Proposta de Trabalho

Preenchimento e identificação de palavras em matrizes de caracteres (Sopa de Letras)

Linguagens Programação I

Rui Silva Moreira

rmoreira@ufp.edu.pt

Beatriz Gomes

argomes@ufp.edu.pt

Algoritmos e Estruturas Dados I

José Torres

rmoreira@ufp.edu.pt

André Pinto

apinto@ufp.edu.pt

Outubro 2018

Universidade Fernando Pessoa

Faculdade de Ciência e Tecnologia

Definição do problema

Pretende-se desenvolver um programa, organizado em funções, que implemente uma variante do jogo/passatempo tradicional de descobrir palavras numa matriz de letras. A matriz de caracteres poderá ser fornecida por um utilizador ou gerada de diversas formas (e.g., aleatoriamente, recorrendo a heurísticas ou de acordo com algoritmos específicos). Dada a matriz pretende-se encontrar determinadas palavras na matriz, formadas pelos caracteres existentes. Tradicionalmente as palavras contêm sequências contíguas de caracteres mas sem repetições dos mesmos caracteres. Contudo, nesta versão, poderemos começar numa letra e, movimentando-nos nas diferentes direções dos pontos cardeais, reutilizar os mesmos caracteres várias vezes na construção de uma palavra (e.g., a palavra "AMAR" pode ser encontrada na sequência "RAM"; começando no 'A'-Este -> 'M'-Oeste -> 'A'-Oeste -> 'R').

Tabuleiro

	0	1	2	3	4	5	6	7	8	9	10
0	J	D	C	P	C	P	X	O	A	A	
1	Z	X	V	O	V	X	F	R	V	V	
2	N	D	L	E	I	R	B	I	E	A	
3	Y	T	R	Q	O	M	O	I	I	O	
4	F	Z	Z	A	P		E	R	T	Q	
5	X	A	U	E	O	E	O	O	T	O	
6	P	O	R	T	U	O	A	Z	L	Z	
7	C	Z	N	O	Q	U	P	U	O	P	

Dicionário

- Porto
- Aveiro
- Coimbra
- Faro

Porto - [6, 0] -> [5, 5] - Direcção: [E] -> [E] -> [E] -> [E] -> [NE]
Aveiro - [0, 10] -> [5, 8] - Direcção: [S] -> [SW] -> [SW] -> [S] -> [S]
Coimbra - [0, 2] -> [0, 9] - Direcção: [SE] -> [SE] -> [SE] -> [NE] -> [NE] -> [NE]
Faro - [4, 0] -> [7, 4] - Direcção: [SE] -> [SE] -> [SE]

Suponha a matriz de caracteres alfabéticos da Figura 1. Neste exemplo podemos encontrar a palavra PORTO a partir da posição [6, 0] da matriz. A palavra começa no P, segue para a direita (Este - E) e passa pelo O, segue novamente para a direita (Este - E) pelo R, continua na mesma direção (Este – E) para o T, por fim passa para a diagonal (Nordeste - NE) até ao O. Da mesma forma se poderão encontrar as outras palavras do dicionário proposto (cf. Aveiro, Coimbra e Faro).

Requisitos funcionais

Pretende-se que os alunos proponham duas abordagens/implementações para o problema proposto: uma solução baseada em matrizes e outra baseada em listas encadeadas. Assim, para cada uma das abordagens, deve especificar-se e implementar-se as estruturas de dados e os algoritmos necessários à resolução do problema proposto. Em particular pretende-se:

1. Permitir a inicialização das estruturas de dados (cf. matriz de caracteres e conjunto/dicionário de palavras a procurar). A inicialização deverá ser feita por várias formas: geração aleatória da matriz de caracteres; leitura da matriz a partir de um ficheiro de texto; algoritmos de inserção/pesquisa a partir de um conjunto de palavras pré-existente. A lista de palavras/dicionário deverá ser obtida por diferentes métodos: inserida pelo utilizador; lida a partir de um ficheiro de texto; conseguida a partir de análise textual.
2. Permitir ao utilizador atualizar (adicionar ou remover) palavras ao conjunto de palavras a procurar;
3. Permitir efetuar a pesquisa de palavras, guardando para cada palavra a sua posição inicial (x, y) e respetiva sequência de movimentos, como descrito anteriormente;
4. A pesquisa deve ser efectuada pela ordem de inserção.
5. Com base no ponto anterior, o programa deve guardar para cada palavra encontrada, as estatísticas referentes ao processo de pesquisa (cf. número de movimentos parciais e o total de movimentos). Por exemplo, para a palavra PORTO temos no total 4 movimentos: 3 movimentos para Este (E) e 1 para Nordeste;
6. A pesquisa deve ser efectuada pela ordem de inserção.
7. Uma dada palavra pode encontrar-se na matriz a partir de diferentes posições iniciais pelo que o programa deverá encontrar todas as ocorrências dessa palavra;
8. Permitir procurar um conjunto de caracteres que se encontram ao longo de um determinado caminho. Esse caminho é caracterizado por um conjunto de movimentos.
9. Criar mecanismos que melhorem a eficiência na pesquisa de palavras.
10. Permitir gerar um ficheiro de texto com a matriz de caracteres e os respetivos resultados das pesquisas referentes a cada palavra;
11. Permitir gerar um ficheiro binário com a matriz de caracteres e os respetivos resultados das

pesquisas para cada palavra.

