Programação Estruturada

Exercícios — folha para 2 aulas práticas

2000/2001

Nesta aula deverá implementar os seguintes exercícios que foram propostos nas aulas teórico-práticas (ver página da disciplina):

1. Máquina de pilha para manipular fracções
2. Factoriais com impressões intermédias
3. Ordenação/pesquisa com árvore binária

Exercício de revisão

1. Dados: Na primeira linha é dado o inteiro n $n \geq 0$; nas n linhas seguintes são dados os n inteiros de uma lista.

   Resultado Os inteiros da lista que aparecem repetidos pelo menos uma vez na lista dada.

   Exemplo

   ```
   6
   1
   2
   1
   10
   2
   1

   1
   2
   ```

   As duas últimas linhas constituem o resultado.

   Deverá resolver este exercício utilizando dois métodos distintos.

(a) Guardar num vector `a[ ]` o conjunto dos inteiros que já foram lidos pelo menos uma vez. Note-se que é necessário distinguir o caso de um inteiro ter ocorrido 1 vez do de ter ocorrido 2 ou mais vezes. Assim, sugere-se que cada elemento de `a[ ]` seja uma estrutura como indicado:

```
#define MAX_ELS 1000
typedef char boolean;
struct elemento{
  int valor;          // Um inteiro lido
  boolean impresso;   // Verdade se já ocorreu 2 ou mais vezes
} a[MAX_ELS];
```

(b) Neste método supõe-se que todos os inteiros lidos estão compreendidos entre 0 e 1000. Utiliza-se uma vector `oc[ ]` de "presenças"

```
char oc[1001]
```

com o seguinte conteúdo

$$\texttt{oc[i]} = \begin{cases} 0 & \text{se o elemento } \texttt{i} \text{ ainda não ocorreu} \\ 1 & \text{se o elemento } \texttt{i} \text{ ocorreu 1 vez} \\ 2 & \text{se o elemento } \texttt{i} \text{ ocorreu 2 ou mais vezes} \end{cases}$$

Assim, por exemplo, se é lido o elemento `i` e `oc[i]==1`, imprime-se `i` e faz-se `oc[i]=2`.

2. Um vector `lista[ ]` de nomes e telefones tem a seguinte estrutura e é inicializado da forma indicada:

```
#define MAX_LISTA 1000
struct pessoa{
  char * nome;
  char * telefone;
} lista[MAX_LISTA] = {{"rui","226664441"},{"ana","214444444"}};
int n = 2;     // Número de entradas na lista
```

(a) Escreva uma função com o seguinte protótipo que pesquisa um nome `pal` na lista, indicando o índice em que se encontra ou `-1` caso não exista.

```
int pesquisa(char *pal);
```

(b) Escreva uma função semelhante à da alínea anterior mas que só procura o nome `pal` a partir do índice `i` (parâmetro da função) até ao fim da lista:

```
int pesqui(int i, char *pal);
```

(c) Escreva um programa que utilizando a função `pesqui`, leia repetidamente um nome e imprima os telefones correspondentes ou "`não existe!`".

(d) Altere o seu programa por forma a aceitar os seguintes comandos:

`p <nome>` — imprime o(s) telefone(s) correspondente(s) ou "`não existe!`".

`i <nome> <tel>` — Insere um novo nome e telefone na lista (a não ser que esse par já exista).

`d <nome>` — Elimina da lista todas as ocorrências de pares (nome, telefone) com o nome indicado.

<u>Exemplo</u>

```
p rui
  226664441
i rui 888888
p rui
  226664441
  888888
d rui
p rui
  não existe!
```