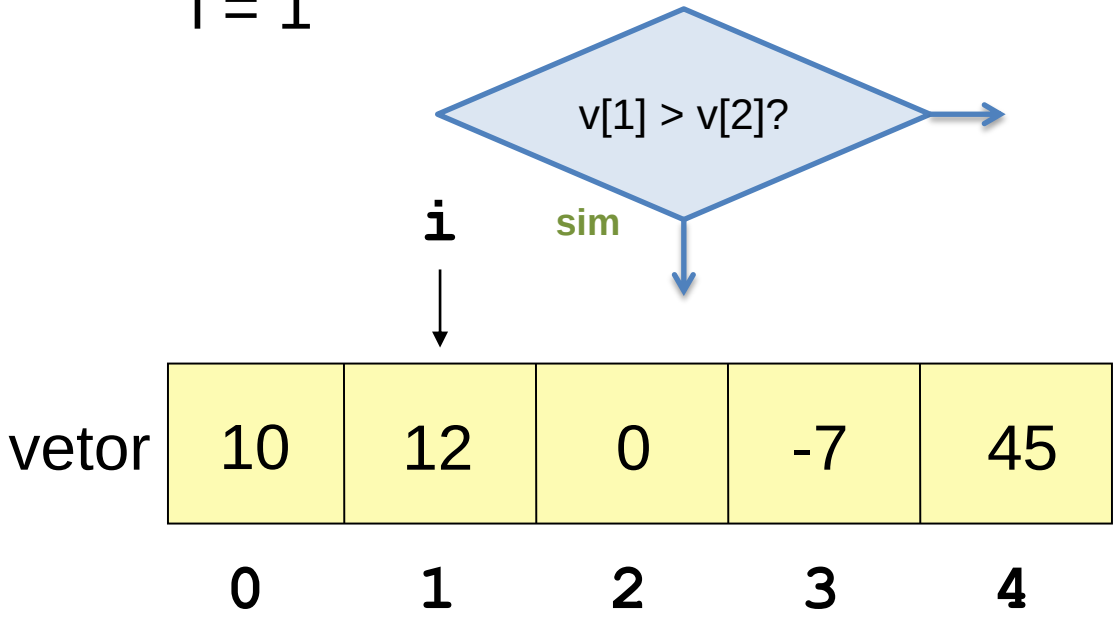

# Bubble Sort (simplificado)

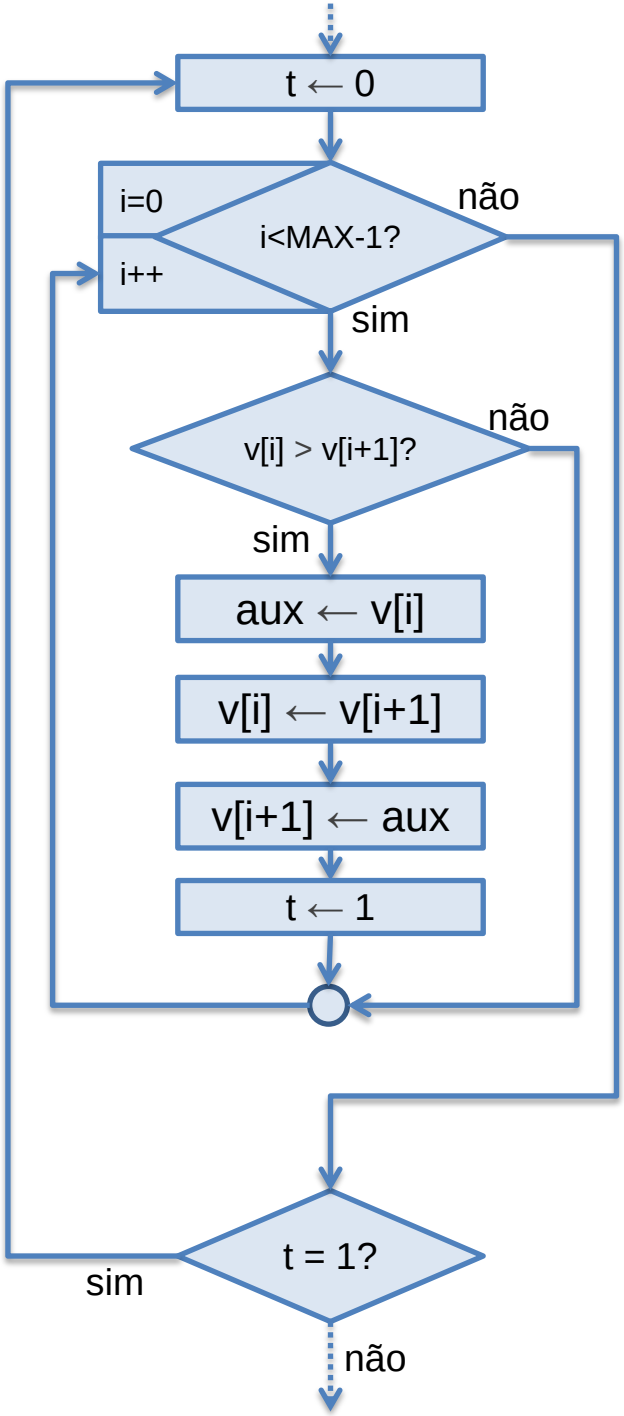
**Iteração do-while 2**

t = 0

**Iteração for 2**

i = 1






# Bubble Sort (simplificado)

**Iteração do-while 2**

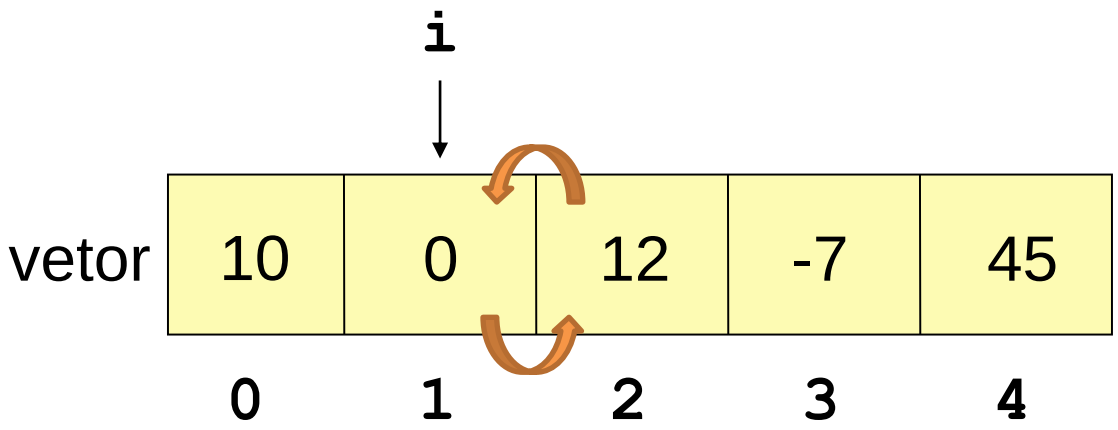
t = 0

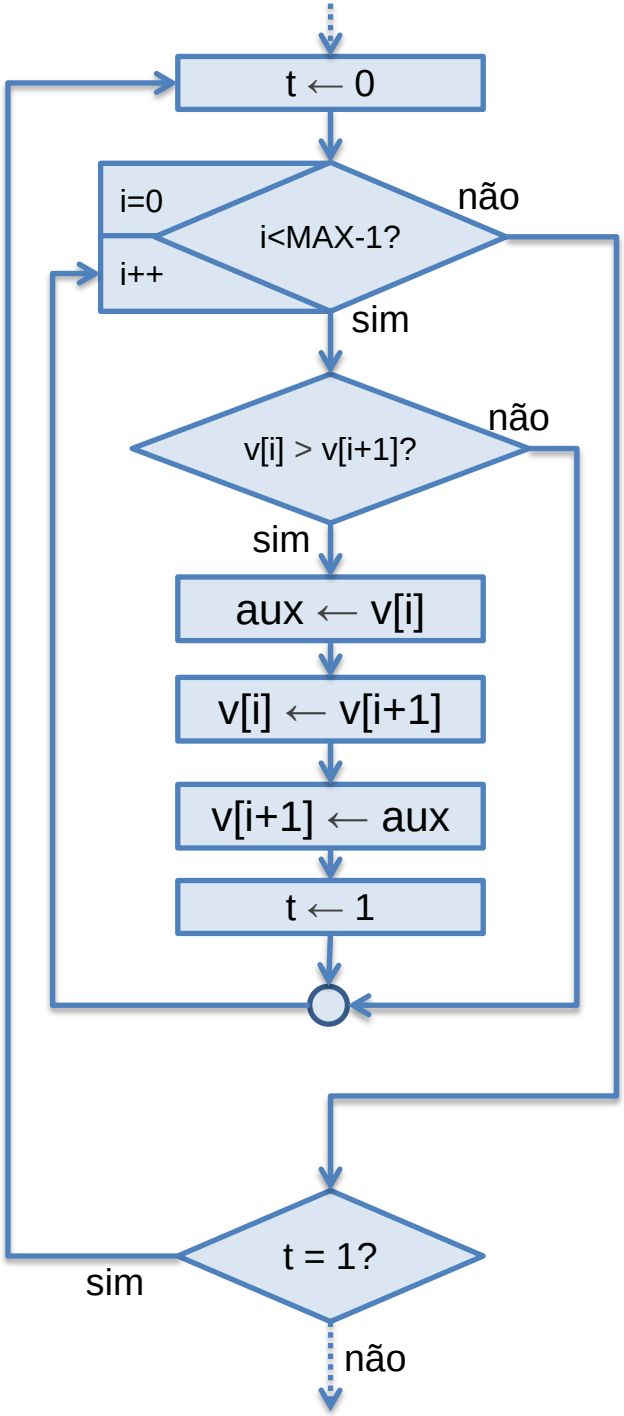
**Iteração for 2**

i = 1

