

Trilha Algoritmo

Encontro 07 - Selecionando e ordenando.



Recapitulação

1. Encontrando o menor valor.
2. O problema.
3. A solução.
4. Exemplos.
5. Atividades.



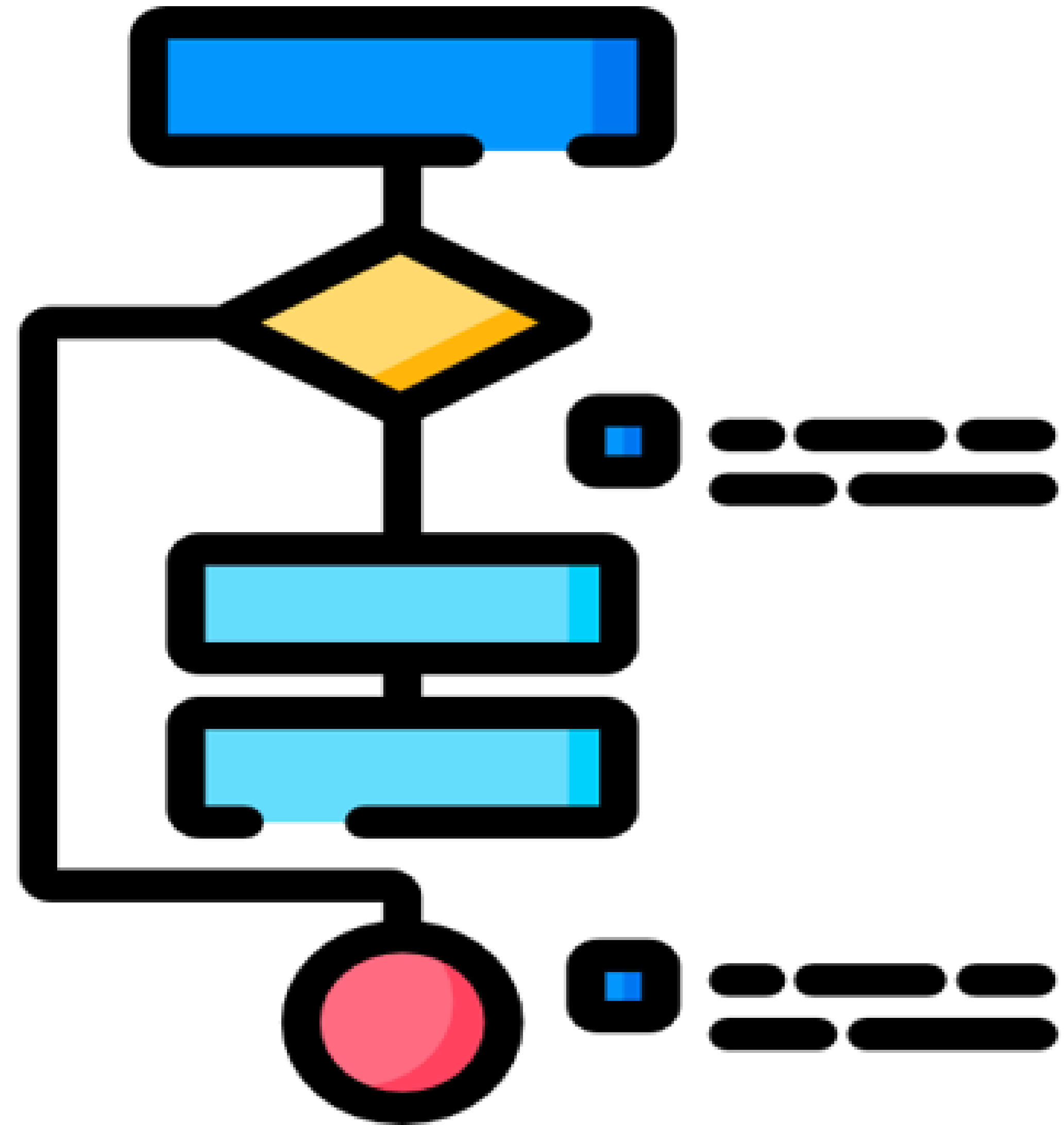
Agenda

1. Como a ordenação pode ajudar?
2. Selecionando e ordenando.
3. Insertion Sort.
4. Selection Sort.
5. Exemplos.
6. Atividades.



Como a
ordenação
pode ajudar?

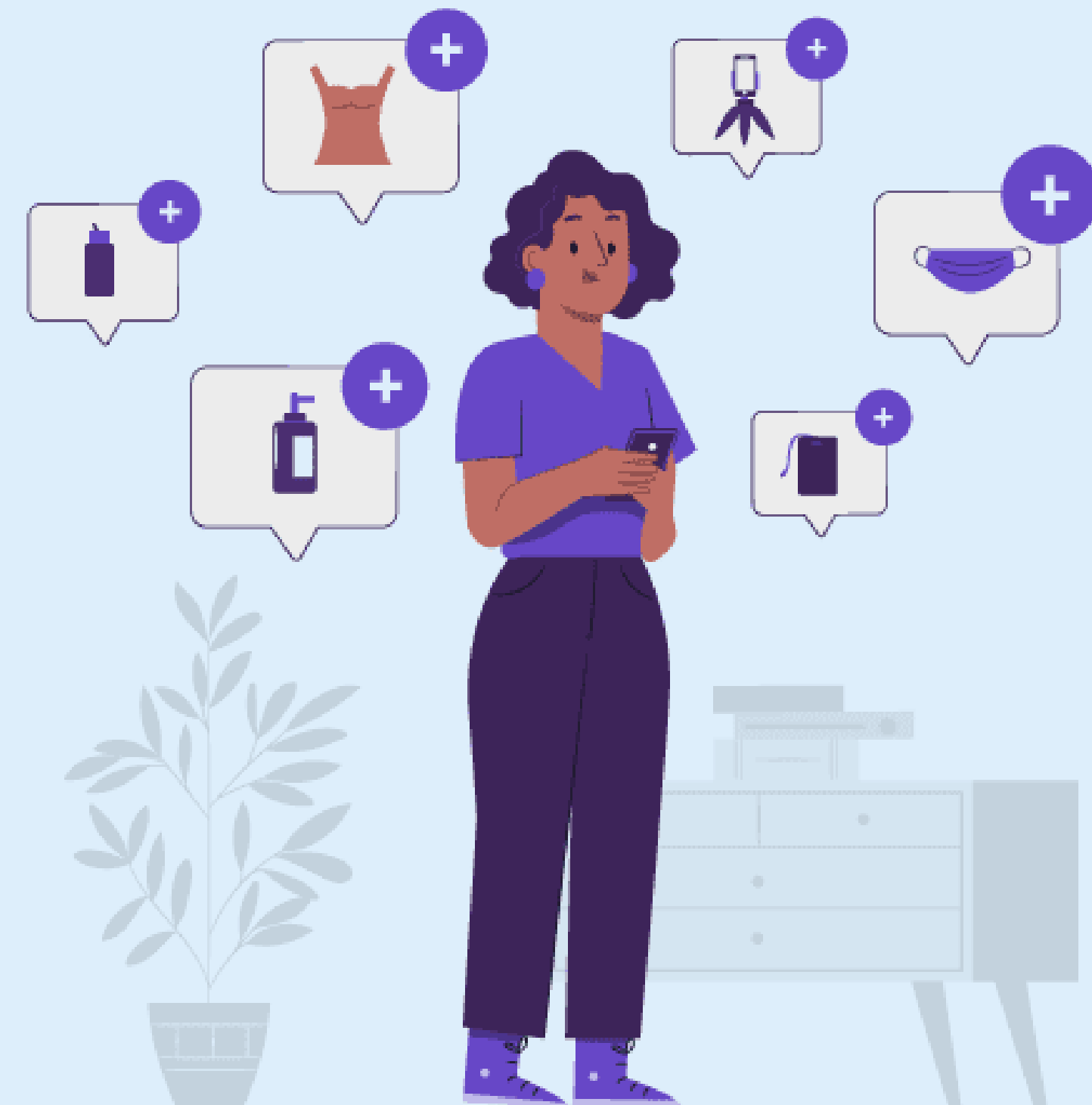
Os N mais baratos



Como a ordenação pode ajudar?

