



## Solutions modèles en Pascal

### Problème I - LISTES

Programme encore relativement facile. C'était l'exercice de réchauffement!

```
program Listes;
{$APPTYPE CONSOLE}

uses SysUtils;

var FL1,FL2 : text;
    L1,L2 : string;

function CONT (L1,L2 : string) : boolean;
var I,P : integer;
    TEMP : boolean;
begin
    TEMP := true;
    for I := 1 to length(L1) do
        begin
            P := pos(copy(L1,I,1),L2);
            if P = 0
            then TEMP := false
            else delete(L2,1,P)
            end;
        CONT := TEMP
    end;

function MIRR (STR : string) : string;
var TEMP : string;
    I : integer;
begin
    TEMP := '';
    for I := length(STR) downto 1 do
        TEMP := TEMP + STR[I];
    MIRR := TEMP
end;

begin
    assign(FL1,'Liste1.txt');
    assign(FL2,'Liste2.txt');
    reset(FL1);
    reset(FL2);
    readln(FL1,L1);
    readln(FL2,L2);
    close(FL1);
    close(FL2);
    writeln('L1 : ',L1);
    writeln('L2 : ',L2);
    if CONT(L1,L2) or CONT(MIRR(L1),L2)
    then writeln('L1 est contenu dans L2')
```

```

else writeln('L1 n''est pas contenu dans L2');
readln
end.

```

## Problème II - RECTANGLE

Ce programme calcule toutes les sommes possibles. Si les dimensions du rectangle donné sont x et y, il y a à calculer

$$\sum_{i=1}^x i \cdot \sum_{i=1}^y i$$

sommes. Pour le rectangle maximal ce sont

$$\sum_{i=1}^{100} i \cdot \sum_{i=1}^{100} i = \frac{100 \cdot 101}{2} \cdot \frac{100 \cdot 101}{2} = 5050 \cdot 5050 = 25502500$$

sommes!

Complexité:  $O(x^2 \cdot y^2)$ ; soit de l'ordre  $x^4$ !

Voilà pourquoi ce programme nécessite environ 1 minute pour trouver la somme maximale dans un rectangle de 100x100.

```

program SommeMaximale;
{$APPTYPE CONSOLE}

uses SysUtils;

var f : text;
    x,y,i,j,k,p : integer;
    a : array[1..100,1..100] of integer;
    maxSum,Sum : integer;
    c,x1,y1,x2,y2 : integer;

procedure EVALSOMME (x1,y1,x2,y2 : integer;
                    var S : integer);
var i,j : integer;
begin
    s := 0;
    for i := x1 to x2 do
        for j := y1 to y2 do
            s := s + a[i,j]
        end;
    end;
end;

begin
    assign(f, 'NOMBRES.TXT');
    reset(f);
    readln(f, x, y);
    for i := 1 to x do
        begin

```

```

        for j := 1 to y do
            read(f,a[i,j]);
        readln(f)
    end;
writeln('Les dimensions : ');
writeln('x : ',x);
writeln('y : ',y);
writeln;
writeln('Les nombres : ');
for i := 1 to x do
    begin
        for j := 1 to y do
            write(a[i,j], ' ');
        writeln
    end;
writeln;
maxSum := -32728;
c := 0;
for i := 1 to x do
    for j := 1 to y do
        for k := i to x do
            for p := j to y do
                begin
                    EVALSOMME(i, j, k, p, SUM);
                    c := c + 1;
                    if SUM > MaxSum
                        then begin
                            MaxSum := SUM;
                            x1 := i;
                            y1 := j;
                            x2 := k;
                            y2 := p
                        end
                end;
writeln('Nombre de sommes : ',c);
writeln('Somme maximale : ',maxSum);
writeln('dans le rectangle : ',x1,' ',y1,' ',x2,' ',y2);
readln
end.

