ATAD 2017/18

Algoritmos e Tipos Abstratos de Dados

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Projeto de Recurso

GeoCaching

O *Geocaching* é uma atividade ao ar livre que consiste numa espécie de caça ao tesouro com uso de equipamento GPS. Envolve a procura de tesouros físicos chamados de *geocaches (*ou simplesmente *caches*) que são escondidos em localizações idealmente interessantes. Uma *cache* típica consiste num contentor à prova de água contendo um livro de registo e alguns pequenos bonecos para troca.

O mais relevante da modalidade consiste no seguinte: a atividade física necessária para conseguir chegar às *caches*; ficar a conhecer os locais visitados; assinar o *logbook* para provar que se esteve lá e registar a visita num site específico.

A atividade do *Geocaching* tem noções e vocabulário próprios, e.g., tipos de caches, cache sizes, difficulty/terrain, etc..

Informação adicional, incluindo significados de termos, pode ser obtida em:

- http://www.geocaching.com/about/glossary.aspx
- https://en.wikipedia.org/wiki/Geocaching

0. Regras de Elaboração

1. O projeto de recurso é **elaborado individualmente**;

2. Qualquer situação de **plágio** (parcial ou integral; inclui desenvolvimento conjunto) levará à imediata anulação dos projetos envolvidos.

3. A **data de entrega** é dia **16 de Julho de 2018 até às 11h**, via Moodle.

4. O projeto é **entregue** na data definida neste enunciado. Aplicam-se descontos de 0,5 valores por cada hora de atraso na entrega (é considerado 1h a partir de 10min volvidos da hora anterior), até a um limite de 2h; após este limite ser ultrapassado o projeto não é considerado.

1. Requisitos do Projeto

Pretende-se desenvolver um programa em C para extrair informação útil de um ficheiro de caches. Para tal o aluno deverá desenvolver um conjunto de TADs apropriados e segundo os requisitos deste enunciado.

O programa desejado consiste num interpretador de comandos que o utilizador usa para obter diversos tipos de informação, principalmente informação estatística.

1.1 Representação das Caches em Memória

Cada cache é representada, obrigatoriamente, pela estrutura de dados apresentada na Figura 1, sendo *Date* um tipo de dados apropriado para guardar uma data.

```c
#include "date.h"

typedef struct {        // Exemplo:
    char code[11];        // "GCK1JY"
    char name[101];       // "Atlantis [Pico]"
    char state[61];       // "ARQUIPÉLAGO DOS AÇORES"
    char owner[101];      // "Joao&Olivia"
    double latitude;    // 38.468917
    double longitude;   // -28.3994
    char kind[12];        // "TRADITIONAL"  opcoes: "MULTI", "PUZZLE", etc.
    char size[11];        // "REGULAR"  opcoes: "MICRO", "SMALL", "LARGE", etc.
    double difficulty; // 2.0  opcoes: 1.0, 1.5, 2.0, ..., 4.5, 5.0
    double terrain;     // 4.5  opcoes: 1.0, 1.5, 2.0, ..., 4.5, 5.0
    char status[11];      // "AVAILABLE" opcoes: "AVAILABLE", "DISABLED"
    Date hidden_date;   // "2004/07/20"
    int founds;         // 196
    int not_founds;     // 25
    int favourites;     // 48
    int altitude;       // 2286  (-9999999 significa "altitude desconhecida")
} Cache;
```

Figura 1: Representação de uma Cache em memória.

1.2 Representação das Caches em Ficheiro

As caches encontram-se descritas num ficheiro CSV, uma por linha, no seguinte formato:

```
<code>; <name>; <state>; <owner>; <latitude>; <longitude>; <kind>; <size>; <difficulty>; <terrain>; <status>; <hidden_date>;
<founds>; <not_founds>; <favourites>; <altitude>
...
...
```

Os valores correspondentes aos campos de cada cache encontram-se na ordem apresentada no tipo de dados da Figura 1. Nessa figura também são apresentados exemplos de possíveis de valores para esses campos.

O valor <*hidden_date*> encontra-se no formato "aaaa/mm/dd".

Desconhece-se a altitude de algumas caches e utiliza-se o valor especial -9999999 (7 noves) para representar esta situação.

