

# Atividade Prática 02

## Keysorting - Marvel/DC Heroes

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados 2 - EDCO4B  
Prof. Dr. Luiz Fernando Carvalho  
Prof. Dr. Rafael Gomes Mantovani

---

### Instruções:

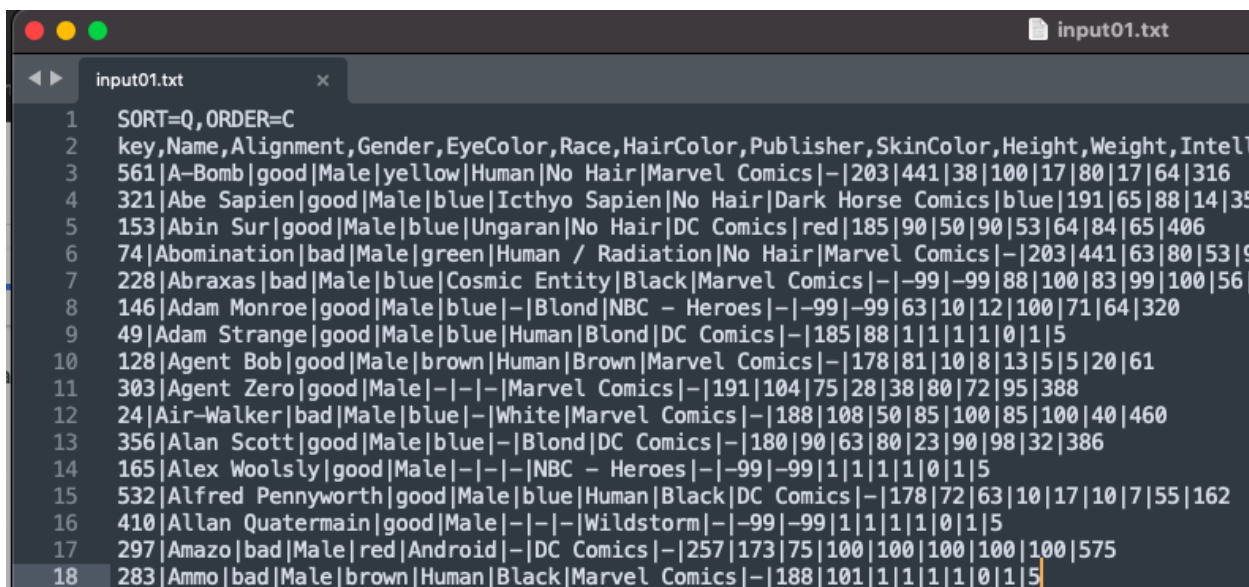
- Leia todas as instruções corretamente para poder desenvolver sua atividade/programa;
- Evite plágio (será verificado por meio de ferramentas automatizadas). Faça seu programa com os seus nomes de variáveis e lógica de solução. Plágios identificados anularão as atividades entregues de todos os envolvidos.
- Adicione comentários nos códigos explicando seu raciocínio e sua tomada de decisão. Porém, não exagere nos comentários, pois a própria estrutura do programa deve ser auto-explicativa.
- Salve sua atividade em um arquivo único, com todas as funções e procedimentos desenvolvidos. É esse **arquivo único** que deverá ser enviado ao professor.

## 1 Descrição da atividade

Como todo bom “computeiro”, o professor M é um nerd convicto e gosta muito de quadrinhos e heróis em geral. De todas as HQs, quadrinhos, animes e outros materiais que ele já consumiu, ele mantém uma lista com as principais informações dos heróis que mais curte. Porém, essa lista não está organizada, pois ele sempre adiciona novos valores conforme seus gostos vão mudando ao longo dos anos. Bom, a ideia do professor M é fazer uma solução que ordene o arquivo com as informações dos heróis. Como vocês são alunos espertos, já ouviram nas aulas do professor que o método de ordenação de chaves (*Keysorting*) realiza a mágica necessária. O desafio é implementar o método, porém ele reaproveita grande parte dos conceitos que vocês já aprenderam em sala. Sendo, assim, mãos a obra. Ajudem o professor M e implementem um programa que faça o *Keysorting* dos registros de herói e grave o resultado da operação em um novo arquivo ordenado.

