

Atividade Prática 02

Simulação de Leitura/Escrita de Bytes em Estruturas de Arquivos

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 tentando inovar. Depois de uns semestres apenas lecionando, ele desejou ajudar na gestão do curso onde trabalha. E para isso, quer agregar e armazenar a informação de todos os professores do curso, para que seja mais fácil entrar em contato com cada um deles. Mas, sem tempo irmão! O professor M não consegue resolver toda essa treita sozinho, e precisa de ajuda de alguns engenheiros de computação, pelo menos no sistema de gravação/recuperação de informação em disco.

Coincidentemente (ou não), vocês são esses engenheiros, ajudem o professor M. Para isso, façam um programa que consiga persistir e recuperar do disco as informações de um número variado de professores. Nesse sistema, vocês irão **simular** o funcionamento da escrita/leitura de bytes por meio de arquivos texto. Esse sistema irá guardar as seguintes informações de um professor:

- código identificador: inteiro de até 3 dígitos;
- nome: de até 30 caracteres;
- sexo: um caractere, f para feminino, m para masculino, n para não informado;
- idade: inteiro de 2 dígitos;
- área de especialidade: string com até 30 caracteres;
- telefone: descrito no formato (XX)XXXXX-XXXX, ou seja, com 14 caracteres.

2 Entradas do programa

O programa receberá três arquivos texto como parâmetros de entrada:

- **arquivo de entrada:** um arquivo texto contendo os registros das pessoas/professores. Durante a execução podem ser fornecidos **N** registros. Esse número é variável. Após os registros, existirá uma linha com um inteiro único, especificando qual método de leitura/escrita será usado:
 1. uso de registros de tamanho fixo;
 2. indicadores de tamanho para cada registro;
 3. uso de delimitadores entre registros.
- **arquivo de saída:** um arquivo texto onde deverá ser impresso os objetos na codificação desejada de acordo com o correspondente método especificado na entrada;
- **arquivo de persistência:** um arquivo texto onde deverá ser impresso os objetos recuperados pelos métodos de leitura, mostrando a persistência da informação no disco e recriando a informação do arquivo de entrada.

Considerem que para qualquer um dos métodos de organização dos registros, os correspondentes campos estarão separados por delimitadores fixos. Use o caractere pipe (|) para separar campos de um mesmo registro. Além da necessidade de definir uma **struct** para organizar os dados, a solução pode seguir a sugestão de implementação das seguintes funções:

Exemplos de arquivos de entrada e correspondentes saídas são apresentados na Figura 1.

Dica: 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:

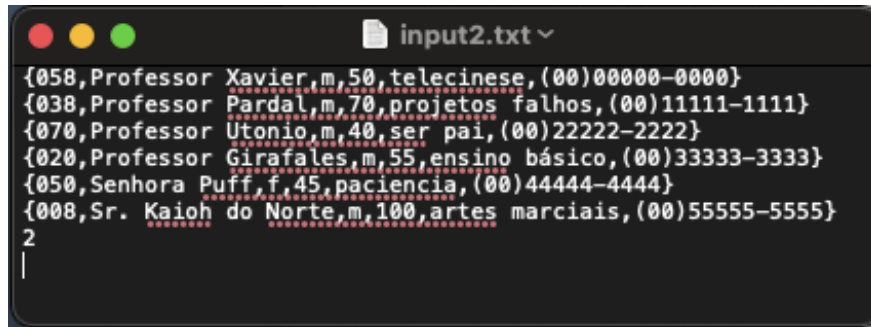
```
[nome do programa] [arquivo de entrada] [arquivo de saída] [arquivo de
                        persistência]
```

Exemplo de execução de um programa chamado **teste.c**:

```
./teste entrada.txt saida.txt persistencia.txt
```

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

Função
<code>void escreverRegistrosTamanhoFixo(FILE* arq, Professor p);</code>
<code>void escreverRegistrosIndicadoresTamanho(FILE* arq, Professor p);</code>
<code>void escreverRegistrosDelimitadores(FILE* arq, Professor p);</code>
<code>void lerRegistrosTamanhoFixo(FILE* arq, Professor p);</code>
<code>void lerRegistrosIndicadoresTamanho(FILE* arq, Professor p);</code>
<code>void lerRegistrosDelimitadores(FILE* arq, Professor p);</code>

