Instructions répétitives
les boucles

6. Instructions répétitives (les boucles)

Introduction

Prenons l'exemple d'un **algorithme** qui demande à l'utilisateur la **saisie d'une valeur** au clavier. Par exemple, on pose **une question** à laquelle **l'utilisateur** doit répondre soit par **O (Oui)** ou bien **N (Non)**.

Mais **l'utilisateur est maladroit :** au lieu de taper **O** ou **N,** il tape autre chose, par exemple **X**,

Lorsque **l'utilisateur** entre des données **incorrectes ou incomplètes**, le programme peut soit :

1. **Se bloquer (erreur d'exécution)** puisque le type de réponse ne correspond pas au type de la variable.
2. **Se dérouler normalement,** mais en produisant des **résultats faux.**

Commet éviter ce problème de fonctionnement?

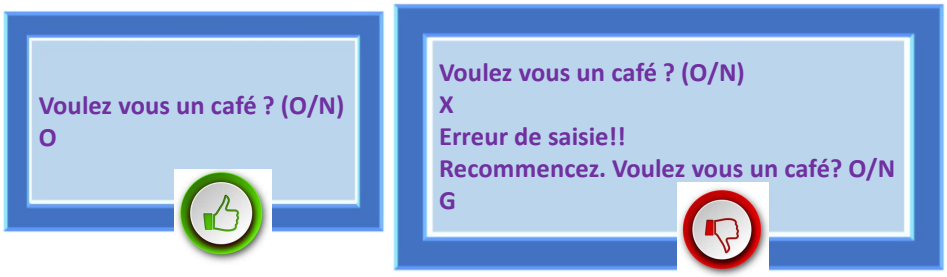
6. Instructions répétitives (les boucles)

On peut mettre en place un **contrôle de saisie** afin de **vérifier que la valeur entrée** correspond bien à celles attendue par l'algorithme.

On pourrait faire ça en utilisant une **Structure conditionnelle Si**
Voyons voir ce que ça donne :

```
Algorithme saisie;
   Var
      Rep : Caractère;
   Début
      Ecrire("Voulez vous un café ? (O/N)");      // Première demande
      Lire(Rep);
      Si ((Rep <>"O") ET (Rep <> "N")) alors
         Ecrire("Erreur de saisie!!");            // Deuxième demande
         Ecrire("Recommencez. Voulez vous un café? O/N");
         Lire(Rep);
      Finsi
   Fin
```

6. Instructions répétitives (les boucles)



1. **Ça marche** tant que l'utilisateur ne se trompe pas ou bien se trompe une seule fois, et rentre une valeur correcte à la deuxième demande.
2. S'il y a une **deuxième erreur de saisie**, il faudrait rajouter une autre structure **SI**. Et ainsi de suite, on peut rajouter des centaines de structure **SI**.

   ➔ Cela **ne résout pas complétement le problème**. La seule issue à ce problème est **l'utilisation d'une boucle.**

6. Instructions répétitives (les boucles)

- **Une boucle** permet d'exécuter **plusieurs fois** de suite une même séquence d'instructions.
- Cette ensemble d'instructions s'appelle le **corps de la boucle**.
- **Chaque exécution** du corps d'une boucle s'appelle une **itération (1,2,3, …)** , ou encore un passage dans la boucle.

6. Instructions répétitives (les boucles)

Pour que la **structure d'une boucle** soit **correcte**, il faut qu'elle soit composée de **quatre blocs élémentaires :**

1. **Bloc d'initialisation de la boucle**: sert comme point de départ des itérations;
2. **Bloc de processus itératif ou corps de la boucle**: contient toutes les instructions à répéter à chaque itération;
3. **Bloc d'incrémentation ou décrémentation** : c'est un compteur de la boucle ou une instruction qui fait évoluer la boucle;
4. **Bloc de test de continuation**: effectue le contrôle pour décider la continuation ou l'arrêt de la boucle (récurrence).

6. Instructions répétitives (les boucles)

Les **instructions itératives** constituent **des boucles** qui **répètent** l'exécution **d'un même bloc d'instructions** un certain **nombre de fois**.

Ce **nombre d'itération ou de répétition** est soit:

1. **Connu et précisé avant l'exécution de l'algorithme,**
2. **N'est pas connu** avant l'exécution de l'algorithme**,** mais dépend de l'évolution d'une condition.

Il existe **trois types** de boucle :

1. **TantQue … FinTantQue**
2. **Répéter … Jusqu'à**
3. **Pour … FinPour**

Chacune de ces boucles a ses **avantages** et ses **inconvénients**.

