Haute École de Bruxelles
École Supérieure d'Informatique

TD Alternatives

Résumé

Ce TD a pour but d'aborder les structures alternatives qui permettent de conditionner des parties d'algorithmes, de code. Elles ne seront exécutées que si une condition est satisfaite.

1 si - alors - sinon

Cette structure permet d'exécuter une partie de code ou une autre en fonction de la valeur de vérité d'une condition.

1.1 si-alors-fin si

En pseudo-code :

```
si condition alors
      // instructions à réaliser si la condition est VRAIE
fin si
```

La **condition** est une expression délivrant un résultat **booléen** (vrai ou faux) ; elle associe des variables, constantes, expressions arithmétiques, au moyen des opérateurs logiques ou de comparaison. En particulier, cette condition peut être réduite à une seule variable booléenne.

Dans cette structure, lorsque la **condition est vraie**, il y a exécution de la séquence d'instructions contenue entre les mots **alors** et **fin si** ; ensuite, l'algorithme continue de façon séquentielle.

Lorsque la **condition est fausse**, les instructions se trouvant entre **alors** et **fin si** sont tout simplement **ignorées**.

```
module Test
    nombre1 : entier
    lire nombre1
    si (nombre1 < 0) alors
      afficher nombre1, " est négatif"
    fin si
fin module
```

En Java :

```
if (condition) {
     // instructions si la condition est VRAIE
}
```

La **condition** est une expression délivrant un résultat **booléen** (true ou false).

Dans cette structure, lorsque la **condition est vraie**, il y a exécution de la séquence d'instructions contenue entre les mots **et** ; ensuite, le code continue de façon séquentielle.

Lorsque la **condition est fausse**, les instructions se trouvant entre **et** sont tout simplement **ignorées**.

```
import java. util .Scanner;
public class Test {
    public static void main(String [] args) {
      Scanner clavier = new Scanner(System.in);
      int nombre1;
      nombre1 = clavier. nextInt ();
      if (nombre1 < 0) {
        System.out.println (nombre1 + " est negatif");
      }
    }
}
```

1.2 si-alors-sinon-fin si

En pseudo-code :

```
si condition alors
      // instructions à réaliser si la condition est VRAIE
sinon
      // instructions à réaliser si la condition est FAUSSE
fin si
```

Dans cette structure, une et une seule des deux séquences est exécutée.

```
// Lit un nombre et affiche si ce nombre est positif (zéro inclus) ou strictement né
module signeNombre()
    nb : entier
    lire nb
    si nb < 0 alors
      afficher "le nombre", nb, " est négatif"
    sinon
      afficher "le nombre", nb, " est positif ou nul"
    fin si
fin module
```

En Java :

```
if (condition) {
      // instructions si la condition est VRAIE
} else {
      // instructions si la condition est FAUSSE
}
```

```
import java. util .Scanner;
public class Test {
    public static void main(String [] args) {
      Scanner clavier = new Scanner(System.in);
      int nombre1;
      nombre1 = clavier. nextInt ();
      System.out. println (nombre1 + " est un nombre ");
      if (nombre1 < 0) {
        System.out. println (" negatif ");
      } else {
        System.out. println (" positif ");
      }
    }
}
```

2 selon que

Avec ces structures, plusieurs branches d'exécution sont disponibles. L'ordinateur choisit la branche à exécuter en fonction de la valeur d'une variable (ou parfois d'une expression) ou de la condition qui est vraie.

2.1 selon que (version avec listes de valeurs)

En pseudo-code :

```
selon que variable vaut
    liste_1 de valeurs séparées par des virgules :
      // instructions lorsque la valeur de la variable est dans liste_1
    liste_2 de valeurs séparées par des virgules :
      // instructions lorsque la valeur de la variable est dans liste_2
    ...
    liste_n de valeurs séparées par des virgules :
      // instructions lorsque la valeur de la variable est dans liste_n
    autres :
      // instructions lorsque la valeur de la variable
      // ne se trouve dans aucune des listes précédentes
fin selon que
```

Notez que le cas `autres` est facultatif.

Dans ce type de structure, comme pour la structure `si-alors-sinon`, une seule des séquences d'instructions sera exécutée. On veillera à ne pas faire apparaitre une même valeur dans plusieurs listes. Cette structure est une simplification d'écriture de plusieurs alternatives imbriquées.