aux = 12

*Realiza a Troca...*






# Bubble Sort (simplificado)

**Iteração do-while 2**

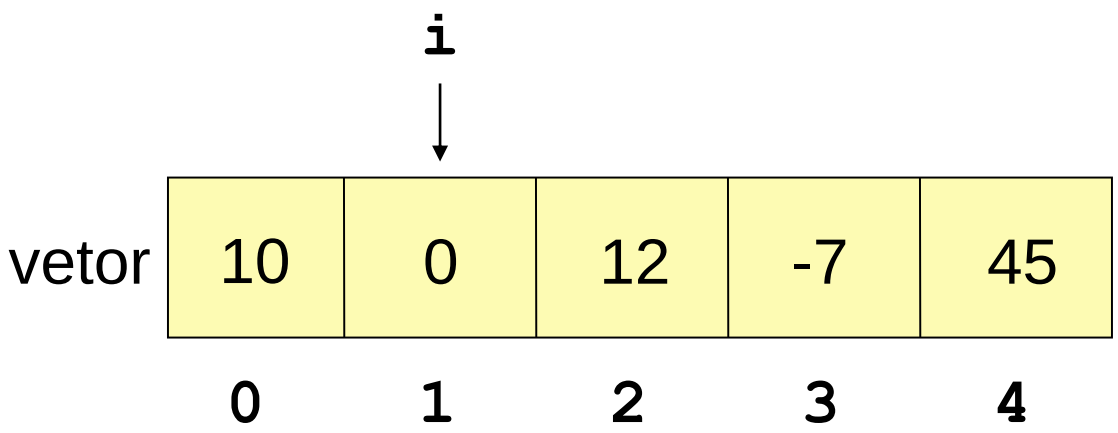
t = 1

Houve troca, então muda o sinalizador

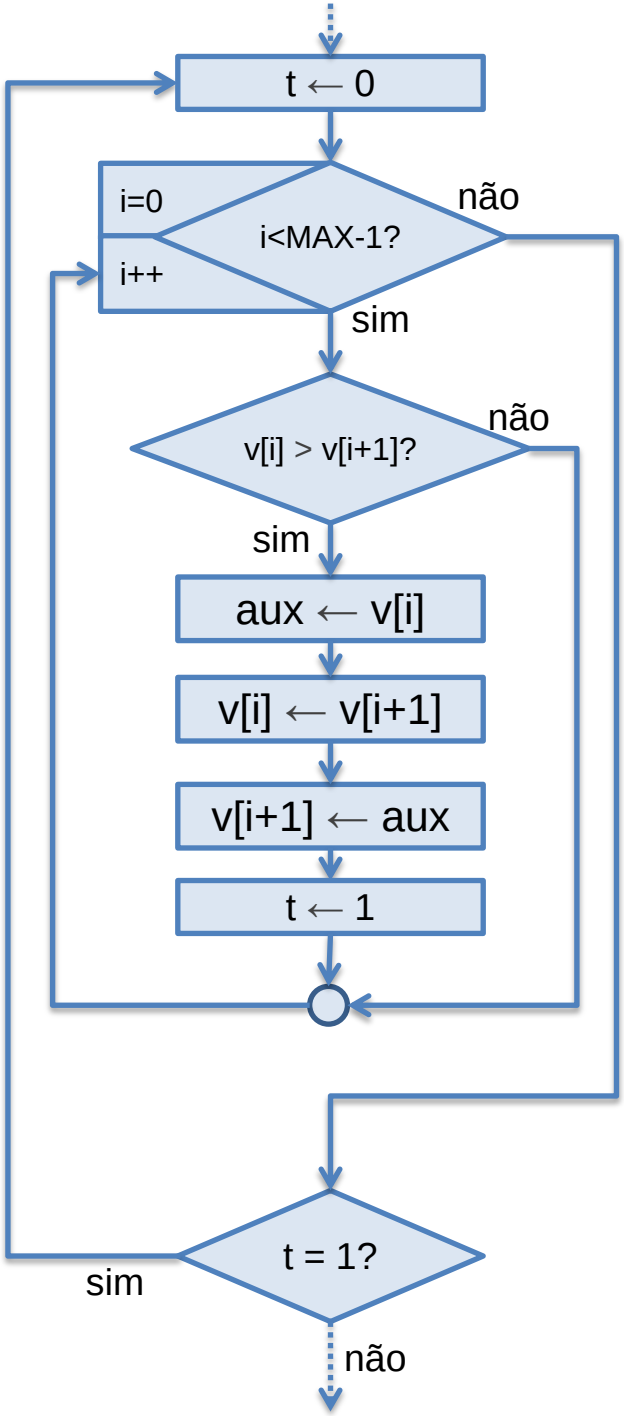
**Iteração for 2**

i = 1

aux = 12



Segue executando o mesmo procedimento durante mais algumas iterações...