6. Instructions répétitives (les boucles)

La boucle **Tant que … fin tant que**

La **syntaxe** d'une boucle **Tant que** est la suivante:

```
TantQue (condition) faire
        Bloc d'instructions;
FinTantQue
```

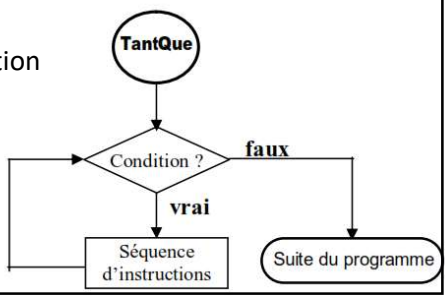
- La **condition de contrôle** est évaluée **avant chaque itération (passage** dans la boucle),
- Si la **condition est Vraie**, on exécute le bloc d'**instructions** (corps de la boucle), puis, on retourne tester la condition. Si elle encore Vraie on répète l'exécution, …
- Si la **condition est Fausse**, on sort de la boucle et on exécute l'instruction qui est après **FinTantQue**.
- Si **dès le début** la condition est **fausse**, le traitement (les instructions de la boucle) **ne sera jamais exécuté**.
- La boucle **Tantque** peut s'exécuter **0, 1 ou n fois**.

6. Instructions répétitives (les boucles)

La boucle : **Tant que … Fin tant que**

Le principe de la **boucle TantQue….FinTantQue** est simple :

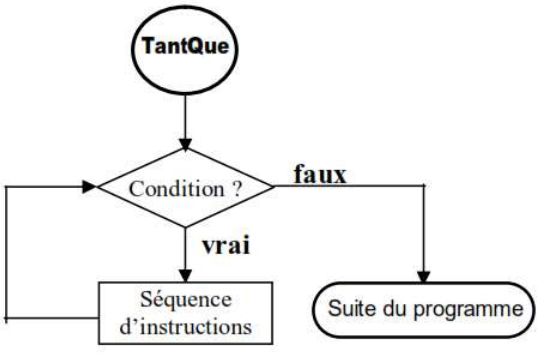
1. **L'algorithme** arrive sur la **ligne de l'instruction TantQue**
2. Il examine alors la valeur de la **condition.**
3. Si cette valeur est **VRAI**, l'**algorithme** exécute les instructions de la boucle, jusqu'à ce qu'il rencontre la ligne **FinTantQue**.
4. Il retourne ensuite sur la ligne du **TantQue**, procède au même examen, et ainsi de suite.
5. **On ne s'arrête** que lorsque la condition prend la valeur **FAUX**.



6. Instructions répétitives (les boucles)

La boucle **TantQue … FinTantQue**

Organigramme



- Le **nombre d'itérations** dans une boucle **TantQue** **n'est pas connu** à l'avance. Il dépend de **l'évolution** de la **valeur de la condition**.

➔ **La valeur de la condition doit absolument être évolué de Vrai à Faux, sinon le programme tourne indéfiniment: Boucles infinies**

6. Instructions répétitives (les boucles)

Exemple d'une boucle infinie

```
Algorithme boucle_infinie;
Var
        A: réel;
Début
      A ← 5;
  Tantque  (A<20)  Faire
        Ecrire(A);

  FinTantQue
Fin
```

➔ L'algorithme **affiche toujours** la valeur **5** indéfiniment.

BOUCLE INFINIE

➔ La **condition (5<20)** est toujours **VRAI,** puisque la valeur de la variable **A** est toujours égale à **5**.

SOLUTION : Une des **instructions** du corps de la boucle doit absolument **changer la valeur de la condition de VRAI à FAUX** (après un certain nombre d'itérations). Par exemple, incrémenter la variable A : **A ← A+1;**

Bloc d'incrémentation ou de décrémentation

1. **Incrémentation:** c'est **l'augmentation** de la valeur d'une variable numérique (compteur) de 1 unité.
2. **Décrémentation:** c'est **la diminution** de la valeur d'une variable numérique (compteur) de 1 unité.

Syntaxe :

x ←x+step; où "**step**" est l'unite (ou le pas) d'incrémentation (ou de décrémentation).

Exemple :

```
x ←5;        //Initialisation de la variable x à 5
x ←x+1;      //Incrémentation de la variable x [résultat x=(5+1)=6]
x ←x-2;      //Décrémentation de la variable x [résultat x=(6-2)=4]
```

6. Instructions répétitives (les boucles)

La boucle Tant que … FinTantQue

Exemples d'application :

- **Exemple 1**: Afficher par exemple tous les nombres de **1 à 10 dans l'ordre croissant.**
- **Exemple 2**: Ecrire un algorithme permettant d'afficher respectivement les **nombres 10, 20, 30, 40 et 50**.
- **Exemple 3**: Ecrire un algorithme qui lit un **entier positif n puis affiche tous ses diviseurs.**
- **Exemple 4**: Ecrire un algorithme qui lit un entier positif n puis calcule et affiche son factoriel selon la **formule n! = 1*2*… *n (Voir TD N°3)**

