

# IFT 1010 - Programmation 1

## Boucles 1

Professeurs:  
Sébastien Roy & Balázs Kégl

B. Kégl, S. Roy, F. Duranleau  
Département d'informatique et de recherche opérationnelle  
Université de Montréal

automne 2004

1

## Au programme

- Boucles en général
- Boucles `while`, `do ... while` et `for`
- Indentation
- `Keyboard.error()` et `Keyboard.isEof()`
- Boucles infinies
- Boucles imbriquées

[Tasso:4], [Niño: 12.3, 12.6, 12.7]

2

## Les boucles de style "recette"

- Livre de cuisine:  
"pour chacun des 8 oeufs, séparer le jaune du blanc"  
au lieu de  
"prendre le 1<sup>er</sup> oeuf, séparer le jaune du blanc"  
"prendre le 2<sup>e</sup> oeuf, séparer le jaune du blanc"  
...  
"prendre le 8<sup>e</sup> oeuf, séparer le jaune du blanc"

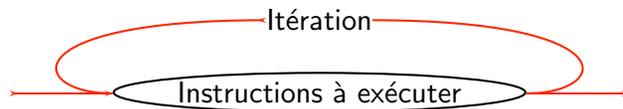
3

## Les boucles de l'ordinateur

- Calculer le prix de 100 billets d'avions:
  - répéter le code du calcul 100 fois?
  - et si l'utilisateur peut déterminer le nombre de billets lors de l'exécution?
- Boucles (structures de répétition)
  - moyens d'effectuer des répétitions

4

## Principe fondamental d'une boucle



- Itération
  - un processus dans lequel une **opération** est effectuée **plus d'une fois**
- Combien de fois?
  - il faut éventuellement **arrêter l'itération**
  - quand une **condition** devient fausse: boucle **while** (tant que)
  - **n fois**: boucle **for** (pour)

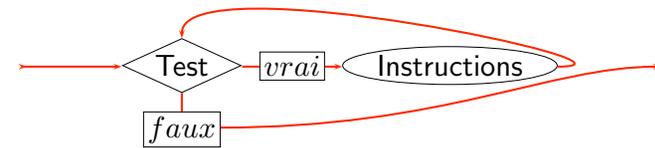
5

## Exemple

- Machine à café
  1. Poser la question: "Désirez-vous du sucre?"
  2. Attendre la réponse.
  3. WHILE la réponse est "OUI":
    4. Prendre un morceau de sucre.
    5. Déposer le morceau dans la tasse.
    6. Poser la question:
      7. "Désirez-vous un autre morceau de sucre?"
      8. Attendre la réponse.

7

## La boucle **while**



- Si le test est
  - **vrai** ⇒ on **entre dans la boucle** (une première fois ou à nouveau)
  - **faux** ⇒ on **contourne la boucle** et l'exécution poursuit son cours
- Dans la boucle, après l'exécution de ses instructions, on **retourne** à l'évaluation du **test**

6

## Boucler à l'infini

- Si l'évaluation du test est **toujours vrai**, on **bouclera à l'infini**
  - **erreur difficile** à trouver
- Pour **éviter**
  - il doit y avoir une **dépendance** entre le **test** et les **instructions** de la boucle
  - dans l'exemple: le test dépend de la réponse de l'utilisateur

8

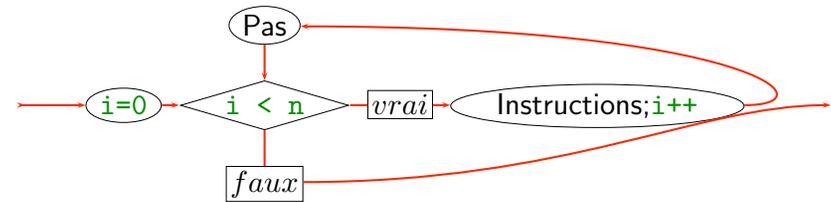
## Exemple

- Machine à café, avec clavier numérique

1. Poser la question:  
"Combien de morceaux de sucre désirez-vous?"
2. Attendre la réponse.
3.  $n = 0$ .
4. WHILE  $n < \text{réponse}$ :
5. Prendre un morceau de sucre.
6. Déposer le morceau dans la tasse.
7.  $n = n + 1$ .