Na prática, será que só estamos interessados em saber o valor mais barato ou o mais caro de **um único** produto?

Ou também estamos interessados em descobrir **o grupo** dos produtos mais baratos, **o grupo** dos mais caros, **o grupo** dos que fizeram mais pontos.



Como a ordenação pode ajudar?

Relembrando
a pesquisa na internet
sobre os valores de
carros.

HONDA CIVIC: R\$ 130.000

MOBI: R\$ 52.000

FUSION: R\$ 200.000

PEUGEOT 208: R\$ 71.000

HB20: R\$ 85.000

Como a ordenação pode ajudar?

Como podemos alterar o processo para marcar quais são os dois carros mais baratos?

Qual é o processo que faremos ao analisar cada um dos elementos para descobrir quais dos produtos são:

maisBarato

segundomaisBarato



Como a ordenação pode ajudar?

Mentalmente concluimos:
MaisBarato: MOBI
SegundomaisBarato: 208



Como a ordenação pode ajudar?

Quando as coisas estão **ordenadas**, o processo de busca se torna **mais rápido**.

Essa é a grande sacada **ordenação**: ela nos permite **resolver** tarefas do cotidiano quase que **imediatamente**.



Como a ordenação pode ajudar?

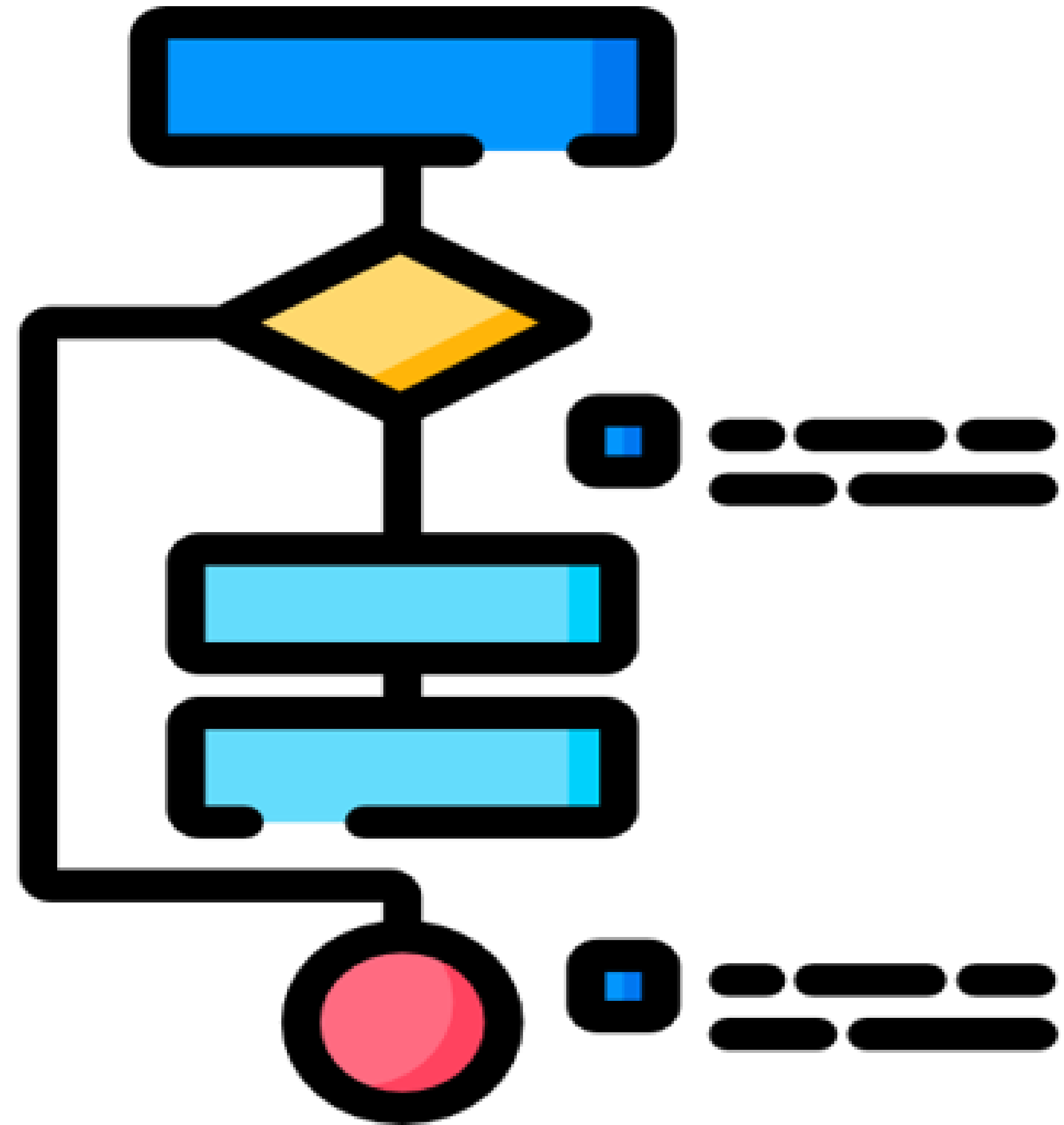
Desafio:

Criar uma ordem para os produtos que estão desorganizados.



Seleccionando e Ordenando

Insertion Sort



Insertion Sort

Como ordenar?

A classificação por **inserção** é um algoritmo de classificação que coloca um elemento **não classificado** em seu local adequado em **cada iteração**.

A classificação por inserção funciona da mesma forma que **classificamos** as cartas em nossa mão em um **jogo de cartas**.

Insertion Sort

Assumimos que o **primeiro cartão** já está **classificado**, então selecionamos um cartão **não classificado**.

Se a **carta não classificada** for **maior** que a carta na mão, ela é colocada à **direita**, caso contrário, à **esquerda**.

Da mesma forma, outras cartas não classificadas são retiradas e colocadas em seus devidos lugares.



Insertion Sort

Imagine **ordenar** a seguinte matriz, relacionado ao preço dos carros.

130	52	200	71	85
-----	----	-----	----	----

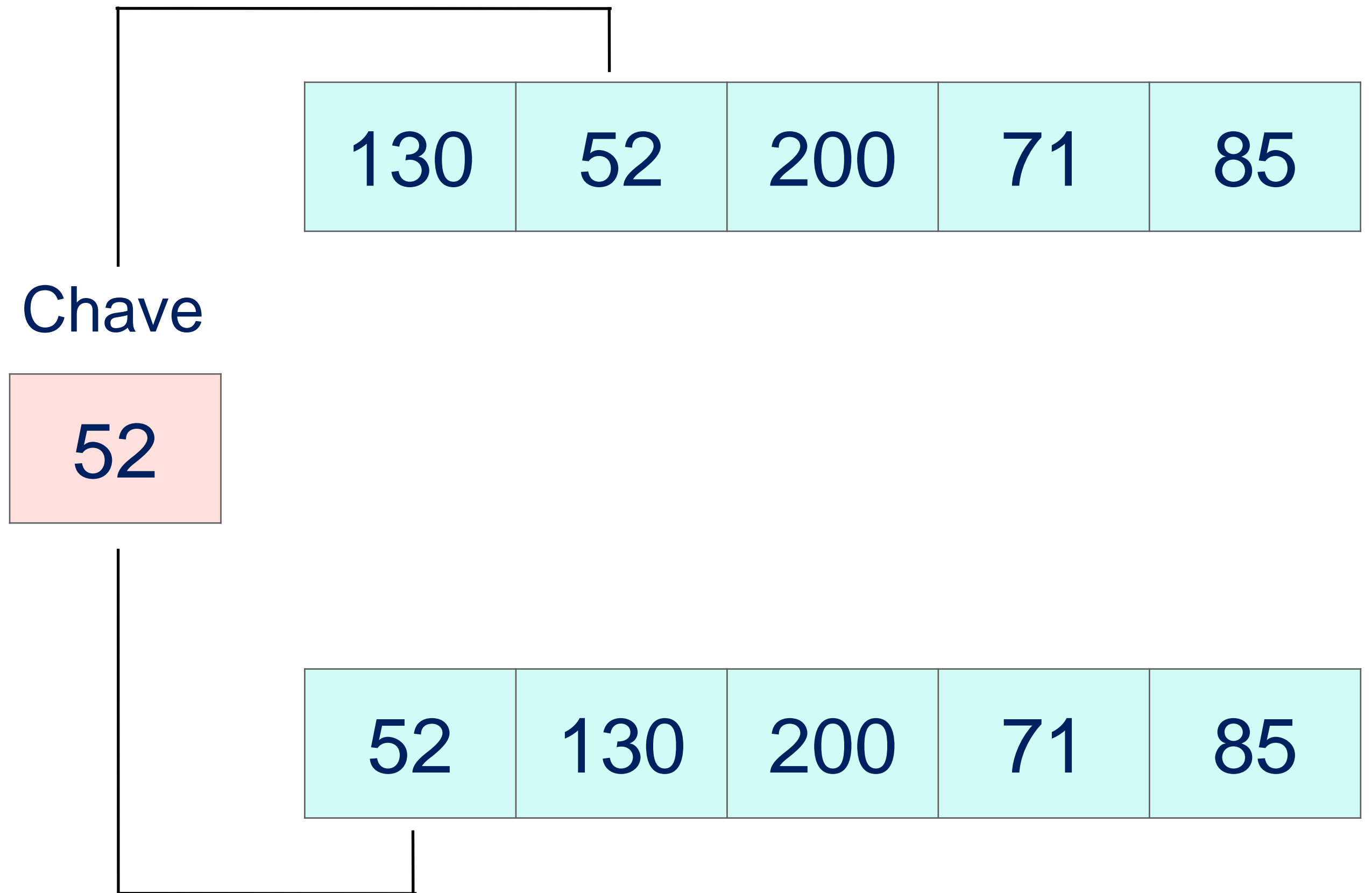
O **primeiro elemento** na matriz é considerado **classificado**.

