AFP 2023/24

Algoritmos e Fundamentos de Programação

Projeto de Época Normal

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 época normal é **elaborado em grupos de, no máximo, dois estudantes**;

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

    a. Todos os projetos (de todas as turmas) serão alvo de deteção automática de plágio, utilizando a plataforma MOSS.

3. As **entregas** serão feitas exclusivamente via Moodle, nas datas indicadas na secção 3.

    a. Aplicam-se descontos de 0,5 valores por cada hora de atraso na entrega (é considerado 1h a partir de 10min volvidos da hora agendada), até a um limite de 3h; 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 localizadas em Portugal. O programa desejado consiste num interpretador de comandos que o utilizador usa para obter diversos tipos de informação, principalmente informação estatística.

1.1 Informação de caches

As caches encontram-se descritas num ficheiro CSV. Cada cache é descrita pela seguinte informação (nomes das colunas):

- **`code`** – código da cache;
- **`name`** – nome da cache;
- **`state`** – distrito onde a cache está localizada;
- **`owner`** – utilizador que escondeu a cache;
- **`latitude`** – latitude aproximada da cache (localização);
- **`longitude`** – longitude aproximada da cache (localização);
- **`kind`** – tipo de cache. Pode ser um dos seguintes valores:
  - EARTHCACHE
  - LETTERBOX
  - MULTI
  - PUZZLE
  - TRADITIONAL
  - VIRTUAL
  - WEBCAM
- **`size`** – tamanho da cache. Pode ser um dos seguintes valores:
  - MICRO
  - SMALL
  - REGULAR
  - LARGE
  - OTHER_SIZE
  - VIRTUAL
  - NOT_CHOSEN
- **`difficulty`** – dificuldade prevista de localizar a cache. Um valor em [1, 5];
- **`terrain`** – caracterização do terreno onde se encontra a cache. Um valor em [1, 5];
- **`status`** – estado da cache. Pode ser um dos seguintes valores:
  - AVAILABLE
  - DISABLED
- **`hidden_date`** – data em que a cache foi escondida. No formato "aaaa/mm/dd";
- **`founds`** – quantos caçadores encontraram a cache;
- **`not_founds`** – quantos caçadores não encontraram a cache;
- **`favourites`** – quantos caçadores marcaram a cache como favorita;
- **`altitude`** – altitude em que se encontra a cache;

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

Pode 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.

1.1.1 Ficheiros Disponibilizados

Juntamente com este enunciado são disponibilizados 2 ficheiros de entrada:

- `caches_small.csv`
- `caches_all.csv`

Cada ficheiro possui um número arbitrário de caches, contendo caches duplicadas; pode assumir que nunca existirão mais de 2000 caches num ficheiro.

Cada cache tem um código respetivo (campo **code**). Pode detetar caches duplicadas quando um código de cache ocorrer mais que uma vez. Caches duplicadas deverão ser ignoradas durante o processo de importação.

O programa deverá funcionar corretamente para um qualquer ficheiro de entrada.

Adicionalmente, é fornecido o ficheiro `caches_all.xlsx` que contém a informação do ficheiro `caches_all.csv` em formato Excel. Pode utilizar este ficheiro para inspecionar a informação de antemão e para validar o resultado de alguns comandos posteriormente.

1.2 Representação da informação em memória

Deverão ser definidos tipos compostos apropriados para representar caches e conjuntos de caches. É obrigatório representar uma data através de um tipo composto apropriado.

Sugere-se a reprodução das estratégias apresentadas nos Labs 9 e 10.

Pode definir outros tipos auxiliares que julgue necessitar para a implementação de certos comandos.

É obrigatória a utilização correta de modularidade para estes tipos de dados.

1.3 Interpretador de Comandos

Deverá existir um interpretador de comandos que apresenta ao utilizador os comandos existentes (ver secção 1.4) e que permita ao utilizador executar um dado comando.

Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas (*case-insensitive*).

1.4 Comandos

Há exatamente 13 comandos que o programa deve implementar, que serão apresentados de seguida.

Quando a memória de caches estiver vazia (ainda não houve carregamento de dados), a maioria dos comandos limitam-se a escrever a mensagem "<no cache data>".

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**
  - Solicita o nome dum ficheiro, abre o ficheiro e carrega-o em memória.
  - No final da importação deve ser mostrado ao utilizador quantas caches "únicas" foram importadas, e.g., "<X unique caches loaded>".

    Os restantes comandos passarão a atuar sobre este conjunto de dados.
  - Se o ficheiro não puder ser aberto, escreve "<File not found>".
  - Se já existirem dados carregados em memória, deverá apresentar a mensagem "<Cache data exists. Please clear it first.">

- ✓ **CLEAR**
  - Limpa os dados importados da memória; isto irá permitir carregar outro ficheiro. Apresenta uma mensagem de confirmação.

- ✓ **QUIT**
  - Sai do programa.

