

Ordenação com “Radix Sort”

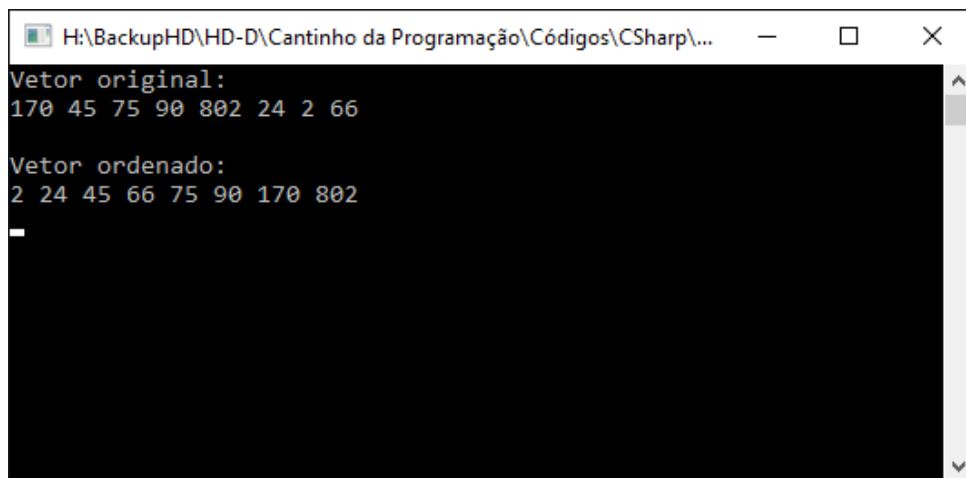
Mário Leite

...

Uma das técnicas mais importantes para os programadores é a Ordenação de dados, segundo alguma tipo de classificação: seja crescente ou decrescente. Na maioria das linguagens de programação existem até funções internas para isto: até em linguagens de 4ª. Geração, como SQL; por exemplo para listar todos alunos em ordem decrescente de média: **SELECT * FROM Alunos ORDER BY media DEC**; A cláusula **DEC** faz com que todos os alunos sejam listados por ordem decrescente de *médias*. Por outro lado, existem várias técnicas e métodos para classificar uma lista de valores: numéricas ou textos. A ordenação pelo “Método da Bolha” (Bubble Sort) é a técnica mais conhecida e a mais fácil, baseando na troca de posições e usando uma variável auxiliar, por exemplo, veja o algoritmo para Ordenação Bubble Sort, em ordem crescente de valores...

```
...
Se (valorX > valorY) Então
    aux = valor;
    valor = valorY;
    valorY = aux;
FimSe
...
```

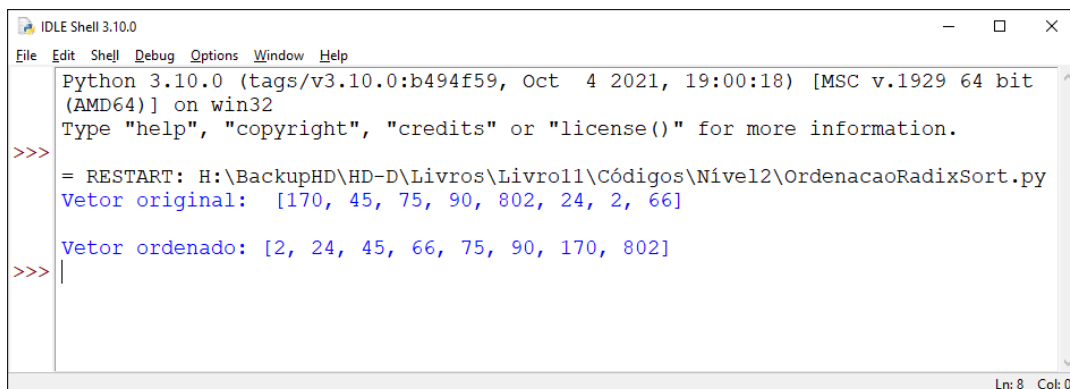
Onde **valorX** e **valorY** são elementos de um vetor que deve ser ordenado em ordem crescente., ficando **valorX** à frente de **valorY**. As **figuras 1 e 2** mostram as saídas do programa “**OrdenacaoRadix**”, codificado em C# e em Python, respectivamente.



```
H:\BackupHD\HD-D\Cantinho da Programação\Códigos\CSharp\...
Vetor original:
170 45 75 90 802 24 2 66

Vetor ordenado:
2 24 45 66 75 90 170 802
```