Para simplificar os preços dos carros foram atribuídos sem os zeros após o ponto.

Insertion Sort

Pegue o **segundo elemento** e guarde-o separadamente na **chave**.

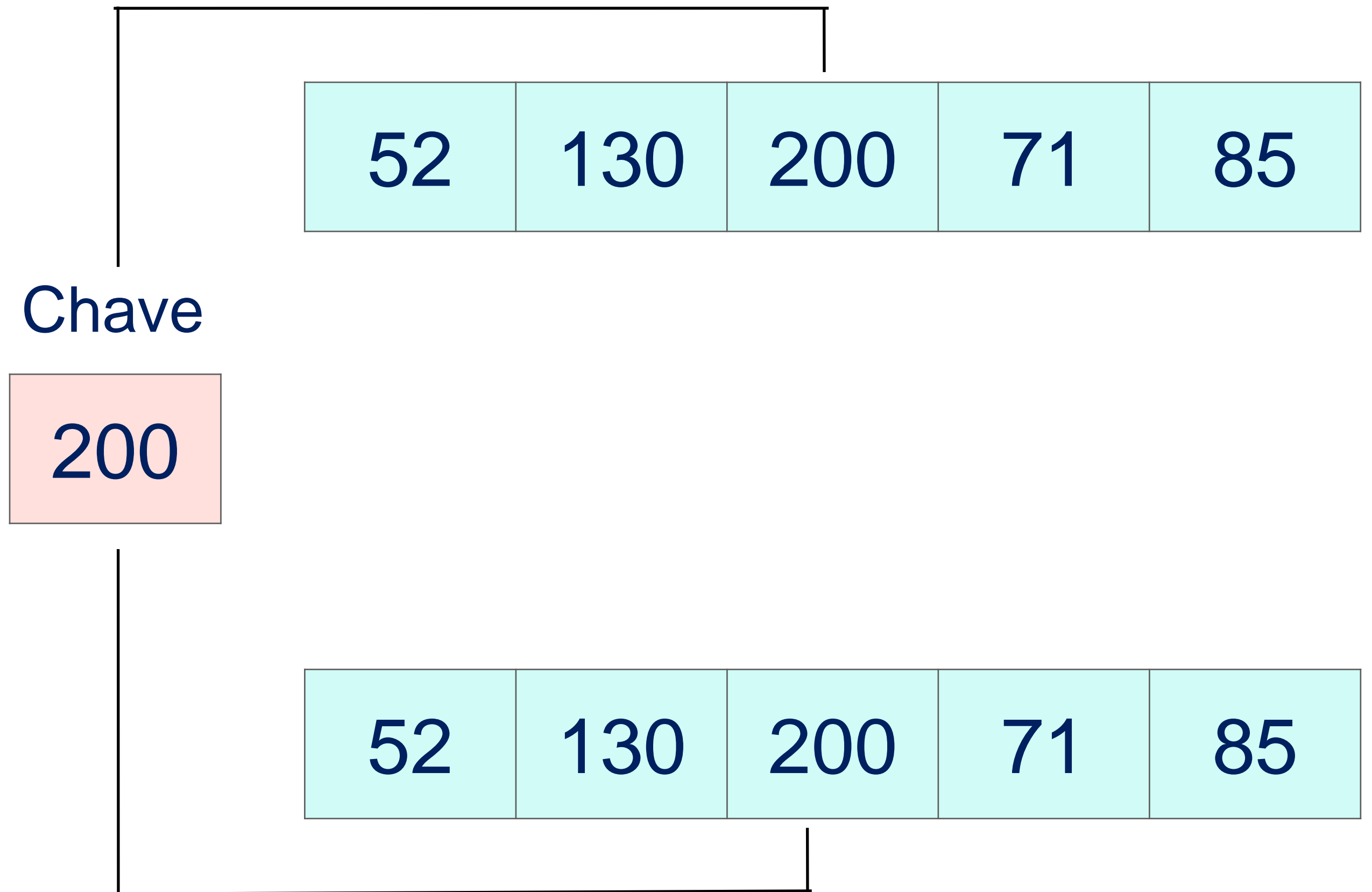
Compare a chave com o primeiro elemento. Se o **primeiro elemento** for **maior** que **chave**, então chave é colocado na frente do primeiro elemento.



Insertion Sort

Agora, os **dois primeiros** elementos **são classificados**.

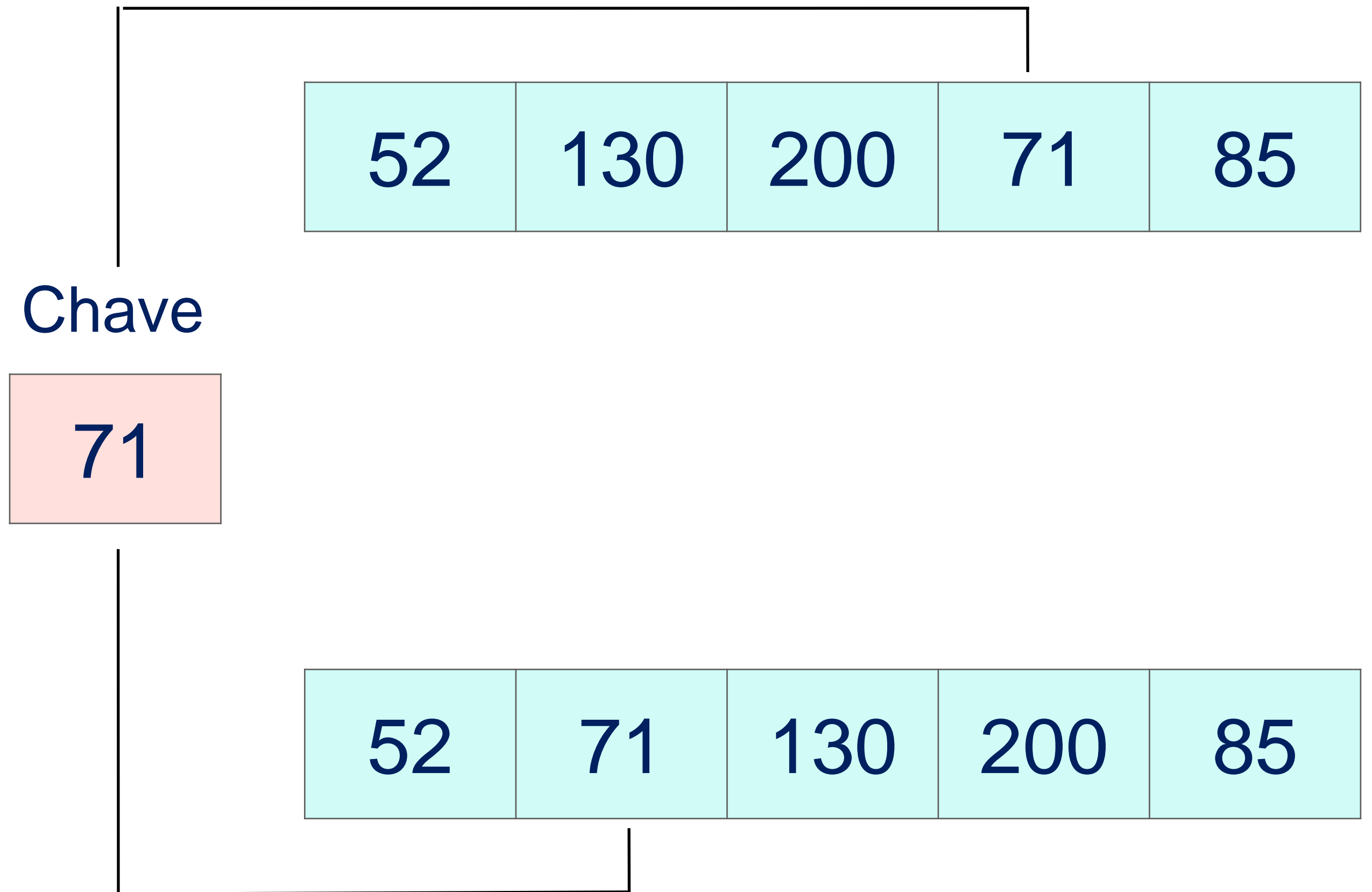
Pegue o **terceiro** elemento e **compare-o** com os elementos à **esquerda** dele. Se não houver nenhum elemento menor que ele, deixe-o no mesmo lugar.