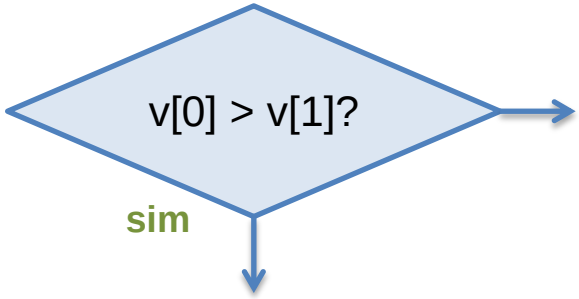

# Bubble Sort (simplificado)

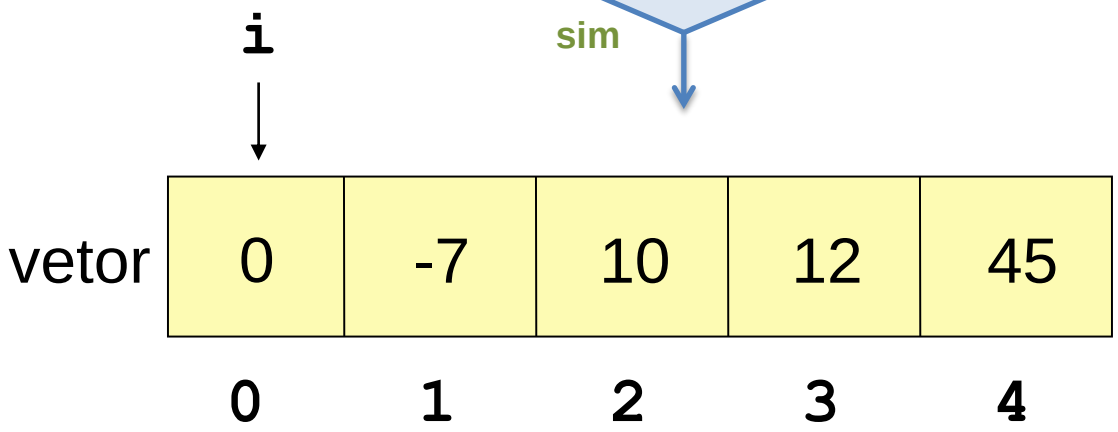
**Iteração do-while 4**

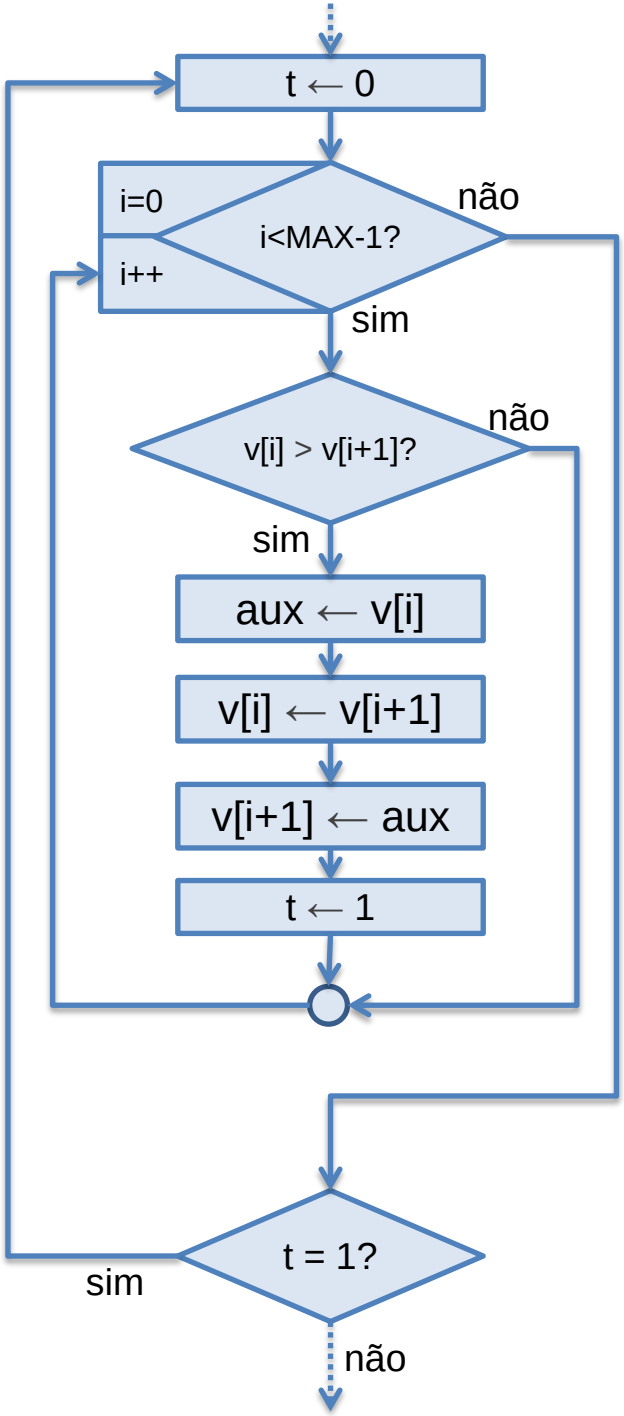
t = 0

**Iteração for 1**

i = 0






# Bubble Sort (simplificado)

**Iteração do-while 4**

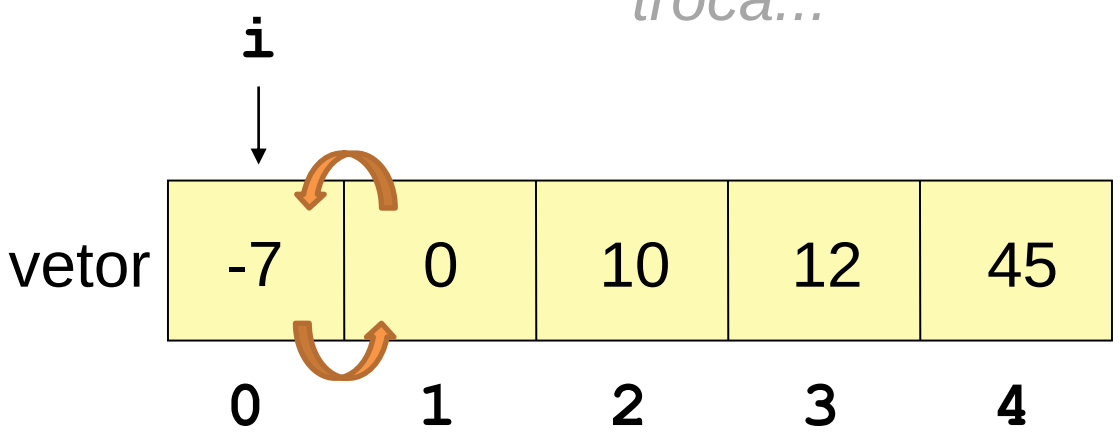
t = 0