Pode-se assumir que não existem ficheiros "mal-formados", i.e., todas as linhas contêm 16 valores separados por ponto-e-vírgula. Contudo, pode haver linhas em branco.

Atenção que um ficheiro vazio também é um ficheiro válido!

1.2.1 Ficheiros Disponibilizados

Juntamente com este enunciado é disponibilizado 1 ficheiro de entrada:

- *caches.csv*

Este ficheiro possui um número "irrelevante" de caches; a aplicação deverá suportar um número arbitrário de caches de entrada.

1.3 Tipos Abstratos de Dados

Uma das componentes deste projeto visa a implementação e uso de tipos abstratos de dados para armazenar a informação necessário.

As caches deverão ser importadas para uma instância de cada um dos TAD descritos nesta secção.

Os TAD cujo programa deve utilizar são:

- **SetList**;
- **Map**;

SetList

Este TAD é uma variante do TAD List que não permite duplicados; o aluno deverá definir a sua interface, tipos de dados necessários e a sua implementação (escolha a estrutura de dados mais eficiente).

A sua definição e implementação é **obrigatória**.

Map

Este TAD já foi lecionado nas aulas. Contudo pretende-se uma implementação utilizando tabelas de dispersão e resolução de colisões utilizando a técnica de **Open Addressing**.

Utilize a interface lecionada e forneça a sua implementação utilizando uma estrutura de dados apropriada.

Caso não consiga, pode optar por uma implementação com listas duplamente ligadas com sentinelas, apesar de uma forte penalização.

1.4 Comandos

Há exatamente 10 comandos que o programa deve implementar, que serão apresentados de seguida.

Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas, não importa.

Quando a memória de caches estiver vazia, a maioria dos comandos limitam-se a escrever a mensagem "Nao existem caches".

O formato de output de cada comando fica ao critério do aluno. Os resultados obtidos deverão ser validados autonomamente, e.g., através de EXCEL.

Os comandos são os seguintes:

- ✓ **LOAD**
  - Pede o nome dum ficheiro, abre o ficheiro e carrega-o em memória (ver Secção 2.3), mostrando o número de caches importadas. Os restantes comandos passarão a atuar sobre o novo conteúdo dos tipos abstratos de dados. Se o ficheiro não puder ser aberto, escreve "Ficheiro nao existe..." e os tipos abstratos de dados ficam vazios.

    É esperado que a instância de ListSet "filtre" automaticamente caches duplicadas (pelo seu código) contidas no ficheiro de entrada.

- ✓ **CLEAR**
  - Limpa os dados importados.

- ✓ **FOUNDR**
  - Mostra a percentagem de vezes que cada cache foi encontrada.

- ✓ **CENTER**
  - Mostra as seguintes estatísticas, pela ordem indicada:
    - média das latitudes seguida do desvio padrão;
    - média das longitudes seguida do desvio padrão;

- ✓ **SORT**
  - Mostra as caches ordenadas, requerendo ao utilizador a forma de ordenação (atenção aos critérios de desempate):
    - 1 – Por *owner* (crescentemente, desempate por *favourites*);
    - 2 – Por altitude (decrescentemente e ignorando altitudes desconhecidas);
    - 3 – Pela decrescente distância (euclidiana) relativa à média das latitudes/longitudes.

    Este comando deverá utilizar obrigatoriamente uma instância auxiliar do TAD List para efetuar as ordenações.

✓ **DATES**
- Mostra a cache mais antiga e a mais recente, por esta ordem, juntamente com a diferença em meses entre ambas (ignore os dias).

✓ **SIZES**
- Mostra a contagem de caches para cada tamanho diferente (campo *size*); não pode assumir que apenas existem os tipos contidos no ficheiro disponibilizado.

✓ **M81P**
- Mostra as percentagens da "matriz 81". A matriz 81 trata-se de uma matriz 9x9 com o número de caches para cada uma das 81 combinações de terreno/dificuldade. A Figura 2 mostra um exemplo (ignore as cores). Pretende-se que o valor apresentado em cada célula seja a percentagem relativa ao total.

