Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Estruturas de Dados II
Prof. Tiago A. Almeida <talmeida@ufscar.br>

LISTA 01
REVISÃO DE ARQUIVOS & ESTRUTURAS DE DADOS

- **Prazo para entrega: 07/09/2016 – 23:55**
- **Atenção:**
  1. **Padrões:** os exercícios devem ser implementados em `ANSI C`. Para cada exercício, é permitido o uso de **um** único arquivo `.h` com o mesmo nome do arquivo `.c`;
  2. **Arquivos:** o nome do arquivo referente ao código-fonte de cada exercício deverá seguir o seguinte padrão: <número do RA>_**L**<número da lista>**EX**<número do exercício>.c. Exemplo: `123456_L01EX01.c`;
  3. **Submissão:** submeta apenas um arquivo zip contendo todos os arquivos-fonte. Use um diretório para cada exercício. O nome do arquivo compactado deverá respeitar o seguinte padrão: <número do RA>_**L**<número da lista>.zip. Exemplo: `123456_L01.zip`;
  4. **E/S:** tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas;
  5. **Identificadores de variáveis:** escolha nomes apropriados;
  6. **Documentação:** inclua cabeçalho, comentários e indentação no programa.
  7. **Certifique-se de que todos os arquivos descompactam e compilam no Linux.**
- **Exercícios**
  1. (0,1) Faça um programa para ler um arquivo texto qualquer (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa) e imprimir o seu conteúdo na tela.
  2. (0,1) Faça um programa que gere uma cópia de um arquivo texto qualquer (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa).
  3. (0,1) Escreva um programa que conte o número de caracteres e de palavras de um arquivo texto qualquer (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa).
  4. (0,1) Escreva um programa que lê um arquivo texto qualquer e copia apenas os caracteres alfabéticos (letras) para um arquivo de destino (ambos fornecidos na chamada do programa). Números e caracteres especiais devem ser desconsiderados.

5. (0,1) Dado um arquivo texto (`arq.in`) com duas colunas de valores inteiros (separadas por um espaço), faça uma função que leia estes dados e gere um arquivo com 3 colunas (`arq.out`), sendo a terceira coluna o valor da soma das outras duas. O número de linhas do arquivo deve permanecer o mesmo.
6. (0,5) Dado um arquivo texto qualquer, faça um programa para substituir as palavras (traduzir o texto). Crie um arquivo texto (`dicionario.dic`). Leia cada palavra do texto (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa), verifique se ela existe no dicionário e então faça a substituição, caso contrário, não substitua a palavra.
7. (1,0) Crie um programa para manter um cadastro de Alunos. O seu programa deverá oferecer as seguintes funcionalidades (menu):

   1. `Cadastrar`
   2. `Alterar`
   3. `Remover`
   4. `Buscar`
   5. `Listar`
   6. `Sair`

   Todos os dados deverão ser salvos no arquivo-texto `Alunos.dat` com os registros ordenados em ordem crescente de RA. Caso o arquivo já exista, o seu programa deverá ler os dados já existentes. Mantenha os dados dos alunos em uma estrutura `Aluno` com os seguintes campos: RA (`int`), nome (*string* - 100 posições), ano de ingresso (`int`) e quantidade de créditos cursados (`int`).

   Cada opção deverá executar o seguinte procedimento:

   - `Cadastrar`: solicita todos os dados do aluno. Caso o RA informado já exista, informar na tela que o aluno já está cadastrado e retornar ao menu;
   - `Alterar`: solicitar o RA do aluno. Caso ele seja encontrado, solicitar novamente os campos: nome, ingresso e quantidade de créditos cursados. Caso contrário, emitir mensagem de aluno não cadastrado e retornar ao menu.
   - `Remover`: solicitar o RA do aluno. Caso ele seja encontrado, remover o registro. Caso contrário, emitir mensagem de aluno não cadastrado e retornar ao menu.
   - `Buscar`: solicitar o RA do aluno. Caso ele seja encontrado, exibir todos os campos do registro. Caso contrário, emitir mensagem de aluno não cadastrado e retornar ao menu.
   - `Listar`: imprimir na tela todos os campos de todos os registros existentes em ordem crescente de RA.
   - `Sair`: sobrescrever o arquivo com dados atualizados, liberar memória e fechar o programa.

   O seu programa deverá carregar/criar o arquivo na inicialização, manipular os dados em memória (sempre mantendo os registros em ordem crescente de RA) e sobrescrever o arquivo na finalização.

Os campos dos registros deverão ser separados pelo delimitador '@' e cada registro deverá ocupar uma linha do arquivo de dados.

Trate os possíveis erros que o usuário possa cometer, tais como: informar valores não numéricos nos campos `int`, deixar campos em branco, digitar uma opção inválida no menu, etc.

8. (0,5) Faça um programa para calcular o tamanho de um arquivo qualquer (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa).
9. (1,0) Escreva um programa para carregar e listar os dados de uma agenda armazenados em um arquivo sequencial `arq.in` (sem a informação da quantidade de registros). O registro de entrada do arquivo possui os seguintes campos: NOME, RUA, NÚMERO, TELEFONE, CIDADE e ESTADO. Use lista linearmente encadeada.
10. (1,0) Uma instituição de pesquisa recolheu amostras em três regiões a respeito do nível de vida da população. Cada amostra constitui um registro com os seguintes campos: SEXO, IDADE, SALÁRIO, ESTADO CIVIL, NÚMERO DE DEPENDENTES, VALOR DO PATRIMÔNIO, QUANTIDADE DE CALORIAS ABSORVIDAS POR DIA e GRAU DE INSTRUÇÃO. Em cada região, os dados foram armazenados em um arquivo texto (`reg1.dat, reg2.dat, reg3.dat`), sendo os registros colocados em ordem crescente de idade. Escreva um programa que leia os três arquivos e intercale-os de modo que o arquivo resultante (`regAll.dat`) permaneça ordenado.
11. (0,5) Escreva um programa que leia um arquivo qualquer (o usuário deverá fornecer o nome do arquivo junto com a chamada do programa) e copie o seu conteúdo para uma variável `buffer`. Imprima o conteúdo de `buffer`.
12. (1,0) Escreva um programa que solicite um número n, depois gere n números inteiros aleatórios e armazene-os em uma lista encadeada ordenada (use o `quicksort` nativo da linguagem C). Salve a quantidade de elementos e o conteúdo da lista em um arquivo texto (`arq.out`).
13. (1,0) Faça um programa que leia o arquivo (`arq.out`) gerado no exercício anterior, carregue a lista em memória, inverta-a, imprima-a na tela (um número por linha) e exiba o valor do maior e do menor elemento (um por linha).
14. (1,0) Escreva um programa que solicite um número n, depois gere n números inteiros aleatórios e armazene-os em uma árvore de busca binária. Salve a árvore em um arquivo texto (`arq.out`). Faça uma rotina para carregar a árvore do arquivo e imprimir a soma dos números primos.
15. (2,0) Escreva um programa que solicite um número n, depois gere n números inteiros aleatórios e armazene-os em uma árvore AVL. Salve a árvore em um arquivo texto (`arq.out`). Faça uma rotina para carregar a árvore do arquivo e imprimir a soma dos valores pares.

- **Cuidados**

1. **Erros de compilação:** nota **zero** no exercício
2. **Tentativa de fraude:** nota **zero** na média para todos os envolvidos.