Elle est équivalente à :

```
si variable = une des valeurs de la liste_1 alors
    // instructions lorsque la valeur est dans liste_1
sinon
    si variable = une des valeurs de la liste_2 alors
      // instructions lorsque la valeur est dans liste_2
    sinon
      ...
      si variable = une des valeurs de la liste_n alors
        // instructions lorsque la valeur est dans liste_n
      sinon
        // instructions lorsque la valeur de la variable
        // ne se trouve dans aucune des listes précédentes
      fin si
    fin si
fin si
```

Écrivons un algorithme qui lit un jour de la semaine sous forme d'un nombre entier (1 pour lundi, . . ., 7 pour dimanche) et qui affiche en clair ce jour de la semaine.

```
// Lit un nombre entre 1 et 7 et affiche en clair le jour de la semaine correspondan
module jourSemaine()
    jour : entier
    lire jour
    selon que jour vaut
      1 : afficher "lundi"
      2 : afficher "mardi"
      3 : afficher "mercredi"
      4 : afficher "jeudi"
      5 : afficher "vendredi"
      6 : afficher "samedi"
      7 : afficher "dimanche"
    fin selon que
fin module
```

En Java :

```
switch (variable){
    case val1 :
      // instructions lorsque la valeur de la variable est val1
      break;
    case val2 :
    case val3 :
    case val4 :
      // instructions lorsque la valeur de la variable est val2 ou val3 ou val4
      break;
    ...
    case valN  :
      // instructions lorsque la valeur de la variable est valN
      break;
    default :
      // instructions lorsque la valeur de la variable
      // ne se trouve dans aucune des listes précédentes
}
```

Notez que le cas `default` est facultatif.

Notez le `break` à la fin de chaque (groupe de) `case`.

La variable peut être de type `byte`, `short`, `char`, `intString` et les types énumérés que nous verrons plus tard.

Elle est équivalente à :

```
if (variable == val1){
      // instructions lorsque la valeur de la variable est val1
} else if (variable == val2 || variable == val3 || variable == val4){
      // instructions lorsque la valeur de la variable est val2 ou val3 ou val4
} else if (variable == valN){
      // instructions lorsque la valeur de la variable est valN
} else {
      // instructions lorsque la valeur de la variable
      // ne se trouve dans aucune des listes precedentes
}
```

Par exemple :

```
import java.util.Scanner;
public class Test{
  public static void main(String[] args){
      Scanner clavier = new Scanner(System.in);
      String produit = clavier.next();
      switch(produit) {
        case "Coca" :
```

```
        case "Sprite" :
        case "Fanta" :
        prixDistributeur = 60;
        break;
      case "IceTea" :
        prixDistributeur = 70;
        break;
      default :
        prixDistributeur = 0;
        break;
      }
      System.out.println(prixDistributeur);
    }
}
```

2.2 selon que (version avec conditions)

En pseudo-code :

```
selon que
    condition_1 :
      // instructions lorsque la condition_1 est vraie
    condition_2 :
      // instructions lorsque la condition_2 est vraie
    ...
    condition_n :
      // instructions lorsque la condition_n est vraie
    autres :
      // instructions à exécuter quand aucune
      // des conditions précédentes n'est vérifiée
fin selon que
```

Comme précédemment, une et une seule des séquences d'instructions est exécutée. On veillera à ce que les conditions ne se «recouvrent »pas, c'est-à-dire que deux d'entre elles ne soient jamais vraies simultanément.

C'est équivalent à :

```
si condition_1 alors
    // instructions lorsque la condition_1 est vraie
sinon
    si condition_2 alors
      // instructions lorsque la condition_2 est vraie
    sinon
      ...
```

```
      si condition_n alors
        // instructions lorsque la condition_n est vraie
      sinon
        // instructions à exécuter quand aucune
        // des conditions précédentes n'est vérifiée
      fin si
    fin si
fin si
```

Par exemple :

```
// Lit un nombre et affiche si ce nombre est strictement positif , strictement négat
module signeNombre()
    nb : entier
    lire nb
    selon que
      nb < 0 :
        afficher "le nombre", nb, " est négatif"
      nb > 0 :
        afficher "le nombre", nb, " est positif"
      autres :
        afficher "le nombre", nb, " est nul"
    fin selon que
fin module
```

En Java :

Il n'existe pas de `switch` avec condition, il faut l'écrire comme une succession de `if`.

```
if (condition_1){
      // instructions lorsque la condition_1 est vraie
} else if (condition_2){
      // instructions lorsque la condition_2 est vraie
} ...
} else if (condition_n){
      // instructions lorsque la condition_n est vraie
} else {
      // instructions a executer quand aucune
      // des conditions precedentes n est verifiee
}
```

Par exemple :

```
import java. util .Scanner;
public class Test {
```

```
    public static void main(String [] args) {
      Scanner clavier = new Scanner(System.in);
      int nb = clavier.nextInt();
      if (nb>0) {
        System.out.println("␣positif␣");
      } else if (nb==0) {
        System.out.println("nul");
      } else {
        System.out.println("␣negatif␣");
      }
    }
}
```

3 Exercices

Maintenant, mettons tout ça en pratique.