9

## Un schéma fréquent de boucles



- Boucle **for**: spécialisation d'une boucle **while**
  - but: exécuter les instructions  $n$  fois
  - **compteur**  $i = 0, 1, \dots, n - 1$ : initialisé, incrémenté
  - **variantes**:  $i = \text{initVal}, i--, i \geq 0, i += 2, i *= 10$

10

## Exemple

- Style "recette" → pseudocode

1. Poser la question:  
"Combien de morceaux de sucre désirez-vous?"
2. Attendre la réponse.
3. FOR  $i = 1$  TO réponse:
4. Prendre un morceau de sucre.
5. Déposer le morceau dans la tasse.

- L'initialisation et le test de fin de boucle sont "camouflés" dans la spécification de l'intervalle  $i = 1$  TO réponse
- L'incrément est implicite

11

## Autre exemple

- Afficher les nombres pairs entre 0 et  $n$  un (inclusivement)

1. FOR  $i = 0$  TO  $n$  STEP 2
2. Afficher  $i$ .

- Équivalent à

1.  $i = 0$
2. WHILE  $i \leq n$ :
3. Afficher  $i$ .
4.  $i += 2$

12

## Savoir compter

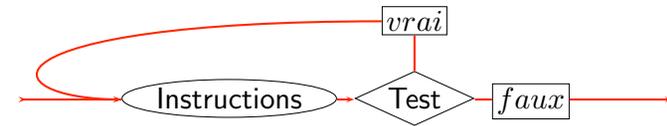
- Afficher les nombres pairs entre 0 et  $n$  **exclusivement** pour  $n$

```
1. FOR i = 0 TO n-1 STEP 2
2.   Afficher i.
```

```
1. i = 0
2. WHILE i <= n-1 ou WHILE i < n:
3.   Afficher i.
4.   i += 2
```

13

## La boucle do ... while



- Entrer dans la boucle **inconditionnellement**
  - exécuter les instructions de la boucle **au moins une fois**
- Exemple: saisie au clavier avec **vérification**

```
1. DO
2.   Demander l'entrée.
3.   Attendre la réponse.
4. WHILE réponse est invalide
```

14

## Et tout ceci, en **Java**?

## La boucle while

- Objectif
  - répéter un groupe d'instructions **tant que la condition est satisfaite**
- Syntaxe

```
while (<condition>)
  <instruction>
```

- normalement `<instruction>` = **bloc d'instructions**

15

16

## La boucle do ... while

- Objectif
  - répéter un groupe d'instructions tant que la condition est satisfaite,
  - mais au moins une fois

- Syntaxe

```
do
    <instruction>
while (<condition>)
```

- utilisé rarement

17

## La boucle for

- Schéma fréquent

```
int i = start;
while (i < end) {
    <séquence d'instructions>
    i++;
}
```

- Même programme avec for

```
for (int i = start; i < end; i++) {
    <séquence d'instructions>
}
```

18

## La boucle for

- Objectif
  - répéter un groupe d'instructions  $n$  fois où  $n$  est pré-spécifié

- Syntaxe

```
for (<initialisation>; <condition>; <itération>)
    <instruction>
```

19

## La boucle for

- En principe: for peut contenir n'importe quoi
- Le plus souvent:
  - <initialisation> initialise un compteur:  
`int i = 0`
  - <itération> in/décrémente le compteur:  
`i++` ou `i -= 5`
  - <condition> vérifie si le compteur est plus grand/plus petit qu'un seuil:  
`i <= 10` ou `i > 0`

20

## Attention!

```
int n = 0;
while( n < 10 );
    System.out.println( n );
    n = n + 1;
```

21

## Attention!

```
int n = 0;
while( n < 10 );
    System.out.println( n );
    n = n + 1;
```

- Boucle infinie!!

