Trabalho nº 1 de ADA

Sistema de Gestão de Chamadas para Call Center de Suporte Técnico

Abril de 2004

Pretende-se desenvolver um sistema de gestão de chamadas para um *call center*. O sistema deve permitir identificar o cliente a partir do seu número de telefone, bem como aceder aos dados de todas as chamadas anteriores do cliente, incluindo o operador que atendeu e os dados do problema que levou o cliente a telefonar. Assim é possível um melhor diagnóstico dos problemas do cliente, por exemplo: se um problema é reincidente, tentar dar uma solução melhor que da última vez, ou mesmo tentar redireccionar para o operador que atendeu da última vez.

1 Operações a implementar

1. Inserção de dados:

   (a) `void novo_cliente(char *telef, char *nome, char *morada)`
   Insere um novo cliente na(s) estrutura(s) de dados. Não verifica se o cliente já existe ou não.

   (b) `void nova_chamada(char *telef, char *operador, char *problema)`
   Insere uma nova chamada de um cliente na(s) estrutura(s) de dados.

2. Listagens e pesquisas:

   (a) `struct Cliente *mostra_cliente(char *telef)`
   Devolve os dados de um cliente, se este existir. Esta função deve ser chamada sempre que for necessário saber se um cliente existe ou não.

   (b) `struct Chamada **lista_chamadas_cliente(char *telef)`
   Devolve a lista de todas as chamadas telefónicas do cliente cujo nº de telefone é 'phonenumber', ordenadas cronologicamente. A lista é uma sequência de apontadores (struct Chamada *), terminada por um apontador para NULL.

Exemplo:

...

```
struct Chamada **lst;

// alocar a lista:
lst = (struct Chamada **) malloc (sizeof (*lst) * 3); /* 2 items */

// alocar cada elemento da lista:
lst[0] = (struct Chamada *) malloc (sizeof (**lst));
lst[1] = (struct Chamada *) malloc (sizeof (**lst));
lst[2] = NULL;  /* fim da lista */

// dar valores:
lst[0].telef = strdup(``111222333'');
```

...

(c) `struct Cliente **lista_clientes(short ord)`
Devolve a lista de clientes, ordenada de uma das seguintes formas:

- Se `ord == 1`, ordena por ordem crescente de nº de telefone
- Se `ord == 2`, ordena por ordem alfabética de nomes

(d) `struct Chamada **lista_chamadas(short ord)`
Devolve a lista de chamadas, ordenada de uma das seguintes formas:

- Se `ord == 1`, ordena por ordem crescente de nº de telefone e data da chamada
- Se `ord == 2`, ordena por ordem alfabética de nome de cliente e data
- Se `ord == 3`, ordena por ordem alfabética de nome de operador e data

(e) `struct Cliente **lista_clientes_dia(char *data)`
Devolve a lista dos clientes que ligaram no dia 'data'

(f) `struct Cliente **lista_clientes_nvezes(int nvezes)`
Devolve a lista dos clientes que:

- Se `nvezes > 0`, ligaram *mais de* 'nvezes' vezes
- Se `nvezes < 0`, ligaram *menos de* 'abs(nvezes)' vezes

2 Entrega

- O trabalho deve ser entregue até 30/05/2004 (inclusivé), na forma de um ficheiro .tar.gz, enviado para o docente das práticas por email.
- O nome do ficheiro (excluindo a extensão) deve ser igual ao mínimo dos números de aluno dos elementos do grupo.
- O ficheiro deve descompactar para uma directoria com o mesmo nome do ficheiro.
- O assunto do email deve ser "ADA: trabalho".
- Recomenda-se que os trabalhos sejam enviados a partir das contas de email da universidade, para evitar que sejam "apanhados" por filtros de spam (acontece frequentemente).
- O ficheiro enviado deve conter o seguinte:
  - Relatório (em .txt) onde é explicada e justificada a escolha das estruturas de dados utilizadas e a complexidade temporal obtida nas operações implementadas;
  - Makefile a funcionar e que inclua um target 'teste:' (make test deve executar a aplicação);
  - Ficheiro 'main.c', que contém apenas a função 'main()';
  - Todos os outros ficheiros de código necessários ao funcionamento do trabalho.
- Em casos de atraso na entrega, serão descontados 2^{n-1} valores na nota do trabalho, sendo n o número de dias em atraso.