Correction de l'exemple 1:

Algorithme qui affiche tous les nombres de **1 à 10 dans l'ordre croissant.**

```
Algorithme  nombreCroissant;
  Var  i: entier;
  Début
    i ← 1;
    TantQue (i<= 10) faire
       Ecrire( i);
       i ← i + 1;
    FinTantQue
Fin
```

Initialisation

Test

Corps de la boucle

Incrémentation

- Cet algorithme **initialise** i à 1 et tant que la valeur de **i n'excède pas 10**, cette valeur est **affichée** puis **incrémentée.**
- **Les instructions** se trouvant dans le **corps de la boucle** sont donc exécutées **10 fois de suite**.
- La **variable i**, appelée **compteur de la boucle**, est incrémentée à chaque fois ou la **boucle est exécutée**; on sort de la boucle une fois i est supérieur à 10.

Attention: **L'initialisation** du compteur est très **importante** ! Si vous n'initialisez pas i explicitement, alors cette variable prendra n'importe quelle valeur et votre algorithme ne se comportera pas du tout comme prévu.

6. Instructions répétitives (les boucles)

Correction de l'exemple 2 :

Algorithme permettant d'afficher les nombres 10, 20, 30, 40 et 50.

```
Algorithme   exemple2;
  Var
        i: entier;
  Début
        i ←1;
        TantQue (i<=5) faire
            Ecrire(i*10);
            i ←i + 1;
        FinTantQue
  Fin
```

Initialisation

Test

Corps de la boucle

Incrémentation

6. Instructions répétitives (les boucles)

Correction de l'exemple 3 :

Algorithme qui lit un **entier positif n puis affiche tous ses diviseurs.**

```
Algorithme diviseurs;
  Var
        n, i : Entier;
  Début
        Ecrire("Entrer un entier positif : ") ;
        Lire(n);
        i ← 1;
        TantQue (i <= n) Faire
            Si (n Mod i = 0) Alors
                Ecrire(i);
            FinSi
            i← i + 1;
        FinTantQue
  Fin
```

Initialisation

Test

Corps de la boucle

Incrémentation

Instructions répétitives (les boucles)

La boucle : Répéter … jusqu'à

La **syntaxe** d'une boucle **Répéter** est :

```
Répéter
        Bloc d'instructions;
Jusqu'à (Condition)
```

Son **fonctionnement** est analogue à celui de la boucle **TantQue,** à quelques détails prés :

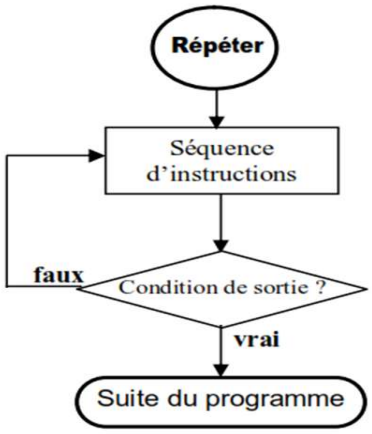
- la **condition** est évaluée **après** chaque **passage** dans la **boucle**.
- On exécute le **corps** de la boucle **jusqu'a** ce que la **condition** soit **vérifiée**, donc **tant que** la **condition** est **FAUSSE**.

Une boucle **Répéter**...**Jusqu'à** est donc exécutée au moins **une fois**.

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Organigramme



Le **principe** de cette boucle **est simple** :

1. Toutes les **instructions écrites** entre **Répéter** et **Jusqu'à** sont exécutées **au moins une fois**,
2. Leur exécution est **répétée** jusqu'à ce que la **condition soit satisfaite**.

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Prenons l'exemple d'une saisie au clavier:

```
Algorithme saisie;
    Var
        Rep: Caractère;
    Début
      Ecrire( "Voulez vous un café ? (O/N) ");
      Répéter
         Lire (Rep);
         Si ((Rep <>"O") ET (Rep <>"N")) Alors
               Ecrire ("Saisie erronée. Recommencez");
         FinSi
       Jusqu'a ((Rep ="O") OU (Rep ="N"));
    Fin
```

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Correction de l'exemple 1 (avec la boucle Répéter)

Algorithme qui affiche tous les nombres de **1 à 10 dans l'ordre croissant.**

```
Algorithme  nombreCroissant;
  Var
        i: Entier;
  Début
        i ← 1;
        Répéter
            Ecrire( i);
            i ← i + 1;
        Jusqu'à (i> 10);
  Fin
```

Initialisation

Corps de la boucle

Incrémentation

Test

- De la **même façon** que pour la boucle **Tant que**, le **compteur** est **initialisé** avant le premier **passage** dans la **boucle**,
- **Par contre**, la **condition** de sortie de la boucle **n'est pas la même**: On sort de la boucle une fois la **valeur 10** est affichée et **i** est **supérieure à 10**.

