Atividade Prática 03
Manipulação de Listas Duplamente Encadeadas

Universidade Tecnológica Federal do Paraná (UTFPR), campus Dois Vizinhos
Curso de Engenharia de Software
Disciplina de Estrutura de Dados - ED23S
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 é frequentemente procurado pelos amigos para ajudar a solucionar problemas com a tecnologia. Em uma dessas necessidades, diferentemente de instalar uma impressora ou modem, o Mr. Ugly, amigo do professor, precisava de ajuda para organizar os arquivos de sua clínica de ortodontia particular. A clínica precisa urgentemente organizar o registo de seus pacientes, de uma forma que seja fácil encontrar a ficha de um paciente em específico, ou imprimir a listagem dos mesmos em ordem crescente ou decrescente.

Sem tempo para tentar resolver sozinho o problema, o professor tem procurado um(a) programador(a) que consiga colocar a "ordem na casa" e gerar um sistema que o ajude a obter as informações dos pacientes. Por uma (in)felicidade do destino, você é esse(a) programador(a)! Desta forma, faça um programa que implemente o funcionamento de um sistema para a clínica, recebendo as informações dos pacientes e as organizando em uma **lista duplamente encadeada com comportamento circular**. Esse sistema será capaz de realizar algumas operações:

1. consultar se um registro de um paciente está contido no sistema (lista). Imprimir se existir, caso contrário indicar que não foi encontrado;

2. imprimir todos os registros de pacientes em ordem crescente;

3. imprimir todos os registros de pacientes em ordem decrescente.

Além de um código único, cada paciente é representado no sistema por um conjunto de informações como: nome, sexo (m ou f), idade, peso, altura, e telefone para contato. Claro, que todas essas informações precisam ser impressas/mostradas pelas operações do sistema.

2 Lista duplamente encadeada circular

Uma lista duplamente encadeada é um arranjo de dados onde cada elemento é também um tipo abstrato de nó de lista (`NoLista`) que guarda dois ponteiros, como mostrado na Figura 1:

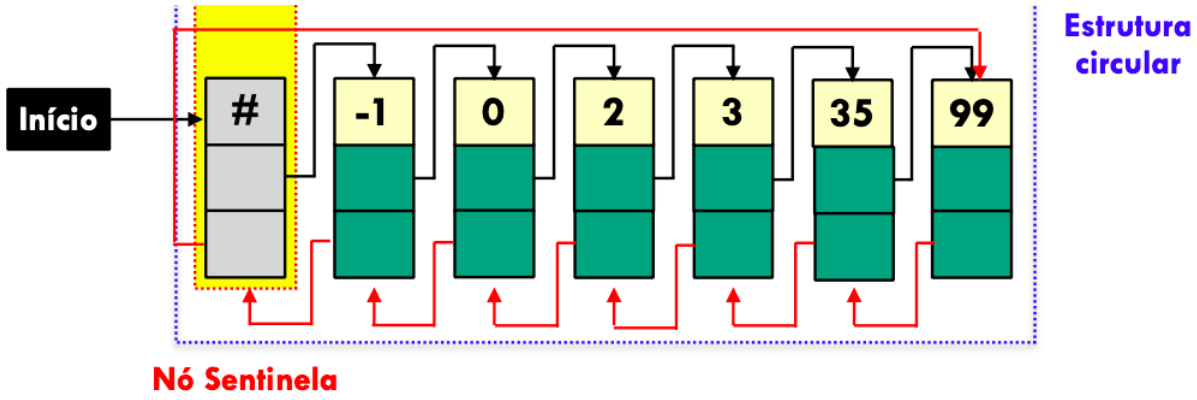


Figura 1: Diagrama representativo de uma lista duplamente encadeada circular com 6 elementos. O Nó sentinela (em cinza) é o primeiro elemento acessível a partir do ponteiro de início da estrutura e permite o acesso fácil ao primeiro e último elementos do arranjo.

- **anterior**: ponteiro que aponta para o elemento anterior na lista ordenada;
- **próximo**: ponteiro que aponta para o elemento posterior na lista ordenada.

Uma lista duplamente encadeada com comportamento circular é uma lista com um nó sentinela/cabeça, que controla o acesso imediatado tanto do primeiro como do último elemento da lista. Esse nó possui uma chave com um valor de controle, diferente de qualquer valor de chave de elementos válidos. Esse valor deve ser escolhido/definido durante a implementação. Assim, use as implementações das estruturas já desenvolvidas em sala para criar sua implementação de lista duplamente encadeada circular.

3 Entradas do programa

O programa receberá dois arquivos texto como parâmetros de entrada:

- **arquivo de entrada:** um arquivo texto contendo os registros/cadastros dos pacientes. Cada linha contém a informação de um paciente, na ordem: código, nome, sexo, idade, peso, altura e telefone para contato. Durante a execução podem ser fornecidos **N** pacientes. Esse número é variável. Após os registros, existirá uma linha com um inteiro único, especificando qual operação será realizada:

  1. impressão na ordem crescente dos registros (segundo o código);
  2. impressão na ordem decrescente dos registros (segundo o código);
  3. consulta se um determinado paciente existe ou não nos registros da clínica.

  No caso 3 em específico, haverá mais uma linha com um inteiro único correspondente ao código que será consultado na lista. Perceba que o código consultado pode ou não existir nos registros, e é sua tarefa lidar com ambas as situações.

