Relatório Técnico - Unidade 1

Disciplina: Introdução às Técnicas de Programação
Projeto: Duarte Cadastro de Alunos
Aluno: Felipe Augusto de Lima Duarte
Matrícula: 20250025417

1. Introdução e Contexto

O projeto implementa um Sistema CRUD de Gerenciamento Escolar em linguagem C. O objetivo é permitir o Cadastro, Consulta, Atualização e Exclusão de registros de alunos, utilizando uma interface de linha de comando (CLI).

O projeto resolve o problema do gerenciamento básico de cadastros de alunos, controlando de forma segura os dados (nome, idade, matrícula, série) e permitindo que os registros desativados sejam preservados no sistema, garantindo o histórico e a possibilidade de reativação imediata.

A escolha desse projeto se justifica porque o desenvolvimento de um sistema CRUD é uma forma eficaz de englobar os conceitos fundamentais da Unidade 1 (Funções, Vetores, Condicionais e Laços de Repetição). Por já ter trabalho em um projeto WEB tive contato com o CRUD e pude desenvolver e aprender mais ainda sobre o tema, apesar de não utilizar um sistema de armazenamento persistente como um banco de dados, impossibilitando que os cadastros fiquem guardados mesmo após fechar o terminal, tentei simular esse armazenamento utilizando vetores e tive um resultado positivo.

2. Análise Técnica

2.1 Metodologia e Ferramentas

- **Compilador:** MinGW - GCC
- **Editor:** VSCode
- **Execução:** Terminal do VSCode

2.2 Conceitos Aplicados

Estruturas Condicionais

As condicionais (`if`, `else`, `switch`) foram usadas extensivamente para o controle de fluxo e validação:

- **Validação de Entrada:** Usadas em todas as funções para verificar a validade das opções do menu e para checar se o índice de consulta ou alteração está dentro dos limites do vetor.
- **Filtro de Soft Delete:** Aplicadas em `consultar_alunos` e `alterar_aluno` para verificar se `lista[i].ativo == 1` antes de listar ou modificar um registro.

Lógica das Estruturas de Repetição Implementadas

Os laços de repetição garantem a interatividade e o processamento de dados:

- **Menu Principal (`do-while`):** Utilizado na função `main` para manter o sistema rodando até que o usuário escolha a opção de sair.
- **Submenus de Ação (`do-while`):** Usado em `consultar_alunos`, `alterar_aluno` e `deletar_aluno` para que o usuário possa realizar múltiplas ações (consultas, alterações) sem sair da função.
- **Iteração de Dados (`for`):** Usado em `consultar_alunos` e na função `cadastros_ativos` para percorrer o vetor de alunos de índice `0` até `total_alunos - 1`.

Aplicação dos Vetores

Os vetores foram usados para a coleção de dados e para o armazenamento de strings:

- **Lista Principal (`Aluno lista_alunos[100]`):** Armazena todos os registros do tipo `struct Aluno`.
- **Strings:** Vetores de caracteres (`char[]`) foram utilizados para armazenar o **nome** e a **matrícula**.

Organização e Função das Funções Criadas

O projeto está dividido em **7 funções** principais:

1. **`main`:** Gerencia o fluxo do menu principal e exibe o status de alunos ativos/registros.
2. **`limpar_buffer`:** Função auxiliar para tratar o caractere `\n` após o uso de `scanf`.
3. **`cadastros_ativos`:** Função auxiliar que calcula dinamicamente o número de alunos com o status `ativo == 1`.
4. **`cadastro_aluno` (Create):** Insere um novo aluno e define seu status inicial como `ativo = 1`.
5. **`consultar_alunos` (Read):** Exibe registros dos alunos, mostrando suas informações cadastradas.
6. **`alterar_aluno` (Update):** Modifica campos de um aluno ativo, sendo possível modificar qualquer atributo de um aluno.
7. **`deletar_aluno` (Delete/Reativar):** Altera o campo `ativo` entre 0 (inativar) e 1 (reativar).

2.3 Estrutura de Dados

- **Struct Aluno:** Estrutura principal que agrupa dados de diferentes tipos para representar um único cadastro (nome, idade, série, matrícula e o campo ativo).
- **Strings (char[ ]):** Usadas para armazenar o nome e a matrícula.
- **Inteiro (int):** Usado para a idade, série escolar, o campo de status, as variáveis de opções do menus e a variável contadora de alunos.