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Correction de l'exemple 1 (avec la boucle Répéter)

Algorithme qui affiche tous les nombres de **1 à 10 dans l'ordre croissant.**

```
Algorithme nombreCroissant;
  Var
        i: entier;
  Début
        i ← 1;
        Répéter
            Ecrire( i);
            i ← i + 1;
        Jusqu'à (i> 10);
  Fin
```

- Or, **i** est **incrémentée** après l'affichage, par conséquent **i** aura **la valeur 11** à la fin de l'itération pendant laquelle la valeur **10 aura été affichée.**
- On sort de la boucle une fois **i dépasse strictement la valeur 10.**

Remarque : Un des usages les plus courant de la boucle **Répéter ... jusqu'a** est le **contrôle de saisie.**

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Correction de l'exemple 2 (avec la boucle Répéter)

Algorithme permettant d'afficher les nombres **10, 20, 30, 40 et 50.**

```
Algorithme   exemple2;
 Var   i: entier;
 Début
        i ←1;
        Répéter
            Ecrire(i*10);
            i ←i + 1;
        Jusqu'à(i>5);
 Fin
```

Initialisation

Corps de la boucle

Incrémentation

Test

Remarque: Il faut toujours s'assurer que la condition de sortie sera valide après un nombre fini de parcours. Sinon, on aura une ***boucle infinie.***

Instructions répétitives (les boucles)

La boucle **Répéter … jusqu'à**

Correction de l'exemple 3 (avec la boucle Répéter)

Algorithme qui lit un **entier positif n puis affiche tous ses diviseurs.**

```
Algorithme diviseurs;
Var  n, i : Entier;
Début
     Ecrire("Entrer un entier positif : ") ;
     Lire(n);
     i ← 1;                          Initialisation
     Répéter
          Si (n Mod i = 0) Alors     Corps de la boucle
               Ecrire(i);
          FinSi
          i← i + 1;                  Incrémentation
     Jusqu'à (i > n);
Fin                                  Test
```

Instructions répétitives (les boucles)

La boucle **Pour … finPour**

La syntaxe d'une boucle **Pour** est la suivante.

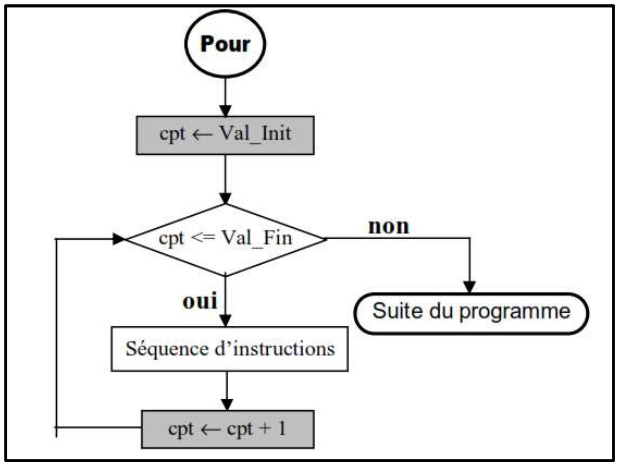
```
Pour  variable ← valeur initiale à valeur finale [pas]  faire
          Bloc d'instructions
FinPour
```

- La boucle **Pour** fait varier la valeur du compteur <**variable**> entre <**valeur initiale**> et <**valeur finale**>.
- Le <**pas**> est **optionnel** et permet de préciser la variation du compteur entre chaque itération.
  - Le **pas par défaut est 1** et correspond donc à une **incrémentation**.
- Une **boucle Pour** peut être exécutée **0, 1 ou n fois.**
- Toute boucle **Pour** peut être réécrite avec une boucle **Tantque**.

Instructions répétitives (les boucles)

La boucle **Pour … finPour**

Le **compteur** (variable de contrôle) prend la **valeur initiale** au moment de l'accès à la boucle puis, à chaque parcours, il passe **automatiquement à** la valeur suivante jusqu'à atteindre la **valeur finale** :

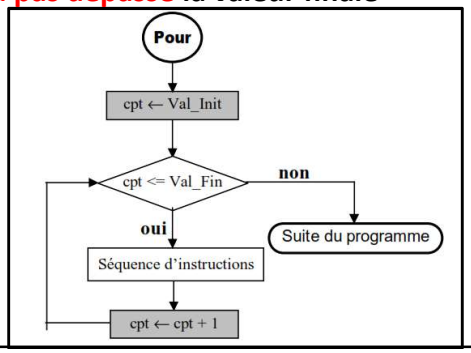


Instructions répétitives (les boucles)

La boucle **Pour … finPour**

Le principe de la boucle **Pour** est simple :

1. On initialise le **compteur cpt** au début par la **valeur initiale**
2. On test **si le compteur cpt** a dépassé la **valeur finale**
3. On exécute la séquence d'instructions (**corps de la boucle**)
4. On incrémente le compteur (**cpt ← cpt + 1**)
5. On test **si le compteur n'a pas dépassé la valeur finale**
6. etc.



