Sistema de Gerenciamento de Idiomas

Alunos: André Santos Gomes e Bernardo Abrahão Mantovani

Descrição sucinta do Sistema:

Este relatório descreve o desenvolvimento de um sistema de gerenciamento de dados desenvolvido na linguagem C++, cujo objetivo é manipular informações sobre "Idiomas" armazenadas em arquivos CSV. Dentre as características do programa, temos: capacidade de armazenamento de novos idiomas, atualização deles, exclusão, leitura e ordenação. A capacidade de leitura do sistema inclui também diversos filtros para facilitar a visualização dos dados, desde alteração na quantidade de dados impressos até filtragem de campos específicos. Toda a leitura e escrita de dados é feita com o usuário através do terminal do Linux, onde eles acessam diversos menus através de dígitos numéricos e escolhem o que desejam fazer.

Estruturas de dados e Lógica do programa:

Para cumprir as necessidades de armazenamento dinâmico e redimensionamento eficiente, o programa foi baseado em duas estruturas de dados principais (structs):

1. Struct idioma: representa a unidade básica de informações. Contém os campos de texto (nome, famLing, principalPais), numéricos (id, numfalantes) e flags de controle lógico (del para sinalizar a remoção lógica e valido para controle de erros).

2. Struct arrayListIdioma: funciona como um vetor dinâmico feito manualmente. Esta estrutura guarda um ponteiro para o vetor de idiomas (*db), bem como números inteiros que controlam a capacidade total alocada e o tamanho lógico atual (número real de elementos).

O programa funciona num ciclo de vida dividido em fases lógicas:

- Inicialização e Leitura: No início, o programa abre o ficheiro db.csv. Ele lê o primeiro registo para saber o número de linhas e aloca dinamicamente na memória. Depois, ignora o cabeçalho e preenche o vetor com os dados extraídos (separados por ponto e vírgula).

- Gestão de Memória: Foi implementada uma função de resize. Quando, ao inserir um elemento, o vetor atinge a sua capacidade máxima, o programa aloca um novo vetor com mais 10 posições, copia os dados antigos e liberta a memória antiga, garantindo a escalabilidade requerida.

- Menu Interativo: O utilizador interage através de um menu numerado que o leva para as funções de manipulação (Ver, Inserir, Atualizar, Apagar e Sair).

- Consulta e Visualização: Permite ver todos os dados ou só os que correspondem a um dado campo. Para a pesquisa, é usada a Busca Binária. Como os dados podem não estar ordenados de acordo com o campo em que se quer procurar, então o programa, antes de procurar, ordena (Quicksort) com base nesse campo conforme a escolha do utilizador (Nome, Família, País,...). Para aqueles campos, como 'País Principal', 'Família Linguística' em que podem aparecer vários valores, foi implementada uma variação da busca binária que permite encontrar a primeira e última ocorrência do termo, podendo assim retornar e imprimir todos os registos existentes nesse intervalo.

- Manipulação (Inserção e Atualização): A inserção insere elementos no fundo do vetor lógico. A atualização localiza um registo pelo id (via busca binária), e permite alterar os campos, validados para não existir nomes duplicados ou números negativos.

- Eliminação Lógica: Uma “remoção” não apaga a informação imediatamente da memória. O elemento é marcado apenas com um flag del = true. O sistema foi pensado para ignorar os elementos com este flag nas operações de visualização e pesquisa.

- Finalização e Persistência: Ao sair, o programa percorre o vetor e sobrescreve o ficheiro db.csv, ignorando os itens marcados para eliminação, efetivando desta forma a “remoção” e guardando as novas inserções e atualizações.

Observações e Desafios durante o Desenvolvimento:

Esse projeto, apesar de ser simples, requer do aluno diversas habilidades: bom controle de ponteiros, uma boa lógica de programação aplicada, conhecimento das bibliotecas e funções base oferecidas pela linguagem e muita paciência.

Durante o seu desenvolvimento encontramos 4 grandes problemas, sendo eles:

1. Como armazenar os idiomas na memória do sistema de maneira conveniente
2. Como retornar um conjunto de respostas da busca binária ao invés de um elemento só

3. Escolher entre mergesort e quicksort para ordenar os vetores
4. Ignorar os elementos marcados para serem deletados enquanto não eram salvos no arquivo

Para solucionar o primeiro problema, nós utilizamos um struct que mimetiza os atributos de uma classe de lista baseada em arrays, ou seja, um array que pode acrescentar novos elementos além do seu tamanho base e ao mesmo tempo controla seu próprio tamanho, tanto lógico quanto real. Assim pudemos evitar um malabarismo de valores entre as funções, facilitando o acompanhamento das mudanças no valor e a legibilidade do código.

A solução do segundo foi realizar duas buscas binárias, as quais não param enquanto não encontram a primeira e a última aparições de um valor no vetor, assim temos a localização da faixa de valores que correspondem ao pedido do usuário.

No terceiro problema, após estudar a eficiência de cada um dos algoritmos – seus pontos fortes e fracos – chegamos à conclusão de que o quicksort é mais barato na memória, principalmente para arrays, do que o mergesort, apesar de poder precisar de mais operações no pior caso possível.

Para solucionar o quarto problema, não houve desafio lógico, mas sim falta de planejamento, já que percebemos essa necessidade após entrarmos nos estágios finais do desenvolvimento. Então foi árduo atualizar as funções de busca binária e de escrita no terminal para ignorar as instâncias que possuíam `.del` como verdadeiro.

Esses quatro problemas nos ajudaram a entender etapas cruciais na criação de um sistema: precisamos planejar o algoritmo sempre levando em consideração todas as condições impostas pelo enunciado, e não simplesmente executar com base na memória ou na impressão passada pelo problema.

Conclusão:

O projeto foi uma ótima oportunidade de aprendizado da linguagem C++, uma vez que precisamos revisar todos os conceitos passados anteriormente no curso e aplicá-los por toda a extensão do código.

No fim, atingimos o objetivo traçado e construímos um sistema de CRUD que: salva seus dados num arquivo CSV na memória secundária, carrega os dados de um arquivo csv em um vetor alocado dinamicamente, consegue inserir novos elementos no vetor, remove elementos, realiza ordenações e mostra resultados na tela.

O código pode ser acessado juntamente ao relatório no arquivo `.zip`, o qual conta com comentários e com instruções de leitura no arquivo readMe.md.