Atividade Prática 02
Reuso de Espaços em Arquivos de Registros

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Disciplina de Estrutura de Dados 2 - EDCO4B
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

O Professor M está cansado. Depois de vários semestres apenas lecionando, ele decidiu tirar um tempo para jogar alguns games e relaxar. Como todo bom gamer, ele acabou acumulando uma grande quantidades de games e não lembra mais quais jogos jogou ou deseja zerar. Dessa forma, o passo inicial seria organizar e contabilizar todos os jogos disponíveis em seu acervo, para depois definir quais jogos zerar. **Mas ... , sem tempo irmão!** O professor M não consegue resolver toda essa treta sozinho, e precisa de ajuda de alguns engenheiros de computação. Coincidentemente (ou não), vocês são esses engenheiros, ajudem o professor M!

Para isso, façam um programa que consiga manipular do disco as informações de um número variado de games. Nesse programa, vocês irão **simular** o funcionamento da escrita/leitura de bytes por meio de arquivos texto. Esse sistema irá guardar as seguintes informações de um game:

- Título do game (até 50 caracteres);

- Produtora: quem desenvolveu o game (até 40 caracteres);
- Gênero: qual o gênero do game (até 25 caracteres);
- Plataforma: para qual tipo de maquina foi desenvolvido (PC, PS4, Xbox, etc...). Se forem para vários, colocar "Multiplataforma" (até 15 caracteres);
- Ano: data de lançamento do game (4 caracteres);
- Classificação: classificação etária do game (até 12 caracteres);
- Preço: média de preço para o produto (até 7 caracteres);
- Tipo de mídia: se esta disponível em formato físico, digital, ou em ambos (até 8 caracteres);
- Tamanho: tamanho da mídia (em GB) (até 7 caracteres);

O problema é que alguns jogos são doados, vendidos ou acabaram sendo perdidos. Assim, o programa precisa ser robusto o suficiente para remover as informações dos games que partiram, e adicionar as informações dos novos. Mas existe um porém: sempre tentando otimizar ao máximo o tamanho dos arquivos usados para gravar a informação, e para isso, **manter um processo dinâmico para reuso dos espaços livres do arquivo é imprescindível**. Sendo assim, implemente um programa que manipule os arquivos do sistema do Professor M, e faça remoções e inserções (*updates*) persistindo os registros em arquivo. Os eventuais espaços disponíveis no arquivo serão reutilizados por uma abordagem dinâmica.

2 Entradas do programa

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

- **arquivo de entrada:** um arquivo texto contendo os registros das games (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 com campos de tamanho variável**. Há um registro de cabeçalho (*header*) que guarda a quantidade de registros válidos no arquivo, e também o endereço do primeiro elemento da Pilha de Disponibilidade (*Avail List*) em termos de RRN, identificado com a palavra-chave **TOP**;
- **arquivo de operações:** um arquivo texto onde cada linha descreve uma operação a ser simulada pelo programa (Figura 1b). Apenas duas operações podem ser realizadas: remover um elemento indicando uma chave para procura; e adicionar um novo registro. A sintaxe desses comandos é explicada abaixo:

input1.txt

```
REG.N=15 TOP=-1
Hollow Knight|Team Cherry|Metroidvania|Multiplataforma|2017|Everyone 10+|24.97|Digital|5.3
Devil May Cry 5|Capcom|Hack and Slash|Multiplataforma|2019|Mature 17+|99.90|Ambos|35.0
The Binding of Isaac: Rebirth|Nicalis|Roguelike|PC|2014|Mature 17+|27.99|Digital|0.449
Blasphemous|Team17|Metroidvania|Multiplataforma|2019|Mature+18|57.99|Digital|0.85388
Pokemon Emerald|Nintendo|RPG de turno|GBA|2004|Everyone|34.99|Ambos|0.016
Final Fantasy XV|Square Enix|Action RPG|Multiplataforma|2016|Teen|125.00|Ambos|100.0
Rocket League|Psyonix|Sports|Multiplataforma|2015|Everyone|Free|Digital|20.0
Fallout 4|Bethesda Game Studios|RPG|Multiplataforma|2015|Mature 17+|59.99|Ambos|36.23
Borderlands 3|Gearbox Software|RPG|Multiplataforma|2020|Mature 17+|119.90|Ambos|94.38
Arma 3|Bohemia Interactive|Simulacao|PC|2013|Mature 17+|29.99|Digital|40.57
Squad|Offworld Industries|Indie|PC|2020|Mature 17+|70.49|Digital|74.15
Halo 4|343 Industries|FPS|Xbox 360/One/Series|2012|Mature 17+|49.00|Ambos|55.0
Halo 3|343 Industries|FPS|Xbox 360/One/Series|2007|Mature 17+|49.00|Ambos|11.0
Bleach Brave Souls|KlabGames|Action RPG|Multiplataforma|2016|12+|00.00|Digital|5.0
Counter Strike Global Offensive|Valve|First-person Shooter|PC|2012|16+|F2P|Digital|29.51
```

(a) Exemplo de arquivo de entrada.

op1.txt

```
i, Resident Evil 4,Capcom,Action Horror,Multiplataforma,2005,Mature 17+,190.00,Ambos,15.0
i, Hollow Knight,Team Cherry,Metroidvania,Multiplataforma,2017,Everyone 10+,24.97,Digital,5.3
d, HOLLOWKNIGHT2017
d, SQUAD2020
d, HALO32007
```

(b) Exemplo de arquivo de operações.

temp1.txt

```
REG.N=13 TOP=12
*-1|ow Knight|Team Cherry|Metroidvania|Multiplataforma|2017|Everyone 10+|24.97|Digital|5.3
Devil May Cry 5|Capcom|Hack and Slash|Multiplataforma|2019|Mature 17+|99.90|Ambos|35.0
The Binding of Isaac: Rebirth|Nicalis|Roguelike|PC|2014|Mature 17+|27.99|Digital|0.449
Blasphemous|Team17|Metroidvania|Multiplataforma|2019|Mature+18|57.99|Digital|0.85388
Pokemon Emerald|Nintendo|RPG de turno|GBA|2004|Everyone|34.99|Ambos|0.016
Final Fantasy XV|Square Enix|Action RPG|Multiplataforma|2016|Teen|125.00|Ambos|100.0
Rocket League|Psyonix|Sports|Multiplataforma|2015|Everyone|Free|Digital|20.0
Fallout 4|Bethesda Game Studios|RPG|Multiplataforma|2015|Mature 17+|59.99|Ambos|36.23
Borderlands 3|Gearbox Software|RPG|Multiplataforma|2020|Mature 17+|119.90|Ambos|94.38
Arma 3|Bohemia Interactive|Simulacao|PC|2013|Mature 17+|29.99|Digital|40.57
*0|ad|Offworld Industries|Indie|PC|2020|Mature 17+|70.49|Digital|74.15
Halo 4|343 Industries|FPS|Xbox 360/One/Series|2012|Mature 17+|49.00|Ambos|55.0
*10| 3|343 Industries|FPS|Xbox 360/One/Series|2007|Mature 17+|49.00|Ambos|11.0
Bleach Brave Souls|KlabGames|Action RPG|Multiplataforma|2016|12+|00.00|Digital|5.0
Counter Strike Global Offensive|Valve|First-person Shooter|PC|2012|16+|F2P|Digital|29.51
Resident Evil 4,Capcom,Action Horror,Multiplataforma,2005,Mature 17+,190.00,Ambos,15.0
```

(c) Exemplo de arquivo temporário **antes** do *storage compaction*.