22

## Attention!

```
int n = 0;
while( n < 10 )
    System.out.println( n );
    n = n + 1;
```

23

## Attention!

```
int n = 0;
while( n < 10 )
    System.out.println( n );
    n = n + 1;
```

- Boucle infinie [encore!!](#)

24

## Attention!

```
int n = 0;
while( n < 10 ) {
    System.out.println( n );
    n = n + 1;
}
```

- Toujours utiliser des blocs

25

## Indentation

- Toujours faire attention de bien indenter en fonction des blocs

### Bon exemple

```
int nb;
boolean valide = false;
do
{
    System.out.print(
        "Entrez un nombre entre 1 et 100: " );
    nb = Keyboard.readInt();
    if( (nb < 1) || (nb > 100) )
    {
        System.err.println( "Entrée invalide." );
    }
    else
    {
        valide = true;
    }
}
while( !valide );
```

### Mauvais exemple

```
int nb;
boolean valide = false;
do
{
    System.out.print(
        "Entrez un nombre entre 1 et 100: " );
    nb = Keyboard.readInt();
    if( (nb < 1) || (nb > 100) )
    {
        System.err.println( "Entrée invalide." );
    }
    else
    {
        valide = true;
    }
}
while( !valide );
```

26

## Directives pour bien indenter

- Toute instruction dans un même bloc doit débiter à la même colonne
- Indentation d'au moins deux espaces (par rapport au bloc englobant)

```
englobant
{
    indenté
    ...
}
...
```

27

## Directives pour bien indenter

- Même nombre d'espaces pour tous les blocs

```
englobant1
{
    indenté1
    ...
    englobant2
    {
        indenté2
        ...
    }
}
...
```

28

## Directives pour bien indenter

- Instruction **coupée**: la suite doit être indentée.

```
grandTotal = total + fraisBase +  
    (int)((total + taxe) * TVQ + 0.5) + etc;  
  
System.out.println( "Le cercle de rayon " +  
    r + " a un perimetre de " +  
    p );
```

29

## Exemple 1

- Saisie au clavier avec **vérification**
  - lire un entier au clavier
  - si l'utilisateur entre **autre chose** qu'un entier, recommencer la lecture
  - **tant que** ce **n'est pas** un entier **valide**
- Quel type de boucle?
  - on doit lire **au moins une fois** ⇒ **do ... while**

31

## Enfin, plus d'exemples

30

## Exemple 1

- L'algorithme (pseudocode):
  1. **DO**
  2.     **Lire n.**
  3. **WHILE** n n'est pas un entier.

32

## À propos de Keyboard

- `Keyboard.error()`: expression booléenne
  - `true` si la dernière lecture faite avec `Keyboard.read...()` a généré une erreur
- Exemples
  - `Keyboard.readDouble()`, l'utilisateur a tapé une lettre
  - `Keyboard.readInt()`, l'utilisateur a tapé un nombre réel

33

## Exemple 1: code

- Premier essai

```
int n;
do
{
    n = Keyboard.readInt();
}
while (Keyboard.error());
```

34

## Exemple 1: code

- Premier essai

```
int n;
do
{
    n = Keyboard.readInt();
}
while (Keyboard.error());
```

- Messages?

35

## Exemple 1: code

- Deuxième essai

```
int n;
do
{
    System.out.print( "Entrez un entier: " );
    n = Keyboard.readInt();
}
while (Keyboard.error());
```

36

## Exemple 1: code

- Deuxième essai

```
int n;
do
{
    System.out.print( "Entrez un entier: " );
    n = Keyboard.readInt();
}
while (Keyboard.error());
```

- Message d'erreur?