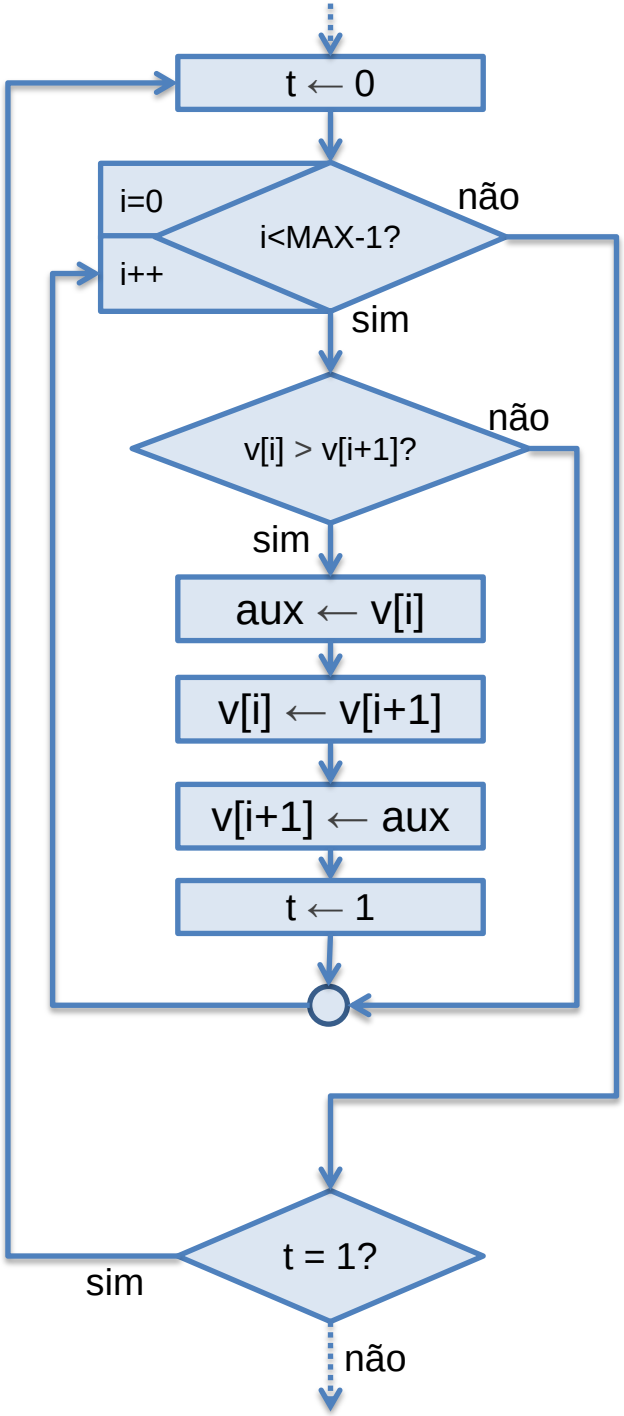
**Iteração for 1**

i = 0

aux = 0

*Realiza a última troca...*






# Bubble Sort (simplificado)

**Iteração do-while 4**

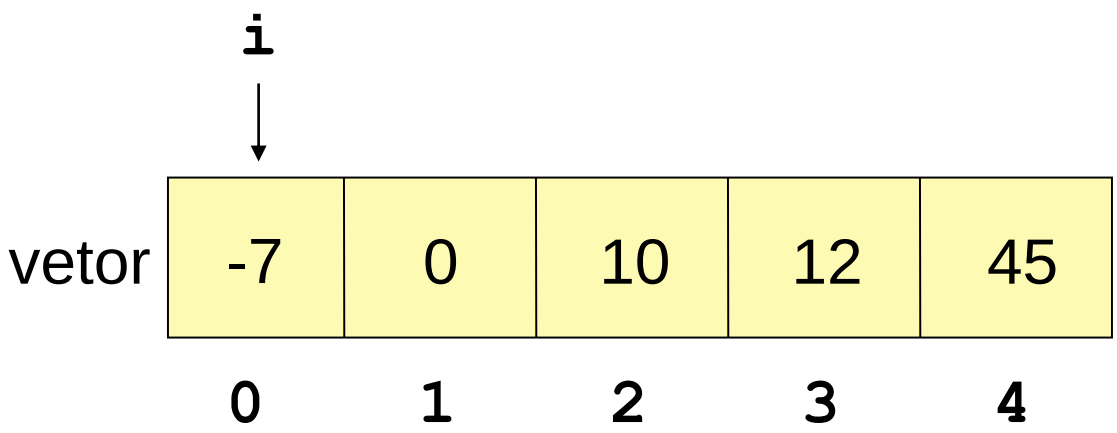
t = 1

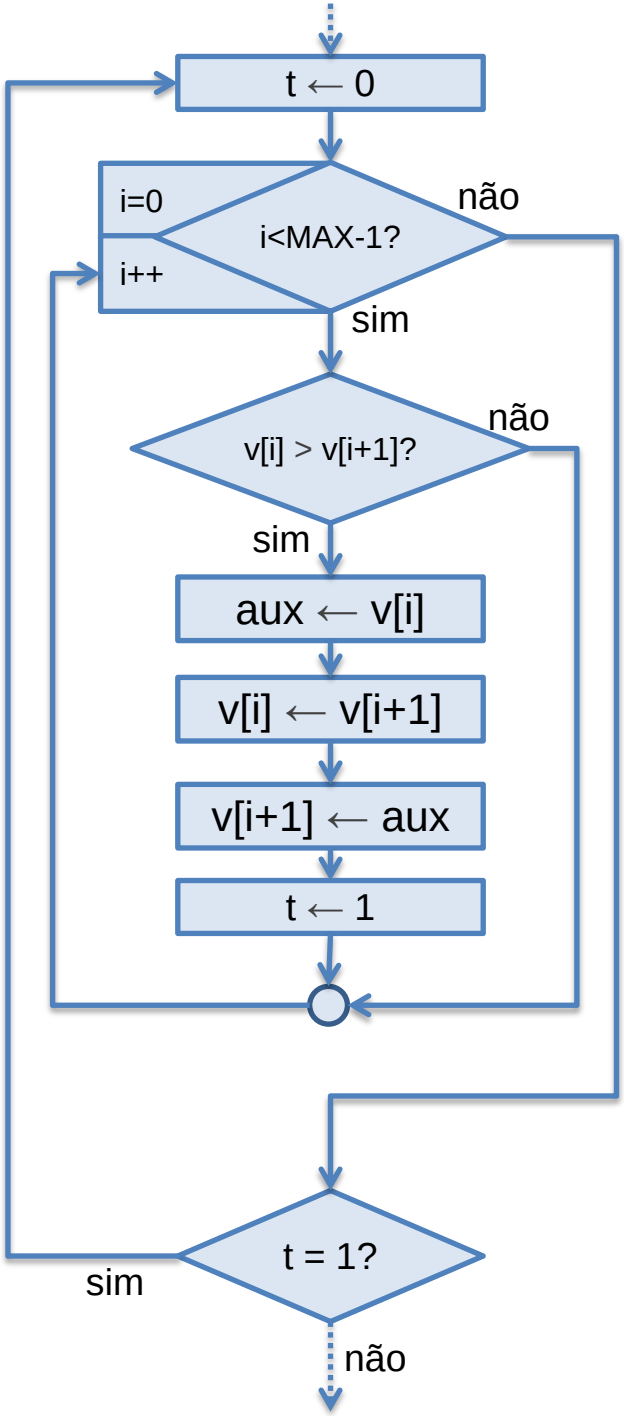
Houve troca, então muda o sinalizador

**Iteração for 1**

i = 0

