Structures itératives

Dr Khadim DRAME

Département d'Informatique
UFR des Sciences et Technologies
Université Assane Seck de Ziguinchor

18 juin 2021

Plan

Itération

- Une **itération** est un processus dans lequel un traitement est répété plusieurs fois.
- Les structures itératives sont aussi appelées structures **répétitives** ou **boucles**.
- C'est une des structures de base de la programmation.
- Il existe 3 types de boucles classer en 2 catégories :
  - Les boucles à borne définie : le nombre d'itérations est **connu à priori** (boucle **for**) ;
  - Les boucles à borne indéfinie : le nombre d'itérations n'est **pas connu à l'avance** (boucles **while**, **repeat**).

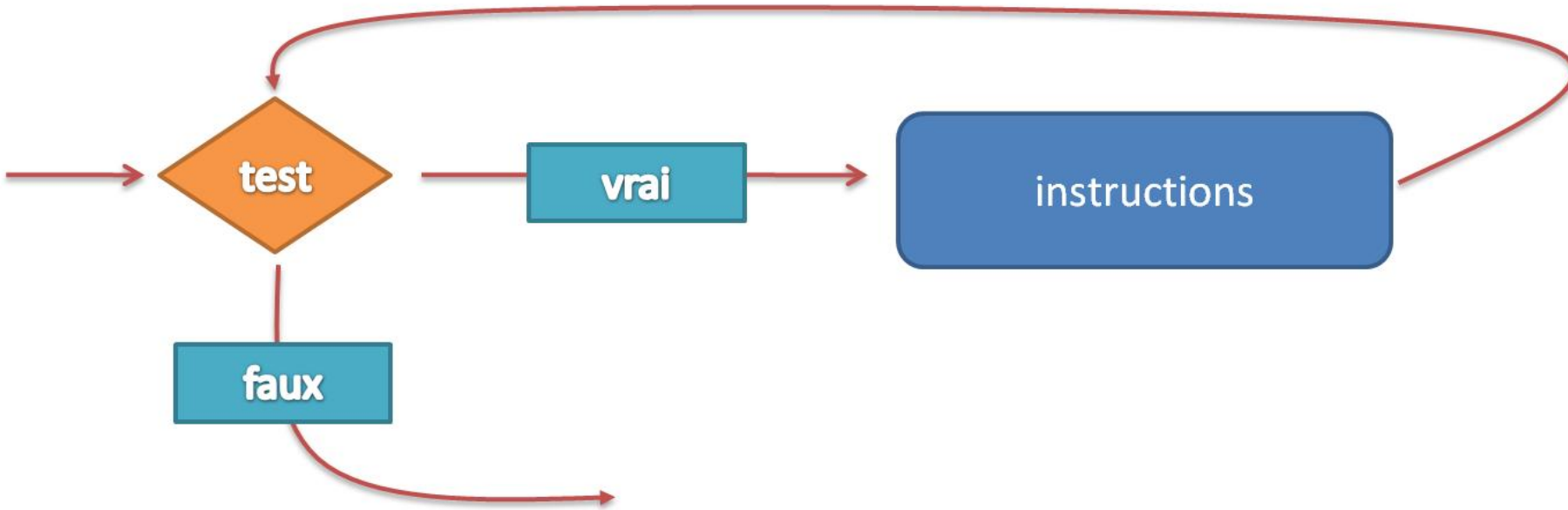
Itération

- Exemples d'itérations
  - Calcul de la moyenne arithmétique de plusieurs valeurs entrées par l'utilisateur :
    - Calcul de la somme des valeurs ;
    - Opération à répéter : addition.
  - Calcul de la puissance :
    - $x^n = x * x * ... * x$ (n fois) ;
    - Opération à répéter : multiplication.
  - Calcul la somme des N premiers termes de la série harmonique :
    - $S_n = 1 + 1/2 + 1/3 + ... + 1/N$;
    - Opération à répéter : addition.

Plan

Boucle while

- Elle permet de répéter un traitement (une ou plusieurs instructions) un nombre de fois indéterminé à l'avance.



