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

Exercícios 11 :: 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 verifica se o arquivo de caminho informado existe (retorna 1). Caso contrário, retorna 0.
   `int file_exists(const char* filepath)`

   Exemplo de uso da função:
   ```
   int res = file_exists("C:\documentos\programa.exe"); // retorna 1 se existir
   ```

2. Escreva uma função que receba um caminho de arquivo. Se o mesmo existir, faz nada. Se o arquivo não existir, tenta criá-lo. A função deve retornar 1 (um) caso arquivo exista ou tenha sido criado e, 0 (zero), caso contrário.
   `int check_or_create(const char *filepath)`

3. Escreva uma função que receba um caminho de arquivo e imprima seu conteúdo no terminal. Retorna 1 para sucesso ou 0, caso o arquivo não exista.
   `int print_content(const char *filepath)`

4. Escreva uma função que receba um caminho de arquivo e retorna a quantidade de linhas de texto contidas no arquivo. Dica: o caractere '\n' define a quebra de linha no texto.
   `int count_lines(const char *filepath)`

5. Escreva uma função que receba um caminho de arquivo e escreva uma string no mesmo. A função deve (re)criar o arquivo especificado. Retorna 1 para sucesso ou 0, em caso de erro.
   `int save_string(const char *filepath, const char* text)`

6. Escreva uma função que receba um caminho de arquivo e acrescenta uma string no mesmo. A função deve criar o arquivo especificado, caso não exista. Retorna 1 para sucesso ou 0, em caso de erro. Dica: observe o modo de abertura 'a' (slide no. 5 do material).
   `int append_string(const char *filepath, const char* text)`

7. Escreva uma função que receba um caminho de arquivo e escreva linhas de strings no mesmo. Ela deve (re)criar o arquivo especificado. A função recebe um vetor de strings e cada uma deve ser escrita em uma linha do arquivo. Retorna 1 para sucesso ou 0, em caso de erro.
   int save_lines(const char *filepath, int n, const char text_lines[n][51])

8. Escreva uma função que receba um caminho de arquivo e retorna uma string (vetor de caracteres terminado em nulo) alocada em heap contendo o conteúdo do arquivo. Caso ocorra algum problema, retorna NULL.
   char* get_content(const char *filepath)

9. Escreva uma função que receba um caminho de arquivo e retorna um vetor de strings contendo todas as palavras do arquivo. Considere como separadores de palavras o caractere de espaço e a quebra de linha '\n'. O número de palavras encontradas e colocadas no vetor de strings deve ser retornado via parâmetro **words_count** (por endereço). Caso ocorra algum problema, a função retorna NULL.
   char get_words(const char *path, int* words_count)**

   Exemplo:
   Arquivo 'texto.txt':
   C is a general-purpose programming language, widely used and very influential. By design, C reflects the capabilities of the targeted CPUs, such as types, operations and memory access.

   Retorno da função:
   {"C", "is", "a", "general-purpose", "programming", "language,", "widely", "used", "and", "very", "influential.", "By", "design,", "C", "reflects", "the", "capabilities", "of", "the", "targeted", "CPUs,", "such", "as", "types,", "operations", "and", "memory", "access."}

10. Escreva uma função que receba um caminho de arquivo e retorna um vetor de strings contendo todas as palavras do arquivo. A string de separadores é passada como argumento à função. O número de palavras encontradas e colocadas no vetor de strings deve ser retornado via parâmetro **words_count** (por endereço). Caso ocorra algum problema, a função retorna NULL.
    char get_words_sep(const char *path, const char* separators, int* words_count)**

    Considerando exemplo do exercício anterior, este seria o retorno da função para os separadores " ,;.:?!-":
    {"C", "is", "a", "general-purpose", "programming", "language", "widely", "used", "and", "very", "influential", "By", "design", "C", "reflects", "the", "capabilities", "of", "the", "targeted", "CPUs", "such", "as", "types", "operations", "and", "memory", "access."}