aux = 0






# Bubble Sort (simplificado)

**Iteração do-while 4**

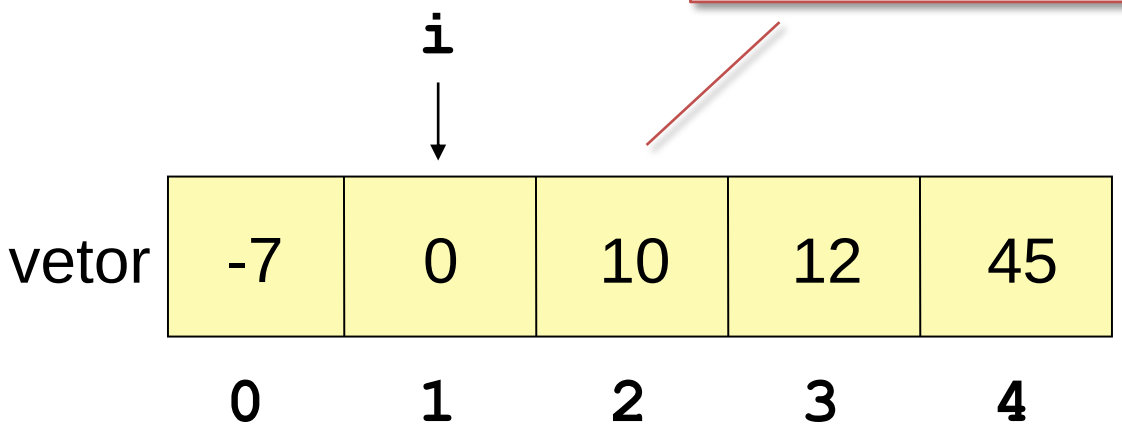
t = 1

**Iteração for 2**

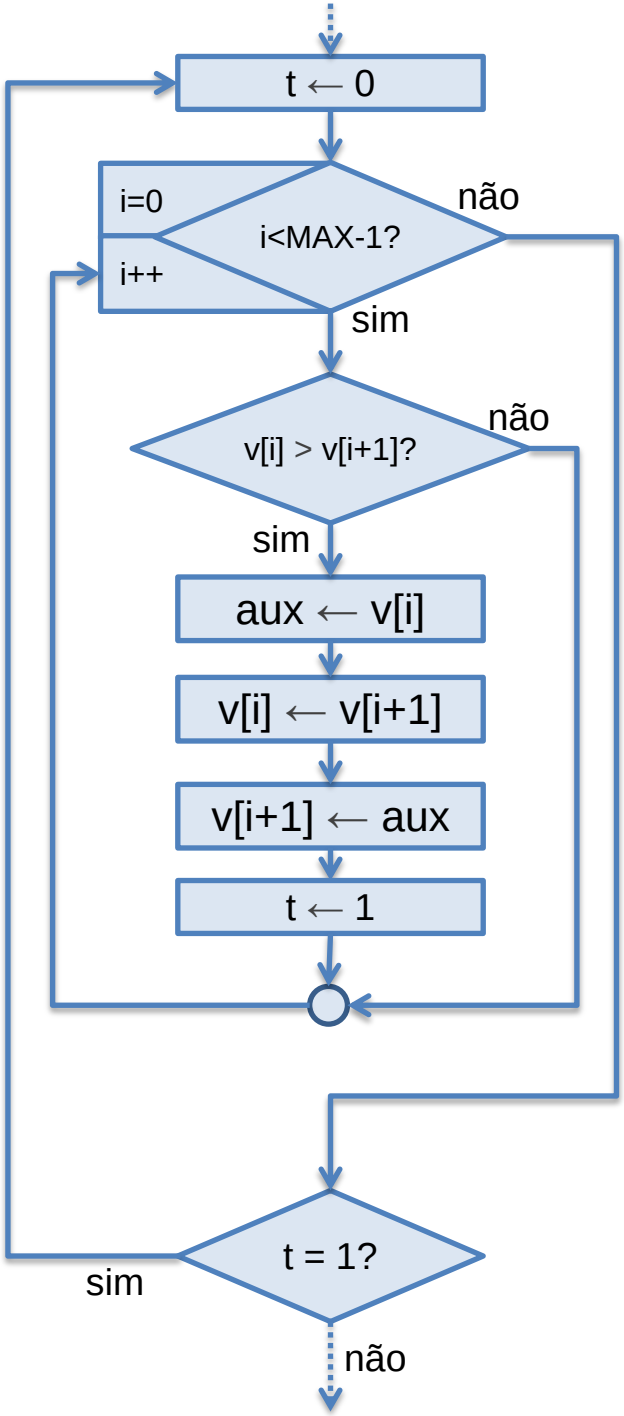
i = 1

aux = 0

Veja que o vetor já está ordenado, porém nessa implementação pouco otimizada todos os valores ainda serão testados até o final