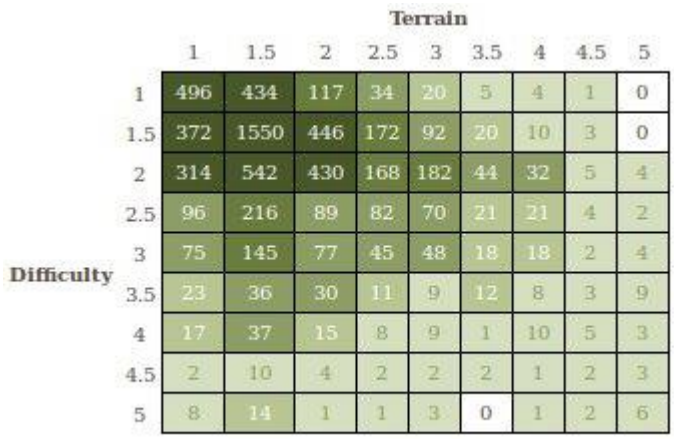

Figura 2: Exemplo de uma "matriz 81". Pretende-se que o valor efetivamente mostrado seja a percentagem relativa.

✓ **SEARCH**
- Pede um código de cache (string) e mostra a cache correspondente. Se não existir, deve mostrar a mensagem "Cache nao encontrada.".
  Esta operação deve obrigatoriamente utilizar a instância do TAD Map.

✓ **QUIT**
- Sai do programa, libertando toda a memória alocada para os tipos abstratos de dados.

2 Relatório

No relatório deverão constar as seguintes secções:

a) Descrição dos tipos abstratos de dados usados e respetivas implementações;
   - Deverá colocar uma tabela com as complexidades algorítmicas de cada operação do TAD;

b) Para cada comando (exceto LOAD/QUIT) fornecer:
   - O código/função respeitante à funcionalidade;
   - A complexidade algorítmica da respetiva implementação, tendo em conta a complexidade algorítmica das funções dos TADs utilizadas na mesma;

c) Limitações:
   - Quais os comandos que apresentam problemas ou não foram implementados;

d) Conclusões:
   - Análise crítica do trabalho desenvolvido.

3 Tabela de Cotações e Penalizações

A avaliação do trabalho será feita de acordo com os seguintes princípios:

- Estruturação: o programa está estruturado de uma forma modular e procedimental;
- Correção: o programa executa as funcionalidades, tal como pedido.
- Legibilidade e documentação: o código está escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.

A nota final obtida, cuja tabela de cotações se apresenta a seguir, será ponderada de acordo com os princípios acima descritos.

Descrição	**Cotação (valores)**
LOAD+CLEAR+QUIT	2
Comandos FOUNDR	1
Comando CENTER	1
Comando SORT	2
Comando DATES	1
Comando SIZES	2
Comando M81P	2
Comando SEARCH	2
Definição e Implementação do TAD ListSet	1,5
TAD Map	
-- Implementação c/ Lista Ligada	1
-- Implementação c/ Open Addressing	3,5
Relatório	2
TOTAL	20

A seguinte tabela contém penalizações a aplicar:

Descrição	**Penalização (valores)**
Uso de variáveis globais	até 2
Não separação de funcionalidades em funções	até 2
Fraca modularidade	até 2

4 Instruções e Regras Finais

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 respetivo docente de laboratório de ATAD.

Regras:

a) O Projeto é ser elaborado por **individualmente**.

b) A nota do Projeto será atribuída após a discussão. As discussões poderão ser orais e/ou com perguntas escritas. A não comparência na discussão atribuirá ao projeto a nota zero.

c) 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.

d) No rosto do relatório e nos ficheiros de implementação deverá constar o número e nome do autor.

e) O trabalho deverá ser submetido no moodle, no link do respetivo docente de laboratórios criado para o efeito, até às **11h do dia 16 de Julho de 2018 (segunda-feira)**. Para tal terão que criar uma pasta com o nome: **nomeAluno1_númeroAluno1**, onde colocarão o ficheiro do relatório em formato pdf e a pasta do projeto Visual Studio. O aluno terá de submeter essa pasta compactada. Apenas será permitido submeter um ficheiro.

f) As datas das discussões serão publicadas após a entrega dos trabalhos.

(fim de enunciado)