TP n°2 - Les fonctions (suite)

Retrouvez tous les énoncés et les corrections des TPs sur ma page personnelle :

`http://perso.ens-lyon.fr/hadrien.croubois/`

Repondre, pour chacun des programmes, aux question suivants :
- Quelle fonctionnalitées sont presentées dans le programme (generalement une par fonction/procédure) ?
- Pouvez vous penser à une situation reelle ou une telle fonctionnalitée serait-utile ?
- Remplir les espaces laissés vide.
- Quelles informations seront afficher à l'ecran au cours de l'excution ?
- Discuter de l'efficacité des fonctions. Pouvez vous faire mieux ?

Programme 1 : cercle

```
PROGRAM cercle(output);
CONST
  PI = 3.141592653589793238462 6;
VAR
  rayon : REAL;
FUNCTION aire(r:....):....;
BEGIN
  aire := PI*r*r;
END;

FUNCTION perimetre(r:....):....;
BEGIN
  perimetre := 2.*PI*r;
END;

BEGIN
  rayon := 17.;
  writeln(rayon, '->', perimetre(rayon), ',', aire(rayon));
END.
```

Programme 2 : minmax

```
PROGRAM minmax(input, output);
VAR
  a : INTEGER;
  b : INTEGER;
FUNCTION min(i:INTEGER; j:INTEGER):INTEGER;
BEGIN
  ..............................
END;

FUNCTION max(i:INTEGER; j:INTEGER):INTEGER;
BEGIN
  ..............................
END;

BEGIN
  write('nombre 1: ');
  readln(a);
```

```
  write('nombre 2: ');
  readln(b);
  writeln('min : ', min(a,b), ', max : ', max(a,b));
END.
```

Programme 3 : recursion

```
PROGRAM recursion(output);
FUNCTION fact(.......) : .......;
BEGIN
  if n = 0 then
    fact := 1
  else
    fact := n * fact(n-1);
END;

FUNCTION fibo_naif(n : INTEGER) : longint;
BEGIN
  if n = 0 then
    fibo_naif := 0
  else if n = 1 then
    fibo_naif := 1
  else
    fibo_naif := fibo_naif(n-1) + fibo_naif(n-2);
END;

FUNCTION fibo_memory(n : INTEGER; var memory : array of longint) : longint;
BEGIN
  if n = 0 then
    memory[n] := 0
  else if n = 1 then
    memory[n] := 1
  else if memory[n] = -1 then
    memory[n] := fibo_memory(n-1, memory) + fibo_memory(n-2, memory);
  fibo_memory := memory[n];
END;

VAR
  i :       INTEGER;
  memory :  array[0..100] of longint;
BEGIN
  for i := 0 to 100 do
    memory[i] := -1;
  for i := 0 to 20 do
    writeln('fact (', i, ') : ', fact(i));
  for i := 0 to 42 do
    writeln('fibo (', i, ') : ', fibo_naif(i));
  for i := 0 to 42 do
    writeln('fibo_memory (', i, ') : ', fibo_memory(i, memory));
END.
```

Programme 4 : tableau

```
PROGRAM tableau(output);
CONST
  SIZE =    10;
  MAX_INT = 100;
VAR
  table : array[0..SIZE] of INTEGER;
  i : INTEGER;
```

```
PROCEDURE randomize_table(var table : array of INTEGER; size : INTEGER);
VAR
  i : INTEGER;
BEGIN
  for i := 0 to size do
    table[i] := random(MAX_INT);
END;

PROCEDURE display_table(table : array of INTEGER; size : INTEGER);
VAR
  i : INTEGER;
BEGIN
  write('(');
  for i := 0 to size-1 do
    write(table[i], ',');
  writeln(table[size], ')');
END;

FUNCTION min_table(table : array of INTEGER; size : INTEGER) : INTEGER;
VAR
  i, r : INTEGER;
BEGIN
  r := table[0];
  for i := ... to ... do
    if r > table[i] then
      r := ........;
  min_table := r;
END;

FUNCTION rmax_table(table : array of INTEGER; size : INTEGER) : INTEGER;
VAR
  i, j : INTEGER;
BEGIN
  j := 0;
  for i := ... to ... do
    if table[j] < table[i] then
      j := i;
  rmax_table := j;
END;

PROCEDURE swap(var i : INTEGER; var j : INTEGER);
VAR
  t : INTEGER;
BEGIN
  t := i;
  i := j;
  j := t;
END;

PROCEDURE selectionsort_table(var table : array of INTEGER; size : INTEGER);
VAR
  i : INTEGER;
BEGIN
  ........................
    swap(table[i], table[rmax_table(table, i)]);
END;

PROCEDURE quicksort_table(var table : array of INTEGER; a : INTEGER; b : INTEGER);
VAR
  p : INTEGER;
  i : INTEGER;
```

```
BEGIN
  if (a < b) then
    begin
      p := a;
      for i := a+1 to b do
        if table[i] < table[a] then
          begin
            p := p+1;
            swap(table[i], table[p]);
          END;

      swap(table[a], table[p]);
      quicksort_table(table, a, p-1);
      quicksort_table(table, p+1, b);
    END;

END;

BEGIN
  for i := 0 to SIZE do
    table[i] := 0;
  randomize();
  display_table(table, SIZE);
  randomize_table(table, SIZE);
  display_table(table, SIZE);
  // selectionsort_table(table, SIZE);
  quicksort_table(table, 0, SIZE);
  display_table(table, SIZE);
END.
```