Quando chegar ao final da
**Iteração do-while 4**




# Bubble Sort (simplificado)

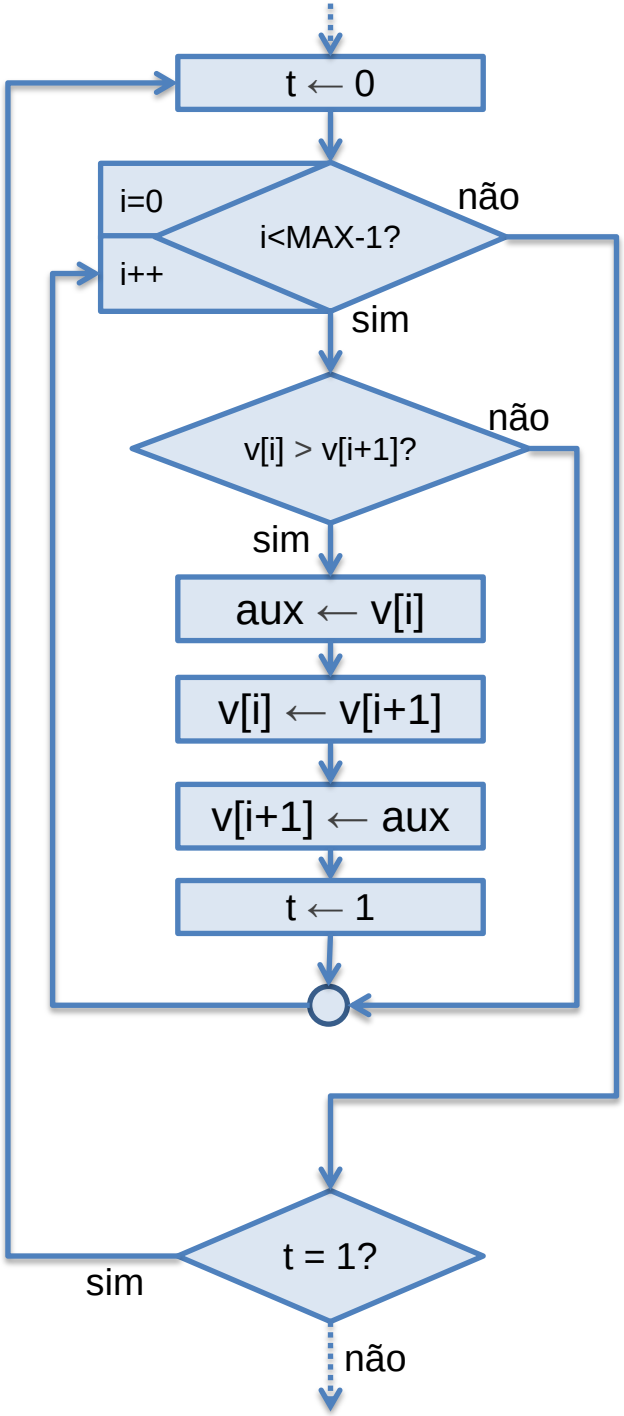
**Iteração do-while 5**

t = 0

Volta o sinalizador para zero indicando que ainda não houve troca nessa passada

| vetor | -7 | 0 | 10 | 12 | 45 |
|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** |