3. Implementação e Reflexão

3.1 Dificuldades Encontradas

- A dificuldade inicial do projeto foi na utilização de ponteiros, por não se tratar de um assunto pertinente à unidade, precisei procurar materiais alternativos. Porém, apesar de ser um conteúdo de difícil assimilação, a partir da implantação no projeto foi possível ter uma melhor compreensão do assunto.

- Outra dificuldade encontrada foi na manipulação de entradas, pelo tipo Aluno ser composto por tipos de dados diferentes, foi necessário que cada um recebesse um tratamento diferente na função de cadastro, por exemplo o nome, que por conter espaços em branco teve que ser lido pelo “fgets”. Outro problema foi lidar com o buffer de entrada, que deixa sempre um “\n” residual.

- Por último, um dos problemas que tive foi na adaptação a linguagem C. Aprendi a programar utilizando a linguagem Python, que é uma linguagem de alto nível e menos “verbosa”, diferente do C. Ao fazer um projeto semelhante na matéria de Pensamento Computacional utilizei a estrutura de dicionários, que abriga um par de chave e valor, entretanto, tal estrutura não existe no C, sendo necessário utilizar a estrutura struct, que, em termos práticos, cria um novo tipo de dados que vai conter alguns atributos, quase como uma classe, que se diferencia por não conter métodos.

3.2 Soluções Implementadas

- Aprendi o conceito básico de ponteiros para poder aplicar no projeto, foi desafiador devido ao tempo curto, entretanto foi bastante enriquecedor também.

- Para lidar com a manipulação de entradas eu precisei pesquisar até desenvolver uma função que “limpa” o buffer de entrada do scanf, e utilizar a função strcspn que localiza e “limpa” o “\n” deixado no final da string pelo “fgets”.

- Por fim, por ter uma lógica de programação consolidada, não foi tão difícil acostumar com a sintaxe da linguagem C. Foram necessárias pesquisas e estudos para entender melhor algumas estruturas do C, como o struct que veio a suprir minhas necessidades, combinado aos vetores, de uma estrutura como os dicionários do Python.

3.3 Organização do Código

O código foi estruturado com base em funções específicas para cada funcionalidade do acrônimo CRUD, dessa forma foi possível realizar testes individualizados, e em caso de erros, a correção e a manutenção se dava de forma simples, bastando depurar a função utilizada nos testes. Entretanto, ainda há um problema a ser resolvido, em breve irei modularizar o projeto, transferindo as funções para arquivos separados, tal ação vai facilitar ainda mais a manutenção do código, trazendo também uma melhor legibilidade de cada parte do programa.

3.4 Conclusão

Foi possível aplicar todos os conceitos estudados durante a primeira unidade do curso, sendo eles: Condicionais, laços de repetição, funções e vetores. Aprendi bastante sobre programação ao decorrer do projeto e das listas, me sinto cada dia mais capaz de aplicar meus conhecimentos de forma prática no que for necessário. A ideia de não trazer prova, mas sim um projeto, foi inicialmente desafiadora, entretanto extremamente benéfica para a consolidação do aprendizado, o aprendizado foi realmente posto à prova, e mesmo com a correria do semestre foi possível desenvolver, até o momento, uma ideia interessante.

3.5 Possíveis Melhorias

- **Tratamento de erros:** O primeiro ponto a ser melhorado é o tratamento de erros de entradas, por ser uma linguagem de baixo nível o tratamento de erros em C é um pouco complicado, diferente do Python que possui uma estrutura como o try…except. Porém alguns pontos ainda podem ser melhorados nesse sentido.

- **Função de atribuição de notas e matérias:** Os alunos além de possuírem os atributos iniciais de cadastro também irão poder ser matriculados em matérias e receber notas dentro dessas matérias. Pretendo criar funções para atribuir novas matérias aos alunos, assim como atribuir, ler, alterar e deletar as notas dessas matérias.

- **Atribuição de matrículas automático:** Dentro do sistema de cadastro que criei as matrículas possuem uma lógica, entretanto a sua atribuição ainda é manual, pretendo criar também uma função que vai atribuir automaticamente a matrícula a cada aluno com base nos dados inseridos pelo usuário.

