

# Atividade Prática 03

## Manipulação de Listas

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados 1 - EDCO3A  
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, além de professor de computação, é também médico e frequentemente atende pacientes em sua clínica particular. Porém, a clínica do professor M tem tido muita dificuldade em organizar os registros de seus pacientes. Sem tempo para tentar resolver sozinho o problema, o professor tem procurado um programador que consiga colocar a "ordem na casa" e gerar um sistema que o ajude a obter as informações dos pacientes.

Você é esse(a) programador(a)! 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** por meio de **um código único do paciente**. Esse sistema será capaz de realizar algumas operações:

1. consultar se um registro está contido no sistema (lista). Imprimir se existir, caso contrário indicar que não foi encontrado;
2. imprimir todos os registros em ordem crescente;

3. imprimir todos os registros em ordem decrescente.

Além do código, cada paciente também possui informações como: nome, sexo (m ou f), peso e altura. Claro, que todas essas informações precisam ser impressas/mostradas pelas operações do sistema.

## 2 Listas duplamente encadeada

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:

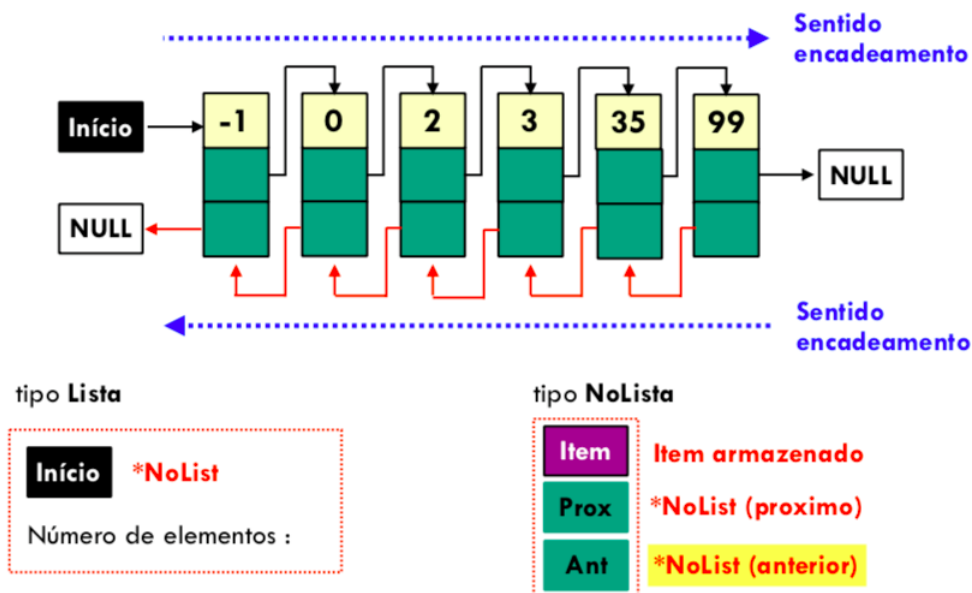


Figura 1: Diagrama representativo de uma lista duplamente encadeada com 6 elementos. No diagrama é possível também ver uma representação gráfica dos tipos abstratos de dados envolvidos em sua codificação.

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

Use as implementações das estruturas já desenvolvidas em sala para criar sua implementação de lista duplamente encadeada.

## 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, peso e altura. 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**:

