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: `n(n-1)/2` $(= (\texttt{n}-1) + (\texttt{n}-2) + \cdots + 2 + 1)$
   Escreva uma função

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

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

Nota. Embora não fosse pedido, incluimos o `main`.

```
int subidas(int v[],int n){
  int i,j,cont=0;
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if(v[i]<v[j]) cont++;
  return cont;
}
main(){
  int v[]={1,2,2,3,4,5},n=6;
  printf("%d\n",subidas(v,n));
}
```

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).

Nota. Embora não fosse pedido, incluimos o `main`.

```
#define DIGITO(C) (C>='0' && C<='9')

void sem_digs(char s[]){
  int i=0,j=0;
  while(s[j])
    if(DIGITO(s[j]))
      j++;
    else
      s[i++]=s[j++];
  s[i]=0;
}

int main(){
  char s[]="5sem2004! Ai! OOh!";
  printf("%s\n",s);
  sem_digs(s);
  printf("%s\n",s);
  return 0;
}
```

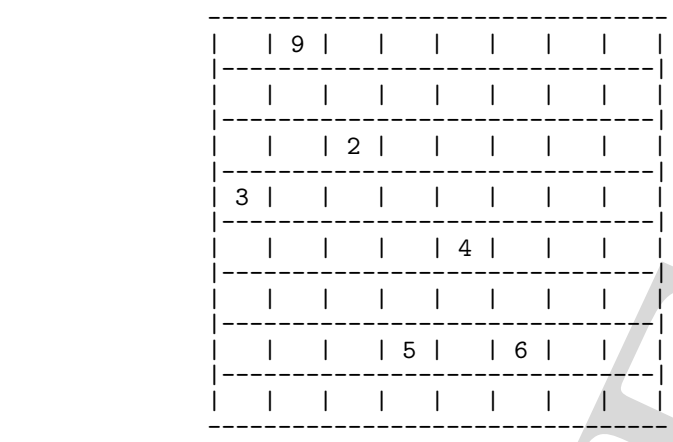
3. Cavalos no tabuleiro
Um tabuleiro de xadrez contém alguns cavalos. O tabuleiro é representado pela variável global

```
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 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;
```

```
#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}};


int absv(x){
  return x>0?x:-x;
}

//-----------------------------------------------------
//--se (i,j) ataca (i1,j1), retorna 1, senão retorna 0
//-----------------------------------------------------
int ataca(int i,int j,int i1, int j1){
  return
    i >=0 && i <MAX && j >=0 && j <MAX &&
    i1>=0 && i1<MAX && j1>=0 && j1<MAX &&
    (absv(i-i1)==1 && absv(j-j1)==2 ||
     absv(i-i1)==2 && absv(j-j1)==1);
}

//-----------------------------------
main(){
  int i,j,i1,j1;
  for(i=0;i<MAX;i++)
    for(j=0;j<MAX;j++)
      if(tab[i][j])                  // se há cavalo em (i,j)
        for(i1=i+1;i1<MAX;i1++)
          for(j1=0;j1<MAX;j1++)
            if(tab[i1][j1])          // se há cavalo em (i1,j1)
              if(ataca(i,j,i1,j1)) // ... se se atacam...
                printf("%d ataca %d\n",tab[i][j],tab[i1][j1]);
}
```

(a) Implemente e teste o programa pedido. R: `Ver o programa anterior.`

(b) Explique porque é que, no programa esquemático apresentado em cima, cada "ataque" é impresso uma e uma só vez.
R: `Se o cavalo (i,j) ataca o cavalo (i1,j1) existe também o ataque de (i1,j1) a (i,j). Mas o programa só considera o caso em que i1>i; assim um e um só dos dois ataques é impresso.`

(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 os pares (i1, ji) com $i1 > i$.
R: `Bastaria, em vez de fazer ciclos em i1 e j1, considerar para cada par (i,j) apenas as posições seguintes`

$$(i+1, j-2),\ (i+1, j+2),\ (i+2, j-1),\ (i+2, j+1)$$

`que estiverem dentro do tabuleiro.`