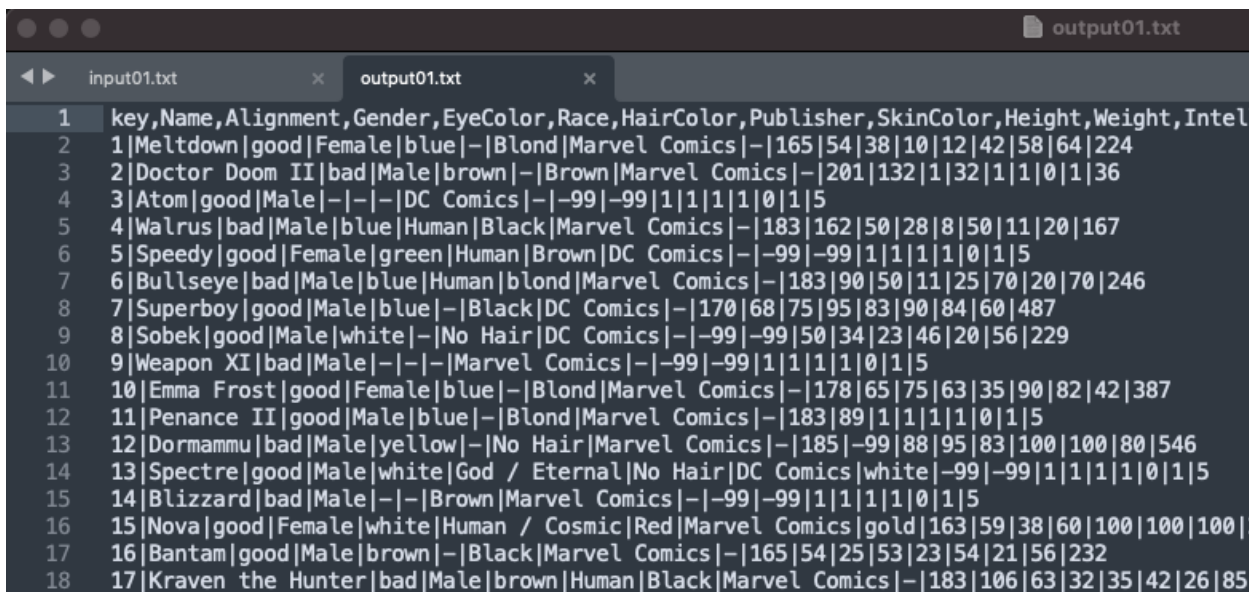
## 2 Entradas do programa

O programa receberá **dois** arquivos texto como parâmetros: um de entrada e um de saída. Exemplos de arquivos manipulados pela aplicação podem ser vistos na Figura 1. Abaixo, iremos detalhar cada um deles.



```
1 SORT=Q, ORDER=C
2 key,Name,Alignment,Gender,EyeColor,Race,HairColor,Publisher,SkinColor,Height,Weight,Intell
3 561|A-Bomb|good|Male|yellow|Human|No Hair|Marvel Comics|-|203|441|38|100|17|80|17|64|316
4 321|Abe Sapien|good|Male|blue|Ichthy Sapien|No Hair|Dark Horse Comics|blue|191|65|88|14|35
5 153|Abin Sur|good|Male|blue|Ungaran|No Hair|DC Comics|red|185|90|50|90|53|64|84|65|406
6 74|Abomination|bad|Male|green|Human / Radiation|No Hair|Marvel Comics|-|203|441|63|80|53|9
7 228|Abraxas|bad|Male|blue|Cosmic Entity|Black|Marvel Comics|-|-99|-99|88|100|83|99|100|56|
8 146|Adam Monroe|good|Male|blue|-|Blond|NBC - Heroes|-|-99|-99|63|10|12|100|71|64|320
9 49|Adam Strange|good|Male|blue|Human|Blond|DC Comics|-|185|88|1|1|1|1|0|1|5
10 128|Agent Bob|good|Male|brown|Human|Brown|Marvel Comics|-|178|81|10|8|13|5|5|20|61
11 303|Agent Zero|good|Male|-|-|Marvel Comics|-|191|104|75|28|38|80|72|95|388
12 24|Air-Walker|bad|Male|blue|-|White|Marvel Comics|-|188|108|50|85|100|85|100|40|460
13 356|Alan Scott|good|Male|blue|-|Blond|DC Comics|-|180|90|63|80|23|90|98|32|386
14 165|Alex Woolly|good|Male|-|-|NBC - Heroes|-|-99|1|1|1|1|0|1|5
15 532|Alfred Pennyworth|good|Male|blue|Human|Black|DC Comics|-|178|72|63|10|17|10|7|55|162
16 410|Allan Quatermain|good|Male|-|-|Wildstorm|-|-99|-99|1|1|1|1|0|1|5
17 297|Amazo|bad|Male|red|Android|-|DC Comics|-|257|173|75|100|100|100|100|100|575
18 283|Ammo|bad|Male|brown|Human|Black|Marvel Comics|-|188|101|1|1|1|1|0|1|5
```

