Algoritmos e Tipos Abstratos de Dados

Mini-Projeto 1

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Caça ao Tesouro

1. Requisitos

As aplicações terão de ser desenvolvidas na linguagem de programação C e usando as técnicas adequadas lecionadas nas aulas teórico-práticas e laboratoriais.
O IDE a utilizar fica ao critério dos alunos mas, caso não utilizem o IDE usado nas aulas (o Codeblocks), terão que (antes de submeterem) criar os respectivos projetos no IDE Codeblocks.

2. Objetivo

Compreensão das estruturas FIFO (First In First Out) e LIFO (Last In First Out). Implementação estática e dinâmica de filas e pilhas.

3. Descrição do Problema

Dado um mapa com a localização de tesouros e a representação de um percurso, determinar em que localização o percurso termina, a distância percorrida, o número de tesouros encontrados ao longo do percurso, e em que localizações o percurso passa mais de uma vez e quantas vezes passa.
Pretende-se obter duas soluções para o problema recorrendo-se à implementação estática e à implementação dinâmica de filas e pilhas.

Observações:

1. Por localização entende-se um par (x, y) de números inteiros maiores ou iguais a zero que indicam as coordenadas de uma posição (localização) numa matriz (mapa) em termos de linhas e colunas. A localização (0,0) encontra-se no canto superior esquerdo do mapa/matriz.
2. Por percurso entende-se uma sequência de localizações adjacentes, sendo o número máximo de localizações cem (100). O percurso tem sempre inicio na localização (0,0) e a última localização visitada indica o final do percurso. Não é possível avançar de uma localização para outra na diagonal, isto é, avança-se sempre ou na horizontal ou na vertical.

4. Desenvolvimento do Trabalho

Para o desenvolvimento do trabalho será necessário implementar duas aplicações (uma para a implementação estática e outra para a dinâmica) de consola (interfaces visuais não são consideradas para a avaliação) que resolvam o problema descrito acima e a escrita de um relatório, que são descritos nas secções seguintes. Observação: não é permitido alterar a estrutura dos ficheiros nem acrescentar informação adicional, como por exemplo, caracteres de terminação.

4.1. Dados de Entrada

A aplicação receberá os dados de entrada através de dois ficheiros de texto:

- Um ficheiro, chamado “mapa.txt”, que contém:
  - Primeira linha: n m (onde 5≤**n**≤10 e 5≤**m**≤10), que são dois inteiros que representam a dimensão do mapa (**n** indica o número de linhas e **m** o número de colunas).

- Nas linhas seguintes: em cada linha, um par **(x, y)** onde 0≤**x**<**n** e 0≤**y**<**m**, que representa a localização de cada tesouro no mapa.

- Um ficheiro, chamado "movimentos.txt", que contém a representação de um percurso na forma de uma sequência de movimentos onde cada linha corresponde a um movimento. Os movimentos são sempre válidos e produzem localizações válidas.
  Movimentos possíveis:
  - **MOVE(d)** : avançar para a localização imediatamente a **d**, onde **d** pode ser **1** (Norte), **2** (Sul), **3** (Este) ou **4** (Oeste).
  - **JUMP(x,y)** : avançar para a localização **(x, y)**, percorrendo o caminho mais curto (se existirem vários, fica ao critério dos alunos escolher um). Observação: o movimento "JUMP" pode representar uma linha diagonal entre posições da matriz, para a qual não existe um caminho definido. Há que calcular (a forma de cálculo fica ao critério dos alunos) que localizações são visitadas aquando de um movimento "JUMP" para ser possível construir o percurso.
  - **LOOP(d, n)** : avançar **n** vezes na direção **d**, onde **d** pode ser **1** (Norte), **2** (Sul), **3** (Este) ou **4** (Oeste).
  - **UNDO(n)** (onde **n** ≥ 1) : anular os **n** movimentos "MOVE" e "LOOP" anteriores. Assume-se que se existe um movimento UNDO(n) então os **n** movimentos anteriores são todos movimentos "MOVE" ou LOOP".

4.2 Dados de Saída (Resultados a Obter)

Os resultados a obter deverão ser armazenados num ficheiro de texto, chamado "resultados.txt" que contém:

- Primeira linha: um par de números inteiros (x, y) que representa a localização final do percurso.
- Segunda linha: um valor inteiro que indica a distância percorrida no percurso. O movimento de uma localização para outra localização contígua possui um valor unitário (1) de comprimento. A distância de um percurso é a soma dos comprimentos referentes às localizações visitadas.
- Terceira linha: um valor inteiro que indica o número de tesouros encontrados ao longo do percurso (não é possível encontrar cada tesouro mais de uma vez).
- Linhas seguintes: em cada linha, um trio de números inteiros (x, y, n) que representa as localizações (x, y) onde o percurso passa mais de uma vez e n representa o número de passagens.