- **arquivo de saída:** um arquivo texto onde deverá ser impressa a saída desejada:

  1. os registros impressos, um por linha, em ordem crescente de código;
  2. os registros impressos, um por linha, em ordem decrescente de código;
  3. se o código consultado existir, imprimir ele no arquivo de saída. Caso não exista, imprimir uma mensagem indicando que o código não existe/não foi encontrado.

Exemplos de arquivos de entrada e correspondentes saídas são apresentados na Figura 2. **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>

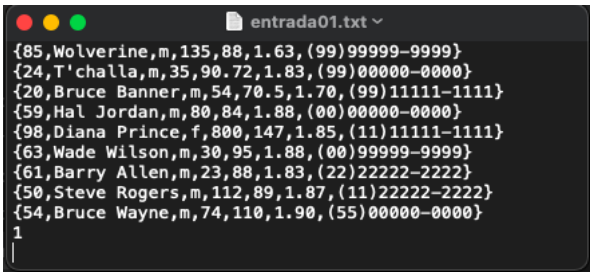
Exemplo de execução de um programa chamado `teste.c`:

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

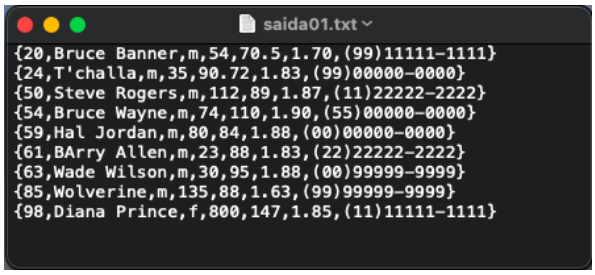
4 Orientações gerais

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

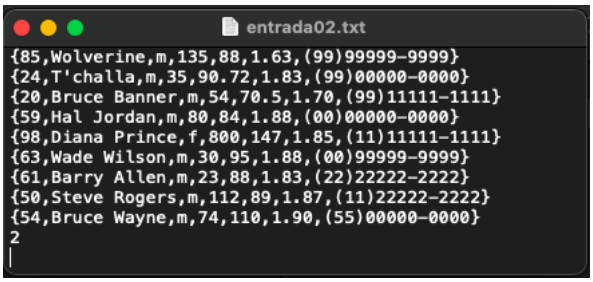
- problemas nas aberturas dos arquivos de entrada e saída;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado (registros não formatados no padrão de leitura, opções inválidas para uso da lista, etc);



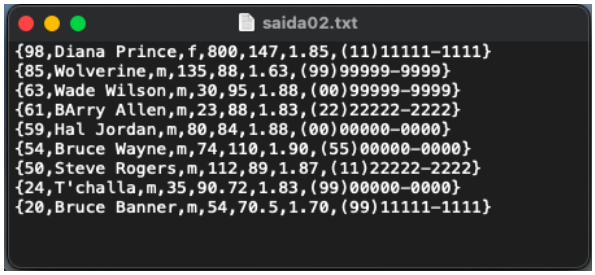
(a) Exemplo de arquivo de entrada para impressão dos registros em ordem crescente de código.



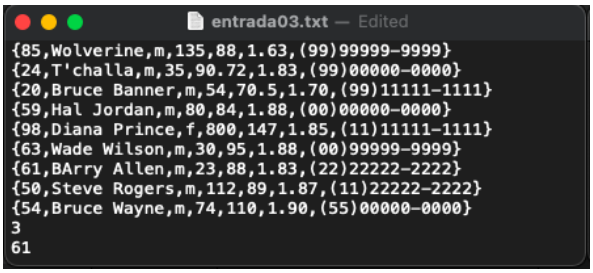
(b) Exemplo de arquivo de saída com os registros impressos em ordem crescente de código.



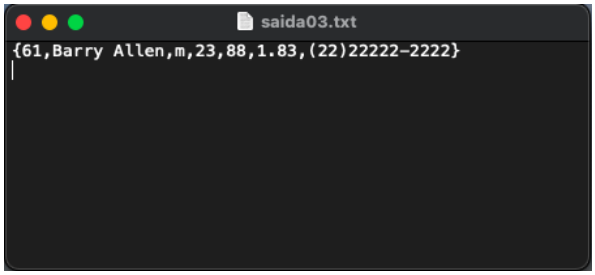
(c) Exemplo de arquivo de entrada para impressão dos registros em ordem decrescente de código.



(d) Exemplo de arquivo de saída com os registros impressos em ordem decrescente de código.



(e) Exemplo de arquivo de entrada para consulta de um registro em específico.



(f) Exemplo de arquivo de saída com impressão do registro encontrado.

Figura 2: Valores de entrada e correspondentes arquivos de saída gerado pelo programa.

- etc.

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

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. o código compila e executa;
4. uso de `argc` e `argv` para controle dos arquivos de teste;
5. completude da implementação (tudo que foi solicitado é entregue usando as estruturas corretamente);
6. implementar a entrada dos dados via leitura do arquivo texto;
7. implementação correta da estrutura necessária (lista duplamente encadeada com nó sentinela/cabeça);
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, etc);
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:

```
ED1-<ANO>-<SEMESTRE>-AT03-Clinica-<NOME>.c
```

Exemplo:

```
ED1-2022-1-AT03-Clinica-RafaelMantovani.c
```

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

5 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] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.

[2] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.

[3] Adam Drozdek. Estrutura De Dados E Algoritmos Em C++. Cengage, 2010.