Atividade Prática 03
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

Graças aos seus esforços você conseguiu ajudar o Professor M na tarefa de gerenciar a informação de todos os professores do curso onde trabalha. Aliás, o professor M ficou extremamente satisfeito com suas habilidades e lhe propõe uma nova *quest*: incrementar o sistema para que possa substituir alguns registros do arquivo. O problema é que alguns professores acabam saindo de férias, ou se transferem para outros lugares, e novos professores são recebidos para suprir essas vagas. Assim, o sistema precisa ser robusto o suficiente para remover as informações dos professores que partiram, e adicionar as informações dos novos professores. 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. Lembrando que os arquivos do sistema guardam informações de um professor:

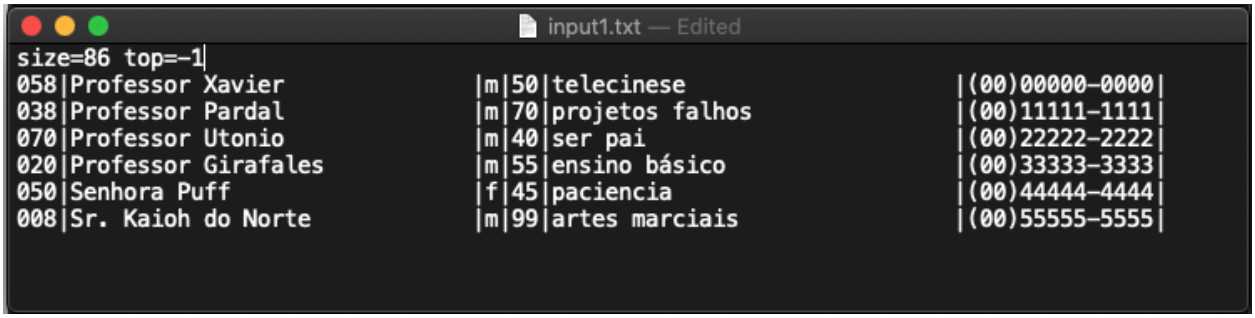
- código identificador: inteiro de até 3 dígitos;
- nome: de até 30 caracteres;
- sexo: um caractere, f para feminino, m para masculino, n para não informado;
- idade: inteiro de 2 dígitos;
- área de especialidade: string com até 30 caracteres;
- telefone: descrito no formato (XX)XXXXX-XXXX, ou seja,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

2 Entradas do programa

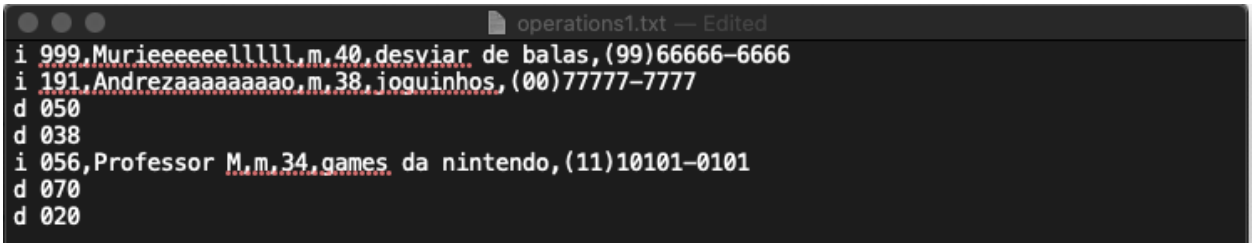
O programa receberá quatro arquivos texto como parâmetros de entrada, dois para entrada e dois para saída. Exemplos de arquivos manipulados pela aplicação podem ser vistos na Figura 1. Abaixo, iremos detalhar cada um deles:

- **arquivo de entrada:** um arquivo texto contendo os registros das pessoas/professores (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**. Há um registro de cabeçalho (*header*) que indica o tamanho do registro em bytes (caracteres no nosso caso), e guarda também o endereço do primeiro elemento da Pilha de Disponibilidade (*Avail List*) em termos de RRN;
- **arquivo de operações:** um arquivo texto onde cada linha descreve uma operação a ser simulada pelo programa (Figura 1b). Há um total de duas operações que podem ser realizadas: remover um elemento indicando uma chave para procura; e adicionar um novo registro. A sintaxe desses comandos é explicada abaixo:
  - remoção: <d> <chave> - um caractere único **d** seguido da chave associada ao registro que será excluído (pode existir ou não);
  - inserção: <id> <registro> - um caractere único **i** seguido do novo registro, com os campos separados por vírgulas;
- **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.

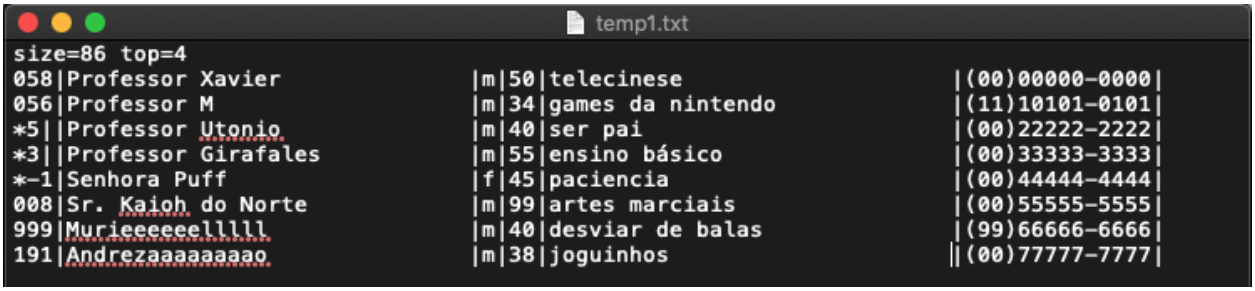
Considerem que na representação dos registros, os correspondentes campos estarão separados por delimitadores fixos. Use o caractere pipe (|) para separar campos de um mesmo registro, e um caractere especial de quebra de linha para identificar o fim de um registro.



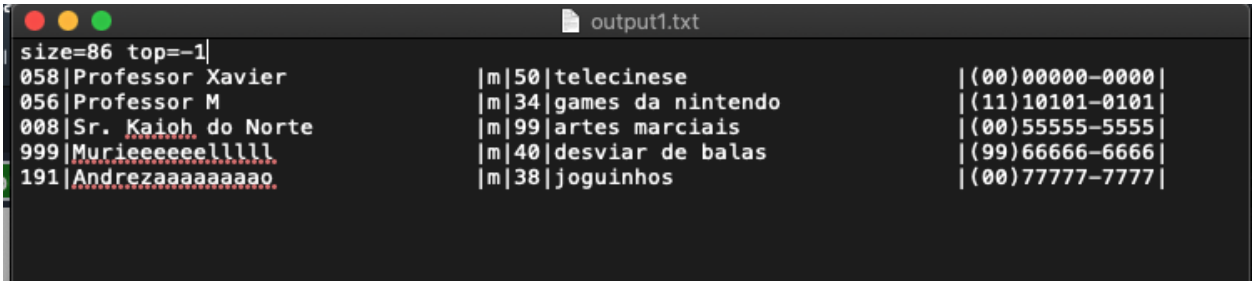
(a) Exemplo de arquivo de entrada.



(b) Exemplo de arquivo de operações.



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



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

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

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 operações] [arquivo de saída temporário] [arquivo de saída final]`

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

`./teste 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`.

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>-AT03-UpdateRegistros-<NOME>.c`

Exemplo:

`ED2-2021-2-AT0AT03-UpdateRegistro-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.