1. Introduction :

Il arrive souvent dans un algorithme que la même action soit répétée plusieurs fois. Pour gérer ces cas, on fait appel à des instructions en boucle qui ont pour effet de répéter plusieurs fois une même action. Deux formes existent : la première, si le nombre de répétitions est connu avant l'exécution de l'instruction de répétition, la seconde s'il n'est pas connu.

2. Répétitions inconditionnelles :

Il est fréquent que le nombre de répétitions soit connu à l'avance, et que l'on ait besoin d'utiliser le numéro de l'itération afin d'effectuer des calculs ou des tests. Le mécanisme permettant cela est la boucle **Pour**.

• Forme de la boucle Pour :

Pour variable **de** valeur initiale **à** valeur finale **faire**

Séquence d'instructions

fpour

• Exemple :

Pour i **de** 1 **à** 100 **faire**
écrire('Bonjour') ;
FinPour

• Commentaires :

1) Les mots faire et finpour encadrent les instructions qui doivent être exécutées plusieurs fois. On précise entre pour et Répéter comment seront contrôlées les répétitions. On y définit une variable appelée variable de contrôle (dite aussi compteur) (i dans l'exemple) et les valeurs que prendra cette variable: une première valeur ou valeur initiale indiquée après le mot de, une dernière valeur ou valeur finale indiquée après le mot à. La variable de contrôle est initialisée à la première valeur. Avant chaque exécution du corps de la boucle, la valeur de la variable de contrôle est comparée à la valeur finale. Si la variable de contrôle ne dépasse pas cette valeur, on exécute le corps de la boucle, sinon on passe à l'instruction qui suit le mot finpour. Après chaque exécution du corps de la boucle, la variable de contrôle est augmentée d'une unité.
2) Dans l'exemple ci-dessus, la variable de contrôle sert uniquement de compteur du nombre d'exécutions du corps de la boucle. Si on le souhaite, on peut utiliser cette valeur dans le corps de la boucle, entre autres pour faire un calcul fondé sur cette

valeur. Dans tous les cas, il faut éviter de modifier la valeur de cette variable.

3) La première valeur, la dernière valeur et l'incrément peuvent être des expressions numériques. Les instructions du corps de la boucle ne peuvent en aucun cas modifier ces valeurs.
4) Il est théoriquement possible de modifier la valeur de la variable de contrôle à l'intérieur de la boucle mais cette technique est dangereuse et à ne pas utiliser.

3. Répétitions inconditionnelles en Pascal:

- **Forme Générale de la boucle Pour en Pascal :**

For compteur :=Valeur_Initiale **to** Valeur_Finale **do**
Begin
Instruction(s) ;
End ;

- **Exemple :**

For i :=1 **to** 100 **do**
Begin
write('Bonjour') ;
End ;

4. Répétitions conditionnelles :

L'utilisation d'une boucle pour nécessite de connaître à l'avance le nombre d'itérations désiré, c'est-à-dire la valeur finale du compteur. Dans beaucoup de cas, on souhaite répéter une instruction tant qu'une certaine condition est remplie, alors qu'il est a priori impossible de savoir à l'avance au bout de combien d'itérations cette condition cessera d'être satisfaite. Le mécanisme permettant cela est la boucle Tant que.

A. La boucle " Tant que " :

- **Forme de la boucle Tant que :**

Tant que condition **Faire**

liste d'instructions

FinTantQue

- **Exemple :**

Algorithme X;

Déclaration

k : entier;

Début

k ← 1;

Tant que k <= 100 **faire**

écrire (‘Bonjour) ;

k := k + 1;

FinTantQue

Fin.

- **Commentaires :**

1) Les mots Répéter et FinTantQue encadrent les instructions qui doivent être exécutées plusieurs fois. On indique entre Tant que et Répeter les conditions dans lesquelles on doit exécuter le corps de la boucle.
2) Une boucle "Tant que" se présente donc comme suit: tant que expression logique Répéter séquence d'instructions FinTantQue En premier lieu, l'expression logique est évaluée (il faut donc veiller à sa valeur lors de l'entrée dans la boucle): si sa valeur est vrai, le corps de la boucle est exécuté puis l'expression logique est réévaluée (il faut donc qu'elle puisse changer de valeur pour sortir de la boucle) et si elle a la valeur faux, on exécute l'instruction qui suit FinTantQue
3) Il est à noter qu'il est préférable d'exprimer l'expression logique sous la forme NOT (condition(s) d'arrêt). Il est en effet plus simple de déterminer les raisons d'arrêter le processus répétitif que celles de continuer. La forme de ce type de boucle devient donc: tant que NON condition(s) d'arrêt Répéter séquence d'instructions FinTantQue

Remarques

- Les variables de la condition doivent être initialisées avant tant que, pour que au premier passage la condition puisse être évaluée.
- La boucle tant que continue de boucler tant que B n'est pas faux. Pour éviter une boucle infinie, qui < plante > le programme, il faut obligatoirement que dans les instructions il y aie une sous-instruction rendant la condition faux à un moment donné.

B. La boucle " répéter ... jusqu'à " :

Comme la boucle "tant que", ce type de répétitive est utilisé lorsque le nombre de fois que la séquence d'instructions à répéter est inconnu au moment où cette séquence est abordée pour la première fois mais le corps de la boucle est toujours exécuté au moins une fois.

- **Sa formulation générale est:**

Répéter
séquence d'instructions ;

Jusqu'à (expression logique) ;

- **Exemple :**

Algorithme X;

Déclaration

k : entier;

Début

k ← 1;

Répéter

écrire (‘Bonjour) ;

k := k + 1;

Jusqu'à k>100 ;

Fin.

- **Commentaires :**

1) L'expression logique est évaluée après l'exécution du corps de la boucle: si sa valeur est faux, le corps de la boucle est exécuté à nouveau puis l'expression logique est réévaluée (il faut donc qu'elle puisse changer de valeur pour sortir de la boucle) et si elle a la valeur vrai, on exécute l'instruction qui suit jusqu'à.
2) Il faut remarquer que dans tant que expression logique faire séquence d'instructions fintantque expression logique exprime les raisons de continuer et dans répéter séquence d'instructions jusqu'à (expression logique) expression logique exprime les raisons d'arrêter.

5. Répétitions conditionnelles en Pascal :

- **Forme Générale de la boucle Tant Que … en Pascal :**

```
While (Condition) do
   Begin
      Instruction(s) ;
   End ;
```

- **Exemple :**

```
PROGRAM X;

VAR

k : integer;

BEGIN

k := 1;

while k <= 100 do

   begin

      write('Bonjour) ;

      k := k + 1;

   end;

END.
```

- **Forme générale de la boucle Répéter … Jusqu'à en Pascal :**

```
Repeat
Instruction(s) ;
Until (Condition) ;
```

- **Exemple :**

```
PROGRAM X;

VAR

k : integer;

BEGIN
```

```
k := 1;
while k <= 100 do
    begin
        write('Bonjour) ;
        k := k + 1;
    end;
END.
```