# Bubble Sort (simplificado)

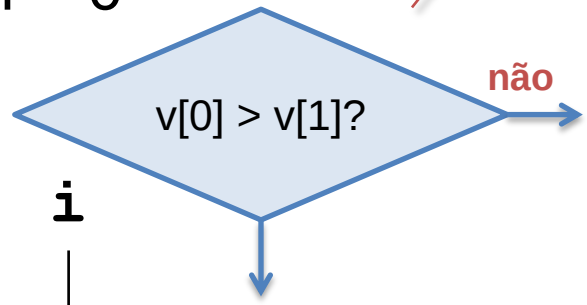



**Iteração do-while 5**

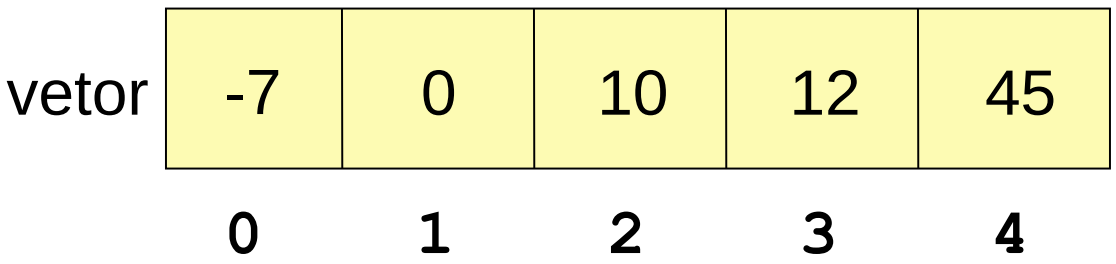
t = 0

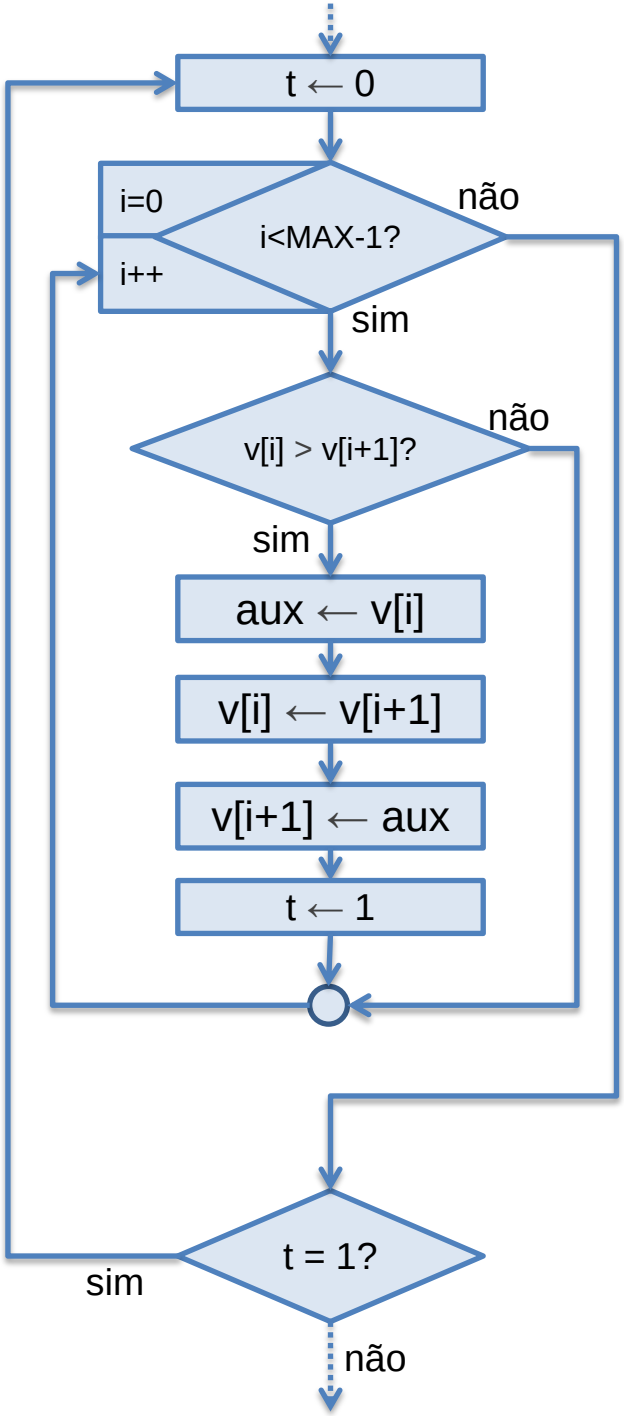
**Iteração for 1**

i = 0




Agora o teste **v[i] > v[i+1]** será sempre **falso**, então **t** continuará com valor **zero** até o fim da **Iteração for**

