UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
BCC2001 :: Algoritmos

**Exercícios 12 :: Arquivos de texto++**

Instruções Gerais

- Faça todos os exercícios em um único arquivo .c. Utilize a função main() para fazer chamadas de testes às funções solicitadas pelos exercícios.
- Utilize a extensão .c, o compilador gcc e o editor de sua preferência: VS Code, Dev C++, etc.
    - Alternativamente, utilize https://replit.com/languages/c.

**Lembretes:**

- **Ao testar cada função na main(), lembre-se de liberar a memória manualmente alocada pela(s) função(ões), caso isso ocorra, para evitar vazamentos de memória;**
- **Feche o ponteiro do arquivo ao término da função: `fclose(file)`.**

1. Escreva uma função que conta todas as ocorrências de uma dada palavra, em um arquivo.
   **`int count_word(const char* filepath, const char* word)`**

2. Escreva uma função que substitui todas as ocorrências de uma dada palavra por outra, em um arquivo. A função deve retornar a quantidade de substituições realizadas.
   **`int replace_word(const char* filepath, const char* old_str, const char* new_str)`**

3. Escreva uma função que imprime a contagem de palavras por quantidade de letras, de um arquivo. A contagem deve ser informada em ordem crescente de quantidade de letras.
   **`void report_by_chars(const char* filepath)`**

   Exemplo de saída da função:
   ```
   Arquivo: relatorio_final.txt
   Contagem de palavras por quantidade de letras:
   > 1 letra:  9
   > 2 letras: 17
   > 3 letras: 15
   > 5 letras: 8
   > 9 letras: 2
   ```

4. Escreva uma função que imprime a quantidade de ocorrências de cada palavra, em um arquivo.
   **`void count_all_words(const char* filepath)`**

5. Escreva uma função que retorna um vetor contendo todas as palavras (strings) que possuam um dado número de letras em um arquivo. O vetor deve ser alocado em heap.
   **`char** get_words(const char* filepath, int letters)`**

6. Considerando o tipo estruturado Product, escreva uma função que grava o conteúdo de um vetor de produtos em um arquivo no formato CSV (valores separados por vírgula). No arquivo, os registros dos produtos são armazenados em sequência, com seus campos separados por vírgula. A função deve retornar true (1) para sucesso ou false (0), caso contrário. OBS: ao salvar o campo Product::name no arquivo, substitua os espaços por '@'. Dica: Utilize a função fprintf() para facilitar a implementação.

```
typedef struct {
    int id;
    char name[51];
    float price;
} Product;
```

**int save_records(const char* filepath, int n, const Product* prods)**

Exemplo de formato no arquivo:
`4,Monitor@Axer@23,970.000000,2,Mouse@Optron@,102.000000,...`

Curiosidade: após criado o arquivo, abra-o em um editor de planilhas, como Microsoft Excel, LibreOffice Calc ou Google Sheets.

7. Considerando o tipo estruturado Product, escreva uma função que lê um arquivo no formato do exercício anterior e retorna um vetor de Produtos com o conteúdo do arquivo. A função deve retornar o endereço do vetor de Produtos, alocado em heap. O comprimento do vetor deve ser preenchido pelo parâmetro de endereço 'n'. Se não for possível abrir o arquivo ou o mesmo estiver vazio, a função deve retornar NULL e n=0. OBS: ao carregar o campo Product::name do arquivo, substitua os '@' por espaços.
**Product* load_records(const char* filepath, int* n)**

8. Escreva uma função que faz a leitura de um arquivo em formato INI e retorna um vetor de pares (tipo Pair) preenchido com os dados constantes no arquivo. O formato INI define pares "**chave = valor**" em cada linha do arquivo. A função deve retornar o endereço do vetor de Pares, alocado em heap. O comprimento do vetor deve ser preenchido pelo parâmetro de endereço 'n'. Se não for possível abrir o arquivo ou o mesmo estiver vazio, a função deve retornar NULL e n=0. OBS: poderão haver espaços, tanto na **chave**, quanto no **valor.** Espaços que estiverem antes e depois devem ser removidos. Sugestão: utilize a função fgets() para leitura das linhas do arquivo.

```
typedef struct {
    char key[101];
    char value[101];
} Pair;
```

**Pair* load_ini_file(const char* filepath, int* n)**

Exemplo de formato INI:
target = 192.0.14.662
file = vm_params.cfg
name = John Doe
description = Configuration file for Neptune VM