Figura 1 - Saído do programa em C#



```
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: H:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\OrdenacaoRadixSort.py
Vetor original: [170, 45, 75, 90, 802, 24, 2, 66]
Vetor ordenado: [2, 24, 45, 66, 75, 90, 170, 802]
>>> |
Ln: 8 Col: 0
```

Figura 1 - Saído do programa em Python

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OrdenacaoRadix
{
    class Radix
    {
        //Função de ordenação
        static void RadixSort(int[] VetArr)
        {
            int maxElem = GetMax(VetArr);

            for (int exp = 1; maxElem / exp > 0; exp *= 10)
            {
                CountingSort(VetArr, exp);
            }
        }
        //-----

//=====
        //Programa principal
        static void CountingSort(int[] VetArr, int exp)
        {
            int n = VetArr.Length;
            int[] saida = new int[n];
            int[] cont = new int[10];

            // Inicializa o array de contagem
            for (int i = 0; i < 10; i++)
            {
                cont[i] = 0;
            }

            // Armazena a contagem de ocorrências em cont[]
            for (int i = 0; i < n; i++)
            {
                cont[(VetArr[i] / exp) % 10]++;
            }

            // Atualiza cont[i] para que cont[i] contenha
            // a posição real deste dígito em saida[]
            for (int i = 1; i < 10; i++)
            {
                cont[i] += cont[i - 1];
            }

            // Cria o vetor de saída
            for (int i = n - 1; i >= 0; i--)
            {
                saida[cont[(VetArr[i] / exp) % 10] - 1] = VetArr[i];
                cont[(VetArr[i] / exp) % 10]--;
            }

            // Copia o vetor de saída para VetArr[] para que contenha números
            // classificados de acordo com o dígito atual
            for (int i = 0; i < n; i++)
            {
                VetArr[i] = saida[i];
            }
        }
    }
}
//=====

```

```

// Usa a função para obter o valor máximo de VetArr[]
static int GetMax(int[] VetArr)
{
    int max = VetArr[0];
    for (int i = 1; i < VetArr.Length; i++)
    {
        if (VetArr[i] > max)
        {
            max = VetArr[i];
        }
    }
    return max;
}

//-----
//Imprime o vetor
static void PrintArray(int[] VetArr)
{
    int n = VetArr.Length;
    for (int i = 0; i < n; i++)
    {
        Console.Write(VetArr[i] + " ");
    }
    Console.WriteLine();
}

//-----
//Testa o método
public static void Main()
{
    int[] VetArr = { 170, 45, 75, 90, 802, 24, 2, 66 };
    int n = VetArr.Length;

    Console.WriteLine("Vetor original:");
    PrintArray(VetArr);

    // Faz a ordenação com a função RadixSort()
    Console.WriteLine();
    RadixSort(VetArr);

    Console.WriteLine("Vetor ordenado:");
    PrintArray(VetArr);

    Console.ReadLine();
}

//-----
} //Fim da classe "Radix"
} //Fim da aplicação "OrdenacaoRadix.CS" -----

```

```
'''
OrdenacaoRadix.py
-----
Faz a ordenação dos elementos de um vetor de inteiros utilizando
o "Método Radix Sort", juntamente com "Counting Sort", e mostra o
vetor original e ordenado.
-----
'''

def OrdenarRadixSort(LstVet):
    maxElem = max(LstVet) #pega o elemento de maior valor
    exp = 1

    while (maxElem // exp > 0):
        OrdenarCountingSort(LstVet, exp)
        exp *= 10

    return LstVet #retorna o vetor ordenado
#-----
def OrdenarCountingSort(LstVet, exp):
    n = len(LstVet)
    saida, LstCont = [0]*n, [0]*10
    LstCont = [0]*10 #assume base 10 (0-9)

    for i in range(n):
        LstCont[(LstVet[i] // exp) % 10] += 1

    for i in range(1, 10):
        LstCont[i] += LstCont[i-1]

    i = n - 1
    while (i >= 0):
        ind = (LstVet[i] // exp) % 10
        saida[LstCont[ind] - 1] = LstVet[i]
        LstCont[ind] -= 1
        i -= 1

    LstVet[:] = saida #atualiza 'LstVet' com o resultado
#=====
#Programa principal
VetArr = [170, 45, 75, 90, 802, 24, 2, 66]
print("Vetor original: ", VetArr)
result = OrdenarRadixSort(VetArr)
print()
print("Vetor ordenado:", result)
#Fim do programa "OrdenacaoRadix.Py" -----
```