Les structures de contrôle répétitives

Les structures répétitives nous permettent de répéter un traitement un nombre fini de fois. Nous avons trois types de structures itératives (boucles):

a. Boucle Pour (For)

La structure de contrôle répétitive pour (for en langage PASCAL) utilise un indice entier qui varie (avec un incrément = 1) d'une valeur initiale jusqu'à une valeur finale. À la fin de chaque itération, l'indice est incrémenté de 1 d'une manière automatique (implicite). La syntaxe de la boucle pour est comme suit:

```
pour <indice> ← <vi> à <vf> faire
      <instruction(s)>
finPour;
```

```
for <indice>:=<vi> to <vf> do
 begin
      <instruction(s)>;
end;
```

<indice> : variable entière

<vi> : valeur initiale <vf> : valeur finale

La boucle **pour** contient un bloc d'instructions (les instructions à répéter). Si le bloc contient une seule instruction, le **BEGIN** et **END** sont facultatifs.

Le bloc sera répété un nombre de fois égale à (<vf> - <vi> + 1) si la valeur finale est supérieure ou égale à la valeur initiale. Le bloc sera exécuté pour <indice> = <vi>, pour <indice> = <vi>+1, pour <indice> = <vi>+2, ..., pour <indice> = <vf>.

Il ne faut jamais mettre de point-virgule après le mot clé **DO**. (Erreur logique)

b. Boucle Tant-que (WHILE)

La structure de contrôle répétitive tant-que (**WHILE** en langage PASCAL) utilise une expression logique ou booléenne comme condition d'accès à la boucle : si la condition est vérifiée (elle donne un résultat vrai : **TRUE**) on entre à la boucle, sinon (la condition n'est pas vérifiée : donne un résultat faux : **FALSE**) on quitte la boucle.

La syntaxe de la boucle tant-que est comme suit :

```
Tant-que <condition> faire
      <instruction(s)>
finTant-que;
```

```
WHILE <condition> do
 begin
      <instruction(s)>;
 end;
```

<Condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions tant que la condition est vraie. Une fois la condition est fausse, on arrête la boucle, et on continue l'exécution de l'instruction qui vient après fin Tant que (après end). Comme la boucle for, il faut jamais mettre de point-virgule après **DO**.

Toute boucle pour peut être remplacée par une boucle tant-que, cependant l'inverse n'est pas toujours possible.

c. Boucle Répéter (REPEAT)

La structure de contrôle répétitive répéter (REPEAT en langage PASCAL) utilise une expression logique ou booléenne comme condition de sortie de la boucle : si la condition est vérifiée (elle donne un résultat vrai : TRUE) on sort de la boucle, sinon on y accède (on répète l'exécution du bloc).

La syntaxe de la boucle répéter est comme suit :

```
Répéter                          repeat
      <Instruction>                    <instruction(s)>;
jusqu'à <condition>;             until <condition>;
```

<condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions jusqu'à avoir la condition correcte. Une fois la condition est vérifiée, on arrête la boucle, et on continue l'exécution de l'instruction qui vient après jusqu'à (après UNTIL). Dans la boucle REPEAT on n'utilise pas BEGIN et END pour délimiter le bloc d'instructions (le bloc est déjà délimité par REPEAT et UNTIL).

La différence entre la boucle répéter et la boucle tant-que est :

– La condition de répéter et toujours l'inverse de la condition tant-que : pour répéter c'est la condition de sortie de la boucle, et pour tant-que c'est la condition d'entrer.

– Le teste de la condition est à la fin de la boucle (la fin de l'itération) pour répéter. Par contre, il est au début de l'itération pour la boucle tant-que. C'est-à-dire, dans tant-que on teste la condition avant d'entrer à l'itération, et dans répéter on fait l'itération après on teste la condition.

Exercice 1 : Soit l'algorithme suivant :

```
Algorithme Exo1
Variables S,i,n:entier
DEBUT
Lire(n)
S←0
Pour  i=1 à n Faire
  S←S+2i
  Ecrire ('la somme à l'itération' , i , 'égale à' , s)
FinPour i
Ecrire('la somme finale =' , s)
FIN
```

(1) Dérouler l'algorithme pour n=5

(2) Déduire l'expression générale de S calculée par l'algorithme en fonction de n.

(3) Traduire l'algorithme en programme Pascal.

(4) Résoudre l'exercice en utilisant la boucle **WHILE** et la boucle **REPEAT**.

a) utiliser une boucle de 0 à n avec un pas d'incrémentation du compteur de 1.

b) utiliser une boucle de 1 à 2n avec un pas d'incrémentation du compteur de 2.