Programação Imperativa – Trabalho do RA2

Prof. Alcides Calsavara

Considere o documento intitulado "Tabela Brasileira de Composição de Alimentos" em anexo. A Tabela 1 desse documento, da página 24 à página 65, apresenta a composição de 597 alimentos com respeito a diversos itens, incluindo umidade, energia, proteína e outros, sendo esses alimentos classificados em 15 categorias, a saber:

1. Cereais e derivados
2. Verduras, hortaliças e derivados
3. Frutas e derivados
4. Gorduras e óleos
5. Pescados e frutos do mar
6. Carnes e derivados
7. Leite e derivados
8. Bebidas (alcoólicas e não alcoólicas)
9. Ovos e derivados
10. Produtos açucarados
11. Miscelâneas
12. Outros alimentos industrializados
13. Alimentos preparados
14. Leguminosas e derivados
15. Nozes e sementes

Etapas do Trabalho:

A. Transcreva os seguintes dados de ***todos*** os alimentos da tabela para um arquivo texto de formato *livre* ou em formato **csv**:
   - Número do alimento
   - Descrição do alimento
   - Energia (Kcal)
   - Proteína (g)
   - Categoria do alimento

B. Escreva um programa **P1** na linguagem **C** que leia os dados sobre alimentos do arquivo construído na etapa A e armazene esses dados em ***formato binário*** num arquivo denominado **dados.bin**.

C. Escreva um programa **P2** na linguagem **C** que:
   a. Leia os dados sobre alimentos do arquivo **dados.bin** gerado na etapa B e armazene todos esses dados em memória, usando estruturas de **lista ligada** a fim de permitir que tanto o número de categorias como o número de alimentos em cada categoria sejam totalmente variáveis.

1. Deve haver uma lista denominada **Categorias** na qual cada elemento corresponde a uma categoria de alimentos. Nessa lista, as categorias devem ser armazenadas em ordem alfabética.
2. Cada elemento da lista **Categorias** deve conter, além da identificação da categoria, uma lista na qual cada elemento corresponde a um alimento da categoria. Na lista de alimentos de cada categoria, os alimentos devem ser armazenados em ordem alfabética da sua descrição.

b. Para cada categoria de alimentos, devem ser geradas duas estruturas em forma de *árvore binária* da seguinte forma:
    1. Uma árvore deve indexar os dados sobre **energia** dos alimentos da categoria.
    2. A outra árvore deve indexar os dados sobre **proteína** dos alimentos da categoria.
c. Forneça ao usuário uma interface com as seguintes opções em *loop*:
    1. Liste todas as categorias de alimentos na ordem constante na lista **Categorias**.
    2. Liste todos os alimentos de certa categoria na ordem constante na lista de alimentos dessa categoria.
    3. Liste todos os alimentos de certa categoria em ordem decrescente com respeito à **energia** (em Kcal) dos alimentos. (A implementação dessa operação deve utilizar a correspondente árvore binária de indexação.)
    4. Liste todos os alimentos de certa categoria em ordem decrescente com respeito à quantidade de **proteína** (em gramas) dos alimentos. (A implementação dessa operação deve utilizar a correspondente árvore binária de indexação.)
    5. Liste todos os alimentos de certa categoria cuja **energia** esteja entre um valor ***mínimo*** e um valor ***máximo*** escolhidos pelo usuário. (A implementação dessa operação deve utilizar a correspondente árvore binária de indexação.)
    6. Liste todos os alimentos de certa categoria cuja **proteína** esteja entre um valor ***mínimo*** e um valor ***máximo*** escolhidos pelo usuário. (A implementação dessa operação deve utilizar a correspondente árvore binária de indexação.)
    7. Remova uma categoria de alimentos.
    8. Remova um alimento específico. (Essa operação implica na atualização das árvores binárias de indexação da correspondente categoria. Essa atualização pode ser feita por meio da criação novas árvores.)
    9. Encerre o programa, fazendo com que uma versão atualizada do arquivo **dados.bin** seja gerada caso alguma categoria ou algum alimento tenha sido removido.

Requisitos de implementação:

1. A representação das categorias de alimentos deve ser implementada por meio de um **enumerado**.
2. Deve-se estruturar o programa em funções a fim de facilitar o desenvolvimento do programa, bem como a sua compreensão.
3. O código dos programas **P1** e **P2** deve ser organizado em arquivos de cabeçalho (extensão **.h**) e de implementação (extensão **.c**) de forma que fique organizado e se evite qualquer duplicação de código. Para todo arquivo de cabeçalho deve existir o respectivo arquivo de implementação, mas nem todo arquivo de implementação requer a existência de um arquivo cabeçalho. Em particular, a função ***main*** de cada programa deve ficar ***isolada*** em um arquivo próprio:
    a. A função ***main*** do programa **P1** deve ficar no arquivo **P1.c**
    b. A função ***main*** do programa **P2** deve ficar no arquivo **P2.c**

    Os arquivos **P1.c** e **P2.c** não devem possuir correspondentes arquivos de cabeçalho.

    Desse modo, os arquivos **P1.c** e **P2.c** contêm somente as respectivas funções ***main*** e incluem os arquivos de cabeçalho necessários e somente os necessários!

    Finalmente, todo arquivo de implementação que possua um correspondente arquivo de cabeçalho deve fazer a sua inclusão, além de poder incluir outros arquivos de cabeçalho.
4. A implementação de lista ligada deve ser feita com uso das funções **malloc**, **calloc**, **realloc** e **free**. Ou seja, deve ser feita alocação dinâmica de memória, usando a área ***heap***.
5. Toda memória alocada dinamicamente (com uso de **malloc** ou **calloc**) deve ser liberada (com uso da função **free**) pelo programa antes do seu término.
6. A ordem alfabética de acordo com a descrição de cada alimento na lista ligada dos alimentos de uma categoria deve ser garantida pelo próprio algoritmo de construção da lista. Isto é, cada inserção de alimento deve ser feita no local apropriado da lista, tal que, quando encerradas todas as inserções, a lista já esteja ordenada. Em outras palavras, não é preciso aplicar um algoritmo de ordenação para rearranjar a lista.
7. A ordenação alfabética da lista de categorias deve seguir a mesma estratégia usada para a lista de alimentos de uma categoria.
8. Cada nó de uma *árvore binária de indexação* de alguma categoria deve conter uma *chave* (valor de **proteína** ou valor de **energia**) e um *apontador* para o correspondente alimento (nó de uma lista ligada dos alimentos da categoria).
9. Não podem ser usados os comandos **break** e **continue** para se controlar laços de repetição.
10. Uma função deve ter, preferencialmente, apenas um comando **return**. Quando ocorrer alguma situação exceptional (por exemplo, falha ao abrir um arquivo), pode-se usar a função **exit** para encerrar a execução do programa.
11. Devem ser usados identificadores bem significativos para variáveis, parâmetros, tipos e nomes de funções.
12. Todo o código deve ser bem documentado por meio de comentários.

OBS: Link para o documento “Tabela Brasileira de Composição de Alimentos”:

https://www.cfn.org.br/wp-content/uploads/2017/03/taco_4_edicao_ampliada_e_revisada.pdf