4.3 Implementação

O código deverá ser escrito tendo em conta as regras de boa programação e de acordo com as convenções seguidas na disciplina.
É da responsabilidade dos alunos decidir quais os tipos abstratos de dados e respectiva implementação que melhor se adequam à informação a armazenar. Projetos que usem tanto pilhas como filas têm melhor cotação do que os que só usem pilhas ou só usem filas.
Há que implementar duas soluções: uma estática e uma dinâmica.
Informação a armazenar:

1. Conjunto de tesouros do mapa, em que para cada tesouro existe a seguinte informação (localização):
   - **Coordenada x** (valor inteiro)

- **Coordenada y** (valor inteiro)

2. Sequência de movimentos, em que para cada movimento existe a seguinte informação:
    - **Tipo de movimento** (texto: “MOVE”, “JUMP”, “LOOP” ou “UNDO”) (observação: a informação que os parâmetros seguintes guardam varia de acordo com o tipo de movimento e no caso dos movimentos “MOVE” e “UNDO” o segundo parâmetro não guarda informação relevante)
    - **Parâmetro1** (valor inteiro)
    - **Parâmetro2** (valor inteiro)
3. Sequência de localizações visitadas no percurso, onde para cada localização existe a seguinte informação:
    - **Coordenada x** (valor inteiro)
    - **Coordenada y** (valor inteiro)
4. Conjunto de localizações por onde o percurso passa mais de uma vez e contabilização do número de passagens. Para cada localização existe a seguinte informação:
    - **Coordenada x** (valor inteiro)
    - **Coordenada y** (valor inteiro)
    - **Número de repetições n** (valor inteiro)

Sugere-se que os alunos criem uma estrutura de dados, chamada “Location” que represente uma localização. Apesar do *output* pretendido ser gravado num ficheiro, aconselha-se os alunos a escreverem um *log* no ecrã com informação relevante às operações que estão a ser realizadas nas estruturas de dados e qual a informação gerada, em especial a sequência de localizações do percurso. Esta informação é útil durante o desenvolvimento do projeto e pode ser útil no momento da discussão em benefício dos alunos dado que permite uma melhor compreensão do que foi implementado.

5. Relatório

No relatório deverão constar as seguintes secções:

- Índice.
- Descrição dos tipos abstratos de dados usados e respectiva implementação.
- Cálculo, e respectiva justificação, da complexidade dos algoritmos de:
    - Procura de tesouros encontrados ao longo do percurso.
    - Cálculo da distância percorrida.
    - Procura das localizações por onde o percurso passa mais de uma vez.
- Conclusões e Limitações.

6. Critérios de Avaliação

É obrigatório que os dados de entrada sejam lidos a partir de ficheiros e que os dados de saída sejam escritos num ficheiro. Implementações que não o façam terão uma forte penalização.
Serão atribuídas as seguintes cotações:

Descrição	Cotação
Implementação estática dos TAD	2 valores
Implementação dinâmica dos TAD	4 valores
Gerar as localizações do percurso	5 valores
Obter a localização final do percurso	1 valor
Obter a distância percorrida	1 valor
Obter o número de tesouros encontrados ao longo do percurso	2 valores
Obter as localizações por onde o percurso passa mais de uma vez e respectiva contabilização	2 valores
Relatório	3 valores

7. Regras e Instruções

O não cumprimento das regras a seguir descritas implica uma penalização na nota do trabalho prático. Se ocorrer alguma situação não prevista nas regras a seguir expostas, essa ocorrência deverá ser comunicada ao docente responsável por ATAD (Profª. Patrícia Macedo), para decisão.
Regras:

- O Mini-Projeto deverá ser elaborado por **dois alunos do mesmo docente**.
- A nota do Mini-Projeto será atribuída individualmente a cada um dos elementos do grupo após a discussão. As discussões poderão ser orais e/ou com perguntas escritas. As orais poderão ser feitas com todos os elementos do grupo presentes em simultâneo ou individualmente.
- A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos com semelhanças, quer tenham sido o original ou a cópia.
- No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente a que se destina.
- O trabalho deverá ser submetido no Moodle, no link do respectivo docente de laboratórios criado para o efeito, até às 23h55 do dia 29 de Abril de 2016. Para tal terão de criar uma diretoria com o nome: nomeAluno1_númeroAluno1-nomeAluno2_númeroAluno2 onde colocarão o ficheiro do relatório em formato pdf e as duas diretorias com os projetos CodeBlocks da implementação das aplicações desenvolvidas. Os alunos terão de submeter essa diretoria compactada no formato zip no Moodle. Apenas será permitido submeter um ficheiro.
- Não serão aceites trabalhos entregues que não cumpram na integra o ponto anterior.
- Após a entrega dos trabalhos, as datas das discussões serão publicadas.