

## Trabalho de Programação

### Sistema de Gerenciamento de Dados de Pacientes – Parte II

Valor: 25 pontos

Deadline: 01 de julho de 2025

*Prof. Thiago M. Paixão*  
**thiago.paixao@ifes.edu.br**

## 1 Objetivo

O trabalho prático de programação consiste em implementar um sistema (**simplificado**) de gerenciamento de dados de pacientes de uma clínica em linguagem C. Os dados são armazenados (persistidos) em um arquivo texto CSV e carregados em uma estrutura de dados específica (memória) para execução do sistema. O sistema deve permitir cadastro de pacientes, consulta, atualização e exclusão de registros.

Para esta entrega específica (Parte II), será entregue um sistema completo baseado no TAD *LinkedList* (lista encadeada). As principais competências abordadas nesse trabalho incluem:

- Uso e implementação filas.
- Alocação dinâmica de memória.
- Modularização e TADs.
- Manipulação de arquivos.
- Documentação da solução.

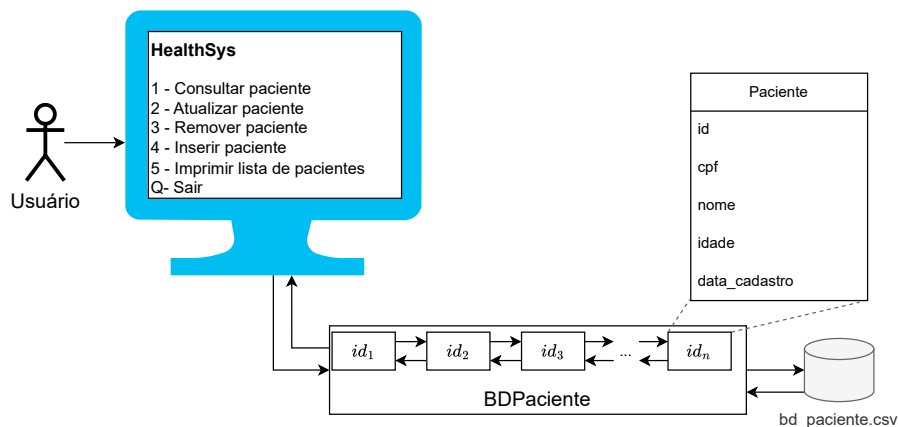


Figura 1: Sistema de gerenciamento de dados de pacientes.

## 2 O Sistema

### 2.1 Comportamento básico

O comportamento básico do sistema é ilustrado na Figura 1. Ao rodar o sistema, o usuário acessa um menu com 5 opções conforme indicado na figura. As opções 1 e 5 são funcionalidades do sistema

(detalhes na Seção 2.3) executadas quando o usuário digita as teclas correspondentes. A opção Q permite que o usuário saia do sistema.

Para este trabalho, simularemos um banco de dados de tabela única que persiste os dados em um arquivo CSV. Ao iniciar o sistema, o banco de dados será carregado do arquivo **bd\_paciente.csv** para uma estrutura de dados baseada em lista encadeada que armazena o banco em memória (**BDPaciente**). O banco em memória funciona como um espelho do arquivo CSV até o ponto em que o estado do banco em memória é alterado por uma requisição do usuário (atualização ou remoção).

## 2.2 Banco de dados

### 2.2.1 Estrutura em arquivo

A estrutura do banco de dados em arquivo (**bd\_paciente.csv**) é definida por uma tabela única que contém informações relevantes sobre os pacientes. Cada entrada na tabela representa um registro de paciente com os seguintes campos:

- **ID**: Um identificador único para cada paciente, garantindo que não existam registros duplicados.
- **CPF**: Um número de registro que identifica o contribuinte no sistema da Receita Federal, utilizado no sistema apenas para efeitos de identificação.
- **Nome**: O nome completo do paciente, fornecendo uma via de reconhecimento.
- **Idade**: A idade do paciente em anos.
- **Data\_Cadastro**: Data em que o usuário foi cadastrado no sistema no formato **AAAA-MM-DD**.

ID	CPF	Nome	Idade	Data_Cadastro
1	123.456.789-09	João Silva	45	2024-12-01
2	987.654.321-00	Maria Oliveira	30	2024-12-02
3	456.789.123-64	Carlos Pereira	65	2024-12-03
4	321.987.654-46	Ana Souza	29	2024-12-04
5	654.321.987-46	Pedro Almeida	50	2024-12-05
6	354.624.978-06	Maria Madalena	50	2024-07-10

Figura 2: Exemplo de banco de dados em arquivo (CSV).

### 2.2.2 TAD BDPaciente

O TAD **BDPaciente** é uma abstração para o armazenamento e a manipulação dos dados dos pacientes. Em vez de manipular diretamente o arquivo do banco de dados, o usuário pode interagir com uma interface que oferece funções específicas para manipulação de dados: consulta (busca), listagem, inserção, atualização e remoção de registros.

Ao iniciar a execução do programa, os registros do arquivo CSV são carregados para a estrutura de dados. É de sua responsabilidade formular o TAD de modo que a estrutura de dados seja (des)alocada e manipulada apenas por meio de funções que abstraem os detalhes de implementação. Para isso, é aconselhável verificar os exemplos de coleções (ex. listas encadeadas) vistos no curso.