output1.txt

```
REG.N=13 TOP=12
Devil May Cry 5|Capcom|Hack and Slash|Multiplataforma|2019|Mature 17+|99.90|Ambos|35.0
The Binding of Isaac: Rebirth|Nicalis|Roguelike|PC|2014|Mature 17+|27.99|Digital|0.449
Blasphemous|Team17|Metroidvania|Multiplataforma|2019|Mature+18|57.99|Digital|0.85388
Pokemon Emerald|Nintendo|RPG de turno|GBA|2004|Everyone|34.99|Ambos|0.016
Final Fantasy XV|Square Enix|Action RPG|Multiplataforma|2016|Teen|125.00|Ambos|100.0
Rocket League|Psyonix|Sports|Multiplataforma|2015|Everyone|Free|Digital|20.0
Fallout 4|Bethesda Game Studios|RPG|Multiplataforma|2015|Mature 17+|59.99|Ambos|36.23
Borderlands 3|Gearbox Software|RPG|Multiplataforma|2020|Mature 17+|119.90|Ambos|94.38
Arma 3|Bohemia Interactive|Simulacao|PC|2013|Mature 17+|29.99|Digital|40.57
Halo 4|343 Industries|FPS|Xbox 360/One/Series|2012|Mature 17+|49.00|Ambos|55.0
Bleach Brave Souls|KlabGames|Action RPG|Multiplataforma|2016|12+|00.00|Digital|5.0
Counter Strike Global Offensive|Valve|First-person Shooter|PC|2012|16+|F2P|Digital|29.51
Resident Evil 4,Capcom,Action Horror,Multiplataforma,2005,Mature 17+,190.00,Ambos,15.0
```

(d) Exemplo de arquivo de saída **depois** do *storage compaction*.

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

  - remoção: <d> <chave> - um caractere único **d** seguido da chave canônica associada ao registro que será excluído (pode existir ou não). A chave canônica é composta da união dos campos "título" e "ano", presentes no registro de games;
  - inserção: <id> <registro> - um caractere único **i** seguido do novo registro, com os campos separados por vírgulas. Só faz a inserção se não existir o registro;

- **arquivo de saída temporário**: um arquivo contendo as edições do arquivo de entrada depois de executar as operações do arquivo de operações (Figura 1c). Esse arquivo mostra o estado do arquivo dos registros **antes** de realizar o *storage compaction*. Perceba que os registros removidos são marcados com um caractere especial (*) e os espaços para reuso são organizados por meio de uma Pilha de RRNs. Isso implica em sempre atualizar o registro de cabeçalho para ter acesso imediato ao espaço liberado mais recententemente;

- **arquivo de saída final:** um arquivo texto contendo o estado resultante do programa após todas as operações listadas no arquivo de operações e após execução do *storage compaction*. **Dica**: faça uma cópia do arquivo de entrada e os manipule durante a execução para criar o arquivo de saída.

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.
Dica: Para rodar o programa por linha de comando, manipular os argumentos via **sys.argv** no script principal, obedecendo o seguinte padrão:

```
[nome do programa] [arquivo de entrada] [arquivo de operações] [arquivo
        de saída temporário] [arquivo de saída final]
```

Exemplo de execução de um programa chamado `teste.py`:

```
python3 teste.py entrada1.txt op1.txt tmp1.txt saida1.txt
```

3 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);
- etc.

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

4 Sugestão de funções

Tabela 1: Sugestão de nomes de funções que poderão ser implementadas.

Função
`def adicionaRegistro(arq, game);`
`def procuraRegistro(arq, game);`
`def removeRegistro(arq, chave);`
`def storageCompaction(arq);`

4.1 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: solução contém tudo que foi solicitado no enunciado;
4. o código executa;
5. uso de parâmetros por linha de comando para execução do *script* com os 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);
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.

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

```
ED2-AT02-UpdateRegistros-<NOME>.py
```

Exemplo:

```
ED2-AT02-UpdateRegistros-RafaelMantovani.py
```

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

5 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.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://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.