Instructions répétitives (les boucles)

La boucle **Pour … finPour**

Exemple :

On veut écrire un algorithme qui affiche le message **"Bonjour à tous"** 100 fois.

Résolution :

Au lieu d'écrire **100 fois** l'instruction : **Ecrire**("Bonjour à tous") ;

On utilise plutôt une **boucle Pour** :

```
Algorithme message;
  var i: Entier;
  Début
     Pour i allant de 1 à 100 faire
          Ecrire("Bonjour à tous");
     FinPour
  Fin
```

Instructions répétitives (les boucles)

La boucle **Pour … finPour**

On peut **améliorer l'algorithme** précédent par :

- L'ajout d'une variable **n** : le nombre de fois que le message s'affichera à l'écran,
- L'affichage de la **variable i** dans la boucle : pour **numéroter les passages dans la boucle.**

```
Algorithme message2;
 Var
        n, i: Entier;
 Début
        Ecrire("entrer le nombre n :");
        Lire(n);
        Pour i allant de 1 à n faire
              Ecrire("Bonjour à tous la ", i," fois");
        FinPour
 Fin
```

Instructions répétitives (les boucles)

La boucle **Pour … finPour**

Exercice1 : Algorithme qui affiche tous les nombres de **1 à 10 dans l'ordre croissant.**

```
Algorithme nombreCroissant;
   Var
       i : entier;
   Début
       Pour i←1 à 10 faire
            Ecrire( i);
       FinPour
   Fin
```

Initialisation + test + Incrémentation

Corps de la boucle

- Notez bien que l'on utilise une **boucle pour** quand on **sait** à l'avance **combien** d'itérations devront être **faites**.

Par exemple, ne pas utilisez une **boucle pour** afin de **contrôler une saisie!**

Instructions répétitives (les boucles)

La boucle Pour … finPour

Remarque : Si le nombre d'itération est connu avant l'exécution de l'algorithme on utilise la boucle **pour,** sinon il faut utiliser soit :

- la boucle Répéter … Jusqu'à.
- Ou bien la boucle Tantque.

Correction de l'exercice 2 (Avec la boucle Pour)

Algorithme permettant d'afficher les nombres **10, 20, 30, 40 et 50.**

```
Algorithme  exemple2;
   Var
       i : Entier;
   Début
       Pour i← 1 à 5 Faire
            Ecrire(i*10);
       FinPour
   Fin
```

Initialisation + test+ Incrémentation

Corps de la boucle

Instructions répétitives (les boucles)

La boucle **Pour … FinPour**

Remarque : Dans une boucle **«Pour»**, l'évolution du compteur peut se faire dans le sens décroissant comme dans l'exemple suivant :

L'exercice 2bis (Avec la boucle pour)

Algorithme permettant d'afficher les nombres **10, 20, 30, 40 et 50.**

```
Algorithme exemple2bis;
   Var
      i : Entier;
   Début
      Pour i← 5 à 1 [pas = -1] Faire
         Ecrire(i*10);
      FinPour
   Fin
```

Initialisation + test+ Décrémentation

Corps de la boucle

Dans ce cas, le compteur i sera **décrémenté** après chaque itération.

Instructions répétitives (les boucles)

La boucle **Pour … FinPour**

L'exercice 3 (Avec la boucle Pour)

Algorithme qui lit un **entier positif n puis affiche tous ses diviseurs.**

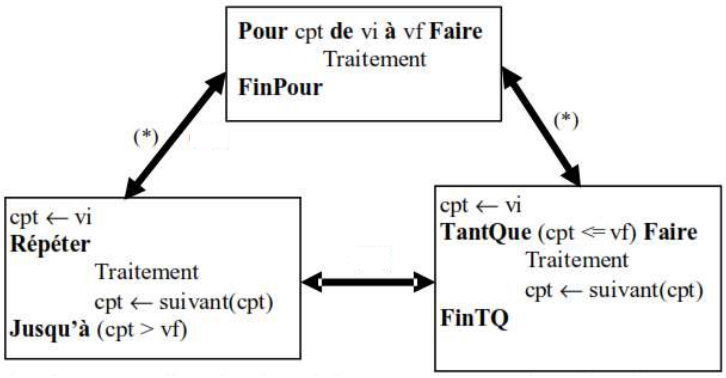
```
Algorithme diviseurs;
Var  n, i : Entier;
Début
 Ecrire("Entrer un entier positif : ");
 Lire(n);
 Pour i allant de 1 à n Faire
    Si (n Mod i = 0) Alors
       Ecrire(i);
    FinSi
 FinPour
Fin
```

