BÁRBARA OLIVEIRA FONSECA (14B)
CARLOS EDUARDO SILVA SARTO (14A)
THIAGO ANTONIO DE SOUZA (14B)

PROJETO PRÁTICO DE INTRODUÇÃO AOS ALGORITMOS

LAVRAS-MG
2025

1 INTRODUÇÃO

O objetivo deste projeto é desenvolver um programa que implemente um sistema de cadastro, com funcionalidades para busca e atualização de dados, além de garantir o armazenamento eficiente das informações em arquivos binários. Com base nesse propósito, optamos por utilizar espécies de plantas como tema para a organização e gerenciamento dos dados.

2 DESCRIÇÃO

Este código implementa um sistema simples de gerenciamento de plantas, utilizando um conjunto de estruturas e funções para executar operações como inserção, busca, listagem, ordenação e remoção de itens armazenados em um arquivo CSV.

2.1 ESTRUTURAS

- struct Plantas: contém os campos de cada planta: id, nome popular, nome científico, número de cotilédones, número de pétalas e classe.

2.2 FUNÇÕES E PROCEDIMENTOS

- importarCSV: essa função lê os dados de um arquivo CSV e os armazena em um vetor de Plantas.
- aumentarCapacidade: expande o tamanho do vetor em 10 posições sempre que ele atingir sua capacidade máxima.
- validaInt: uma função que verifica se o valor inserido pelo usuário é um número inteiro válido, o código limpa o erro da entrada e solicita que o usuário insira novamente um valor válido. Para isso utilizamos a biblioteca limits.
- validaClasse: uma função que , usando a biblioteca limits, verifica se a classe inserida é válida, ou seja, se é M ou D.
- formataString: formata a string para o padrão primeira letra maiúsculas e o resto minúscula, a fim de evitar erros. Para isso utilizamos a biblioteca algorithm.
- inserir: permite adicionar novos itens ao vetor.
- remover: permite remover itens do vetor.
- editar: permite a alteração de itens existentes no vetor.
- shellSortPorNome e shellSortPorID: por meio do método de ordenação shell sort, estas duas funções ordenam o vetor pelo nome popular e id, respectivamente.
- buscaPorNome e buscaPorID: usando uma busca binária, isto é, dividindo o vetor em duas partes até encontrar o elemento desejado, esta função chama a toString para imprimir os dados do item pesquisado pelo usuário, seja por nome ou id.
- toString: exibe os dados de uma variável do tipo Planta. Essa função é utilizada nas buscas para mostrar as informações de somente uma planta pesquisada.
- imprimir: imprime o vetor inteiro, exceto as plantas removidas.

- imprimirItervalo: permite visualizar um intervalo específico de plantas.
- salvarEmBinario: salva os dados do vetor em um arquivo binário.
- importarBin: importa os dados de um arquivo binário e armazena no vetor.
- salvarEmCSV: salva as alterações feitas no vetor, no arquivo CSV.
- menu: apenas imprime o menu.
- main: onde fazemos as chamadas das demais funções.

2.3 DADOS DO ARQUIVO

No arquivo CSV, os dados estão organizados da seguinte forma:

1. ID (int)
2. Nome popular (string que pode conter espaços)
3. Nome Científico (string com espaços)
4. Número de cotiledones (int)
5. Número de pétalas (int)
6. Classe (char)

Os campos são separado por vírgulas, sendo organizados dessa forma: “ID”,“Nome popular”,“Nome científico”,“Número de Cotilédones”,“Número de Pétalas”,“Classe”

3 DESENVOLVIMENTO

Durante o desenvolvimento do projeto, foi necessário tomar algumas decisões importantes. Uma delas foi a inclusão de um identificador único (ID) para cada planta, o que possibilitou a busca e a ordenação das plantas por ID. Além disso, optamos por utilizar o valor de ID igual a zero (0) para marcar as plantas removidas, sendo esse o único valor de ID repetido. Embora fosse possível criar uma flag para indicar se um item foi removido, decidimos adotar a abordagem do ID zero, já que não estamos utilizando um banco de dados que exija chaves primárias ou IDs exclusivos.

Outras duas decisões importantes durante o desenvolvimento das funções de busca foram: primeiro, chamar a função de ordenação dentro da própria função de pesquisa, antes de percorrer o loop de busca. Isso garante que o vetor esteja 100% ordenado, evitando possíveis erros. Em segundo lugar, na busca por nome, optamos por converter a primeira letra para maiúscula e as demais para minúscula, alinhando com o formato no qual os nomes estão escritos tanto no CSV. Dessa forma, ao pesquisar por "cravo", por exemplo, o sistema retornará a planta armazenada como "Cravo". Utilizamos a mesma lógica na hora de inserir e editar os nomes das plantas, chamando a função formataString.

Além disso, enfrentamos um problema ao tentar exportar e importar dados de um arquivo binário. O erro gerado era: Segmentation fault (core dumped), que normalmente ocorre quando tentamos acessar uma área de memória inexistente. Após algumas pesquisas, descobrimos que, para salvar os dados de um vetor em um arquivo binário, era necessário serializar os dados da estrutura Planta antes de armazená-los. Isso envolve salvar primeiro o tamanho das strings (nome e nomeCientifico), para então escrever o conteúdo dessas strings.

Por outro lado, a função importarBin precisava desserializar os dados do arquivo binário e carregá-los no vetor. Ou seja, precisávamos ler o tamanho das variáveis nome e nomeCientifico antes de armazená-las corretamente.

Nas demais funções, tivemos um cuidado especial com o tratamento de erros, realizando verificações antes de executar as operações principais. Por exemplo, verificamos se a planta existe no vetor antes de tentar removê-la ou editá-la, e garantimos que o arquivo foi corretamente aberto antes de realizar qualquer operação de leitura ou escrita. Essas precauções são fundamentais para evitar erros fatais no código, assegurando que o sistema funcione de maneira robusta e confiável, mesmo em situações inesperadas.

No geral, a execução do projeto prático representou um grande desafio em diversos aspectos. O código desenvolvido apresenta uma solução robusta para o gerenciamento de dados de plantas, utilizando estruturas e funções que garantem operações eficientes no arquivo CSV. A escolha de armazenar os dados em arquivos binários também contribui para a persistência das informações entre diferentes execuções do programa.

4 CONCLUSÃO

Em resumo, o projeto prático cumpriu com êxito seu objetivo de desenvolver um sistema eficiente de cadastro de plantas. A utilização de estruturas bem definidas e a implementação de funções específicas refletem uma abordagem organizada e alinhada às necessidades do sistema. A flexibilidade proporcionada pelas funções de ordenação e busca aprimora a experiência do usuário, tornando-a mais rica e intuitiva. Além disso, a capacidade de manipular dados em arquivos CSV, por meio de operações como leitura, escrita, busca e remoção, torna o programa altamente funcional e capaz de atender exatamente aos requisitos propostos.