

UNEB – Universidade do Estado da Bahia
Departamento de Ciências Exatas e da Terra
Linguagem de Programação II
Professor Marco Antônio Chaves Câmara

LISTA DE EXERCÍCIOS II

BÁSICO

1. Com base no programa desenvolvido na Lista 1, desenvolva um programa que lê um arquivo ASCII contendo uma listagem de nomes em maiúsculas e converte o mesmo para outro arquivo contendo a listagem modificada. Esta deve apresentar apenas as primeiras letras de cada nome em maiúsculas.

Dicas : crie o arquivo usando um editor ASCII (Ex.NotePad). Em primeiro lugar, preocupe-se em mostrar o resultado em tela, depois grave o arquivo. Utilize o programa da lista I como função deste novo programa.

2. A partir de um banco de dados familiares com a estrutura abaixo (que pode ser modificada, com acréscimo de novos campos), desenhar árvore genealógica de um determinado familiar :

Estrutura :

FICHA	Int
NOME	Vetor String c/ 40 posições
NASCIMENTO	Vetor String c/ 10 posições
PAI	Int
MAE	Int

Em termos operacionais, o programa deve solicitar o número da ficha do familiar que terá sua árvore analisada. A partir daí, deve imprimir uma listagem no seguinte formato : (dados fictícios)

Geração 0 - Zé Mané

Geração 1 - Pai - João
Mãe – Maria

Geração 2 - **Pai do** Pai – Pedro
Mãe do Pai – Ana

Pai da Mãe – Joaquim
Mãe da Mãe - Angélica

Geração 3 - **Pai do** Pai do Pai ...

Dicas : observe que o prefixo “Pai”, “Mãe”, “do” e “da” se repetem à medida que aumenta o número de gerações. Isto pode ser usado em um laço para simplificar a análise de árvores genealógicas muito extensas. Para efeito de exercício, no entanto, não vale a pena analisar árvores genealógicas muito extensas. É importante, no entanto, que se imprima até a segunda geração no mínimo, para atestar a operação correta do programa.

AVANÇADO

3. Através da análise de um arquivo fonte de um programa em C, gerar as indentações, saltos de linha e espaços necessários à melhor visualização do programa, gravando o resultado em outro arquivo.

Dicas : manter abertura e fechamento de blocos em uma linha separada; indentar todos os sub-itens de palavras reservadas, laços e blocos resultados de testes; espaços antes e depois de cada parâmetro na chamada de funções, nos operadores de teste e atribuições etc.

4. Contar o número de ocorrências de uma palavra em um determinado arquivo. Transformar o programa em um comando de linha e aceitar *wildcards* (* e ?) no nome do arquivo (totalizando o número de ocorrências em todos os arquivos). Retornar 0 se não encontrar nenhuma ocorrência, 1 se encontrar apenas uma e 2 se encontrar mais de uma (para teste via ERRORLEVEL em arquivos *batch*).
5. A partir de um arquivo texto, gerar listagem em ordem alfabética de todas as palavras do texto (sem repetição) e o número de ocorrências de cada uma (usar programa do item anterior como base). Na listagem, mostrar número de ocorrências e palavra, nesta ordem.

Dicas : aproveite o algoritmo “bolha” da lista anterior para colocar a listagem em ordem alfabética antes de apresentá-la no vídeo. Para arquivos muito grandes, talvez seja interessante manter a listagem de palavras e ocorrências em um arquivo, já que o uso de vetores pode facilmente ultrapassar a capacidade máxima de memória do modelo escolhido ... Novamente seria necessário o uso de estruturas para armazenamento e ordenação deste novo arquivo.