- Tant que la condition (test) est vraie, on répète le traitement.
- Dès que la condition est fausse, l'exécution de la boucle est terminée.
- A la sortie de la boucle, la condition est toujours fausse.

Boucle while

- Syntaxe (une instruction à exécuter) :

  while <condition> **do**

  <instruction> ;

- Syntaxe (bloc d'instructions à exécuter)

  while <condition> **do**

  begin

  <bloc_instructions> ;

  end ;

- Le traitement n'est jamais exécuté si la condition est fausse au début.
- Si la condition n'est jamais fausse, la boucle est **infinie**.
- Les variables de la condition doivent être initialisées avant la boucle.

Boucle while

Exécution normale

- Exemple 1

```
1  program exemple1_while;
2  var
3    nb : integer;
4  begin
5    nb:=0;
6    while nb<=0 do
7      begin
8        write('Donner un entier positif ');
9        readln(nb);
10     end;
11   write('Le dernier nombre entre :', nb);
12 end.
```

Boucle while

Condition fausse au début

- Exemple 2

```
1  program exemple2_while;
2  var
3    nb : integer;
4  begin
5    nb:=10;
6    while nb<=0 do
7      begin
8        write('Donner un entier positif ');
9        readln(nb);
10     end;
11 end.
```

Boucle while

Boucle infinie

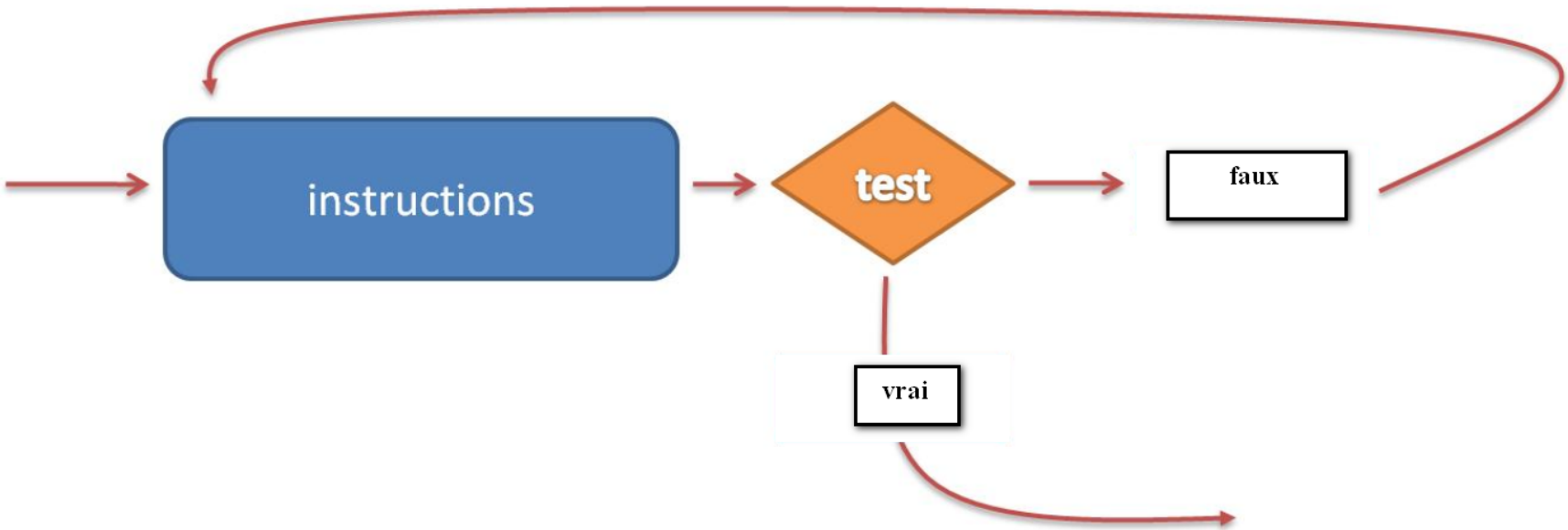
- Exemple 3

```
1 program exemple3_while;
2 var
3   i : integer;
4 begin
5   i:=5;
6   while i<10 do
7       write(i);
8 end.
```

Plan

Boucle repeat

- Comme la boucle while, elle permet de répéter un traitement un nombre de fois indéterminé à l'avance.



