**Métodos de ordenação**

Existem diversos métodos de ordenação de vetores com características e desempenhos diferentes, tais como Bubble Sort, CombSort, QuickSort entre outros.

Um dos métodos mais simples e conhecidos é o Bubble Sort que é um método simples mas eficiente para ordenar pequenas quantidades de elementos em um vetor.

Vamos ver três construções desse método.

**Bubble Sort**

O algoritmo de “ordenação bolha”, ou “Bubble sort”, recebeu este nome pela imagem pitoresca usada para descrevê-lo: os elementos maiores são mais leves, e sobem como bolhas até suas posições corretas. W. Celes e J. L. Rangel

**Bubble 1** – é o mais simples e que pode ser melhorado. Usa dois laços de repetição, sendo o primeiro para contar a quantidade de “passadas” pelo vetor e o segundo para percorrer o vetor comparando a posição atual com a posição posterior.

program Bubble1;

const

TAM = 10;

var

vetor: array[1..TAM] of integer;// = (10, 2, 3, 4, 1, 110, 5, 8, 9, 7);

i, j, aux : integer ;

begin

writeln('Informe os 10 valores para serem ordenados:');

for j:= 1 to TAM do

begin

writeln('Posição ',j,': ');

readln(vetor[j]);

end;

**for i:= 1 to TAM do** // para contar a quantidade de “passadas”

begin

**for j:=1 to TAM-1 do** // para percorrer o vetor da primeira até a penúltima posição

begin

if (vetor[j] > vetor[j + 1]) then // compara se a ordem errada

begin

// troca os valores de posição

aux := vetor[j];

vetor[j] := vetor[j + 1];

vetor[j + 1] := aux;

end;

end;

end;

for j:= 1 to TAM do

begin

write(vetor[j]:5);

end;

readln;

end.

**Bubble 2** - O segundo método traz como melhoria não percorrer o vetor todo sempre, ou seja, a cada “passada”, uma posição a menos será percorrida, fazendo com que os elementos no final do vetor já ordenados não sejam novamente comparados.

program Bubble2;

const

TAM = 10;

var

vetor: array[1..TAM] of integer;// = (10, 2, 3, 4, 1, 110, 5, 8, 9, 7);

i, j, aux : integer ;

begin

writeln('Informe os 10 valores para serem ordenados:');

for j:= 1 to TAM do

begin

writeln('Posição ',j,': ');

readln(vetor[j]);

end;

**for i:= 1 to TAM do**

begin

**for j:=1 to TAM-i do** // para percorrer o vetor não ordenado – diminui 1 a cada passada

begin

if (vetor[j] > vetor[j + 1]) then

begin

aux := vetor[j];

vetor[j] := vetor[j + 1];

vetor[j + 1] := aux;

end;

end;

end;

for j:= 1 to TAM do

begin

write(vetor[j]:5);

end;

readln;

end.

**Bubble 3**: Podemos ainda utilizar um flag para verificar se houve trocas ainda na última passada e caso não tenha ocorrido, significa que o vetor já está ordenada e não precisa mais percorrer

program Bubble3;

const

TAM = 10;

var

vetor: array[1..TAM] of integer;// = (10, 2, 3, 4, 1, 110, 5, 8, 9, 7);

j, aux : integer ;

trocas : boolean;

begin

writeln('Informe os 10 valores para serem ordenados:');

for j:= 1 to TAM do

begin

writeln('Posição ',j,': ');

readln(vetor[j]);

end;

**trocas := TRUE;**

**while(trocas = TRUE)do**

begin

**trocas := FALSE;**

for j:=1 to TAM-1 do

begin

if (vetor[j] > vetor[j + 1]) then

begin

aux := vetor[j];

vetor[j] := vetor[j + 1];

vetor[j + 1] := aux;

**trocas := TRUE;**

end;

end;

end;

for j:= 1 to TAM do

begin

write(vetor[j]:5);

end;

readln;

end.