Perguntas Orientadoras

Quais conceitos da Unidade 1 foram aplicados e onde?

- **Variáveis:** As variáveis estão presentes por todo o código, principalmente nos menus de opção que vão armazenar a opção escolhida pelo usuário, também está presente na struct Aluno, que funciona como um tipo e recebe diferentes tipos de dados, como strings e inteiros.
- **Condicionais:** As condicionais também estão presentes por todo o código, principalmente na verificação de condições que vão quebrar o código, como nos menus de opção ou quando o usuário escolhe uma opção além do cadastro e não há alunos ativos, mostrando uma mensagem de aviso ao usuário.
- **Laços de Repetição:** É uma das principais estruturas utilizadas no código, está presente em todos os menus como um loop que na escolha de opções que só vai ser encerrado com o comando correto. Utilizo também na listagem dos alunos a estrutura “for”, que vai iterar sobre todos os elementos do vetor que contém os alunos, sempre mostrando os alunos com cadastro ativo, trabalhando em conjunto com as condicionais.
- **Funções e Vetores:** As funções são essenciais no código pois ajudam na fácil legibilidade e a manutenção do programa, foram criadas funções principais seguindo o acrônimo CRUD, e funções secundárias com funcionalidades mais pontuais, como a limpeza do buffer de entrada e a contagem de alunos ativos. Já os vetores são também uma das principais partes do código, como o armazenamento persistente, como um banco de dados, não é utilizado, ficou a cargo de um vetor servir como esse armazenamento, eles guarda dados do tipo Aluno, criado com struct, armazenando todos os cadastros, servindo para listagem, alteração e exclusão lógica no programa.

Como a organização em funções facilita a manutenção do código? A divisão em funções é essencial para a manutenção do código, com elas é possível testar individualmente cada funcionalidade e a partir dos testes encontrar erros e problemas a serem solucionados, de forma pontual, basta apenas ir na função e consertar o que for preciso. Outro ponto importante no uso das funções é poder reaproveitar elas em diversos pontos do programa, como na limpeza do buffer de entrada, sem funções seria necessário criar um trecho de código para solucionar a limpeza após cada “scanf”, entretanto, utilizando uma única função se torna muito mais simples de resolver o problema, pois um único trecho de código vai poder ser aplicado em diversos pontos do programa.

Quais foram os principais desafios na implementação das estruturas de repetição? Com toda certeza o maior desafio na implementação das estruturas de repetição foi na função de deletar, não existe uma forma de deletar elementos de vetores por índice no C, é preciso utilizar uma série de artimanhas com as estruturas de repetição para realizar tal feito, o problema então foi resolvido utilizando o conceito de soft delete, que agora não mais apaga o dado do vetor (hard delete), mas apenas altera o campo de ativo do tipo Aluno. Ademais, devido a familiaridade anterior com lógica de programação, não houveram mais dificuldades encontradas na implementação de tais estruturas.

Como os vetores foram utilizados para resolver o problema proposto? Os vetores foram parte essencial do código, servindo como 'banco de dados", armazenando os dados do tipo Aluno, que contém atributos de um aluno, como nome, idade, série escolar e matrícula.

Que melhorias poderiam ser implementadas nas próximas unidades?

- **Tratamento de erros:** O primeiro ponto a ser melhorado é o tratamento de erros de entradas, por ser uma linguagem de baixo nível o tratamento de erros em C é um pouco complicado, diferente do Python que possui uma estrutura como o try…except. Porém alguns pontos ainda podem ser melhorados nesse sentido.

- **Função de atribuição de notas e matérias:** Os alunos além de possuírem os atributos iniciais de cadastro também irão poder ser matriculados em matérias e receber notas dentro dessas matérias. Pretendo criar funções para atribuir novas matérias aos alunos, assim como atribuir, ler, alterar e deletar as notas dessas matérias.

- **Atribuição de matrículas automático:** Dentro do sistema de cadastro que criei as matrículas possuem uma lógica, entretanto a sua atribuição ainda é manual, pretendo criar também uma função que vai atribuir automaticamente a matrícula a cada aluno com base nos dados inseridos pelo usuário.