37

## Exemple 1: code

- Troisième essai

```
int n;
do
{
    System.out.print( "Entrez un entier: " );
    n = Keyboard.readInt();
    if (Keyboard.error())
    {
        System.err.println( "Entrée invalide." );
    }
}
while (Keyboard.error());
```

38

## Exemple 1: code

- Troisième essai

```
int n;
do
{
    System.out.print( "Entrez un entier: " );
    n = Keyboard.readInt();
    if (Keyboard.error())
    {
        System.err.println( "Entrée invalide." );
    }
}
while (Keyboard.error());
```

- Plus efficace?

39

## Exemple 1: code

- Solution finale

```
int n;           // le nombre lu
boolean invalide; // indique une erreur de lecture
do
{
    System.out.print( "Entrez un entier: " );
    n = Keyboard.readInt();

    // un premier test pour signaler l'erreur:
    invalide = Keyboard.error(); // appelée 1 seule fois
    if (invalide)
    {
        System.err.println( "Entrée invalide." );
    }
}
while (invalide);
```

40

## Exemple 2

- Somme d'une séquence
  - lire une **séquence de nombre** au clavier
  - tant qu'une **fin d'entrée** n'est pas lue
  - calculer leur **somme**
- Quel type de boucle?
  - fin d'entrée peut se produire avant le premier nombre ⇒ boucle **while**

41

## Fin d'entrée

- Caractère spécial pour **signaler la fin** des commandes ou entrées
  - **CTRL**-d, si on lit au clavier
  - **fin de fichier**, si on lit d'un fichier
- Comment détecter?
  - **Keyboard.isEof()**: expression booléenne
  - **true** si la fin d'entrée a été lue lors de la **dernière** **Keyboard.read...()**

42

## Exemple 2: algorithmme

- Algorithme (pseudocode):

```
1. somme = 0
2. Lire valeur.
3. WHILE la fin d'entrée n'est pas lu:
4.     somme += valeur
5.     Lire valeur.
```

- OU

```
1. somme = 0
2. DO
3.     Lire valeur.
4.     IF la fin d'entrée n'est pas lu:
5.         somme += valeur
6. WHILE la fin d'entrée n'est pas lu.
```

43

## Exemple 2: code

```
public class Sommaton
{
    public static void main(String[] arg)
    {
        int somme = 0; // initialisation de la somme à 0
        int valeur = Keyboard.readInt(); // valeur courante, 1ere lecture

        // Tant que PAS à la fin d'entrée (d'où le '!')
        while (!Keyboard.isEof())
        {
            somme += valeur; // accumulation de la somme
            valeur = Keyboard.readInt(); // lecture de la valeur suivante
        }

        // Affichage de la somme produite
        System.out.println("La somme est " + somme);
    }
}
```

44

## Rappel

- Affectation + arithmétique

```
n += v;  ≡  n = n + v;  
n -= v;  ≡  n = n - v;  
n *= v;  ≡  n = n * v;  
n /= v;  ≡  n = n / v;
```

- Incrémenter/décrémenter

```
n++;  ≃  ++n;  ≡  n += 1;  ≡  n = n + 1;  
n--;  ≃  --n;  ≡  n -= 1;  ≡  n = n - 1;
```

45

## Exemple 3

- Intérêt

- étant donné le **solde initial**, le **solde souhaité** et le **taux d'intérêt**, **combien d'années** seront nécessaires pour atteindre le solde souhaité
- au lieu d'utiliser la formule, on **simule** le calcul

- Algorithme (pseudocode):

```
1. ans = 0;  
2. WHILE solde n'atteint pas à solde souhaité  
3.   incrémenter ans  
4.   ajouter l'intérêt au solde
```

46

## Exemple 3

- Solution

```
int years = 0;  
double balance = 1000;  
double targetBalance = 1500;  
double rate = 5; // en %  
while (balance < targetBalance) {  
    years++;  
    double interest = balance * rate / 100;  
    balance += interest;  
}  
System.out.println(years + " years are needed");
```

47