12. Todos os requisitos devem ser testados por funções de teste específicas sem recurso à utilização de menus de interação com o utilizador. Poderão, no entanto, ser utilizados parâmetros passados via argc/argv.

Em concreto a aplicação deverá cumprir os seguintes requisitos:

1.1 Fase 1: recurso a arrays/vectores e matrizes dinâmicas

Na primeira fase devem utilizar-se arrays/vetores e matrizes dinâmicas para organizar os dados referentes às palavras a pesquisar e à sopa de letras. Deverão ainda ser desenvolvidos algoritmos de processamento, gestão e pesquisa. A implementação deverá respeitar os seguintes requisitos funcionais:

R1. Permitir a inicialização da estrutura de dados (cf. conjunto de palavras a procurar) através dos seguintes mecanismos:

- **a.** Inserção manual;
- **b.** Leitura da matriz a partir de um ficheiro de texto;

R2. Dada uma matriz, composta por palavras, pretende-se conseguir ordenar, utilizando os algoritmos de ordenação MSD;

R3. Permitir a inicialização da estrutura de dados (cf. Sopa de Letras) através dos seguintes mecanismos:

- **a.** Preenchimento através da geração aleatória de caracteres;
- **b.** Leitura da matriz de caracteres a partir de um ficheiro de texto;

R4. Permitir efetuar a pesquisa de palavras, guardando para cada palavra a sua posição inicial (x, y) e respetiva sequência de movimentos;

1.2 Fase 2: recurso a apontadores e estruturas ligadas

Na segunda fase devem utilizar apontadores e estruturas dinâmicas para agregar os dados anteriores. Deverão ser desenvolvidos ainda algoritmos de processamento, gestão e pesquisa de toda a informação (cf. dicionário de palavras, tabuleiro da sopa de letras).

A implementação deverá, para além da adaptação de todas as estruturas de dados ao funcionamento com estruturas dinâmicas, respeitar os requisitos anteriores bem como endereçar os seguintes requisitos funcionais:

R5. Permitir a inicialização da estrutura ligada (cf. conjunto de palavras a procurar) através dos seguintes mecanismos:

- **a.** Inserção manual;
- **b.** Leitura da matriz a partir de um ficheiro de texto;

Nota: Deve fazer recurso de uma estrutura de dados do tipo Queues.

R6. Dada uma lista, composta por palavras, pretende-se conseguir ordenar, utilizando os algoritmos de ordenação mergesort ou quicksort;

R7. Permitir a criação da estrutura de dados (cf. Sopa de Letras). O tabuleiro deve ter um tamanho bidimensional. A estrutura deve conter espaço para toda a informação necessária ao problema, assim como 8 apontadores para as estruturas das células vizinhas;

R8. Permitir a inicialização da estrutura de dados (cf. Sopa de Letras) através dos seguintes mecanismos:

- **a.** Leitura da matriz de caracteres a partir de um ficheiro de texto;
- **b.** Dado um dicionário de palavras deve ser possível construir o tabuleiro (sopa de letras) que garanta que todas as palavras do dicionário estão presentes no tabuleiro gerado.*;

R9. Permitir efetuar a pesquisa de palavras, guardando para cada palavra a sua posição inicial (x, y) e respetiva sequência de movimentos;

R10. Permitir retornar um conjunto de caracteres, dado um conjunto de movimentos e uma posição inicial;

R11. Implementar um sistema de "cache" que permita em pesquisas posteriores uma maior eficiência na pesquisa de palavras;

R12. Permitir a manipulação das estruturas de dados através da entrada e saída para ficheiros de texto; Pretende-se a definição de um formato textual adequado para representar/suportar a informação dos modelos de dados e a sua exportação e importação para ficheiro;

R13. Permitir a manipulação das estruturas de dados através da entrada e saída para ficheiros binários; Pretende-se a definição de um formato binário adequado para representar/suportar a informação dos modelos de dados e a sua exportação e importação para ficheiro (este requisito refere-se apenas a LP1).

2 Ficheiros e documentos a entregar

2.1 Anotações e comentários no código fonte

As estruturas de dados e os algoritmos definidos devem ser implementados em C com os comentários apropriados, inseridos no código fonte desenvolvido, que facilitem a compreensão do mesmo. Todas as funções devem estar anotadas em formato doxygen incluindo: uma explicação dos algoritmos implementados; uma menção ao desempenho dos algoritmos (quando aplicável) assim como dos testes efetuados/implementados. As principais estruturas de dados e variáveis devem também estar anotadas neste formato. Deverão usar o software doxygen para gerar a documentação com base nos comentários.

Os alunos deverão entregar um ficheiro de texto no formato doxygen (.dox), descrevendo explicitamente: as funcionalidades/requisitos implementados, parcialmente implementados e não implementados. Devem mencionar sempre o número do requisito de acordo com a numeração utilizada neste documento de especificação:

- Funcionalidades implementadas: devem identificar todas as funções desenvolvidas para assegurar os requisitos funcionais solicitados.
- Funcionalidades não implementadas: devem identificar as funcionalidades não implementadas e apontar a justificação e/ou dificuldades que impediram o seu desenvolvimento.

3 Submissão

A aplicação final (código) deve estar depurada de todos os erros de sintaxe e de acordo com os requisitos funcionais pedidos. Só serão considerados os programas que não contenham erros de sintaxe e implementem as funcionalidades pedidas. A documentação, em html ou pdf, juntamente com todo o código-fonte desenvolvido, deve ser submetida num ficheiro zip/rar (project2018.zip) na plataforma elearning.ufp.pt.