Insertion Sort

Agora, os três **primeiros** elementos **são classificados**.

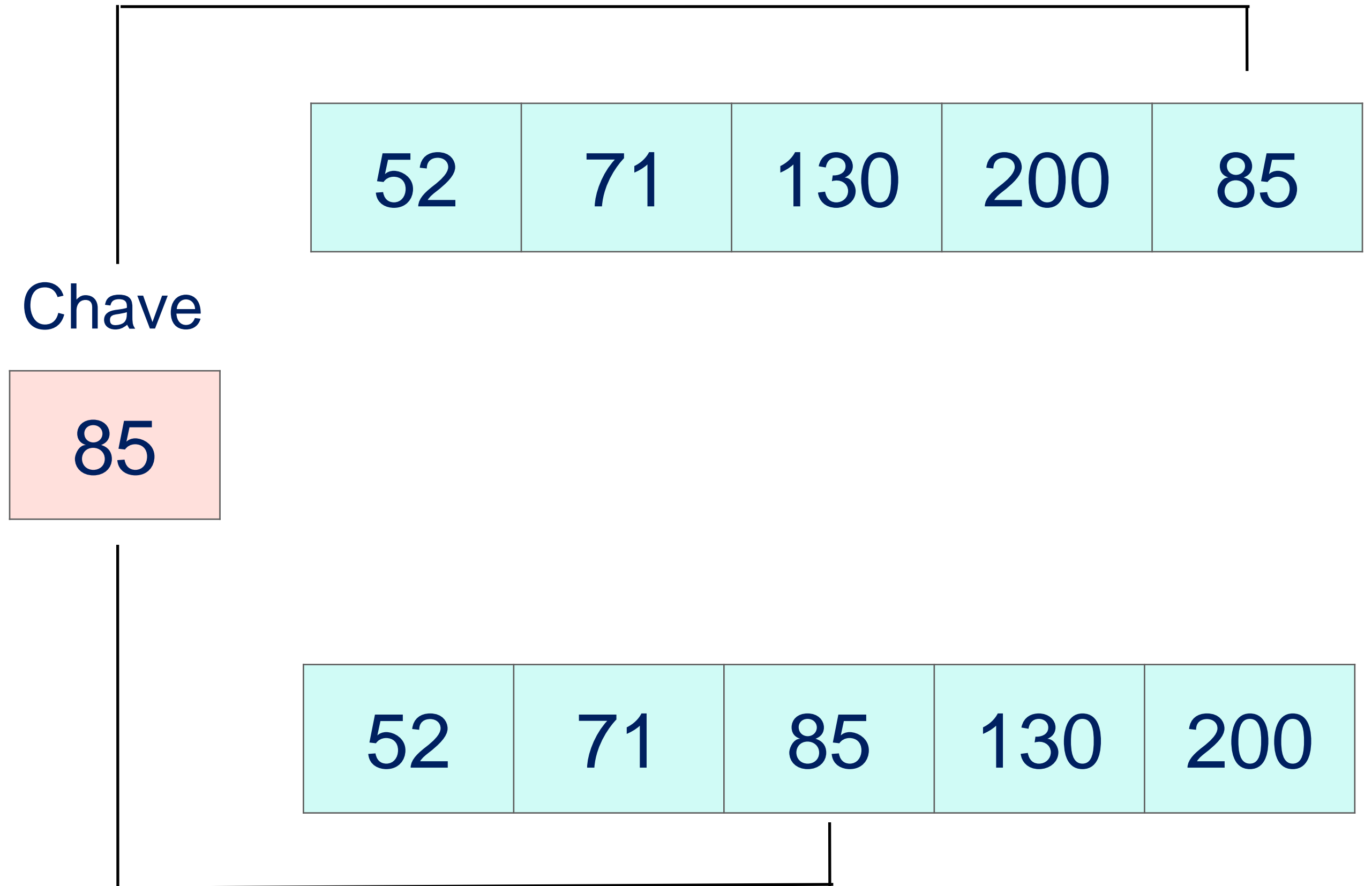
Repita os mesmos passos, para **classificar** os demais elementos.



Insertion Sort

Agora, os quatro **primeiros** elementos **são classificados**.

A **chave** é comparada com todos os elementos e **posicionada corretamente**.



Insertion Sort

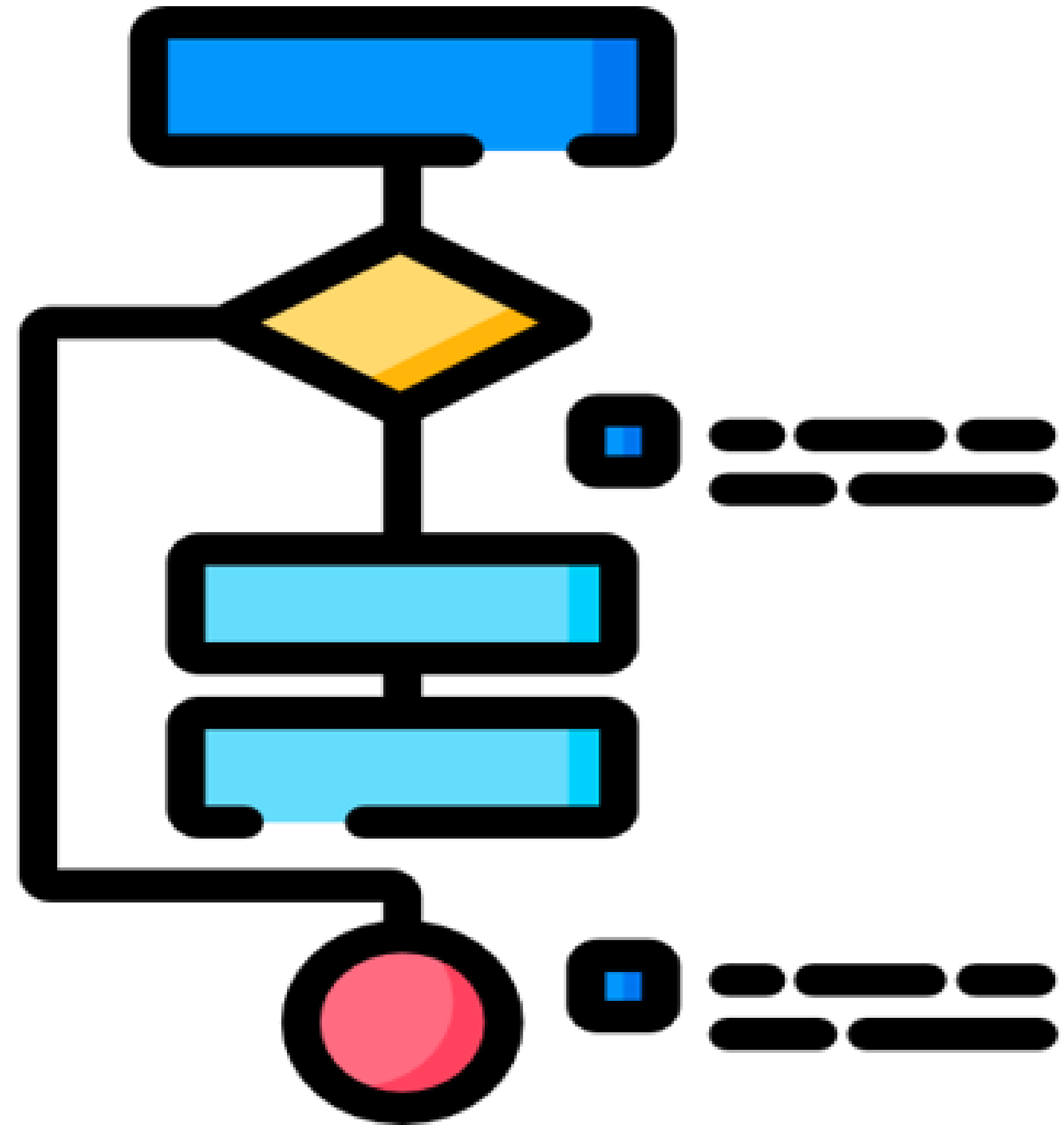
E o código?

