Programação Imperativa – Trabalho 2
Prof. Alcides Calsavara

Estrutura geral da aplicação:

A aplicação deve ser composta por três (3) programas na linguagem C, a saber:

1. **Gerador**:
    - Lê dados de um arquivo texto
    - Armazena internamente esses dados em memória heap
    - Grava esses dados em um arquivo binário

2. **Editor**:
    - Lê dados do arquivo binário
    - Armazena internamente esses dados em memória heap
    - Realiza operações sobre esses dados
    - Gera um novo arquivo binário contendo os dados atualizados

      A geração do arquivo binário pode ocorrer a qualquer momento da execução do Editor (a pedido do usuário), bem como no final da execução (obrigatoriamente, desde que os dados em memória tenham sido atualizados desde a última geração do arquivo binário.)

3. **Exportador**:
    - Lê dados do arquivo binário
    - Gera um arquivo texto contendo os dados, no mesmo formato do arquivo lido pelo **Gerador**.

Requisitos de implementação:

1. Sobre os dados:
    a. Podem se referir a qualquer coisa, à escolha de cada equipe de estudantes. Por exemplo, os dados podem se referir a pessoas ou a veículos.
    b. Devem corresponder a uma sequência de elementos, sendo que, para cada elemento, devem existir, ao menos, dois campos. Por exemplo, se os dados correspondem a uma sequência de pessoas, pode haver, para cada uma delas, o seu nome e o seu CPF.
    c. Um dos campos dos elementos deve ser uma chave única de identificação. No exemplo acima, o CPF seria o campo chave de identificação única de um elemento.
2. Sobre o armazenamento dos dados em memória heap, tanto no **Gerador** quanto no **Editor**:
    a. Deve ser feito por meio de uma lista encadeada que fique o tempo todo ordenada crescentemente com relação ao campo chave dos elementos.

    b. No **Gerador**, a lista deve conter, ao menos, 20 elementos.
    c. No **Gerador**, os dados lidos do arquivo texto estão em ordem aleatória com relação ao campo chave dos elementos.
3. Sobre o **Editor**:
    a. Deve interagir com o usuário para permitir a escolha de uma operação, fazendo as devidas verificações para consistência dos dados fornecidos pelo usuário. Essa interação deve ser contínua, até que o usuário escolha a opção de encerramento do programa.
    b. Deve dar opção ao usuário de **inserir** novos dados, fazendo com que a lista encadeada aumente. Essa inserção pode ocorrer em qualquer ponto da lista, dependendo do campo chave do elemento inserido.
    c. Deve dar opção ao usuário de **remover** dados, fazendo que a lista encadeada diminua. Essa remoção pode ocorrer em qualquer ponto da lista, dependendo do campo chave do elemento removido.
    d. Deve dar opção ao usuário de **exibir** os dados de um elemento específico da lista de acordo com uma chave de busca fornecida pelo usuário.
4. Sobre os arquivos texto e binário:
    a. Os seus nomes devem ser fornecidos aos programas como parâmetros de execução (argc e argv).
    b. Devem ser feitas todas as verificações necessárias sobre o sucesso da operação de abertura de arquivo.
    c. No arquivo texto, não pode haver informação explícita sobre a quantidade de dados, ou seja, a leitura deve, necessariamente, detectar o fim do arquivo.
5. Sobre a codificação:
    a. Deve haver, no mínimo, uma função cujo retorno seja um ponteiro.
    b. Deve haver, no mínimo, uma função recursiva. Por exemplo, a função de busca na lista para exibir os dados de um elemento específico.
    c. Deve respeitar todas as boas práticas de codificação estudadas no TDE 2.
6. Sobre a organização do código:
    a. Deve ser estruturado em funções relativamente pequenas.
    b. Deve ser organizado em arquivos de implementação (arquivo .c) e arquivos cabeçalho (arquivo .h).