## 2.3 Funcionalidades

A seguir, são descritos requisitos mínimos de funcionalidades do sistema. As descrições são intencionalmente fornecidas em um nível mais geral, cabendo ao estudante decidir pela melhor forma de implementação considerando eficiência e usabilidade.

**Consultar paciente** A funcionalidade de consulta permite buscar as informações de um paciente utilizando **Nome** ou **CPF**. Baseado na informação solicitada, deve ser impresso o registro completo dos pacientes de acordo com os campos definidos na Figura 2. Caso o paciente não seja encontrado, o sistema deve informar o usuário com uma mensagem de erro. A busca de pacientes será baseada em prefixo. Isto é, se o usuário fornecer apenas uma parte do nome ou CPF, o sistema deve retornar todos os registros que correspondam ao prefixo informado. A Figura 3 ilustra um possível fluxo de execução desta funcionalidade.

```
[Sistema]
Escolha o modo de consulta:
1 - Por nome
2 - Por CPF
3 - Retornar ao menu principal

[Usuário]
1

[Sistema]
Digite o nome:

[Usuário]
Mari

[Sistema]
ID      CPF      Nome      Idade      Data_Cadastro
2      987.654.321-00  Maria Oliveira  30      2024-12-02
6      354.624.978-06  Maria Madalena  50      2024-07-10
```

Figura 3: Simulando a consulta de pacientes.

**Atualizar paciente** A funcionalidade de atualização permite modificar os dados de um paciente existente no sistema. Para atualizar os dados de um paciente, é necessário inicialmente localizar um único registro de interesse. Dessa forma, a rotina de atualização deve primeiro invocar a rotina de consulta. Uma vez localizado, o registro pode ter alterado um ou mais campos (**CPF**, **Nome**, **Idade** ou **Data\_Cadastro**). A Figura 4 ilustra um possível fluxo de execução desta funcionalidade. No exemplo em questão, apenas o **CPF** está sendo alterado. O **ID** do registro deve ser gerado de maneira automática.

**Remover paciente** A funcionalidade de remoção permite excluir um paciente existente no sistema de forma definitiva. Para garantir que o registro correto seja excluído, o sistema deve solicitar inicialmente a execução de uma consulta para localizar o paciente desejado. Após a seleção do paciente, o sistema deve solicitar a confirmação do usuário antes de realizar a exclusão. A Figura 5 ilustra um possível fluxo de execução desta funcionalidade.

**Inserir paciente** A funcionalidade de inserção permite adicionar um novo paciente ao sistema. Para cadastrar o paciente, o sistema deve solicitar ao usuário que insira todos os campos obrigatórios, como **CPF**, **Nome**, **Idade** e **Data\_Cadastro**. Após o preenchimento, o sistema deve validar os dados fornecidos, como a formatação do **CPF** e o preenchimento dos demais campos. Uma vez validados, os dados devem ser salvos no sistema, e o novo registro deve ser exibido ao usuário como confirmação. A Figura 6 ilustra um possível fluxo de execução desta funcionalidade.

...

[Sistema]

ID	CPF	Nome	Idade	Data_Cadastro
2	987.654.321-00	Maria Oliveira	30	2024-12-02
6	354.624.978-06	Maria Madalena	50	2024-07-10

Digite o ID do registro a ser atualizado:

[Usuário]

6

[Sistema]

Digite o novo valor para os campos CPF (apenas dígitos), Nome, Idade e Data\_Cadastro (para manter o valor atual de um campo, digite '-')

[Usuário]

41257369875

-

-

-

[Sistema]

Confirma os novos valores para o registro abaixo? (S/N)

ID	CPF	Nome	Idade	Data_Cadastro
6	412.573.698-75	Maria Madalena	50	2024-07-10

[Usuário]

S

[Sistema]

Registro atualizado com sucesso.

Figura 4: Simulando a atualização de pacientes com confirmação.

**Imprimir lista de pacientes** A funcionalidade de impressão de lista permite exibir todos os registros de pacientes armazenados no sistema. O sistema deve imprimir todos os registros, apresentando-os de maneira organizada, com os campos **ID**, **CPF**, **Nome**, **Idade** e **Data\_Cadastro** de cada paciente. Caso o sistema tenha um grande número de registros, é recomendado que a impressão seja paginada para não sobrecarregar a interface. A Figura 7 ilustra a saída do sistema quando essa funcionalidade é solicitada.

### 3 Requisitos do programa

Neste trabalho, você terá a flexibilidade de implementar/adequar os módulos e os Tipos Abstratos de Dados (TADs) que considerar necessários para a simulação. No entanto, é fundamental que o programa principal esteja implementado no arquivo **main.c**. Além disso, a estrutura do código deve ser modular. É importante que cada módulo tenha uma responsabilidade clara e que a comunicação entre os diferentes componentes do programa ocorra de forma eficiente.