Vamos tentar entender o **insertion sort** através do código.



Exemplo

Insertion Sort



Exemplo

Dois laços são considerados:

i: 2 até 5

j: i-1 até 5

```
1 Algoritmo "InsertionSort1"
2 Var
3   v: vetor[1..5] de inteiro
4   i, j, aux: inteiro
5 Inicio
6   para i de 1 ate 5 faca
7     Escreval("Digite o", i, "o número:")
8     Leia(v[i])
9   fimpara
10  para i de 2 ate 5 faca
11    aux<-v[i]
12    j<-i-1
13    enquanto (aux < v[j]) faca
14      v[j+1]<-v[j]
15      j<-j-1
16
17
18
19
```

?



Vamos
praticar?

Insertion Sort

Construa um algoritmo que leia uma sequência de **20 números aleatórios** entre 0 e 200.000. Faça sua **ordenação** e apresente o resultado na tela.



Insertion Sort

```
1 Algoritmo "InsertionSort2"  
2 Var  
3   v: vetor[1..20] de inteiro  
4   i, j, aux: inteiro  
5 Inicio  
6   para i de 1 ate 20 faca  
7     Escreval("Digite o", i, "o número:")  
8     aleatorio 0, 200000  
9     Leia(v[i])  
10    aleatorio off  
11    fimpara  
12    para i de 2 ate 20 faca  
13      aux <- v[i]  
14      j <- i-1  
15      enquanto aux<v[j] faca  
16        v[j+1] <- v[j]  
17        j <- j-1  
18        se j=0 entao  
19          interrompa  
20        fimse
```

?

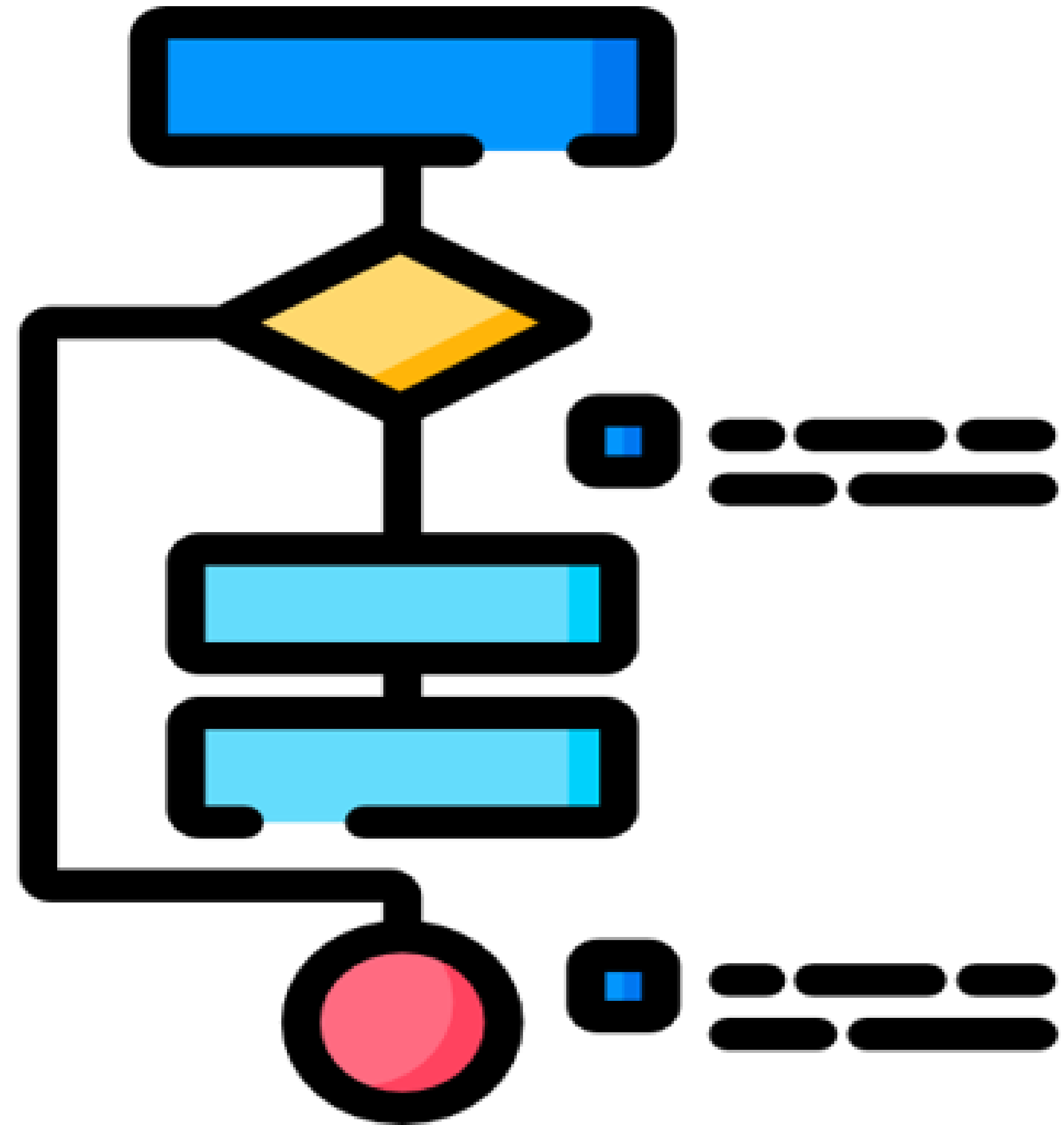


Coffee
time!



Selecionando e Ordenando

Selection Sort



Selecionando e Ordenando

Como ordenar?

Basta **varrer o array** procurando o **produto mais barato** a partir de uma determinada posição.

