Trabalho prático 1 – implementação de biblioteca de vetores de registos

1. Informação geral

O trabalho prático 1 consiste na implementação de alterações e de funções adicionais a incorporar na biblioteca de funções para manipulação de vetores de registos.

Este trabalho deverá ser feito de forma autónoma por cada grupo até à data limite estabelecida. A consulta de informação nas diversas fontes disponíveis é aceitável. No entanto, o código submetido deverá ser apenas da autoria dos elementos do grupo e quaisquer cópias detetadas serão devidamente penalizadas. A incapacidade de explicar o código submetido por parte de algum elemento do grupo implicará também uma penalização.

O prazo-limite para submissão (através do Moodle) é o dia **28 de Março às 21:00h**.

2. Conceito

Os EUA decidiram criar um cofre de sementes semelhante à iniciativa em Svalbard de 2006 (https://en.wikipedia.org/wiki/Svalbard_Global_Seed_Vault). Para tal, precisam de uma base de dados para catalogar e gerir as espécies que guardam.

3. Implementação do trabalho

O arquivo comprimido PROG2_2021_T1.zip contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `plantas.h`: declarações das funções da biblioteca do vetor colecao e dos elementos planta
- `plantas.c`: ficheiro onde deverão ser implementadas as funções pedidas, relativas à biblioteca `plantas.h`
- `colecao-teste.c`: inclui o programa principal que invoca e realiza testes básicos às funções implementadas
- db_small.txt: ficheiro de texto com informações sobre a coleção existente

Notas importantes:

1. **Apenas deverá ser alterado o ficheiro plantas.c que será o único a ser considerado na submissão dos trabalhos.**
2. **Detalhes adicionais sobre as funções (a implementar) poderão ser encontradas junto ao protótipo respetivo em plantas.h**

Os registos de dados `planta` e `colecao` são a base da biblioteca e têm a seguinte declaração:

```
typedef struct
{
  /** identificacao unica no catalogo **/
  char ID[10];

  /** designacao cientifica **/
  char  nome_cientifico[MAX_NAME];

  /** lista de nomes locais dados a especie **/
  char **alcunhas;

  /** numero total de alcunhas da planta **/
  int n_alcunhas;

  /** numero de sementes guardadas no cofre **/
  int n_sementes;

} planta;
```

Neste registo *planta* são guardados: 1) a identificação (`ID`) da planta; 2) o seu nome científico (`nome_cientifico`); 3) um possível *array* de nomes adicionais por que a planta é conhecida (`alcunhas`); 4) um inteiro com o número de alcunhas da planta (`n_alcunhas`); 5) um inteiro com o número de sementes guardadas no cofre (`n_sementes`).

O registo de dados `colecao` aponta para um *vetor* de plantas. Inclui também o número de posições válidas do *vetor* (`tamanho`) e tipo de ordenação presente na coleção (`tipo_ordem`).

```
typedef struct
{
  /** apontador para o array de plantas armazenadas **/
  planta **plantas;


   /** tamanho do vetor de plantas **/
  long tamanho;

   /** indicacao do campo para ordenacao crescente do vetor: 'nome'
   - para nome_cientifico; 'id' - para campo de identificacao unica
   **/
  char tipo_ordem[5];

} colecao;
```

As funções a implementar neste trabalho encontram-se no ficheiro `plantas.c` e são:

1. **planta *planta_nova** (const char *ID, const char *nome_cientifico, char **alcunhas, int n_alcunhas, int n_sementes);
   Cria uma nova instância do registo `planta`*, copiando cada um dos argumentos para o respetivo campo. Todos os campos devem apresentar valores válidos, com a exceção de* `alcunhas`*, que pode ser* `NULL` *caso não haja nomes locais para a planta. A função retorna o apontador para a* `planta` *criada ou* `NULL` *em caso de erro.*
2. **colecao *colecao_nova**(const char *tipo_ordem);
   Cria uma nova instância vazia do registo colecao*. Deve retornar um apontador para o registo. Em caso de erro deverá retornar* `NULL`*.*
3. **int planta_insere**(colecao *c, planta *p);
   Insere a `planta` *apontada por* `p` *na* `colecao` *apontada por* `c`*, na posição correta de acordo com o* `tipo_ordem`*. Caso a planta já exista (tenha o mesmo* `ID`*) deve somar o número de sementes e acrescentar as novas alcunhas, sem duplicação. Caso contrário, deverá acrescentar o registo à coleção. Retorna zero se a planta ainda não existia e foi inserida com sucesso, 1 se a planta já existia e apenas foi atualizada e -1 em caso de erro.*
4. **int colecao_tamanho**(colecao *c);
   Retorna o número de espécies diferentes de plantas armazenadas no registo `colecao` *dado. Deve retornar um inteiro ou -1 em caso de erro.*
5. **colecao* colecao_importa**(const char *nome_ficheiro, const char *tipo_ordem);
   Importa uma coleção presente em `nome_ficheiro`*, retornando-a via argumento de saída. A coleção retornada deve vir ordenada de acordo com o* `tipo_ordem` *dado. Em caso de insucesso retorne* `NULL`*. Cada linha do ficheiro corresponde a uma planta e tem o seguinte formato: ID, nome científico, número de sementes, alcunha0, alcunha1, ..., alcunhaN*
6. **planta* planta_remove**(colecao *c, const char *nomep);
   Remove a `planta` *de nome científico* `nomep` *da* `colecao` *apontada por* `c` *e corrige a coleção no ponto retirado. Deverá retornar um apontador para a planta removida. Em caso de insucesso, deverá retornar* `NULL`*.*
7. **int planta_apaga**(planta *p);
   Elimina o registo `planta` *apontado por* `p` *e liberta toda a sua memória alocada. Em caso de sucesso deverá retornar o valor zero e -1 se insucesso.*
8. **int colecao_apaga** (colecao *c);
   Elimina a coleção `c`*, e todas as suas plantas, libertando toda a memória por ela ocupada. A função deverá retornar zero se bem-sucedida e -1 em contrário.*
9. **int *colecao_pesquisa_nome**(colecao *c, const char *nomep, int *tam);
   Retorna um vetor de índices das plantas que apresentarem a totalidade do `nomep` *no* `nome_cientifico` *ou em* `alcunhas`*. Retorna por referência o tamanho do vetor de índices através de* `tam`*. Em caso de insucesso, deverá retornar* `NULL`*.*
10. **int colecao_reordena**(colecao *c, const char *tipo_ordem);
    Reordena a coleção `c` *de acordo com o* `tipo_ordem` *e atualiza esse campo na coleção* `c`*. Em caso de insucesso retorna -1, se não for necessário reordenar retorna zero ou, se for necessário reordenar, retorna 1 depois de feita a reordenação.*

