Programação Imperativa – Folha das aulas práticas nº 7

Temas: Exercícios variados...

Nome_____ **Ncd**_____

1. Chamemos *subida* de um vector `v` com `n` elementos a um par (i, j) com $0 \leq i < j < n$ e `v[i]` $<$ `v[j]`. Assim, por exemplo, um vector ordenado por ordem decrescente tem 0 subidas e o vector `v[]={1,4,5,2}` tem 4 subidas (pois $1 < 4$, $1 < 5$, $1 < 2$ e $4 < 5$).
   Quantas subidas tem um vector de n elementos distintos ordenado por ordem crescente?
   Resposta: _____

   Escreva uma função

   ```
   int subidas(int v[],int n)
   ```

   que retorna o número de subidas do vector `v` com `n` elementos.

2. Retira alguns...
   Escreva uma função

   ```
   void sem_digs(char s[])
   ```

   que retira os dígitos do "string" `s`.
   Exemplos

String no início	String no fim
`a12b0cdef`	`abcdef`
`abcde`	`abcde`
`100.4`	`.`

   O resultado é um "string". Não pode usar outros "strings" (ou vectores).

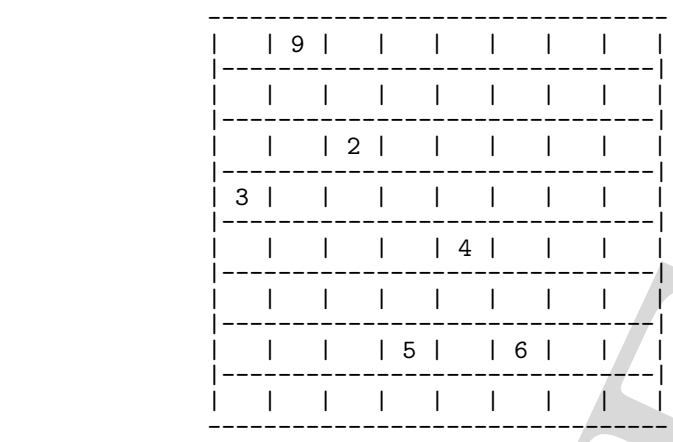
3. Cavalos no tabuleiro
Um tabuleiro de xadrez contém alguns cavalos. O tabuleiro é representado pela variável global

```
int tab[MAX][MAX]
```

em que `MAX` é, por exemplo 8 (para os tabuleiros convencionais); cada posição contém ou o número de um cavalo (um inteiro positivo) ou `0` quando está desocupada.

Escreva um programa que imprime todos os pares de cavalos que se atacam mutuamente (os índices das linhas diferem de 2 em valor absoluto e os índices das colunas diferem de 1 em valor absoluto ou vice-versa).

Exemplo. Para melhor compreensão eliminamos os 0's no seguinte tabuleiro:



Os resultados do programa devem ser, por exemplo

```
9 ataca 2
3 ataca 2
4 ataca 5
6 ataca 4
```

Notas.

- Os "ataques" ser escritos sem repetições; assim, num resultado não deve ocorrer por exemplo "`9 ataca 2`" e "`2 ataca 9`".
- O vector "`tab`" já está definido e inicializado.
- *Nota sobre variáveis indexadas bi-dimensionais (ou "matrizes")*: Num vector bidimensional o elemento da linha `i`, coluna `j` é representado por `v[i][j]`. Assim podemos colocar os cavalos na matriz `tab` da forma seguinte (na página seguinte ilustra-se a *inicialização* da mesma matriz).

```
for(i=0;i<MAX;i++)
  for(j=0;j<MAX;j++)
    tab[i][j]=0;
tab[0][1]=9; tab[2][2]=2; tab[3][0]=3;
tab[4][4]=4; tab[6][3]=5; tab[6][5]=6;
```

Sugestão. (No exame nunca existiria uma sugestão tão "completa"!) Complete o seguinte programa.

```
#define MAX 8
  int tab[MAX][MAX] = {{0,9,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0},
                       {0,0,2,0,0,0,0,0},
                       {3,0,0,0,0,0,0,0},
                       {0,0,0,0,4,0,0,0},
                       {0,0,0,0,0,0,0,0},
                       {0,0,0,5,0,6,0,0},
                       {0,0,0,0,0,0,0,0}};

//-----------------------------------------------------
//--se (i,j) ataca (i1,j1), retorna 1, senão retorna 0
//-----------------------------------------------------
int ataca(int i,int j,int i1, int j1){
  ...
}

//-----------------------------------
main(){
  int i,j,i1,j1;
  for(i=0;i<MAX;i++)
    for(j=0;j<MAX;j++)
      for(i1=i+1;i1<MAX;i1++)
        for(j1=0;j1<MAX;j1++)
          if(ataca(i,j,i1,j1))
            printf(...);
}
```

(a) Implemente e teste o programa pedido.

(b) Explique porque é que, no programa esquemático apresentado em cima, cada "ataque" é impresso uma e uma só vez.

(c) Torne o seu programa mais eficiente, considerando para cada par (`i`, `j`) apenas as posições atacadas (que são, no máximo, 8) em vez de analisar todos oa pares (`i1`, `ji`) com `i1` > `i`.