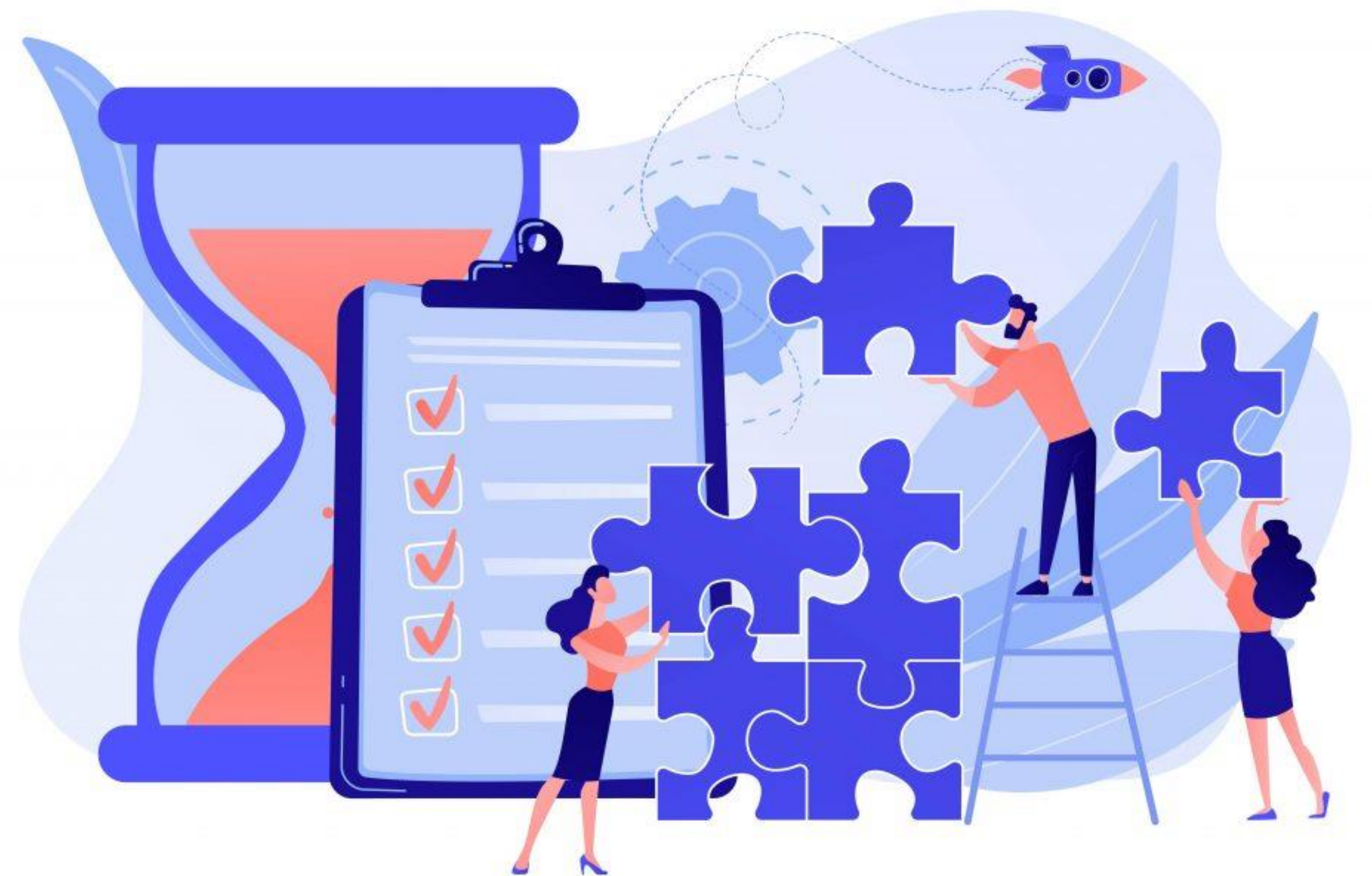
Ao encontrar o produto **mais barato**, o elemento é **trocado** de lugar na lista.

Este procedimento é realizado até que exista uma **ordem final**.

Selecionando e Ordenando

Este **algoritmo** utilizado é chamado de ordenação por seleção (**Selection Sort**).

Ele seleciona o **menor** a partir do instante atual e permite que o **reposicione** na lista.



Selection Sort

1. Defina o **primeiro** elemento como **mínimo**.
2. **Compare** o mínimo com o **segundo elemento**. Se o segundo elemento for menor que o mínimo, atribua o segundo elemento como mínimo.
3. **Compare** o mínimo com o **terceiro elemento**. Novamente, se o terceiro elemento for menor, atribua o mínimo ao terceiro elemento, caso contrário, não faça nada. O processo continua até o último elemento.

Selection Sort

Após cada **iteração**, o **mínimo** é colocado na **frente** da lista não ordenada.

Para cada iteração, a **indexação** começa a partir do primeiro **elemento não classificado**.

Os passos **1 a 3** são **repetidos** até que todos os elementos sejam colocados em suas **posições corretas**.

Selection Sort

O **primeiro** elemento é definido como **mínimo**.

Em seguida são feitas as **comparações**.

130	52	200	71	85
-----	----	-----	----	----

Para simplificar os números foram atribuídos sem os zeros após o ponto.

Selection Sort

Após cada **iteração**,
o **mínimo** é colocado
na **frente** da lista não
ordenada.

52	130	200	71	85
----	-----	-----	----	----

130	52	200	71	85
-----	----	-----	----	----

130	52	200	71	85
-----	----	-----	----	----

130	52	200	71	85
-----	----	-----	----	----

130	52	200	71	85
-----	----	-----	----	----

Selection Sort

Após cada **iteração**, o **mínimo** é colocado na **frente** da lista não ordenada.

52	71	200	130	85
----	----	-----	-----	----

52	130	200	71	85
----	-----	-----	----	----



52	130	200	71	85
----	-----	-----	----	----



52	130	200	71	85
----	-----	-----	----	----



Selection Sort

Após cada **iteração**,
o **mínimo** é colocado
na **frente** da lista não
ordenada.

52	71	85	130	200
----	----	----	-----	-----

52	71	200	130	85
----	----	-----	-----	----



52	71	200	130	85
----	----	-----	-----	----



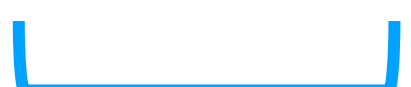
Selection Sort

Após cada **iteração**, o **mínimo** é colocado na **frente** da lista não ordenada.

52	71	85	130	200
----	----	----	-----	-----

Na última **iteração** o mínimo já está na posição **correta**.

52	71	200	130	200
----	----	-----	-----	-----



Selection Sort

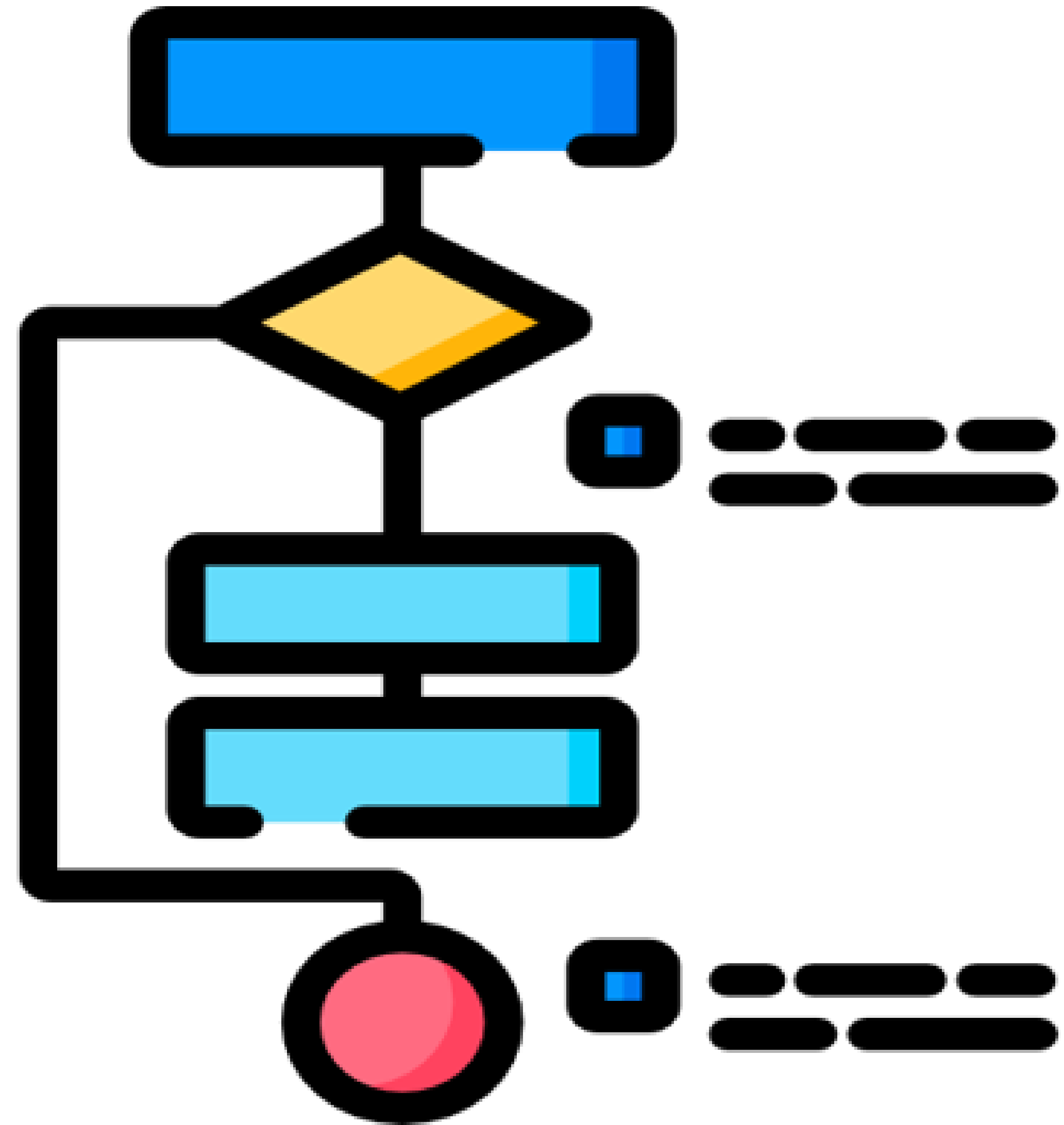
E o código?

