Universidade Federal de Uberlândia

Faculdade de Engenharia Elétrica
Programação Procedimental
Prof. Felipe A. Louza

Lista 15

Manipulação de arquivos

Questão 1

Elabore um programa que aceita um arquivo texto via parâmetros de linha de comando e efetua a troca de vogais minúsculas pelas vogais maiúsculas. O resultado deve ser sobrescrito no mesmo arquivo.

Questão 2

Seja um arquivo texto contendo a descrição de compras de supermercado: em cada linha há o nome do produto e o valor do mesmo.
Faça um programa que receba um arquivo deste tipo e calcule o valor total da compra.

Questão 3

Elabore um programa que receba dois arquivos textos via parâmetros da linha de comando e inverta todas as linhas presentes no primeiro arquivo e armazene-as no segundo arquivo.

Por exemplo, se o arquivo de entrada contém:

```
O rato roeu a roupa do rei de roma
abracadabra pe de cabra
sim salabim
socorram-me subi no onibus em marrocos
```

O arquivo de saída deverá conter:

```
amor ed ier od apuor a ueor otar O
arbac ed ep arbadacarba
mibalas mis
socorram me subino on ibus em-marrocos
```

Questão 4

Elabore um programa que aceita um código-fonte `C` e um arquivo de saída via parâmetros de linha de comando e elimine todos os comentários deste código.

Assuma que o comentário utilizado é do estilo `//` e que a linha que possuir um comentário não possui nada além dele.

O código modificado deverá ser salvo no arquivo de saída.

Questão 5

Faça um programa que leia uma série de alunos de um arquivo texto passado via linha de comando. Cada aluno é composto de:

- Nome (até 30 caracteres).
- Semestre (inteiro).
- Rendimento acadêmico (real).

Considere que cada um dos atributos do aluno está em uma linha separada do arquivo. Por fim, leia um inteiro x do usuário e informe quais os alunos que estão cursando o semestre x.

Questão 6

Suponha que você tenha sido contratado para fazer um sistema de controle de estoque de uma loja de presentes.

Cada presente possui os seguintes atributos:

- Nome (até 30 caracteres).
- Descrição (até 150 caracteres).
- Unidades restantes.
- Preço unitário.

Faça um programa que possibilite:

- Cadastrar uma nova unidade de produto de acordo com o **nome**. Caso o produto já esteja cadastrado, ele deve ter a quantidade de unidades restantes incrementada, se não, uma nova unidade é criada no fim do arquivo e o contador de unidades passa a ser 1.
- Dar baixa em uma nova unidade de produto de acordo com o **nome**. Deve funcionar apenas quando o número de unidades for ≥ 1, decrementando o número de unidades restantes. Se nome do produto não for encontrado ou o número de unidades for 0, nenhuma operação deve ser efetuada e uma mensagem de erro deverá ser impressa na tela.
- Alterar o preço de um produto.
- Verificar o total arrecadado até o momento. A cada baixa de produto este valor é atualizado de acordo com o preço do produto no momento da baixa.

Obviamente o programa deve operar normalmente caso seja reiniciado, e portanto as informações deverão persistir por meio de um arquivo binário.

Dica: utilize a função `fseek` para posicionar o ponteiro de leitura/escrita do arquivo, isso facilitará quando um dado precisar ser lido e sobrescrito logo em seguida, não necessitando portanto de ler o arquivo novamente do começo.

Questão 7

Faça um programa que aceita três arquivos binários via parâmetros de linha de comando. Os dois primeiros arquivos contém, cada um, uma lista de inteiros ordenada. O programa deverá produzir uma única lista ordenada, a partir dos dois arquivos de entrada, e armazená-la no terceiro arquivo.

Questão 8

Um arquivo `CSV` (comma-separated values) é um arquivo texto que armazena cada registro em uma linha. Os campos do registro são separados por algum delimitador, geralmente o caractere ‘,’. Faça um programa que leia um arquivo `CSV` passado via parâmetros de linha de comando e imprima cada registro na tela.

Referência: `https://en.wikipedia.org/wiki/Comma-separated_values`

Dica: você pode utilizar a função `strtok` para separar cada linha de um arquivo `CSV` em campos através do delimitador.

Questão 9

Um arquivo JSON (JavaScript Object Notation) é utilizado frequentemente para transmitir dados. Basicamente trata-se de um arquivo texto no formato `"chave":valor`. Faça um programa que, dado um arquivo JSON passado via parâmetros de linha de comando, interprete-o em um registro em memória. O JSON da questão tem o seguinte formato, podendo assumir valores diferentes a cada arquivo:

```
{
  "nome": "Epaminondas",
  "sobrenome": "da Silva",
  "vivo": 1,
  "idade": 25,
  "endereco": "UFU"
}
```

As chaves podem vir em ordem diferente a cada arquivo, então você deverá utilizar o nome delas para identificar em qual campo do registro os valores devem ser gravados. Após converter um arquivo JSON na sua representação em registro, o seu programa deverá dar a opção de alterar os dados em memória e sobrescrever o arquivo original.

Referência: `https://en.wikipedia.org/wiki/JSON`

Questão 10

Suponha que um arquivo **binário** da Receita Federal esteja populado por vários registros de pessoa física, que consistem dos seguintes campos:

```c
typedef struct pessoa_fisica_t{
    char nome_completo[50], cpf [20];
    double renda_media;
    int nro_bens, nro_dependentes;
} pessoa_fisica_t;
```

Escrevas as seguintes rotinas:

(a) (0.5 points) Realize a abertura deste arquivo no modo leitura + atualização. Suponha que ele se chama `dados.dat` e se encontra no diretório `/home/RF/2019/`.

(b) (0.75 points) Imprima na tela o **nome** e **cpf** das pessoas que possuem uma renda familiar *per capita* (considerando o número de dependentes) superior a $R\$ \ x$ reais. Você deverá implementar uma função com a seguinte assinatura:

```
void imprime_pessoas_renda_per_capita(FILE* arq,double x);
```

(c) (0.75 points) Altere no arquivo o número de bens de todas as pessoas chamadas `str` para 0. Você deverá implementar uma função com a seguinte assinatura:

```
void altera_nro_bens(FILE* arq, char* str);
```

Dica: você pode utilizar a função `fseek` para posicionar a cabeça de leitura/escrita.

```
(...)
fseek(arq,-20,SEEK_CUR) // Recua 20 bytes da posição atual
fseek(arq,20,SEEK_SET) // Avança 20 bytes do inicio do arquivo
fseek(arq,-20,SEEK_SET) // Recua 20 bytes do fim do arquivo
```