(a) Exemplo de arquivo de entrada.



```
1 key,Name,Alignment,Gender,EyeColor,Race,HairColor,Publisher,SkinColor,Height,Weight,Intell
2 1|Meltdown|good|Female|blue|-|Blond|Marvel Comics|-|165|54|38|10|12|42|58|64|224
3 2|Doctor Doom II|bad|Male|brown|-|Brown|Marvel Comics|-|201|132|1|32|1|1|0|1|36
4 3|Atom|good|Male|-|-|DC Comics|-|-99|-99|1|1|1|1|0|1|5
5 4|Walrus|bad|Male|blue|Human|Black|Marvel Comics|-|183|162|50|28|8|50|11|20|167
6 5|Speedy|good|Female|green|Human|Brown|DC Comics|-|-99|-99|1|1|1|1|0|1|5
7 6|Bullseye|bad|Male|blue|Human|blond|Marvel Comics|-|183|90|50|11|25|70|20|70|246
8 7|Superboy|good|Male|blue|-|Black|DC Comics|-|170|68|75|95|83|90|84|60|487
9 8|Sobek|good|Male|white|-|No Hair|DC Comics|-|-99|-99|50|34|23|46|20|56|229
10 9|Weapon XI|bad|Male|-|-|Marvel Comics|-|-99|-99|1|1|1|1|0|1|5
11 10|Emma Frost|good|Female|blue|-|Blond|Marvel Comics|-|178|65|75|63|35|90|82|42|387
12 11|Penance II|good|Male|blue|-|Blond|Marvel Comics|-|183|89|1|1|1|1|0|1|5
13 12|Dormammu|bad|Male|yellow|-|No Hair|Marvel Comics|-|185|-99|88|95|83|100|100|80|546
14 13|Spectre|good|Male|white|God / Eternal|No Hair|DC Comics|white|-99|-99|1|1|1|1|0|1|5
15 14|Blizzard|bad|Male|-|-|Brown|Marvel Comics|-|-99|-99|1|1|1|1|0|1|5
16 15|Nova|good|Female|white|Human / Cosmic|Red|Marvel Comics|gold|163|59|38|60|100|100|100|2
17 16|Bantam|good|Male|brown|-|Black|Marvel Comics|-|165|54|25|53|23|54|21|56|232
18 17|Kraven the Hunter|bad|Male|brown|Human|Black|Marvel Comics|-|183|106|63|32|35|42|26|85
```

(b) Exemplo de arquivo de saída **depois** do *Keysorting*.

Figura 1: Valores de entrada e correspondentes arquivos de saída gerado pelo programa.

## 2.1 Arquivo de entrada

Os arquivos de entrada são arquivos texto contendo os registros dos heróis anotados pelo professor M (Figura 1a). Durante a execução podem ser fornecidos **N** registros, esse número é variável. O arquivo de entrada é codificado usando **registros de tamanho fixo e campos de tamanho variável**. Parte da solução inclui descobrir o maior tamanho de registro e definir esse valor dinamicamente. Preencha os valores excedentes com um caractere especial de sua escolha. No arquivo de entrada, existem duas linhas/registros com meta-dados usados pela solução. No primeiro registro de cabeçalho (*header*), são detalhadas as opções de ordenação, sumarizadas na Tabela 1.

Tabela 1: Parâmetros contidos no cabeçalho do arquivo de entrada

Parâmetro	Descrição	Opções Válidas
SORT	método de ordenação a ser usado pelo processo de <i>KeySorting</i>	{Q, M, H, I}
ORDER	o sentido da ordenação	{C, D}

O primeiro meta-dado chamado **SORT** define o método de ordenação que realizará o processo de *KeySorting*. Os possíveis valores válidos são: Q - Quick Sort; H - Heap Sort; M - Merge Sort; e I - Insertion Sort. Valores diferentes destes indicam opções inválidas para execução do programa. O segundo meta-dado é indicado pela palavra **ORDER**, e especifica o tipo da ordenação: C para crescente ou D para decrescente. Além disso, existe uma segunda linha com o nome dos atributos correspondentes em cada campos dos registros de herói, como detalhado na Tabela 2:

O dataset usado para a atividade é uma versão do dataset *Marvel Superheroes* do Kaggle<sup>1</sup>. Considerem que na representação dos registros, os correspondentes campos estarão separados por delimitadores fixos. Use o caractere pipe (|) para separar campos de um mesmo registro, e um caractere especial de quebra de linha para identificar o fim de um registro.

```
# Sugestão de implementação de objeto do tipo Heroi:
class Heroi:
    __key;          # chave identificadora do heroi
    __name;         # nome completo do herói
    __aligment;     # bom, mau, neutro?
# ... TODO: adicionar demais atributos (campos)
```