Initialisation + test+ Incrémentation

Corps de la boucle

Instructions répétitives (les boucles)

Passage d'une structure itérative à une autre



(*) : Le passage d'une boucle « **Répéter** » ou « **Tantque** » à une boucle « **Pour** » n'est possible que si le nombre de parcours est **connu à l'avance**.

(**) : Lors du passage d'une boucle « **Pour** » ou « **Tantque** » à une boucle « **Répéter** », faire attention le **traitement sera toujours exécuté au moins une fois**.

Méthodologie pour l'écriture d'une boucle :

1. Repérer une **action répétitive**, donc on utilise une **boucle**
2. **Choix** entre une **boucle avec compteur ou sans compteur:**

   Question ? Peut-on prévoir ou déterminer le nombre d'itérations ?

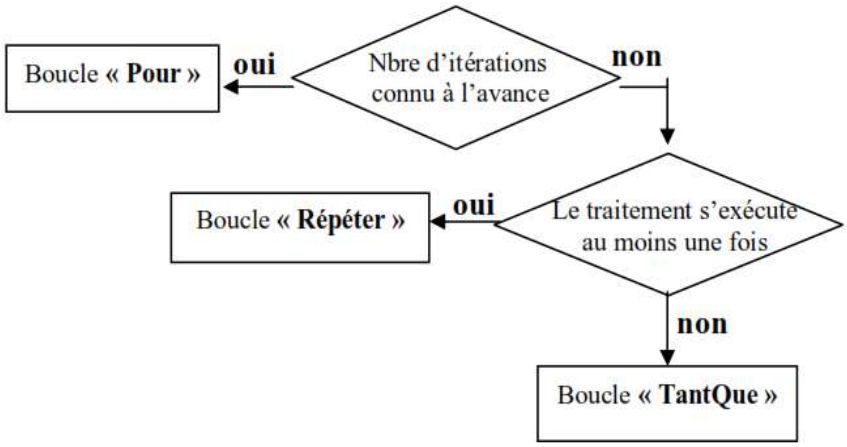
   - Si oui, boucle avec compteur : la **boucle pour** …
   - Si non, boucle sans compteur :

   Est ce que il faut **commencer l'action avant de tester ou l'inverse** ?

   - Si tester d'abord, alors **boucle TantQue**
   - Si action puis tester, alors **Répéter … jusqu'à**
3. Ecrire l'**action répétitive** et l'instruction de la boucle choisie
4. **Initialiser les variables** utilisées (si nécessaires)
5. **Ecrire la condition d'arrêt**, l'évolution de la variable de contrôle.
6. **Tester et exécuter** pour les cas extrêmes et le cas "normal".

Instructions répétitives (les boucles)

Critères de choix de la structure itérative adéquate



Des boucles imbriquées

Exécuter l'algorithme suivant:

```
Algorithme Boucles;
  Var
        i, j : Entier;
  Début
    Pour i allant de 1 à 3 faire
        écrire("Première boucle") ;
    Finpour
    Pour j allant de 1 à 2 faire
        écrire("Deuxième boucle") ;
    Finpour
  Fin
```

Il y aura trois écritures consécutives de **"Première boucle"**, puis deux écritures consécutives de **"Deuxième boucle"** , et ça sera tout.

Des boucles imbriquées

De même qu'une structure **SI … ALORS** peut contenir d'autres structures **SI … ALORS**, une **boucle** peut contenir également d'**autres boucles**.

```
Algorithme Boucles_imbriquées;
Var
   i, j : Entier;
Début
    Pour i allant de 1 à 3 faire
       Ecrire("Première boucle") ;
       Pour j allant de 1 à 2 faire
           Ecrire("Deuxième boucle") ;
       Finpour
    Finpour
Fin
```

Dans cet exemple, l'algorithme affichera une fois **"Première boucle"** puis deux fois de suite **"Deuxième boucle"**, et ceci trois fois en tout.

➔ A la fin, il y aura **3 x 2 = 6 passages** dans la deuxième boucle : le message **"Deuxième boucle"** s'affichera donc 6 fois.

Instructions répétitives (les boucles)

Exercice 1

Ecrire un algorithme permettant de :

1. Lire un **nombre fini** de **notes,** comprises entre **0 et 20** (pas de control de saisie)
2. Afficher la **meilleure note**, la **mauvaise note** et la **moyenne** des notes.