Boucle repeat

- Contrairement à la boucle while, on répète le traitement jusqu'à ce que la condition (test) soit **vérifiée**.
- Le traitement est effectué avant la vérification de la condition.
- Le traitement est exécuté **au moins une fois**.
- Dès que la condition est vraie, l'exécution de la boucle est terminée.

Boucle repeat

- Syntaxe

  repeat

    <instructions> ;

  until <condition> ;

- Si la condition n'est jamais vérifiée, alors la boucle est **infinie**.

Boucle repeat

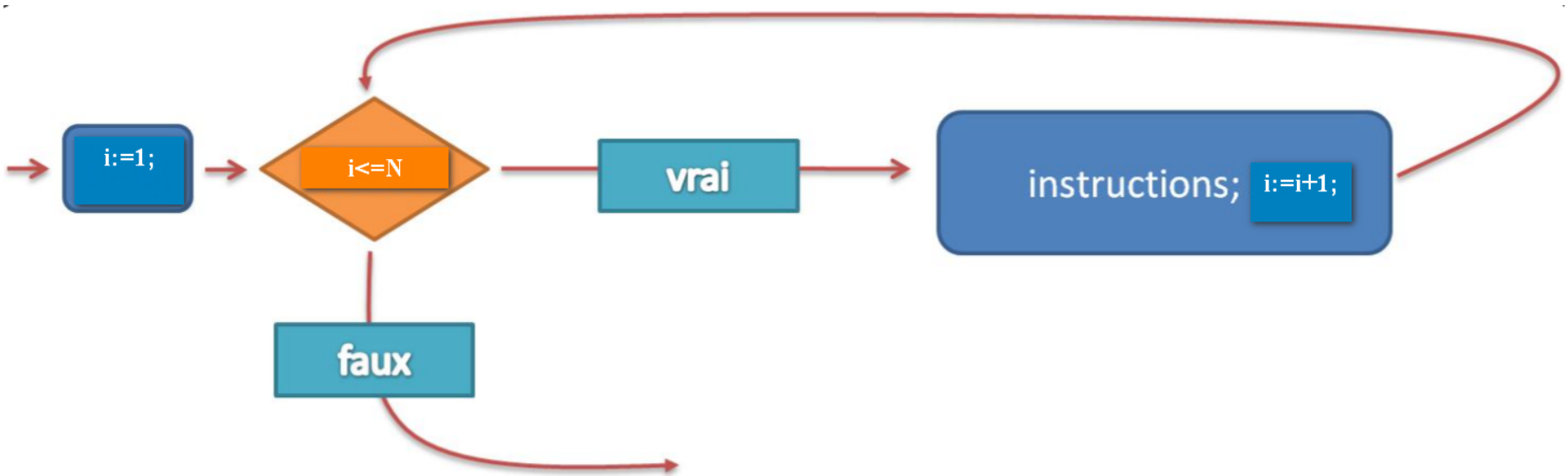
- Exemple

```
1  program exemple_repeat;
2  var
3    nb : integer;
4  begin
5    repeat
6      write('Donner un entier positif ');
7      readln(nb);
8    until nb>0;
9    write('Le dernier nombre entre :', nb);
10 end.
```

Plan

Boucle for

- Elle permet de répéter un traitement un certain nombre de fois (N fois, N connu à l'avance).



Boucle for

- Syntaxe (boucle croissante)

  for <identificateur> := <borne_inf> **to** <borne_sup> **do**

  <instruction>

- Syntaxe (boucle décroissante)

  for <identificateur> := <borne_sup> **downto** <borne_sup> **do**

  <instruction>

- NB : si le code à répéter est un bloc d'instructions, il faut le mettre entre begin et end.