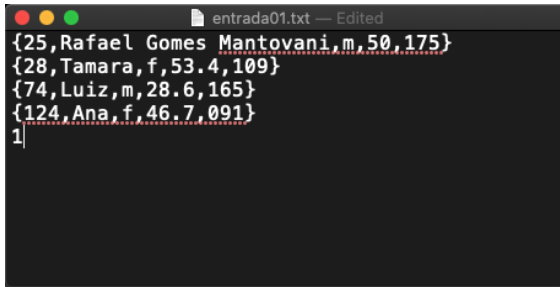
```
./teste entrada.txt saida.txt
```

## 4 Orientações gerais

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

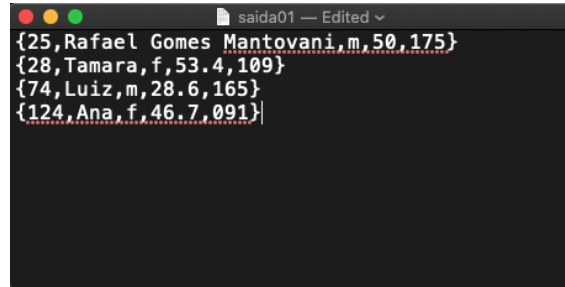
- problemas nas aberturas dos arquivos de entrada e saída;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado (opções inválidas para uso da lista);
- etc.

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



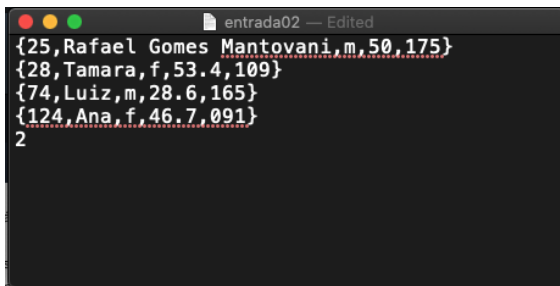
```
entrada01.txt — Edited
{25,Rafael Gomes Mantovani,m,50,175}
{28,Tamara,f,53.4,109}
{74,Luiz,m,28.6,165}
{124,Ana,f,46.7,091}
1
```

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



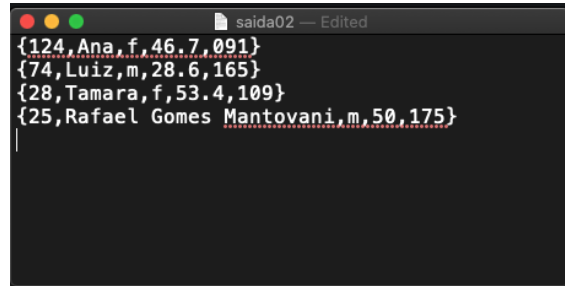
```
saida01 — Edited
{25,Rafael Gomes Mantovani,m,50,175}
{28,Tamara,f,53.4,109}
{74,Luiz,m,28.6,165}
{124,Ana,f,46.7,091}
```

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



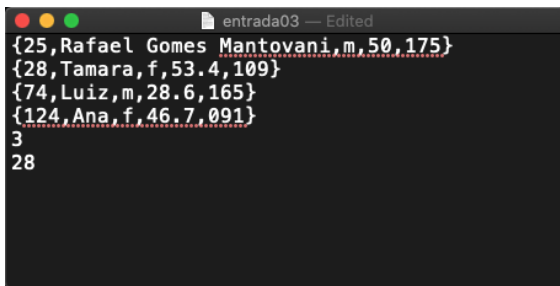
```
entrada02 — Edited
{25,Rafael Gomes Mantovani,m,50,175}
{28,Tamara,f,53.4,109}
{74,Luiz,m,28.6,165}
{124,Ana,f,46.7,091}
2
```

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



```
saida02 — Edited
{124,Ana,f,46.7,091}
{74,Luiz,m,28.6,165}
{28,Tamara,f,53.4,109}
{25,Rafael Gomes Mantovani,m,50,175}
```

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



```
entrada03 — Edited
{25,Rafael Gomes Mantovani,m,50,175}
{28,Tamara,f,53.4,109}
{74,Luiz,m,28.6,165}
{124,Ana,f,46.7,091}
3
28
```

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



```
saida03 — Edited
{28,Tamara,f,53.4,109}
```

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

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

## 4.1 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 da estrutura necessária (lista duplamente encadeada);
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:

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

Exemplo:

ED1-2021-2-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](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.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.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 - 3<sup>a</sup> 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.