Vamos tentar entender o **selection sort** através do código.



Exemplo

Selection Sort



Exemplo

Dois laços são considerados:

i: 1 até 4

j: i+1 até 5

```
7   para i de 1 ate 5 faca
8       Escreva("Digite o", i, "o :")
9       Leia(v[i])
10  fimpara
11  para i de 1 ate 4 faca
12      menor <- i
13      para j de i+1 ate 5 faca
14          se v[j]<v[menor] entao
15              menor <- j
16          fimse
17      fimpara
18      se menor <> i entao
19          aux <- v[i]
20          v[i] <- v[menor]
21          v[menor] <- aux
22      fimse
23
24
25
26
```

?



Vamos
praticar?

Selection Sort

Construa um algoritmo que leia uma sequência de 7 números **aleatórios** entre -200 e 200. Faça sua **ordenação** e apresente o resultado na tela.

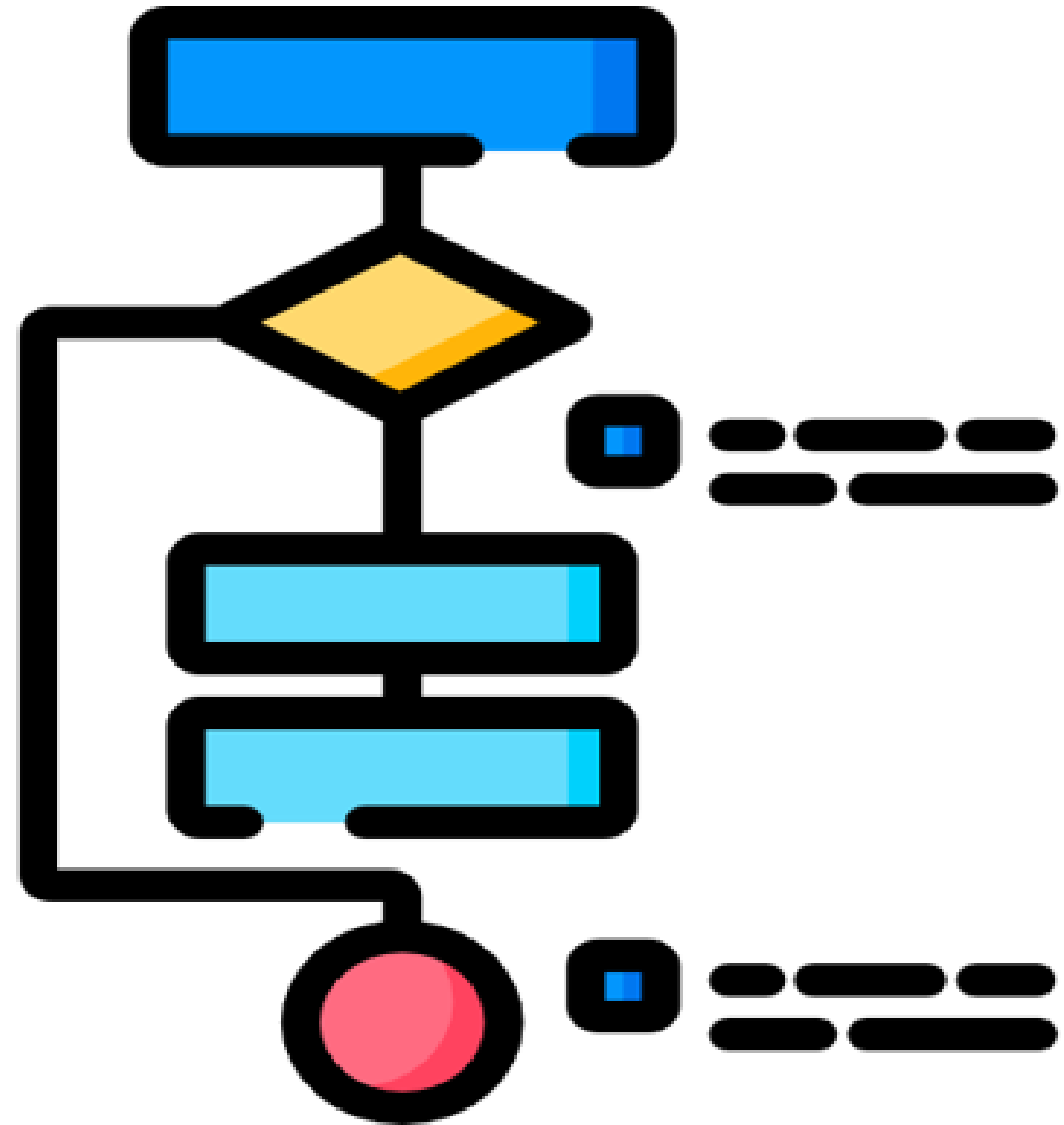


Selection Sort

```
1 Algoritmo "SelectionSort2"
2 Var
3   ordenacao: vetor[1..7] de inteiro
4   c, esq, dir, aux: inteiro
5
6 Inicio
7   para c de 1 ate 7 faca
8     Escreva("Digite o", c, "o :")
9     aleatorio -200, 200
10    Leia(ordenacao[c])
11    aleatorio off
12  fimpara
13  para esq de 1 ate 7 faca
14    para dir de esq+1 ate 7 faca
15      se ordenacao[esq] > ordenacao[dir] entao
16        aux <- ordenacao[esq]
17        ordenacao[esq] <- ordenacao[dir]
18        ordenacao[dir] <- aux
19      fimse
20    fimpara
21  fimpara
```

?

Exercícios



Prática

Exercício 1:

Crie um algoritmo para ordenar um vetor de tamanho 10. Leia todos os valores e depois coloque em ordem crescente.

