Programação - Exame de Recurso

Eng.ª Informática; Eng.ª Informática – Pós-laboral; Eng.ª Informática – Curso Europeu

Duração: 2h30m 1/07/2019

1. No âmbito de um serviço simplificado de aluguer de trotinetes da cidade de Coimbra, pretende-se desenvolver um programa que auxilie a gestão da recolha diária de trotinetes para recarga e manutenção. Através de uma aplicação já existente é disponibilizado um ficheiro de texto com o registo dos locais e estado das diversas trotinetes.

trotinetes.txt

```
23, 40.193, -8.404, 40, ok
15, 40.219, -8.422, 92, avaria
54, 40.201, -8.441, 37, ok
76, 40.213, -8.412, 67, ok
8, 38.712, -9.153, 23, alarme
143, 40.200, -8.434, 38, ok
```

A informação sobre cada trotinete encontra-se numa única linha que contém diversos campos separados por ',' (vírgula), de acordo com o seguinte formato: Id único, latitude, longitude, carga, estado (pode ser *ok*, *avaria* ou *alarme*). O registo das várias trotinetes encontra-se em linhas consecutivas. Na primeira linha do ficheiro exemplificado pode verificar-se que a trotinete 23 está na localização (40.193, -8.404), tem 40% de carga e está no estado ok.

Existe um outro ficheiro que possui informação binária sobre os colaboradores da empresa encarregues pela recolha. A informação sobre cada colaborador está armazenada seguindo o formato da seguinte estrutura:

```
typedef struct {
      int id;           //id único do colaborador
      char nome[50];
      float latBase,longBase; //latitude e longitude do seu local base
      int ativo;  //1 se o colaborador está ao serviço ou 0 se não está
} Colab, *pColab;
```

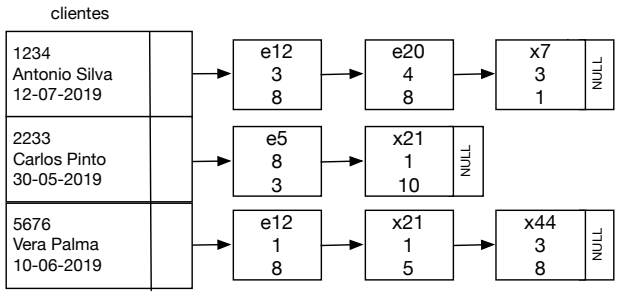
a [3.5 vals.]. Escreva uma função em C que crie um vetor dinâmico de estruturas do tipo *Colab* com informação sobre todos os colaboradores ativos. A função recebe como argumentos o nome do ficheiro binário de colaboradores e o endereço de uma variável inteira onde deve ser colocada a dimensão do vetor dinâmico. Devolve o endereço do vetor criado pela função, ou NULL, em caso de erro.

b [5.5 vals.] Pretende-se criar 2 ficheiros de texto com um resumo da informação das trotinetes que precisam de ser recolhidas. O ficheiro *"avarias.txt"* deve conter uma linha com informação de cada trotinete que se encontra no estado *avaria* ou *alarme*. A linha relativa a cada trotinete deve ter o seguinte formato: *Id único da trotinete, latitude, longitude, estado*. O ficheiro *"recarga.txt"* deve conter uma linha com informação de cada uma das trotinetes que não têm nenhum problema (estão no estado *ok*), mas precisam de ser recarregadas por terem uma carga inferior a 60%. Neste ficheiro, a linha relativa a cada trotinete deve ter o seguinte formato: *Id único da trotinete, latitude, longitude, carga, id do colaborador que vai fazer a recolha*. A seleção do colaborador que vai fazer a recolha obedece a 2 critérios: tem que estar ativo e é o que tem o local base mais próximo do local onde se encontra a trotinete. Pode assumir que existe pelo menos um colaborador ativo.
A função recebe os nomes do ficheiro original de trotinetes e do ficheiro de colaboradores como argumentos.

Ao responder a esta questão pode assumir a existência da função `dist()` que, dadas as latitudes e longitudes de 2 pontos A e B, devolve a distância entre eles.

```
float dist(float latA, float longA, float latB, float longB);
```

2. O ginásio "Sempre em forma" faz a gestão dos treinos dos seus clientes utilizando um array dinâmico de estruturas (1 por cliente). De cada uma destas estruturas sai uma lista ligada de exercícios que constituem o seu treino normal. Na figura ao lado pode consultar um exemplo.



Os tipos estruturados seguintes são usados na implementação:

```
typedef struct e Exercicio, *pExercicio;
typedef struct c Cliente, *pCliente;

struct c{
    int id;             // id inteiro único do cliente
    char nome[80];
    int dia, mes, ano; //data limite para atualizar a lista de exercícios
    pExercicio treino;
};

struct e{
      char id[20];            // id alfanumérico único
      int series, repeticoes; // Número de séries e repetições para este
                              //exercício
      pExercicio prox;
};
```

a [4 vals]. Escreva uma função em C que receba o endereço e dimensão do vetor dinâmico de clientes como parâmetros e que escreva na consola o *id* dos exercícios que apenas surgem no plano de treino de um dos clientes.

b [5 vals]. Escreva uma função em C que receba uma estrutura dinâmica com as características descritas em cima (clientes e planos de treino). Esta função deve criar uma nova estrutura dinâmica com as mesmas propriedades, mas para onde só são copiados os clientes (e respetivas listas de exercícios) cujo limite para atualizar os exercícios seja superior a uma determinada data. A estrutura dinâmica que é recebida não deve ser modificada, ou seja, o espaço necessário para a nova estrutura dinâmica deve ser todo alocado pela função. A função tem o seguinte protótipo:

```
pCliente criaED(pCliente a, int dim, int* nDim, int d, int m, int a);
```

Recebe como parâmetros o endereço e dimensão do vetor dinâmico original, o endereço de uma variável inteira onde deve ser colocada a dimensão do vetor que for criado e a data limite a considerar. Devolve o endereço do novo vetor dinâmico criado.

3 [2 vals.]. Duas árvores binárias são idênticas se tiverem a mesma estrutura e o mesmo conteúdo em cada um dos nós. Escreva uma **função recursiva** em C que verifique se duas árvores binárias constituída por nós do tipo *no* são idênticas.

```
typedef struct valor no, *pno;
struct valor{
      int num;
      pno dir, esq;
};
```

A função recebe os endereços das raízes das 2 árvores. Devolve 1 se forem idênticas, ou 0, caso contrário.