UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE (UFRN)
INSTITUTO METRÓPOLE DIGITAL (IMD)

BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO (BTI)

Disciplina: Introdução a Técnicas de Programação (ITP) - 2025.2

Unidade 1: Relatório Técnico

Projeto: Prontuário Eletrônico de Pacientes

Aluno: Anderson Pedro do Nascimento

Introdução

O objetivo principal deste projeto foi o desenvolvimento de uma aplicação funcional em linguagem C que permitisse avaliar o aprendizado dos conteúdos iniciais da disciplina de Introdução às Técnicas de Programação. A criação do sistema "Prontuário Eletrônico de Pacientes" serviu como o meio prático para aplicar e solidificar o entendimento de conceitos fundamentais, como a manipulação de variáveis e vetores, o controle de fluxo com laços e condicionais, e a modularização do código através de funções, atuando como a principal forma avaliativa para a Unidade 1.

Problemática

O projeto visa solucionar, em pequena escala, o problema da organização de registros de pacientes. Atualmente, muitos profissionais de saúde ainda utilizam prontuários de papel ou anotações manuais. Essa abordagem, por ser manual e descentralizada, pode colaborar para uma gestão inadequada do histórico do paciente, levando à perda de dados e à dificuldade de acesso rápido às informações. Este projeto oferece uma alternativa digital simples, permitindo que os dados de cada paciente sejam cadastrados, consultados e listados de maneira ágil, centralizando as informações e, consequentemente, facilitando o seu acesso.

Justificativa

A escolha do tema "Prontuário Eletrônico" foi motivada por sua capacidade de exigir a aplicação integrada de todos os conceitos da Unidade 1. O gerenciamento de múltiplos pacientes e seus respectivos dados tornou o uso de vetores, funções e laços de repetição uma necessidade essencial para a solução, representando um desafio completo e ideal para a avaliação do aprendizado.

Análise Técnica

Metodologia: Ferramentas Utilizadas

- Linguagem de Programação: C
- Compilador**:** GCC (GNU Compiler Collection), utilizado para compilar o código-fonte em um programa executável.
- Editor de Código: Visual Studio Code, utilizado para a escrita e edição do código-fonte.
- Sistema de Versionamento: Git e GitHub, para controle de versão e publicação do projeto.

Aplicação dos Conceitos da U1

Para construir o Prontuário Eletrônico, a primeira grande decisão foi como armazenar as informações dos pacientes. A solução encontrada e que foi apresentada na última aula da matéria foi aplicar o conceito de vetores, onde três vetores (nomes, idades e sintomas) trabalham em conjunto, usando um mesmo índice para conectar os dados de um único paciente, simulando um pequeno banco de dados que funciona enquanto o programa está rodando. Com os dados tendo um lugar para ficar, o próximo passo foi criar a experiência interativa. Para isso, foi usado uma estrutura de repetição, um laço `do-while` na função `main`, que garante que o menu principal continue aparecendo para o usuário, permitindo que ele realize várias operações em sequência. Dentro desse loop as estruturas condicionais foram fundamentais. Foi usado o switch-case para gerar um pequeno menu numérico onde através dele o usuário conseguiria navegar permitindo ele selecionar o que desejar mediante a oferta de cadastrar, listar, ou verificar detalhes etc. Os comandos if-else foram usados para fazer validações como a checagem do limite de pacientes ou se já existia um paciente cadastrado no sistema. A organização do código foi feita através de funções. Cada parte do trabalho foi delegada a uma função: **`exibirMenu`** desenha a tela, **`cadastrarPaciente`** cuida de todo o processo de entrada de dados, e **`listarPacientes`** percorre os vetores usando um laço **`for`** para mostrar os resultados.