Note que, apesar das sugestões de implementação (fluxo de execução) em alto nível, os detalhes de

```
[Sistema]
ID   CPF           Nome           Idade   Data_Cadastro
2    987.654.321-00  Maria Oliveira  30      2024-12-02
6    412.573.698-75  Maria Madalena  50      2024-07-10
```

Digite o ID do registro a ser removido:

```
[Usuário]
6
```

```
[Sistema]
Tem certeza de que deseja excluir o registro abaixo? (S/N)
```

```
ID   CPF           Nome           Idade   Data_Cadastro
6    412.573.698-75  Maria Madalena  50      2024-07-10
```

```
[Usuário]
S
```

```
[Sistema]
Registro removido com sucesso.
```

Figura 5: Simulando a remoção de pacientes.

```
[Sistema]
Para inserir um novo registro, digite os valores para os campos CPF (apenas
dígitos), Nome, Idade e Data_Cadastro:
```

```
[Usuário]
64037616092
Cristiano Ronaldo
78
2024-10-10
```

```
[Sistema]
Confirma a inserção do registro abaixo? (S/N)
```

```
ID   CPF           Nome           Idade   Data_Cadastro
7    640.376.160-92  Cristiano Ronaldo  78      2024-10-10
```

```
[Sistema]
0 registro foi inserido com sucesso.
```

Figura 6: Simulando a inserção de pacientes.

implementação devem ser decididos por vocês. Você é encorajado a adicionar e/ou modificar funcionalidades de modo a melhorar a experiência do usuário ou otimizar a execução. Não deve, contudo, reduzir a quantidade de funcionalidades já previstas, nem reduzir o escopo do projeto.

[Sistema]				
Imprimindo lista de pacientes...				
ID	CPF	Nome	Idade	Data_Cadastro
1	123.456.789-09	João Silva	45	2024-12-01
2	987.654.321-00	Maria Oliveira	30	2024-12-02
3	456.789.123-64	Carlos Pereira	65	2024-12-03
4	321.987.654-46	Ana Souza	29	2024-12-04
5	654.321.987-46	Pedro Almeida	50	2024-12-05
6	354.624.978-06	Maria Madalena	50	2024-07-10
7	640.376.160-92	Cristiano Ronaldo	78	2024-10-10

Figura 7: Simulando a impressão de lista de pacientes.

## 4 Critérios de avaliação

A nota do trabalho será composta por **25 pontos** atribuídos aos critérios técnicos (1 a 4), multiplicados por fatores que refletem aspectos complementares (5 a 7):

1. Funcionalidades: Até **16 pontos** serão atribuídos à implementação adequada das 5 funcionalidades requeridas (4 pontos por funcionalidade para *atualização*, *remoção* e *inserção*; 2 pontos por funcionalidade para *consulta* e *listagem*).
2. Lógica e organização do código: Até **5 pontos** serão concedidos pela clareza, organização e boas práticas de codificação no projeto. Isso inclui a estruturação adequada do código (modularização), nomes significativos para variáveis e funções, e formatação consistente.
3. Documentação do **README.md**: Até **3 pontos** serão atribuídos à qualidade do arquivo **README.md** presente no repositório. Este arquivo deve ser descritivo e informativo, contendo instruções claras sobre como executar e utilizar o projeto. Deve incluir informações detalhadas sobre a estrutura do repositório, apresentar os principais TADs utilizados e listar as principais decisões de implementação tomadas ao longo do desenvolvimento.
4. Documentação interna do código (comentários): Até **1 ponto** será atribuído à qualidade da documentação incorporada diretamente ao código. Essa documentação deve ser composta por comentários significativos que expliquem a lógica por trás das implementações, facilitando a compreensão do funcionamento do projeto e promovendo a manutenção do código.
5. Apresentação (a ser agendada): A nota de apresentação ( $A \in [0, 1]$ ) será atribuída individualmente aos membros do grupo, refletindo a qualidade da apresentação, bem como a capacidade de explicar e defender o projeto de forma clara e concisa.
6. Robustez: A nota de robustez ( $R \in [0, 1]$ ) será atribuída com base na presença de falhas críticas (por exemplo, falha de segmentação) ou não críticas (por exemplo, *memory leakage*) no sistema.
7. Dias de atraso: Serão contabilizados os dias de atraso ( $D$ ) para efeito de desconto na nota total do trabalho.

A nota final do trabalho será calculada pela equação

$$\text{nota} = \left(1 - \frac{2^D - 1}{31}\right) \times P \times R \times A, \quad (1)$$

onde  $P$  é a soma dos pontos dos critérios 1 a 4. É importante ressaltar que a nota do trabalho será zerada caso o atraso ultrapasse 5 dias.

**Importante:** O programa será testado num ambiente Linux Ubuntu 22.04 com GCC 11. Recomendo FORTEMENTE desenvolver e testar nesse ambiente.

## 5 O que entregar?

1. Um link para um repositório .git com o arquivo **bd\_paciente.csv** e código-fonte do projeto: **Makefile** e arquivos **.c** e **.h**.
2. A documentação/relatório será feita no arquivo **README.md** do repositório e deverá explicar o passo-a-passo para executar o programa, os principais TADs e as principais decisões de implementação.

Bom trabalho!