Elements d'algorithmique

TP 2 : les boucles

1 Introduction

En algorithmique, lorsqu'on a besoin de répéter plusieurs fois un ensemble d'instructions, on fait appel à des boucles. Il existe trois types de boucles en TurboPascal, chacune est plus ou moins spécifique à certains types de problème.

2 La boucle FOR

Une boucle FOR permet d'effectuer une itération d'une ou plusieurs instructions. Ces itérations sont contrôlées par l'évolution d'une variable de type INTEGER qui est soit incrémentée (augmentée de 1), soit décrémentée (diminuée de 1, ou augmentée de -1) à chaque itération. La syntaxe est la suivante:

```
FOR compteur:= valeur_debut TO valeur_fin DO
BEGIN
...
END;
```

ou bien

```
FOR compteur:= valeur_debut DOWNTO valeur_fin DO
BEGIN
...
END;
```

Exemple:

```
FOR i:=1 TO 5 DO
BEGIN
WRITELN('i=',i);
END;
```

Dans cet exemple, i prend d'abord la valeur 1. L'instruction entre BEGIN et END est exécutée une première fois. Ensuite i augmente de 1 et l'instruction WRITELN prend en compte la nouvelle valeur de i. Cette instruction est réalisée une dernière fois quand i a la valeur 5. Le résultat d'une telle boucle affiche à l'écran:

```
i=1
i=2
i=3
i=4
i=5
```

3 La boucle WHILE

Le mot-clé WHILE est la traduction en TurboPascal d'un algorithme du type: tant que (*condition*) faire *instructions* ... La syntaxe en TurboPascal est la suivante:

```
WHILE (condition) DO
BEGIN
....
....
END;
```

- *condition* est une expression booléenne indiquant si la boucle doit être interrompue. Entre BEGIN et END se trouve la suite d'instructions à répéter à chaque tour de boucle.
- Le déroulement d'une boucle de ce type est le suivant:
  - l'expression *condition* est testée
  - si elle est vraie, l'ordinateur sort de la boucle et passe à la prochaine instruction apr'es END;
  - sinon, il effectue les instructions entre BEGIN et END, puis arrivé à END, évalue à nouveau l'expression *condition.*

4 La boucle REPEAT...UNTIL

Les mot-clé REPEAT et UNTIL sont la traduction en TurboPascal d'un algorithme du type: faire *instructions* ... jusquà (*condition*). La syntaxe est la suivante:

```
REPEAT
....
....
UNTIL condition_d_arret;
```

- *conditon_ d_arret* est une expression booléenne indiquant si la boucle doit être interrompue. On remarque l'absence des délimiteurs BEGIN et END. Leur rôle est joué par REPEAT et UNTIL.
- Le déroulement d'une telle boucle est le suivant:
  - l'ordinateur effectue la suite d'instructions
  - puis il teste l'expression *conditon_ d_arret*
  - si elle est vraie il sort de la boucle et passe à l'instruction après UNTIL.
  - si elle est fausse, il reprend à l'instruction qui suit REPEAT et réeffectue la suite d'instructions.

Remarque: la grosse différence entre WHILE et REPEAT est que dans une boucle avec REPEAT, les instructions contenues dans la boucle sont exécutées au moins une fois.

5 Exemple

Soit U_n la suite définie par récurrence:
$U_0 = 1$
$U_{n+1} = 2*U_n + 1$
Ecrire un programme demandant à l'utilisateur de rentrer un entier naturel n et calculant le terme de rang n de la suite U_n.

5.1 Analyse du problème

1. *Quelles sont les données dont je dispose au début du programme ?*
   U_0: valeur de la suite U_n au rang 0
   N: rang de la suite dont l'utilisateur veut connaître la valeur.

2. *Quelles données sont initialisées par l'utilisateur du programme ?*
   N.
   Quelles données l'utilisateur n'a pas le droit d'initialiser ?
   U_0.

3. *Quel résultat doit rendre le programme ?*
   La valeur de la suite U_n au rang N.

4. *Quelle est la relation entre les données et les résultats ?*
   $U_0 = 1$
   $U_{n+1} = 2*U_n + 1$

5. *La suite est définie par récurrence, je dois donc connaître les valeurs de la suite aux rangs précédents N pour pouvoir calculer sa valeur au rang N. Je dois alors utliser une boucle avec compteur qui compte le nombre de rangs auxquels je calcule la valeur U_n jusqu'à n=N. Soit compteur le nom de la variable qui me sert à compter, quelle est sa valeur initiale, quelle est la condition de sortie de la boucle, et quelle boucle est la mieux appropriée à mon problème ?*
   boucle FOR avec initialisation: compteur:=1;
   condition de sortie: compteur=N

6. *Finalement de combien de variables ai-je besoin pour écrire mon programme ?*
   3variables N: INTEGER, U: INTEGER, compteur: INTEGER

5.2 Le programme

```
PROGRAM suite;
VAR N,U,compteur: INTEGER;

BEGIN
U := 1;
WRITELN('Donnez un entier:');
READLN(N);
FOR compteur:= 1 TO N DO
BEGIN
```

```
U := 2 * U + 1;
END;
WRITELN('La valeur de la suite au rang ', N, 'est ', U);
END.
```

5.3 Vérification

Si N=3,

instructions	N	U	*compteur*
$N, U, compteur : INTEGER;$	–	–	–
$U := 1;$	–	1	–
$READLN(N);$	3	1	–
$FOR\_compteur := 1\_TO\_N\_DO$	3	1	1
$U := 2 * U + 1;$	3	3	1
$FOR\_compteur := 1\_TO\_N\_DO$	3	3	2
$U := 2 * U + 1;$	3	7	2
$FOR\_compteur := 1\_TO\_N\_DO$	3	7	3
$U := 2 * U + 1;$	3	15	3
$WRITELN('Lavaleurdelasuiteaurang', N, 'est', U);$	3	15	3

6 Exercices

6.1 Exercice 1

Ecrire un programme demandant à l'utilisateur d'entrer 3 entiers a,b et c, un réel x et affichant la valeur en x des polynômes P et Q définis par:
$P(x) = ax^2 + bx + c$
$Q(x) = xP(x)$
Répéter ce traitement tant que l'utilisateur le désire.

6.2 Exercice 2

Ecrire un programme demandant à l'utilisateur de deviner un nombre entre 0 et 100. Si l'utilisateur tape un nombre plus grand que l'entier cherché, on affiche 'trop grand'; s'il donne un nombre plus petit, on affiche 'plus petit'.
Indiquer à l'utilisateur quand il a gagné.

6.3 Exercice 3

Ecrire un programme demandant à l'utilisateur d'entrer un entier n et affichant tous les nombres impairs entre 0 et 2n.
Remarque: l'opérateur mod donne le reste de la division euclidienne:

```
p mod q
```

est le reste de la division euclidienne de p par q.

6.4 Exercice 4

Écrire un programme qui lit un entier n supposé positif puis calcule la somme :

$$S = 1 + \frac{1}{2} + \frac{1}{3} + ... + + \frac{1}{n}$$