Implementação e reflexão

O processo de desenvolvimento, embora linear, apresentou desafios técnicos significativos que foram cruciais para o aprendizado. O primeiro grande obstáculo foi o correto manuseio do buffer de entrada do teclado em C. Notou-se que, após a leitura de um dado numérico com a função scanf, a subsequente captura de um texto com fgets era ignorada pelo programa. Um segundo desafio foram os erros de compilação, especificamente o erro de "implicit declaration of function", que impedia o programa de ser compilado devido à ordem em que as funções eram declaradas e chamadas. Finalmente, o desafio mais crítico foi um bug de lógica no armazenamento de dados, onde cada novo paciente cadastrado sobrescrevia o anterior, e a função de listagem não encontrava nenhum registro, mesmo após cadastros aparentemente bem-sucedidos.

Cada desafio exigiu uma solução específica. O problema do buffer do teclado foi superado com a implementação sistemática da função getchar() imediatamente após cada chamada à scanf que lia um número. Essa solução "limpa" o caractere de nova linha (\n) residual do buffer, permitindo que a função fgets funcione como esperado. Os erros de compilação foram resolvidos através da introdução de protótipos de funções no cabeçalho do arquivo, declarando a assinatura de cada função antes de serem chamadas pela main. Por fim, o bug de lógica de sobrescrita foi solucionado após uma depuração cuidadosa do fluxo de dados, identificando que a instrução totalPacientes++ a linha mais importante para atualizar o estado do sistema estava faltando na função cadastrarPaciente.

O código-fonte foi intencionalmente estruturado em três seções lógicas para ajudar na manutenção. A primeira seção, no topo do arquivo, contém as configurações globais: inclusão de bibliotecas, definição de constantes, declaração dos protótipos e das variáveis de dados (os vetores). A segunda seção é a função main, que foi mantida enxuta, responsável apenas por controlar o loop principal e delegar tarefas. A terceira e última seção contém a implementação detalhada das funções (exibirMenu, cadastrarPaciente, exibirDetalhePaciente() etc.).

A depuração de erros reais durante o desenvolvimento foi uma experiência de aprendizado inestimável, abrangendo desde problemas de sintaxe e compilação até bugs de lógica. Este processo reforçou a necessidade de extrema atenção aos detalhes que a linguagem C exige.

Como principal melhoria futura, a evolução natural do sistema seria a implementação de uma conexão com um banco de dados. Isso permitiria a persistência real dos dados, superando a limitação atual de armazenamento apenas durante a execução do código e transformando a aplicação em uma ferramenta muito mais robusta e útil.

Perguntas orientadoras

1. Quais conceitos da Unidade 1 foram aplicados e onde?
Vetores paralelos foram usados para armazenar os dados dos pacientes. Um laço do-while controlou o menu principal e um for a listagem. O switch-case gerenciou as opções do menu, if-else validou dados, e a modularização em Funções foi usada para organizar todo o código.

2. Como a organização em funções facilita a manutenção do código?
A organização em funções facilita a manutenção ao isolar as responsabilidades. Se um bug ocorrer no cadastro, a correção é feita apenas na função cadastrarPaciente, sem risco de afetar outras partes do sistema, tornando o código mais fácil de depurar e expandir.

3. Quais foram os principais desafios na implementação das estruturas de repetição?
O principal desafio com o laço `do-while` do menu foi garantir que o estado da aplicação (os dados dos pacientes) fosse mantido entre as iterações. Um bug inicial, onde o contador de pacientes não era incrementado, fazia o estado "resetar" a cada volta, o que foi corrigido para garantir a consistência dos dados.

4. Como os vetores foram utilizados para resolver o problema proposto?
Os vetores foram a solução para simular um banco de dados em memória. A técnica de vetores foi utilizada, onde o mesmo índice em três vetores diferentes (`nomes`, `idades`, `sintomas`).

5. Que melhorias poderiam ser implementadas nas próximas unidades?
A principal melhoria seria a implementação de persistência de dados. Atualmente, os dados são perdidos ao fechar o programa. Nas próximas unidades, o sistema poderia ser aprimorado para salvar as informações em um banco de dados real.