Programação Imperativa 2022/2023 (CC1003), DCC/FCUP

Folha 6

6.1 (**Plataforma codex**) Escreva uma função `int ordenada(int vec[], int size)` que testa se uma variável indexada de inteiros está por *ordem crescente* em sentido lato, isto é, se $vec[i] \leq vec[i+1]$ para todos os índices i de 0 a $size - 2$. O resultado deve ser 1 em caso afirmativo e 0 em caso negativo. A função não deve modificar os valores da variável indexada.

6.2 Escreva uma função `int desordem(int vec[], int size)` que conta quantos pares de valores numa variável indexada estão fora de ordem, isto é, $vec[i] > vec[i+1]$. Exemplo: se `vec` ={`3,1,2,2,4,0`} e `size=6` então o resultado deve ser 2 (porque $3 > 1$ e $4 > 0$).

6.3 Escreva um programa completo que lê uma sequência de inteiros positivo da entrada-padrão terminada por zero, ordena e imprime por ordem crescente.
Sugestão: a função `main` no seu programa deve declarar uma variável indexada com um tamanho máximo (por exemplo, 1000), ler os valores, invocar uma função de ordenação e imprimir a sequência final ordenada.

6.4 Defina uma função `int segundo_menor(int vec[], int size)` que encontra o segundo menor valor de uma variável indexada `vec` com `size` elementos. Pode assumir `size`≥ 2.
Sugestão: para encontrar o segundo menor valor basta efetuar as duas primeiras iterações do algoritmo de ordenação por seleção. A função pode modificar a ordem dos elementos de `vec`.

6.5 (**Plataforma codex**) Defina uma função `void sort_desc(int vec[], int n)` que ordena um vetor de inteiros de comprimento n por *ordem decrescente*.

6.6 Considere a primeira versão do algoritmo de Euclides para calcular o máximo divisor apresentado na aula 5:

```
int mdc(int a, int b)  {
  while (a != b) {
    if(a > b)
      a = a - b;
    else
      b = b - a;
  }
  return a;
}
```

Acrescente uma asserção que exprime as pré-condições para que este algoritmo termine.

6.7 Considere a função que calcula a mediana de três valores a, b, c (exercício 2.6). Modifique a sua solução acrescentando uma asserção correspondente à pós-condição seguinte: seja r o resultado de `mediana(a,b,c)`, então $\min(a, b, c) \leq r \leq \max(a, b, c)$.
Sugestão: defina funções auxiliares para calcular o mínimo e o máximo dos três valores.

6.8 Considere a função pedida no exercício 5.7 (converter uma cadeia de dígitos decimais para um inteiro). Acrescente asserções para exprimir a pré-condição que todos os carateres da cadeia são dígitos decimais (i.e., carateres entre `'0'` e `'9'`).

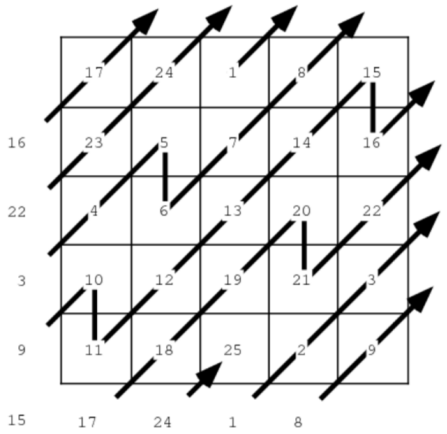
6.9 Defina uma função `int identidade(int mat[N][N])` para verificar se uma matrix $N \times N$ é a *matrix identidade* (i.e., contém 1 na diagonal principal e 0 nas outras posições). O resultado deve ser 1 em caso positivo e 0 em caso negativo. A sua função deve funcionar para qualquer dimensão N declarada como constante usando `#define`.

6.10 (**Plataforma codex**) Um *quadrado mágico* é uma matrix quadrada de números inteiros tal que todas as linhas, colunas e diagonais somam o mesmo valor. No exemplo seguinte cada linha, coluna e diagonal soma o mesmo valor (15 neste caso):

$$\begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}.$$

Escreva uma função `int magico(int a[20][20], int n)` que testa se uma matrix é uma quadrado mágico. A matriz é representada por uma variável indexada `a` com dimensão declarada 20×20; o argumento `n` indica qual sub-matrix a considerar: por exemplo, se `n=3` devemos testar se a sub-matriz 3×3 que contém os valores das primeiras 3 linhas e 3 colunas é um quadrado mágico (e ignorar o resto da matriz). O resultado deve ser um inteiro: 1 se é um quadrado mágico e 0 caso contrário.

6.11 O algoritmo seguinte permite gerar quadrados mágicos de tamanho $n \times n$ para n ímpar: começamos por colocar um "1" no meio da primeira linha; em seguida, vamos colocar os número de 2 a n^2: o próximo número é colocado na posição na diagonal para cima e para a direita se esta se encontrar vazia; se a posição já se encontra preenchida, então passamos para a posição em baixo. Em qualquer dos casos, o movimento é feito considerando que os bordos cima/baixo e esquerdo/direito do quadrado estão ligados (isto é, se sairmos pela direita, re-entramos na posição correspondente à esquerda e se sairmos por cima re-entramos na posição correspondente por baixo).
A figura seguinte ilustra a geração de uma quadrado mágico 5×5.



Escreva um programa que constrói um quadrado mágico de dimensão impar < 50 usando este algoritmo.