Boucle for

- Éléments de la boucle :
  - <identificateur> est une variable appelée **variable d'itération**, de contrôle ou indice de la boucle.
  - <borne_inf> est la **valeur initiale** (borne inférieure).
  - <borne_sup> est la **valeur finale** (borne supérieure).
  - NB : L'identificateur et les bornes doivent être de même type.
- Exécution de la boucle for :
  - <identificateur> prend successivement par pas de 1 les valeurs comprises entre <borne_inf> et <borne_sup> ;
  - Le traitement est effectué pour chaque valeur de <identificateur>.

Boucle for

For avec une instruction

- Exemples

```
1 program exemple1_for;
2 var
3   i : integer;
4 begin
5   for i:=1 to 10 do
6     writeln(i);
7 end.
```

```
1 program exemple2_for;
2 var
3   i : integer;
4 begin
5   for i:=10 downto 1 do
6     writeln(i);
7 end.
```

Boucle for

For avec un bloc d'instructions

- Exemples

```
1  program exemple3_for;
2  var
3    i, som : integer;
4  begin
5    som:=0;
6    for i:=1 to 10 do
7      begin
8        write(i, ' ');
9        som:=som+i;
10     end;
11   writeln('La somme est ', som);
12 end.
```

Plan

Comparaison des boucles

- La boucle while est plus générale.
- Différence de repeat avec while
  - Le traitement est effectué avant la vérification de la condition.
  - Le traitement est exécuté au moins 1 fois.
  - Il n'est pas nécessaire d'encadrer un bloc d'instructions par un begin et un end.
- Différence de for avec while
  - Le pas 1 de la boucle for n'est pas modifiable.
  - Il est totalement interdit de modifier la valeur de la variable de contrôle dans la boucle.

Comparaison des boucles : while vs for

```pascal
program program_for;
var
  i : integer;
begin
  for i:=1 to 5 do
    writeln(i);
end.
```

```pascal
program program_while;
var
  i : integer;
begin
  i:=1;
  while i<=5 do
    begin
      writeln(i);
      i:=i+1;
    end;
end.
```

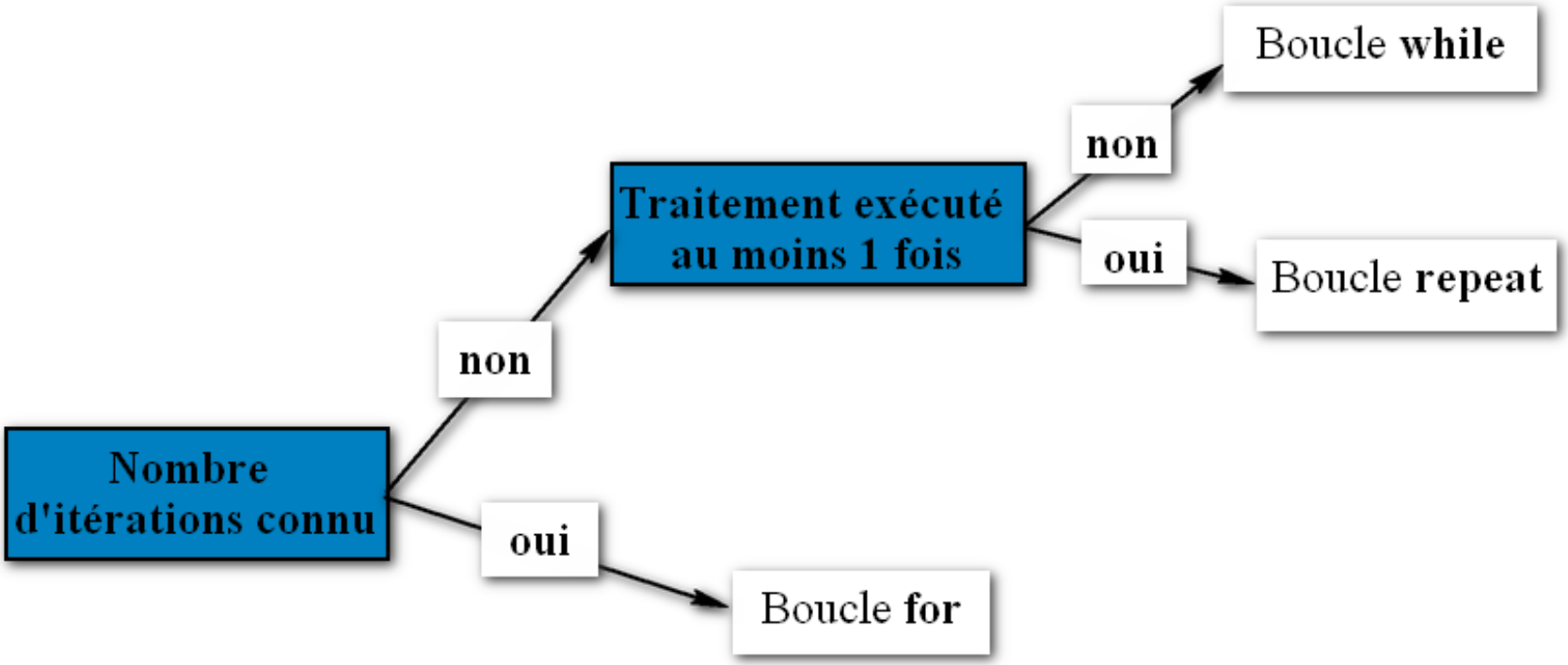
Comparaison des boucles : while vs repeat

```pascal
program program_while;
var
  i : integer;
begin
  i:=1;
  while i<=5 do
    begin
      writeln(i);
      i:=i+1;
    end;
end.
```

```pascal
program program_repeat;
var
  i : integer;
begin
  i:=1;
  repeat
    writeln(i);
    i:=i+1;
  until i>5;
end.
```