```

## Problème III - JEU

Avant de se lancer dans la programmation, il fallait trouver une stratégie de jeu. En voici une. Les nombres dans la liste sont indexés soit par un nombre pair soit par un nombre impair. La stratégie du programme est de choisir soit les uns, soit les autres. Voilà pourquoi le programme analyse la liste avant le jeu. Il calcule la somme de tous les nombres d'indice pair et de tous ceux d'indice impair. Si la somme des nombres d'indice pair est la plus grande, il ne prendra pendant le jeu que des nombres d'indice pair. Dans l'autre cas (la somme des nombres d'indice impair est la plus grande) il choisira les nombres d'indice impair. Ceci est toujours possible vu que le programme démarre le jeu. Considérons un exemple.

La somme des nombres d'indice pair est la plus grande:

1	2	3	4	5	6	7	8
100	1	100	1000	100	1	100	1

La somme des nombres d'indice pair vaut:  $1 + 1000 + 1 + 1 = 1003$

La somme des nombres d'indice impair vaut:  $100 + 100 + 100 + 100 = 400$

Dans cet exemple le joueur qui peut prendre le 1000 gagne! Pour cela il faut prendre tous les 1 et laisser tous les 100 à l'adversaire!

Le programme doit donc choisir les nombres d'indice pair! Au début il choisit donc à droite (indice 8). La liste devient:

1	2	3	4	5	6	7
100	1	100	1000	100	1	100

Peu importe que le joueur humain choisisse à gauche ou à droite, il dévoilera de façon certaine un nombre d'indice pair. S'il choisit à gauche, l'indice 2 devient accessible, sinon s'il choisit à droite l'indice 6 devient accessible. Etc. jusqu'à la fin du jeu.

Les mêmes réflexions sont valides pour l'autre cas (la somme des nombres d'indice impair est la plus grande).

```

program AGAME;
{$APPTYPE CONSOLE}

uses SysUtils;

var f : text;
    n,i : integer;
    Liste : array [1..1000] of integer;
    SommeP,SommeI : integer;
    MaSomme,TaSomme,First : integer;
    choix : string;

begin
    assign(f,'IN.TXT');
    reset(f);
    read(f,n);
    for i := 1 to n do
        read(f,Liste[i]);
    close(f);
    for i := 1 to n do
        write(Liste[i],' ');
    writeln;
    writeln('-----');
    { calcul de la somme des nombres d'indice pair (SommeP) }
    SommeP := 0;
    i := 2;
    while i <= n do
        begin
            SommeP := SommeP + Liste[i];
            i := i + 2;
        end;
    { calcul de la somme des nombres d'indice impair (SommeI) }
    SommeI := 0;
    i := 1;
    while i <= n do
        begin
            SommeI := SommeI + Liste[i];
            i := i + 2;
        end;
    MaSomme := 0; TaSomme := 0;
    First := 1;
repeat

```

```

if SommeP > SommeI
then if n mod 2 = 0
    then begin
        writeln('Je prends à droite');
        MaSomme := MaSomme + Liste[n];
        writeln('Ma somme est : ',MaSomme);
        n := n - 1
    end
    else begin
        writeln('Je prends à gauche');
        MaSomme := MaSomme + Liste[First];
        writeln('Ma somme est : ',MaSomme);
        First := First + 1
    end
else if n mod 2 <> 0
    then begin
        writeln('Je prends à droite');
        MaSomme := MaSomme + Liste[n];
        writeln('Ma somme est : ',MaSomme);
        n := n - 1
    end
    else begin
        writeln('Je prends à gauche');
        MaSomme := MaSomme + Liste[First];
        writeln('Ma somme est : ',MaSomme);
        First := First + 1
    end;
writeln('-----');
for i := First to n do
    write(Liste[i], ' ');
writeln;
write('Vous prenez à (d/g) : ');
readln(Choix);
if Choix = 'd'
then begin
    TaSomme := TaSomme + Liste[n];
    n := n - 1
end
else begin
    TaSomme := TaSomme + Liste[First];
    First := First + 1
end;
writeln('Votre somme est : ',TaSomme);
writeln('-----');
for i := First to n do
    write(Liste[i], ' ');
writeln;
until First >= n;
writeln('Je gagne!');
readln
end.

```