Nota: Os ficheiros de entrada e de teste em que serão avaliados os programas submetidos poderão apresentar conteúdo diferente e incluir casos limite: argumentos de funções com gamas não previstas. Como tal, é sua responsabilidade garantir que os argumentos são devidamente testados de forma a só aceitar quando dentro da gama prevista.

4. Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa `colecao-teste`. Existe um teste por cada função a implementar e que determina se essa função tem o comportamento esperado. Note que os testes não são exaustivos. Por isso, os testes devem ser considerados apenas como um indicador de uma aparente correta implementação das funcionalidades esperadas.

Estando as funções corretamente implementadas, o programa `colecao-teste` quando executado deverá apresentar o seguinte resultado:

```
INICIO DOS TESTES

...verifica_planta_nova: planta nova nao e' NULL (ok)
...verifica_planta_nova: ID coincide com o esperado (= SALAM2) (ok)
...verifica_planta_nova:  Nome cientifico coincide com o esperado (= Sagittaria
lancifolia L. ssp. media (Micheli) Bogin) (ok)
...verifica_planta_nova: Numero de sementes coincide com o esperado (= 60) (ok)
...verifica_planta_nova: Numero de alcunhas coincide com o esperado (= 2) (ok)
...verifica_planta_nova: O conteudo do vetor alcunhas coincide com o esperado (ok)
OK: verifica_planta_nova passou

(...)

...verifica_colecao_pesquisa_nome (teste de um nome que nao existe): retornou NULL
(ok)
...verifica_colecao_pesquisa_nome (teste de um nome que existe): numero de indices
encontrados coincide com o esperado (= 2) (ok)
OK: verifica_colecao_pesquisa_nome passou

...verifica_colecao_reordena: 1ª e 'ultima plantas coincidem com o esperado (= ACSP5
e YUSM) (ok)
OK: verifica_colecao_reordena passou

FIM DOS TESTES: Todos os testes passaram
```

5. Ferramenta de desenvolvimento

A utilização de um IDE ou do Visual Studio Code é aconselhável no desenvolvimento deste trabalho uma vez que permite fazer depuração de uma forma mais eficaz. Poderá encontrar informações sobre a utilização do Visual Studio Code num breve tutorial disponibilizado no Moodle.

6. Avaliação

A classificação do trabalho é dada pela avaliação feita à implementação submetida pelos estudantes. A classificação final do trabalho (T1) é dada por:

T1 = 0.8 Implementação + 0.2 Memória

A classificação da implementação é essencialmente determinada por testes automáticos adicionais (por exemplo, recorrendo a ficheiros de teste de maiores dimensões). No caso de a implementação submetida não compilar, esta componente será 0%.

A gestão de memória também será avaliada, sendo considerados 3 patamares: 100% nenhum memory leak, 50% alguns memory leaks mas pouco significativos, 0% muitos memory leaks.

7. Teste em servidor

Em breve será disponibilizado um servidor para que possam testar o vosso código durante o desenvolvimento. O código submetido neste servidor **NÃO SERÁ AVALIADO**. Apenas a submissão via moodle é válida para efeitos de avaliação.

8. Submissão da resolução

A submissão é apenas possível através do Moodle e até à data indicada no início do documento. Deverá ser submetido um ficheiro *zip* contendo:

- o ficheiro plantas.c com as funções implementadas
- um ficheiro autores.txt indicando o nome e número dos elementos do grupo

Nota importante: apenas as submissões com o seguinte nome serão aceites: `T1_G<numero_do_grupo>.zip`. Por exemplo: "`T1_G999.zip`"