Desafio de Programação - Caça-Palavras

Instruções gerais

Resolver o problema abaixo seguindo, **obrigatoriamente,** as regras:

1) Utilizar vetores e/ou matrizes. (**preferencialmente** dinâmico usando ponteiros);
2) Utilizar estruturas de repetição;
3) Implementar utilizando modularização (no mínimo um módulo de procedimento e dois módulos de ação);
4) Utilizar passagem de parâmetro por valor e referência (**referência é em caso de usar ponteiros**). **É proibido usar variável global!**;
5) Utilizar registros;
6) Utilizar alocação dinâmica de memória (**não é obrigatório, mas desejável**);
7) Trabalho individual.

OBS 1: Apresentar, além do código, **pequeno relatório descrevendo, em detalhes, o método (metodologia) utilizado para resolver o problema**. O relatório deverá conter um resumo entre 400 e 700 palavras, figuras e gráficos que demonstrem a resolução e resultados obtidos. **Importante: os resultados devem ser discutidos e não apenas apresentados**.

OBS 2: Apresentação/entrega até no máximo dia 30/06

Problema (desafio) Caça-palavras:

O asilo Nono Vito adquiriu um conjunto expressivo de passatempos para que os idosos possam se manter intelectualmente ativos. Dentre os passatempos, foram adquiridos jogos caça-palavras (também conhecido por sopa de letras), que é um passatempo que consiste em letras arranjadas aparentemente aleatórias em uma grade quadrada ou retangular.
O objetivo do jogo é encontrar (e circundar) as palavras escondidas na grade, onde as palavras podem estar escondidas verticalmente, horizontalmente ou diagonalmente dentro da grade. As palavras são arranjadas normalmente de modo que possam ser lidas da esquerda para a direita ou de cima para baixo, sendo que os passatempos adquiridos pelo asilo são de maior dificuldade e também podem ocorrer palavras no sentido oposto (de baixo para cima e da direita para a esquerda).
Assim, o asilo pediu sua ajuda para desenvolver um programa que facilite aos cuidadores encontrar as palavras dentro da grade. Para isso, você precisar desenvolver um programa, com um método ou função com a seguinte assinatura em C:

Roi localizaPalavra (char[] grade, char palavra[]); // Exemplo da assinatura em C

onde ***Roi*** significa "Região de interesse" e é um registro (*struct*) contendo dois pares de valores inteiros, os quais indicam a posição inicial (linha, coluna) e final (linha, coluna) de onde a palavra se encontra na grade.

Você receberá como **parâmetro** um **array de char (string)** que representa **a grade (matriz) quadrada de (NxN)** com as letras do caça-palavras, como no exemplo abaixo onde N é igual a 10:

char grade[] = {"SAPAACAVAAAAPADAVUUJTAPAAPASONASJAVMVARAJPOATRRASRAOJALOR
AAAOPAOMOUPSLPAPPMVAUAAOATALSVAPAAOTTRAASOP"};

grade:

	0	1	2	3	4	5	6	7	8	9
0	S	A	P	A	A	C	A	V	A	A
1	A	A	P	A	D	A	V	U	U	J
2	T	A	P	A	A	P	A	S	O	N
3	A	S	J	A	V	M	V	A	R	A
4	J	P	O	A	T	R	R	A	S	R
5	A	O	J	A	L	O	R	A	A	A
6	O	P	A	O	M	O	U	P	S	L
7	P	A	P	P	M	V	A	U	A	A
8	O	A	T	A	L	S	V	A	P	A
9	A	O	T	T	R	A	A	S	O	P

E uma lista com as palavras que constam na grade:

Palavra
SAPATO
UVA
LARANJA
PAO
AMORA
SAPO
SOPA

Para cada palavra que for encontrada nas direções horizontais (tanto no sentido esquerda-direita, quanto direita-esquerda), verticais ou nas diagonais (tanto no sentido cima-baixo, quanto baixo-cima) você deverá retornar as informações de localização da palavra dentro da grade. Exemplo:

Palavra	li	ci	lf	cf
SAPATO	0	0	4	4
UVA	7	1	5	1
LARANJA	6	9	0	9
PAO	6	1	6	3
AMORA	8	3	4	7
SAPO	6	8	9	8
SOPA	0	0	0	0

Nesse caso, a chamada para a função **localizaPalavra (grade, "SAPATO")** deve retornar {0, 0, 4, 4}. Caso a palavra não seja encontrada na grade a função deve retornar {0, 0, 0, 0}, como o exemplo da palavr "SOPA".

Com base nessas informações, desenvolva o algoritmo da maneira mais eficiente possível, de acordo com a assinatura proposta **localizaPalavra (char[] grade, char palavra[])**, que seja capaz de localizar corretamente as palavras na grade.

Exemplos de arquivos texto para validação, com a sequência das grades e as palavras, podem ser encontrados na pasta GDrive.

O que entregar

- Código-fonte em C
- Criar um repositório **privado** no Github
    - Após finalizar o teste adicionar o usuário ***manassesribeiro*** (manasses.ribeiro@ifc.edu.br) como colaborador para que possamos ter acesso ao código;
    - Se o repositório estiver público, será automaticamente desqualificado (e a nota zerada).

- Instruções sobre como executar o programa ou a API (no arquivo README)
    - **OBS:** lembre-se que o programa deve ter interface de uso que facilite o carregamento das sequências

- Relatório técnico descrevendo a metodologia utilizada para resolver o problema, conforme descrito na OBS 1

Observações:

- Tenha em mente que faremos uma série de testes com matrizes válidas e inválidas.
- Considere a performance do algoritmo e o tempo de resposta da aplicação.