Exercice 2

Ecrire un algorithme qui permet d'**afficher les nombres impaires inférieure à 20** et de **calculer leur somme.**

Exercice 3: Ecrire un algorithme permettant de gérer un cinéma. On désire:

1. **Calculer le nombre de personnes** et **la moyenne d'âges des personnes** s'intéressant à voir un film particulier dans un **cinéma**.
2. Avant de finir, **l'utilisateur** doit répondre par **'N'** à la question posée par l'algorithme: **"encore une autre personne (O/N)?".**

Exercice 4 :

Ecrire un algorithme qui lit un entier positif n puis calcule et affiche son factoriel selon la **formule n! = 1*2*… *n**

(Avec la boucle Pour)

```
Algorithme Notes;
  Var
     n, i : Entier;
     note, min, max, som : Réel;
  Début
    Ecrire(”Entrer le nombre de notes : ”);
    Lire(n);  //On suppose que n est toujours supérieur à zéro
    som ←0;
    max ← 0;
    min ← 20;
    Pour i de 1 à n Faire
        Ecrire(”Entrer la note ”, i) ;
        Lire(note);
        som ← som + note;
        Si (note > max) Alors
              max← note;
        FinSi
        Si (note < min) Alors
              min ← note;
        FinSi
    FinPour
              Ecrire(”Meilleure note = ”, max, ”Mauvaise note = ”, min );
              Ecrire(”Moyenne des notes = ”, som/n);
Fin
```

Initialisation + test+ Incrémentation

Corps de la boucle

(Avec la boucle Tanque)

```
Algorithme Notes;
  Var
     n, i : Entier;
     note, min, max, som : Réel;
  Début
     Ecrire(”Entrer le nombre de notes : ”);
     Lire(n); // On suppose que n est toujours supérieur à zéro
     som ←0;
     max ← 0;  min ← 20;
     i←1;
    Tantque( i<=n) faire
        Ecrire(”Entrer une note ”, i) ;
        Lire(note);
        som ← som + note;
        Si (note > max) Alors
            max← note;
        FinSi
        Si (note < min) Alors
            min ← note;
        FinSi
        i←i+1;
    FinTantque
      Ecrire(”Meilleure note = ”,max, ”Mauvaise note = ”,min, ”Moyenne = ”, som/n );
Fin
```

Initialisation

Test de la boucle

Corps de la boucle

incrémentation

Instructions répétitives (les boucles)

Exercice 2 :

Ecrire un algorithme qui permet d'afficher les nombres impaires inférieure à 20 et de calculer leur somme.

(Solution avec Tantque)

```
Algorithme  nombreImpair;
 Var
        i, Som: Entier;
Début
         i←1;
         Som←0;
         Ecrire("Les nombres impairs inférieur à 20 sont: ");
         Tantque  (i<=20)   Faire
              Si(i MOD 2<>0) alors       // on peut utiliser (i MOD 2 = 1)
                 Ecrire(i);
                 Som ← Som + i;
              Finsi
              i← i+1;
        FinTantQue
     Ecrire(" La somme des nombres impairs, inférieur à 20, est ", Som);
Fin
```

Instructions répétitives (les boucles)

Exercice 2 :

Ecrire un algorithme qui permet d'afficher les nombres impaires inférieure à 20 et de calculer leur somme.

(Solution avec Répéter)

```
Algorithme  sommeNbreImpair2;
   Var
        Som, i: Entier;
   Début
        Som←0;
        i←1;
        Répéter
             Si(i MOD 2<>0) alors
                Ecrire(i);
                Som←Som+ i;
             Finsi
             i← i+1;
        Jusqu'à (i>20);
Fin     Ecrire("La somme des nombres impairs < à 20 est : ", Som);
```

Instructions répétitives (les boucles)

Exercice 3: Ecrire un algorithme qui permet de gérer un cinéma. On désire:

1. Compter le **nombre de personnes** et calculer la **moyenne d'âges** des personnes s'intéressant à voir un film particulier dans un cinéma.
2. **Pour finir,** le gérant doit répondre par **'N'** à la question posée par l'algorithme: ***"encore une autre personne (O/N)?"***

Analyse du problème :

- **Valeurs d'Entrées :** l'âge des personnes: **age ,** la réponse à la question: **Reponse**
- **Valeurs de Sorties :** la moyenne d'âges des personnes: **Moy_age**
  compteur du nombre des personnes: **Nbr_Personne**
- **Valeurs Intermédiaires :** la somme d'âges**: Som_age**

- **Les formules :** **Som_age ⟵ Som_age + age**
  Nbr_Personne ⟵ Nbr_Personne + 1 **(Jusqu'à Reponse='N')**
  Moy_age ⟵ Som_age/Nbr_Personne