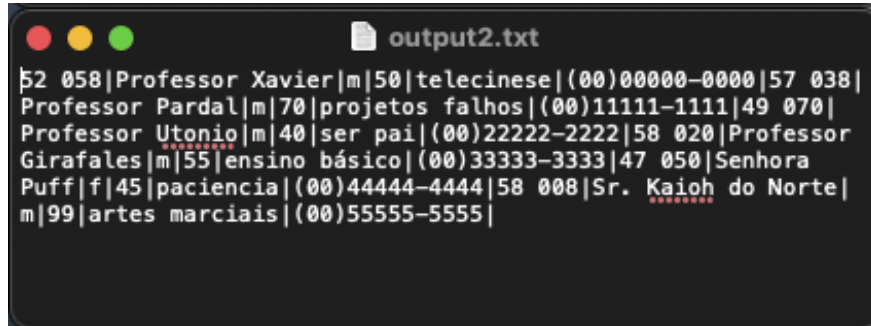


```

input2.txt
{058,Professor Xavier,m,50,telecinese,(00)00000-0000}
{038,Professor Pardal,m,70,projetos falhos,(00)11111-1111}
{070,Professor Utonio,m,40,ser pai,(00)22222-2222}
{020,Professor Girafales,m,55,ensino básico,(00)33333-3333}
{050,Senhora Puff,f,45,paciencia,(00)44444-4444}
{008,Sr. Kaioh do Norte,m,100,artes marciais,(00)55555-5555}
2

```

(a) Exemplo de arquivo de entrada.

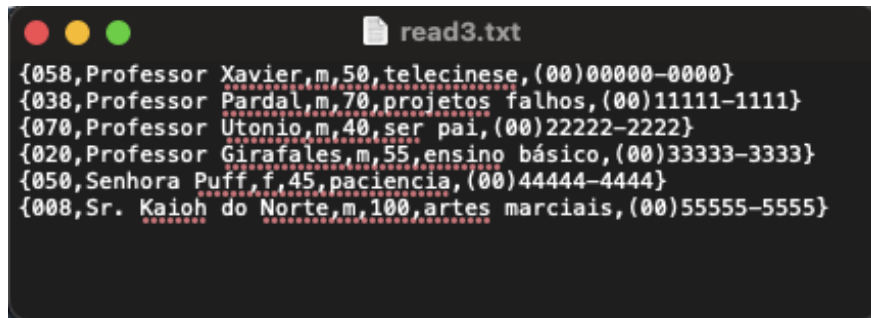


```

output2.txt
52 058|Professor Xavier|m|50|telecinese|(00)00000-0000|57 038|
Professor Pardal|m|70|projetos falhos|(00)11111-1111|49 070|
Professor Utonio|m|40|ser pai|(00)22222-2222|58 020|Professor
Girafales|m|55|ensino básico|(00)33333-3333|47 050|Senhora
Puff|f|45|paciencia|(00)44444-4444|58 008|Sr. Kaioh do Norte|
m|99|artes marciais|(00)55555-5555|

```

(b) Exemplo de arquivo de saída.



```

read3.txt
{058,Professor Xavier,m,50,telecinese,(00)00000-0000}
{038,Professor Pardal,m,70,projetos falhos,(00)11111-1111}
{070,Professor Utonio,m,40,ser pai,(00)22222-2222}
{020,Professor Girafales,m,55,ensino básico,(00)33333-3333}
{050,Senhora Puff,f,45,paciencia,(00)44444-4444}
{008,Sr. Kaioh do Norte,m,100,artes marciais,(00)55555-5555}

```

(c) Exemplo de arquivo de persistência.

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

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;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado (opções inválidas para uso);
- etc.

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

3.1 Critério 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 `argc` e `argv` para controle dos arquivos de teste;
6. implementar o parser para entrada dos dados via arquivo texto;
7. implementação correta das estruturas necessárias;
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. controle de memória: chamar o destrutor e desalocar a memória de tudo se usar estruturas dinâmicas, fechar os arquivos, etc;
11. 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 (.c) seguindo o padrão de nome especificado:

ED2-<ANO>-<SEMESTRE>-AT02-Registros-<NOME>.c

Exemplo:

ED2-2021-2-AT02-Registros-RafaelMantovani.c

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

4 Links úteis

Arquivos em C:

- <https://www.inf.pucrs.br/~pinho/LaproI/Arquivos/Arquivos.htm>
- <https://www.geeksforgeeks.org/basics-file-handling-c/>
- <https://www.programiz.com/c-programming/c-file-input-output>

Argumentos de Linha de comando (argc e argv):

- https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- <http://linguagemc.com.br/argumentos-em-linha-de-comando/>
- http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula19.pdf
- http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-31_Argumentos_linha_comando.html
- <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node145.html>

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.