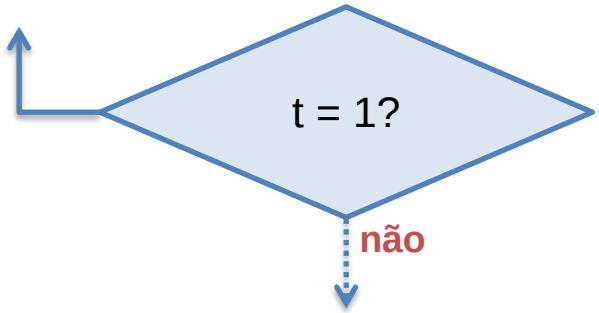





# Bubble Sort (simplificado)

## Iteração do-while 5

t = 0



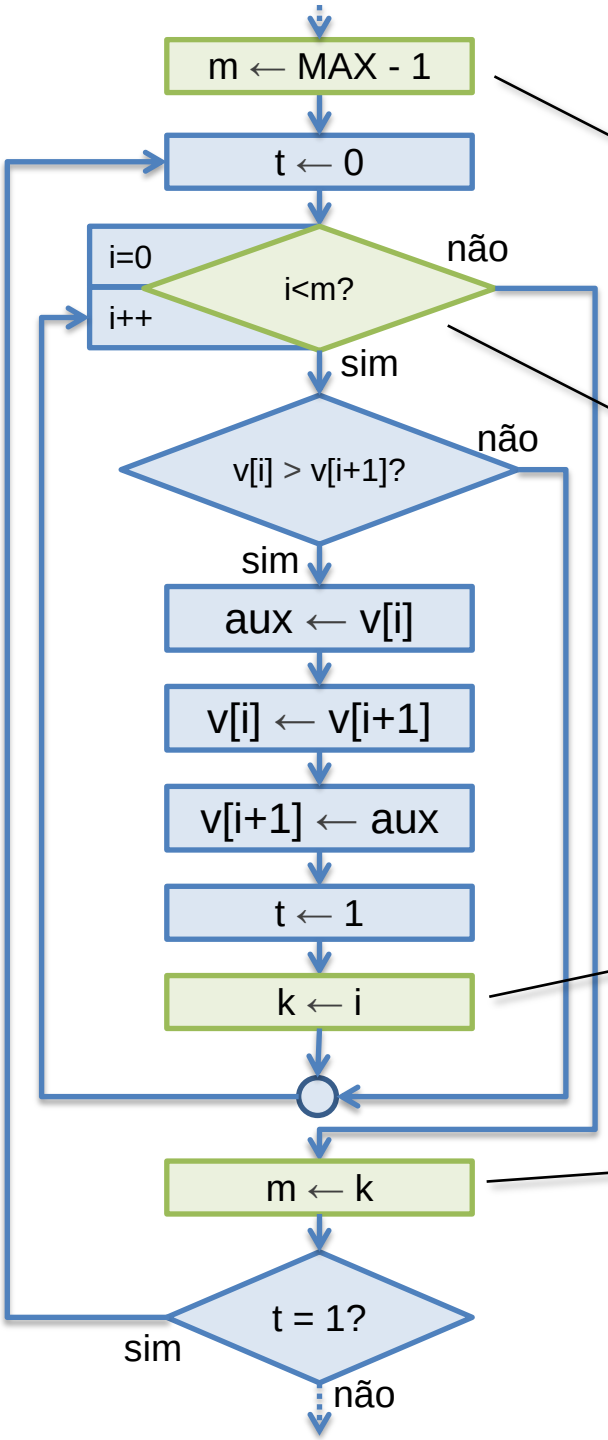

**Problema Resolvido!**
Fora da Iteração do-while o vetor está ordenado

| vetor | -7 | 0 | 10 | 12 | 45 |
|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** |

# Bubble Sort Otimizado

- Como vimos a versão mais simplificada do Bubble Sort apresentada não é muito otimizada
- Otimizar o Bubble Sort é simples
  - Vimos que a cada iteração do-while (laço de fora) os maiores elementos do arranjo são agrupados nas posições finais
  - Basta então **controlar o tamanho da parte desordenada** do vetor (ou saber quanto elementos ainda podem estar fora de posição)
  - Para isso utilizamos duas variáveis auxiliares **m** e **k**

# Bubble Sort (otimizado)

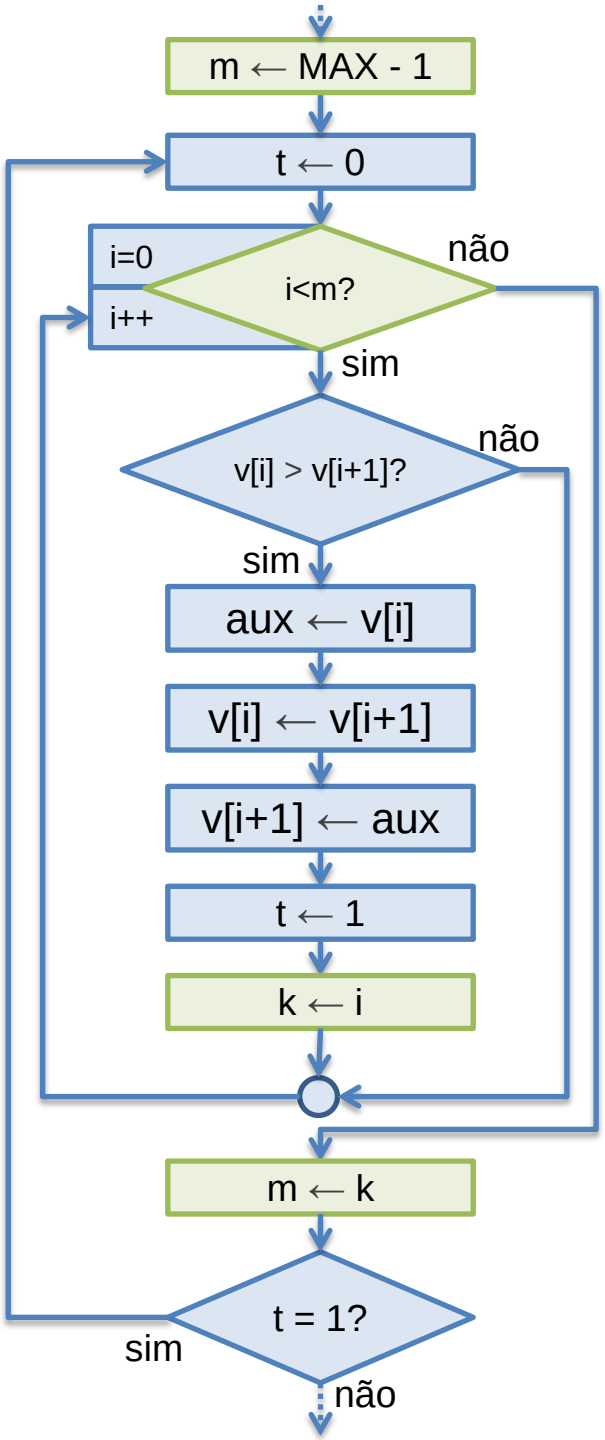



A variável **m** vai controlar o tamanho da parte desordenada, então no começo assumimos que todo o vetor está desordenado

Agora percorremos o vetor somente até o último elemento possivelmente fora de ordem

Houve troca, guardamos a posição onde ocorreu a última troca, esse é um possível elemento fora de ordem

Depois de percorrer o vetor todo trocando elementos, redefinimos o tamanho da parte desordenada




# Bubble Sort (otimizado)

**Iteração do-while 1**

t = 0  m = 5 - 1

Você consegue fazer esse teste de mesa?

Quem sabe escrever esse algoritmo em pseudo-código?

| vetor | 10 | 45 | 12 | 0 | -7 |
|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** |

# Vídeos

- Vídeos dos algoritmos de ordenação para entender melhor
  - LEGO Bubble Sort
    - http://www.youtube.com/watch?v=MtcrEhrt_K0
  - Bubble-sort with Hungarian ("Csángó") folk dance
    - http://www.youtube.com/watch?v=lyZQPjUT5B4
  - 15 Sorting Algorithms in 6 Minutes (Com som) Bubble Sort no minuto 4:00
    - http://www.youtube.com/watch?v=kPRA0W1kECg