Algorithme Cinema; — Avec la Boucle TantQue…

```
Var
      age, Som_age, Nbr_Personne: entier;
      Moy_age : réel:
      Reponse : caractère;
Début
      Som_age ⟵ 0;
      Nbr_Personne ⟵ 0;
      Ecrire(''Encore une autre personne (O/N)?'');
      Lire(Reponse);
   TantQue (Reponse='O') faire
      Ecrire(''Donner l'age de la personne?'');
      Lire(age);
      Nbr_Personne ⟵ Nbr_Personne +1;
      Som_age⟵ Som_age+age;
      Ecrire(''Encore une autre personne (O/N)?'');
      Lire(Reponse);
   FinTantQue
   Moy_age ⟵ Som_age/Nbr_Personne;
   Ecrire(''La moyenne d'ages est ='', Moy_age);
   Ecrire(''Le nombre de personnes saisi  est ='', Nbr_Personne);
Fin
```

Algorithme avec la Boucle REPETER…

```
Algorithme Cinema;
   Var
        age, Som_age, Nbr_Personne: entier;
        Moy_age : réel;
        Reponse : caractère;
   Début
      Som_age ⟵ 0;
      Nbr_Personne ⟵ 0;
      Répéter     // On suppose qu'il y a au moins une seule personne
         Lire(age);
         Nbr_Personne ⟵ Nbr_Personne +1;
         Som_age⟵ Som_age+age;
         Ecrire(''Encore une autre personne (O/N)?'');
         Lire(Reponse);
      Jusqu'à (Reponse='N');
      Moy_age ⟵ Som_age/Nbr_Personne;
      Ecrire(''La moyenne d'ages est ='', Moy_age);
      Ecrire(''Le nombre de personnes saisi  est ='', Nbr_Personne);
   Fin
```

Instructions répétitives (les boucles)

Correction de l'exercice 4:

Exercice 4: Ecrire un algorithme qui lit un entier positif n puis calcule et affiche son factoriel selon la **formule n! = 1*2*… *n**

(Solution avec Pour)

```
Algorithme Factoriel;
  Var
     n, i, f : Entier;
  Début
      Ecrire("Entrer un entier positif : ");
      Lire(n);
      f ←1;
      Pour i ← 2 à n Faire
          f ← f * i;
      FinPour
      Ecrire(n,"! = ",f);
Fin
```

Initialisation + Test+ Incrémentation

Corps de la boucle

Instructions répétitives (les boucles)

Exercice 4: Ecrire un algorithme qui lit un entier positif n puis calcule et affiche son factoriel selon la **formule n! = 1*2*… *n**

Correction de l'exercice 4:

(Solution avec Tantque)

```
Algorithme factoriel;
Var   n, f, i : Entier;
Début
  Ecrire ("Entrer un entier positif : ");
  Lire (n);
  f ← 1;
  i ← 2;
  TantQue (i <= n) Faire
    f ←f * i;
    i ← i + 1;
  FinTantQue
  Ecrire (n,"! = ",f);
Fin
```

Initialisation

Test

Corps de la boucle

Incrémentation

Instructions répétitives (les boucles)

Correction de l'exercice 4:

Exercice 4: Ecrire un algorithme qui lit un entier positif n puis calcule et affiche son factoriel selon la **formule n! = 1*2*… *n**

(Solution avec Répéter)

```
Algorithme factoriel;
Var
     n, f, i : Entier;
Début
    Ecrire("Entrer un entier positif : ");
    Lire(n);
    f ← 1;
    i ← 2;
    Répéter
       f ←f * i;
       i ← i + 1;
    Jusqu'à (i > n) ;
    Ecrire(n,"! = ",f);
Fin
```

Initialisation

Corps de la boucle

Incrémentation

Test

Exercice (exemple de question QCM):

Donner le résultat de l'algorithme suivant, si on saisit N=4 à l'exécution.

```
Algorithme  ExempleBoucle1;
  Var
       N, i: Entier;
       S: Réel;
  Début
    Ecrire(" saisir un entier N:");
    Lire(N);
    S ← 0;
    i ← 0;
    Répéter
              i ← i+ 1;
              S ← S+ 1/i;
    Jusqu'à (i>=N)
              Ecrire(‘’ S=”,S);
  Fin
```

Entrée	Sortie	Test
N=4, S=0, i=0		
i=1	S=0+1/1	1>=4 FAUX
i=2	S=1+1/2	2>=4 FAUX
i=3	S=1+1/2+1/3	3>=4 FAUX
i=4	S=1+1/2+1/3+1/4	4>=4 VRAI ➔ On sort de la boucle
	Affichage du résultat S=1+1/2+1/3+1/4 =2.08	

La simulation de l'exécution