Variáveis indexadas multi-dimensionais

P7.1 Dada uma variável bi-indexada com 1000 linhas e 7 colunas representando o resultado de 1000 sorteios do Totoloto (para valores inteiros não repetidos, entre 1 e 49) escrever funções que determinem:

1. Para cada sorteio o número de pares de números consecutivos;
2. Os 3 números que ocorrem mais vezes;
3. A maior sequência de números consecutivos num sorteio.

P7.2 (Ordenação por "molhos") Escreva uma função `void bucket_sort (int a[],int n)` que ordene a variável `a[]` de inteiros positivos, segundo o seguinte método:

Vamos utilizar uma variável bi-indexada `int aux[10][N]`, com `N` maior que `n` (que inicialmente com zeros).

1. Colocar cada elemento de a[] na variável `aux[][]` na linha correspondente ao seu dígito das unidades (da representação base 10). Por exemplo, 97 é colocado na linha 7, 3 na linha 3 e 100 na linha 0.
2. Percorrer a variável bi-indexada `aux[][]` por linhas e copiar sucessivamente os valores para a variável `a[]`. Para o exemplo anterior ficava `a[0]=100,a[1]=3, a[2]=97`.
3. Repetir o processo para cada um dos dígitos dos elementos de `a[]` (supondo, se necessário que os dígitos à esquerda são 0) e terminar quando o dígito (não nulo) mais à esquerda (mais significativo) do maior elemento de `a[]` for processado. O processo termina quando todos os elementos de `a[]` forem copiados no passo 1 para a 1ª linha de `aux[]`.

Nota: este processo de ordenação requere muito mais espaço que os métodos dados anteriormente mas é mais eficiente em termos de tempo de execução.

P7.3 Simulação do jogo do galo. O jogo do galo joga-se num tabuleiro de 3x3 e dois jogadores. Cada jogador coloca alternadamente um `1` ou um `2` numa das quadriculas e ganha o que colocar 3 peças em linha (horizontal, vertical ou diagonal principal). O tabuleiro é representado por uma variável indexada `a[2][2]` que contém as jogadas feitas, supondo-se que se o jogador `i` jogou numa certa posição então o conteúdo dessa posição é `i`, para `i=1,2`. Se uma posição ainda não foi jogada o conteúdo de `a` nessa posição é `0`. Nenhum jogador pode jogar numa posição já preenchida e o jogo termina empatado se todas as posições estão ocupadas e nenhum jogador ganhou. Em cada jogada deve ser imprimido o tabuleiro, isto é, `a`. No início o programa deve pedir o nome de cada um dos jogadores e imprimir o nome do jogador que ganhar.

P7.4 Simulação do jogo *5 em linha*. Dois jogadores preenchem alternadamente as posições dum tabuleiro $N \times N$, $N > 4$. Um joga com *peças* `0` e o outro com *peças* `1`. Ganha o jogo, o primeiro jogador que conseguir colocar 5 peças consecutivas na mesma direcção: numa linha, coluna ou diagonal (faz *5 em linha*). O jogo termina – empatado – se já não houverem posições para preencher com peças. Supõe-se que inicialmente todas as posições do tabuleiro contêm o símbolo `X`.

Os dois jogadores jogam alternadamente. Em cada jogada, um jogador selecciona uma posição do tabuleiro indicando apenas a coluna c (de 1 a N) em que pretende jogar. Se a coluna c estiver toda preenchida, e, ainda restarem posições livres noutra coluna, terá de escolher uma dessas colunas *livres*. A posição em que ficará a sua peça será a corresponde à linha de maior numeração ainda *livre*, nessa coluna. Isto é, para uma dada coluna c, a primeira posição a ser preenchida é a (c, N), a segunda a $(c, N-1)$, a terceira a $(c, N-2)$, ... No fim de cada jogada, o computador terá de avaliar se o jogador ganhou o jogo. Para isso, basta verificar se com a peça que jogou conseguiu fazer *5 em linha*. O programa a

desenvolver deve, em cada jogada: mostrar o tabuleiro; indicar qual o jogador que deve jogar; pedir ao jogador que seleccione uma coluna; verificar se o jogador fez *5 em linha*; verificar se o jogo terminou e indicar o vencedor; permitir a continuação do jogo.

Strings

P7.5 Escreve uma função em C que dada uma `string` lhe retira todos os caracteres *brancos*.

P7.6 Escreve um programa em C que leia texto dum programa em C e verifique se as chavetas estão bem casadas.

P7.7 Escrever um programa em C que determine a frequência absoluta de todos os caracteres duma *string* e imprima o resultado por ordem decrescente de ocorrência.

P7.8 Considere uma variável global bi-indexada de caracteres `char cruzadex[50][50]`.

Escreva uma função `void encontra(char s[]);` que determina todas as ocorrências da *string* `s` em `cruzadex` numa linha, coluna ou diagonal e em qualquer sentido. Para cada ocorrência devem ser impressos os índices `(linha,coluna)` do primeiro caracter e qual a sua direcção:n,s,e,w,ne,nw,se ou sw.