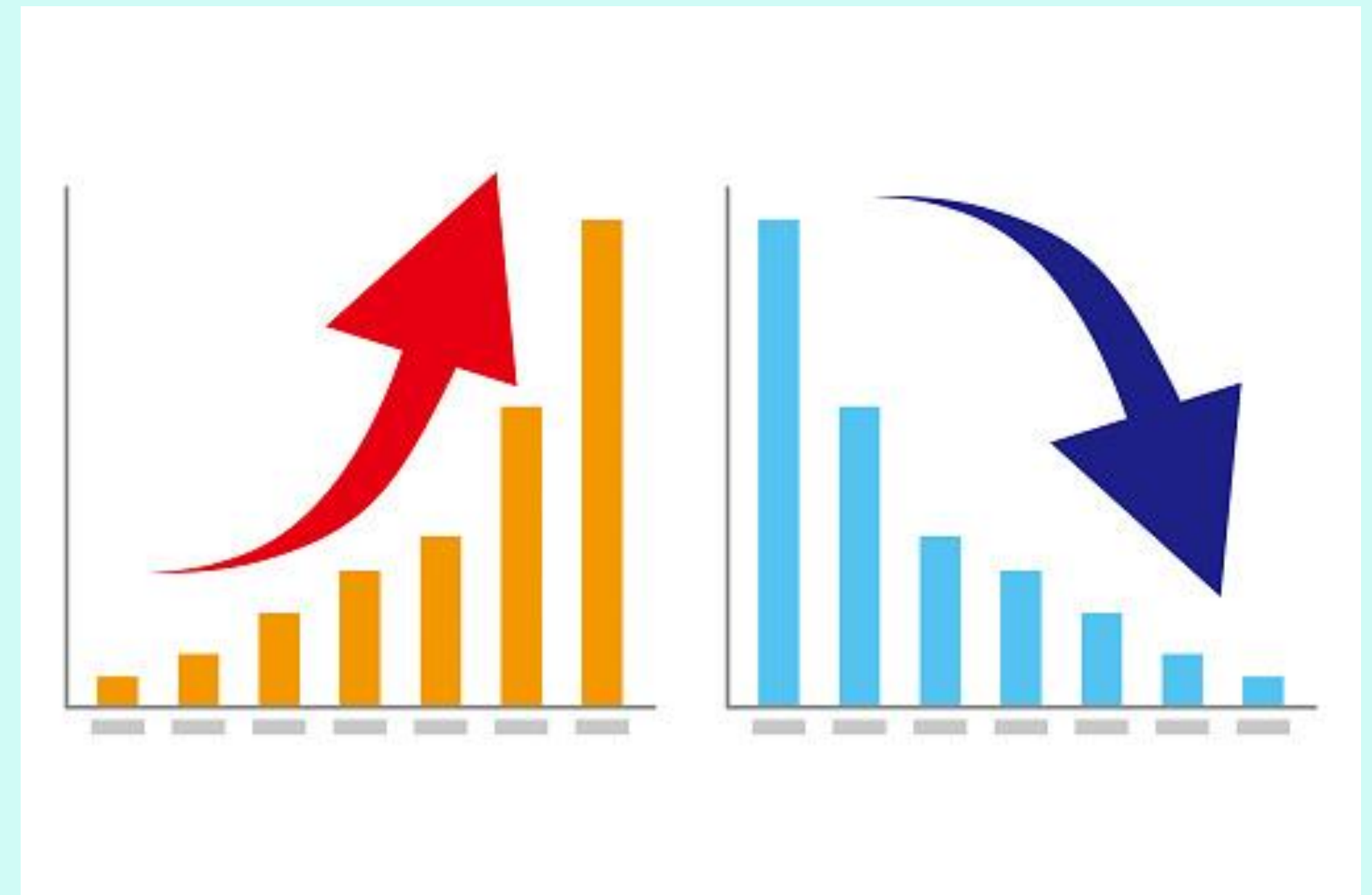


Prática

Exercício 2:

Escreva um algoritmo que leia um vetor com 8 valores numéricos positivos e negativos. Ordene esses números de forma crescente e de forma decrescente. Imprima o seguinte resultado:

- a) Vetor digitado
- b) Vetor em ordem Crescente
- c) Vetor em ordem decrescente



Dica de hoje

Algoritmos "SORT"

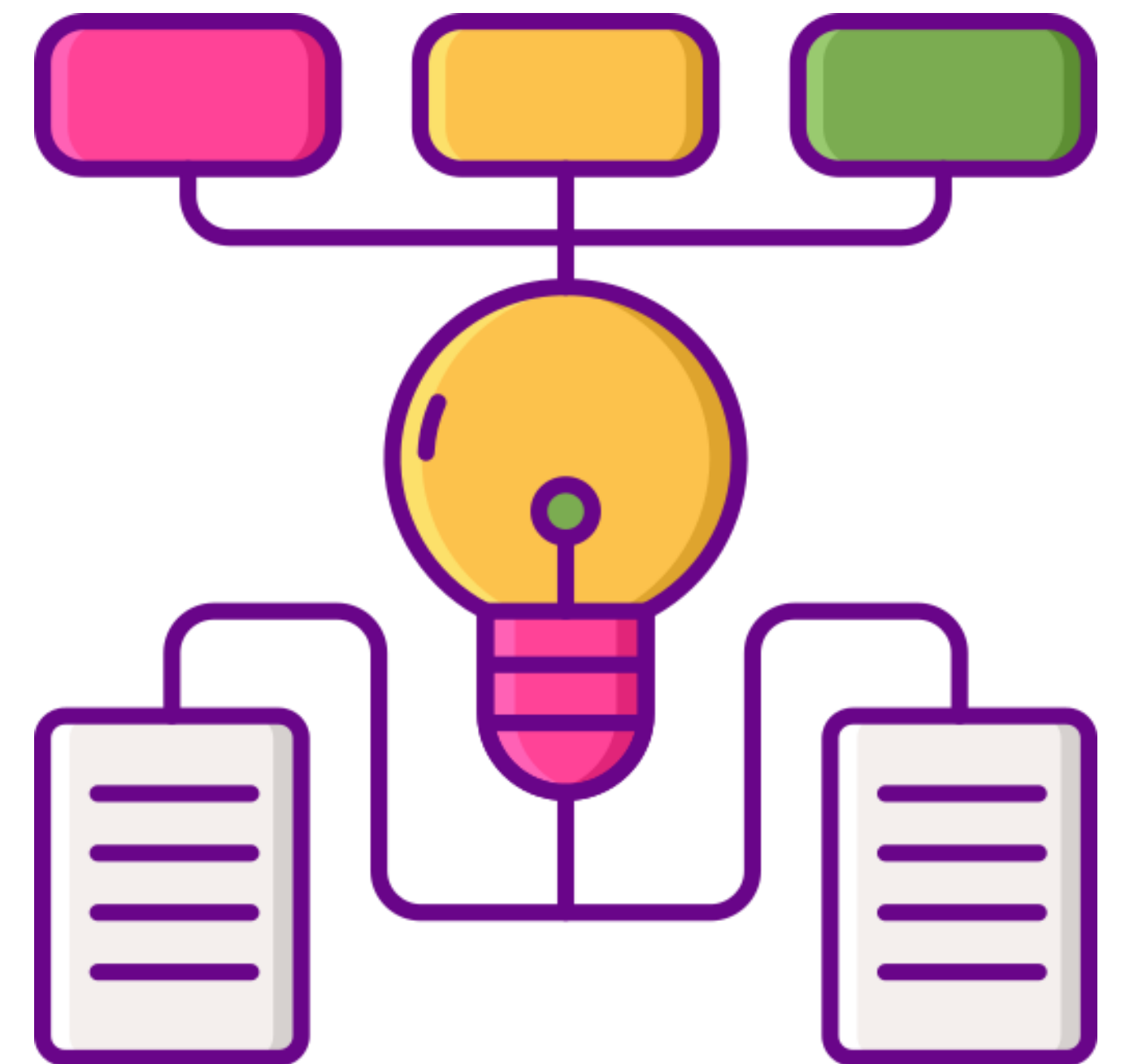
O link abaixo apresenta de forma genérica uma série de tipos de algoritmos que trata sobre o processo de seleção e ordenação:

BubbleSort

MergeSort

Outros.....

<https://www.programiz.com/dsa/merge-sort>





Comunidade VNT



Referências

- [1] A. Goldman, F. Kon, Paulo J. S. Silva; Introdução à Ciência da Computação com Java e Orientação a Objetos (USP). 2006. Ed. USP.
- [2] Algoritmo e lógica de programação. Acessado julho/2022: <https://visualg3.com.br/>
- [3] G. Silveira; Algoritmos em Java; Ed. Casa do Código.
- [4] M. T. Goodrich, R. Tamassia; Estrutura de dados e algoritmos em Java. Ed Bookman. 2007.
- [5] Algoritmo e lógica de programação. Acessado julho/2022: <https://www.cursoemvideo.com/>
- [6] P. Silveira, R. Turini; Java 8 Prático: lambdas, streams e os novos recursos da linguagem. Ed. Casa do Código.
- [7] Linguagem Java: Curso acessado em agosto/2022: <https://www.udemy.com/>
- [8] Linguagem Java: Curso acessado em setembro/2022: <https://www.cursoemvideo.com/>