Synthèse sur les boucles

- Quelle boucle choisir ?



Plan

Exercices d'application

- Exercice 1

  Que fait le programme exo1 si l'utilisateur donne le nombre 2705 ? 426 ?

```
1  program exo1;
2  var
3    n, s : integer;
4  begin
5    write('Donner un entier positif ');
6    readln(n);
7    s:=0;
8    while n>0 do
9      begin
10       s:= s + n mod 10;
11       n:= n div 10;
12     end;
13   writeln('La valeur de s est ', s);
14 end.
```

Exercices d'application

- Exercice 2

  Écrire un programme qui demande à l'utilisateur d'entrer un entier positif n, puis affiche les nombres pairs inférieurs à n.

Exercices d'application

- Exercice 2

  Écrire un programme qui demande à l'utilisateur d'entrer un entier positif n, puis affiche les nombres pairs inférieurs à n.

- Correction :

```
1  program nombres_pairs;
2  var
3    i, n : integer;
4  begin
5    write('Donner un nombre entier positif : ');
6    readln(n);
7    write('Nombres pairs inferieurs a n : ');
8    for i:=1 to n-1 do
9      if (i mod 2) = 0 then
10       write(i, ' ');
11 end.
```

Exercices d'application

- Exercice 3

  Écrire un programme qui demande à l'utilisateur d'entrer un entier positif n, puis indique si c'est un nombre premier ou pas.

Exercices d'application

- Correction de l'exercice 3

```
1  program nombre_premier;
2  var
3    i, n : integer;
4    b : boolean;
5  begin
6    write('Donner un nombre entier positif : ');
7    readln(n);
8    b:=true;
9    for i:=2 to n div 2 do
10     if (n mod i) = 0 then
11       b:=false;
12   if b=true then
13     write(n, ' est un nombre premier')
14   else
15     write(n, ' n est pas un nombre premier');
16 end.
```

Exercices d'application

- Exercice 4 (boucles imbriquées)
  Écrire un programme qui demande à l'utilisateur d'entrer un entier positif n, puis affiche un carré d'étoiles de côté n.

Exercices d'application

- Exercice 4 (boucles imbriquées)
  Écrire un programme qui demande à l'utilisateur d'entrer un entier positif n, puis affiche un carré d'étoiles de côté n.
- Correction :

```
1  program carre_etoile;
2  var
3    i, j, n : integer;
4  begin
5    write('Donner un nombre entier positif : ');
6    readln(n);
7    for i:=1 to n do
8      begin
9        for j:=1 to n do
10         write(' * ');
11       writeln();
12     end;
13 end.
```