- ✓ **LIST**
  - Apresenta uma listagem de todas as caches importadas (preferencialmente numa forma tabular). Deve ser visível toda a informação de cada cache em cada linha.

- ✓ **FOUNDP**
  - Semelhante ao comando LIST, mas no início de cada linha mostra a percentagem de vezes que essa cache foi encontrada.

- ✓ **SEARCH**
  - Pede um código de cache e mostra a informação da cache correspondente. Se não existir, deve mostrar a mensagem "<Cache not found>".

- ✓ **EDIT**
  - Pede um código de cache e permite editar informação dessa cache. Se não existir, deve mostrar a mensagem "<Cache not found>".
  - Deve ser possível editar os seguintes dados (só poderão ser aceites dados válidos!):
    - `owner`
    - `status`
    - `hidden_date`
    - `altitude`

  A informação editada, por enquanto, é mantida apenas em memória.

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

  - Caches com altitudes desconhecidas devem ser ignoradas nestes cálculos.

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

- ✓ **SORT**
  - Mostra uma listagem ordenada das caches, requerendo ao utilizador a forma de ordenação:
    - 1 – Por **`altitude`** (decrescentemente);
    - 2 – Por **`state`** (de A-Z, desempate por **`founds`** de forma decrescente);
    - 3 – Por **`hidden_date`** (mais recente para mais antiga).

- ✓ **STATEC**
  - Mostra a contagem de caches para cada distrito (campo **`state`**). Deve mostrar separadamente as contagens de caches *disponíveis* e *inativas* (campo **`status`**).

- ✓ **M81**
  - Calcula a "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 1 mostra um exemplo (ignore as cores e os valores atuais).

Difficulty \ Terrain	1	1.5	2	2.5	3	3.5	4	4.5	5
1	496	434	117	34	20	5	4	1	0
1.5	372	1550	446	172	92	20	10	3	0
2	314	542	430	168	182	44	32	5	4
2.5	96	216	89	82	70	21	21	4	2
3	75	145	77	45	48	18	18	2	4
3.5	23	36	30	11	9	12	8	3	9
4	17	37	15	8	9	1	10	5	3
4.5	2	10	4	2	2	2	1	2	3
5	8	14	1	1	3	0	1	2	6

Figura 1: Exemplo de uma "matriz 81".

- ✓ **SAVE**
  - Solicita ao utilizador o nome de um ficheiro e exporta a informação de caches atualmente em memória para ficheiro, em formato CSV. Não deve permitir a exportação se já existir um ficheiro com esse nome.

2. Relatório

No relatório deverão constar as seguintes secções:

a) Capa com identificação dos estudantes;

b) Módulos e Tipos de Dados;

- Descrição dos módulos e tipos compostos desenvolvidos.

c) Algoritmos;

- Para os comandos (exceto LOAD, CLEAR, QUIT, SAVE e LIST) fornecer a descrição dos algoritmos implementados:
  - 2 algoritmos em pseudo-código (escolha ao critério dos estudantes)
  - Restantes em "linguagem natural", i.e., passos gerais, não se pretende uma "transcrição" do código para português.

d) Limitações:

- Quais os comandos que apresentam problemas ou não foram implementados;

e) Conclusões:

- Análise crítica do trabalho desenvolvido.

3. Datas de Entrega e Deliverables

1ª FASE (5 janeiro)

- Projeto C com programa funcional que:
  - Define tipos de dados compostos para representação de caches e conjuntos de caches;
  - Interpretador de comandos;
  - Comandos LOAD, CLEAR, QUIT, LIST e FOUNDP.

2ª FASE (26 janeiro)

- Projeto C com programa funcional que:
  - Implementa a totalidade dos comandos;
- Relatório, descrito na secção 2.

4. 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;
- Eficiência: o programa utiliza adequadamente a passagem de parâmetros por valor e por referência.
- 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)**	**Fase**
Definição de estruturas de dados	2	**1**
Implementação de interpretador de comandos	1,5	
Importação de informação (LOAD)	2	
Comandos CLEAR e QUIT	0,5	
Comando LIST	1	
Comando FOUNDP	1	
Comando SEARCH	1	**2**
Comando EDIT	2	
Comando CENTER	1	
Comando AGE	1	
Comando SORT	2	
Comando STATEC	1	
Comando M81	1	
Comando SAVE	1	
Relatório	2	
TOTAL	20	

A seguinte tabela contém penalizações a aplicar:

Descrição	Penalização (valores)
Uso de variáveis globais	até 3
Não utilização de módulos/procedimentos adequados	até 3

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 AFP.

Regras:

a) O projeto é elaborado em grupos de, no máximo, dois alunos.

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) Os projetos são submetidos no Moodle, dentro de um arquivo (formato ZIP), contendo todos os *deliverables*. Apenas será permitido submeter um ficheiro.

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

(fim de enunciado)