## 2.2 Arquivo de saída

Um arquivo texto contendo o estado resultante do programa após realizar a ordenação das chaves do arquivo de entrada. Se o cabeçalho do arquivo possuir ORDER=C, os registros devem ser apresentados na forme crescente. Caso contrário, ORDER=D, apresentar os registros na forma descrente das chaves. A Figura 1b) mostra um exemplo do arquivo de

<sup>1</sup>[https://www.kaggle.com/datasets/danielr/marvel-superheroes?select=charcters\\_stats.csv](https://www.kaggle.com/datasets/danielr/marvel-superheroes?select=charcters_stats.csv)

Tabela 2: Informações contidas no registro de herói

<b>Campo</b>	<b>Descrição</b>
<i>key</i>	chave única para identificação do herói
<i>Name</i>	nome do herói
<i>Alignment</i>	alinhamento (bom, mal, neutro)
<i>Gender</i>	gênero do herói
<i>EyeColor</i>	cor dos olhos do herói
<i>Race</i>	raça do herói
<i>HairColor</i>	cor do cabelo do herói
<i>Publisher</i>	editora (Marvel, DC, etc)
<i>SkinColor</i>	cor da pele do herói
<i>Height</i>	altura do herói
<i>Weight</i>	peso do herói
<i>Intelligence</i>	nível de inteligência do herói [0,100]
<i>Strength</i>	nível de força do herói [0,100]
<i>Speed</i>	nível de velocidade do herói [0,100]
<i>Durability</i>	nível de durabilidade do herói [0,100]
<i>Power</i>	nível de poder do herói [0,100]
<i>Combat</i>	nível de combate do herói [0,100]
<i>Total</i>	nível total do herói [0,100]

saída. Perceba que o cabeçalho com o nome dos campos também é impresso no arquivo de saída.

### 3 Rodando o programa

Para rodar o programa por linha de comando, manipular os argumentos **argc** e **argv** da função **main**. Para executar o programa por linha de comando, deve-se obedecer o seguinte padrão:

```
[nome do programa] [arquivo de entrada] [arquivo de saída]
```

Exemplo de execução de um programa chamado **keysorting.py**:

```
python keysorting.py entrada1.txt saida1.txt
```

### 4 Orientações gerais

Além da funcionalidade desejada, implementar também o controle de erros, para lidar com exceções que possam ocorrer, como por exemplo:

- problemas nas aberturas dos arquivos de entrada e saída;

- arquivos de entrada vazio (sem informação);
- arquivos de entrada fora do padrão esperado (opções inválidas para uso no cabeçalho);
- etc.

Opcionalmente, para acompanhamento do desenvolvimento, pode-se criar um repositório individual no `github`.

## 5 Critérios de correção

A nota na atividade será contabilizada levando-se em consideração alguns critérios:

1. pontualidade na entrega;
2. não existir plágio;
3. completude da implementação (tudo foi feito);
4. o código compila e executa;
5. uso de parâmetros de linha de comando para controle dos arquivos de teste;
6. implementar a leitura dos dados de entrada via arquivo texto;
7. implementação correta das estruturas necessárias (campos, registros e sua manipulação, ordenação das chaves);
8. legibilidade do código (identação, comentários nos blocos mais críticos);
9. implementação dos controles de erros (arquivos de entrada inválidos, e erros no programa principal);
10. executar corretamente os casos de teste.

Em cada um desses critérios, haverá uma nota intermediária valorada por meio de conceitos:

- **Sim** - se a implementação entregue cumprir o que se esperava daquele critério;
- **Parcial** - se satisfizer parcialmente o tópico;
- e **Não** se o critério não foi atendido.

## 6 Padrão de nomenclatura

Ao elaborar seu programa, crie um único arquivo fonte (.c) seguindo o padrão de nome especificado:

ED2-AT02-Keysorting-<NOME>.py

Exemplo:

ED2-AT02-Keysorting-LuizCaravlhho.py

A entrega da atividade será via Moodle: o link será disponibilizado na página da disciplina.

## 7 Links úteis

- Arquivos em Python:

- <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
- [https://www.w3schools.com/python/python\\_file\\_open.asp](https://www.w3schools.com/python/python_file_open.asp)
- <https://www.pythontutorial.net/python-basics/python-read-text-file/>

- Argumentos de Linha de comando no Python:

- [https://www.tutorialspoint.com/python3/python\\_command\\_line\\_arguments.htm](https://www.tutorialspoint.com/python3/python_command_line_arguments.htm)
- <https://realpython.com/python-command-line-arguments/>
- <http://devfuria.com.br/python/sys-argv/>

## Referências

- [1] Michael J. Folk; Bill Zoellick; Greg Riccardi. File Structures, 3rd edition, Addison-Wesley, 1997.
- [2] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.
- [3] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [4] Adam Drozdek. Estrutura De Dados e Algoritmos em C++. Cengage, 2010.