RELATÓRIO - PROJETO PRÁTICO

GAC124 - INTRODUÇÃO AOS ALGORITMOS

Daniel Reis Araújo, 202510364.
Gabriel Lima Leandro, 202510913.
João Vitor Rezende, 202510356.

Lavras, junho de 2025.

1. INTRODUÇÃO:

Este relatório detalha o projeto prático de introdução a algoritmos, cujo foco é a implementação de um sistema de cadastro de carros. O objetivo primordial deste trabalho é aplicar e consolidar conceitos fundamentais de algoritmos e estruturas de dados, por meio do desenvolvimento de funcionalidades essenciais como o registro, a busca e a alteração de dados de veículos. Um aspecto crucial deste projeto é a utilização de arquivos binários para o armazenamento persistente das informações, o que representa um desafio prático na gestão eficiente de dados.

2. ESTRUTURA E LÓGICA DO PROGRAMA

O sistema de gerenciamento de carros foi desenvolvido utilizando C++ e se baseia em uma combinação de estruturas de dados e algoritmos para manipular as informações dos veículos. A persistência dos dados é garantida através do armazenamento em arquivos binários e CSV.

2.1. Estruturas de Dados

A principal estrutura de dados utilizada é um struct chamado carros, que representa um registro individual de um veículo. Ele contém os seguintes campos:

identificador (int): Um número único para identificar cada carro.
ano_lancamento (int): O ano em que o carro foi lançado.
modelo (char[100]): Uma cadeia de caracteres para armazenar o modelo do carro.
marca (char[100]): Uma cadeia de caracteres para armazenar a marca do carro.
descricao (char[500]): Uma cadeia de caracteres mais longa para uma descrição detalhada do veículo.
valor (float): O preço do carro.

Essencialmente, o programa opera com um vetor dinâmico de carros, que é uma coleção desses registros em memória. Esse vetor pode crescer conforme a necessidade, garantindo que o programa possa lidar com um número variável de carros sem esgotar a memória previamente alocada.

2.2. Lógica do Programa (Alto Nível)

O fluxo do programa pode ser dividido nas seguintes etapas e funcionalidades:

2.2.1. Carregamento e Persistência de Dados

Escolha da Fonte de Dados: Ao iniciar, o usuário decide se deseja carregar os dados de um arquivo binário (.bin) ou de um arquivo CSV (.csv).

A leitura de arquivos binários (ler_bin) é otimizada para performance, lendo blocos de dados brutos diretamente para a estrutura carros. Há um mecanismo de redimensionamento dinâmico do vetor embutido para acomodar um número crescente de registros.

A leitura de arquivos CSV (ler_csv) é projetada para interpretar dados textuais delimitados por vírgulas, também com capacidade de redimensionar o vetor conforme novos registros são lidos.
Salvamento de Dados: Ao sair do programa, o usuário é questionado se deseja salvar as alterações. Se confirmado, os dados atualmente em memória (no vetor carros) são gravados de volta em ambos os formatos de arquivo (escrever_bin e escrever_csv), garantindo a persistência das modificações.

2.2.2. Funcionalidades Principais

Após o carregamento dos dados, o usuário entra em um menu de opções para interagir com o sistema:

Busca de Dados:

Permite a busca por identificador, marca, modelo ou ano de lançamento.

Para buscas por identificador, marca e ano, o programa utiliza o algoritmo de ordenação Quicksort no campo relevante antes de aplicar uma Busca Binária. Isso garante eficiência na recuperação de dados, especialmente em grandes volumes.

A busca por marca e ano é mais elaborada, pois pode retornar múltiplos resultados. A Busca Binária é adaptada para encontrar a primeira e a última ocorrência de um valor, exibindo todos os carros dentro desse intervalo.
Filtro por Valor:

Permite que o usuário insira uma faixa de preço (mínimo e máximo).

O algoritmo de ordenação Insertion Sort é aplicado ao campo valor para facilitar a iteração e exibição dos carros que se encaixam na faixa especificada.
Ordenação de Dados:

Oferece opções para ordenar os dados do vetor em memória com base em diferentes critérios: identificador,ano, valor ou marca (em ordem alfabética).

