Exame Final de Algoritmos e Estruturas de Dados I

28/06/2019

O que será avaliado?

- Modularidade: uso de funções e procedimentos, passagem de parâmetros e uso de variáveis locais;
- Clareza, lógica, criatividade;
- Sintaxe, uso correto dos comandos, declaração dos tipos, nomes das variáveis, indentação e o uso equilibrado de comentários no código.

QUESTÃO: (100 pontos)

O **Sudoku** é um jogo de raciocínio e lógica no qual o jogador deve preencher uma matriz 9x9 com números de 1 a 9, tal que:

- nenhum número se repita em cada linha
- nenhum número se repita em cada coluna
- nenhum número se repita nas 9 submatrizes 3x3 geradas a partir da divisão da matriz original em 9 partes.

Por exemplo, a matriz abaixo satisfaz todas as condições e é uma solução para o jogo.

3	4	7	2	1	8	5	6	9
8	2	9	3	5	6	4	7	1
6	1	5	9	7	4	2	3	8
4	6	1	7	9	2	8	5	3
5	8	2	4	3	1	6	9	7
9	7	3	8	6	5	1	4	2
1	9	4	5	8	7	3	2	6
7	5	8	6	2	3	9	1	4
2	3	6	1	4	9	7	8	5

Implemente um programa em *Pascal* que leia uma matriz 9x9 como entrada e escreva como saída:

- `Correto`: caso a matriz satisfaça todas as condições para ser uma solução do Sudoku;
- `Incorreto`: caso alguma condição seja violada.

Seu programa deve utilizar o tipo abstrato de dados `Conjunto`. Os protótipos das funções e procedimentos do TAD Conjunto estão listados abaixo, mas **não** é necessário escrever sua implementação. Considere que eles estão corretamente implementados em uma biblioteca. Apenas resolva o problema *usando* o TAD Conjunto. Lembre-se que conjuntos *não possuem elementos repetidos.* Assim, a união $\{1,2\} \bigcup \{1\} = \{1,2\}$.

```
CONST MAX = 100;
TYPE
     elemento = longint;
     conjunto = record
          tam: longint;
          v: array [1..MAX] of elemento;
     end;

procedure inicializar_conjunto (var c: conjunto);
(* inicializa o conjunto c *)
```

```
function eh_vazio (var c: conjunto): boolean;
(* retorna true se o conjunto c eh vazio *)

function cardinalidade (var c: conjunto): longint;
(* retorna o tamanho - quantidade de elementos - do conjunto c *)

function pertence (x: elemento; var c: conjunto): boolean;
(* retorna true se x pertence ao conjunto c *)

procedure uniao (var c1, c2, uni: conjunto);
(* retorna em uni os elementos que pertencem a c1 ou c2 ou ambos *)

procedure intersecao (var c1, c2, intersec: conjunto);
(* retorna em intersec os elementos que estao tanto em c1 com c2 *)

procedure diferenca (var c1, c2, dif: conjunto);
(* retorna em dif os elementos em c1 que nao estao em c2 *)

function contido (var c1, c2: conjunto): boolean;
(* retorna true se todos os elementos em c1 pertencem a c2 *)

procedure insere (x: elemento; var c: conjunto);
(* insere o elemento x no conjunto c *)

procedure remove (x: elemento; var c: conjunto);
(* remove o elemento x do conjunto c *)
```

Não esqueça que o uso de funções e procedimentos é fundamental nesta prova.