Para identificador, ano e marca, o algoritmo Quicksort é empregado devido à sua eficiência para grandes conjuntos de dados.

Para o valor, o Insertion Sort é utilizado.
Visualização de Dados:

Permite que o usuário exiba todos os registros de carros ou apenas uma parte, especificando um intervalo.
Adição de Novo Carro:

Permite ao usuário inserir novos dados de carro no sistema. O programa lida com o redimensionamento automático do vetor se a capacidade atual for excedida. O novo carro recebe um identificador sequencial baseado no último ID existente.

2.2.3. Funções Auxiliares

tolowerString: Uma função utilitária para converter strings para minúsculas, garantindo que as comparações de texto (como na busca por marca) sejam insensíveis a maiúsculas e minúsculas.
Procedimentos de Partição (particiona_id, particiona_marca,particiona_ano): Componentes essenciais do algoritmo Quicksort, responsáveis por rearranjar o vetor em torno de um pivô.

Em resumo, o programa demonstra a aplicação de estruturas de dados (vetores dinâmicos, structs) e algoritmos de busca e ordenação (Quicksort, Insertion Sort, Busca Binária) para construir um sistema funcional de gerenciamento de informações, com a capacidade de persistir e recuperar dados de arquivos.

3. ORDEM DE ARMAZENAMENTO DE DADOS NOS ARQUIVOS

Durante a execução do programa, os dados dos veículos podem ser ordenados temporariamente por diferentes critérios, como identificador, marca, ano de lançamento ou valor, de acordo com a operação solicitada pelo usuário. No entanto, o programa foi estruturado de forma que, antes de realizar a inserção de novos registros ou efetuar buscas binárias, o vetor de veículos é sempre reordenado pelo campo de identificador. Essa prática garante a consistência das operações de busca e a correta sequência de identificação dos veículos no sistema.

Ao final da execução, quando o usuário opta por salvar as alterações, os dados são gravados tanto no arquivo binário (Carros.bin) quanto no arquivo CSV (Carros.csv). Como o vetor é reorganizado pelo identificador antes dessas operações, a ordem final dos registros armazenados nos arquivos é crescente em relação ao campo de identificador.
Portanto, conclui-se que:

- Os dados armazenados no arquivo permanecem ordenados de forma crescente pelo identificador do veículo.
- Essa ordenação garante integridade e facilita a leitura sequencial ou a aplicação de buscas futuras, visto que a busca binária é baseada nessa ordenação.

Essa organização foi implementada intencionalmente no projeto para manter o desempenho e a consistência dos dados no sistema.

4. ACERTOS E ERROS:

Durante o desenvolvimento do projeto, um dos principais desafios enfrentados foi a leitura dos dados a partir de arquivos CSV. No início, esse processo se mostrou complexo, especialmente no que diz respeito à integração desses dados com os demais procedimentos do sistema, como a inserção, ordenação e manipulação em arquivos binários. Foram necessárias diversas tentativas até que conseguíssemos estruturar a leitura de forma correta e eficiente, garantindo que os dados extraídos do CSV fossem devidamente interpretados e utilizados pelo programa.

Por outro lado, ao longo das aulas, obtivemos avanços significativos. Com o aprofundamento nos conteúdos abordados, conseguimos compreender e aplicar melhor os métodos de ordenação, e também aprimorar a manipulação de arquivos binários. Isso nos permitiu construir um código mais robusto, com uma estrutura mais sólida e funcional.

5. CONCLUSÃO:

O projeto em C++ demonstrou com sucesso a manipulação de dados utilizando arquivos nos formatos CSV e BIN, aplicando operações como leitura, escrita, adição, remoção, ordenação e busca. Entre os principais resultados, destacam-se: o processamento dinâmico de vetores e cabeçalhos, o controle automático de identificadores, a exclusão eficiente de registros com atualização simultânea dos arquivos, e a implementação de algoritmos de ordenação e busca binária. O sistema também permite filtros e visualizações personalizadas, assegurando escalabilidade e confiabilidade na gestão de dados estruturados. O projeto atingiu seus objetivos e consolidou o domínio dos principais conceitos de arquivos e estruturas em C++.