3.1 Compréhension d'algorithme

Pour ces exercices, nous vous demandons de comprendre des algorithmes donnés.

Compréhension

Que vont-ils afficher ?

—
```
module exerciceA()
  a,b : entier
  lire a,b
  si a > b alors
    a ← a+2*b
  fin si
  afficher a
fin module
```
Si les nombres lus sont respectivement 2 et 3 ?
__ Si les nombres lus sont respectivement 4 et 1 ?
__

—
```
module exerciceB()
    a,b,c : entier
    lire b,a
    si a > b alors
      c ← a DIV b
    sinon
      c ← b MOD a
    fin si
```

```
    afficher c
fin module
```

Si les nombres lus sont respectivement 2 et 3 ?

__ Si les nombres lus sont respectivement 4 et 1 ?

__

```
— module exerciceC ()
    x1, x2 : entier
    ok : booléen
    lire x1, x2
    ok ← x1 > x2
    si ok alors
      ok ← ok ET x1 = 4
    sinon
      ok ← ok OU x2 = 3
    fin si
    si ok alors
      x1 ← x1 * 1000
    fin si
    afficher x1 + x2
fin module
```

Si les nombres lus sont respectivement 2 et 3 ?

_____ Si les nombres lus sont respectivement 4 et 1 ?

_____

3.2 Compréhension de codes Java

Pour ces exercices, nous vous demandons de comprendre des codes Java donnés.

Compréhension

Que vont-ils afficher si à chaque fois les deux nombres lus au départ sont successivement 2, 3 et 4 ?

—

```java
import java.util.Scanner;
public class Exercice1 {
    public static void main(String [] args) {
        Scanner clavier = new Scanner(System.in);
        int nb1 = clavier.nextInt();
        int nb2 = clavier.nextInt();
        int nb3 = clavier.nextInt();
        if (nb1 < nb2){
          System.out.print(nb1);
        } else {
          System.out.print(nb2);
        }
    }
```

```
}
```

```
import java.util.Scanner;
public class Exercice2 {
    public static void main(String [] args) {
        Scanner clavier = new Scanner(System.in);
        int nb1 = clavier.nextInt();
        int nb2 = clavier.nextInt();
        int nb3 = clavier.nextInt();
        if (nb1 > nb2 && nb1 > nb3){
          System.out.print(nb1);
        } else {
            if (nb2 > nb3){
              System.out.print(nb2);
            } else {
              System.out.print(nb3);
            }
        }
    }
}
```

```
import java.util.Scanner;
public class Exercice3 {
    public static void main(String [] args) {
        Scanner clavier = new Scanner(System.in);
        int nb1 = clavier.nextInt();
        int nb2 = clavier.nextInt();
        int nb3 = clavier.nextInt();
        switch (nb1){
          case 1 : System.out.print("premier"); break;
          case 2 : System.out.print("deuxieme"); break;
          case 3 : System.out.print("troisieme"); break;
          default : System.out.print("pas␣dans␣le␣trio");
        }
    }
}
```

```
import java.util.Scanner;
public class Exercice3 {
    public static void main(String [] args) {
        Scanner clavier = new Scanner(System.in);
        int nb1 = clavier.nextInt();
        int nb2 = clavier.nextInt();
        int nb3 = clavier.nextInt();
        switch (nb1){
          case 1 : System.out.print("premier");
          case 2 : System.out.print("deuxieme");
          case 3 : System.out.print("troisieme");
          default : System.out.print("pas␣dans␣le␣trio");
        }
    }
}
```

3.3 À vous de jouer...

Il est temps de se lancer et d'écrire vos premiers modules et programmes Java correspondant. Voici quelques conseils pour vous guider dans la résolution de tels problèmes :

- il convient d'abord de bien comprendre le problème posé ; assurez-vous qu'il est parfaitement spécifié ;
- déclarez ensuite les variables (et leur type) qui interviennent dans l'algorithme ; les noms des variables risquant de ne pas être suffisamment explicites ;
- **mettez en évidence les variables «données », les variables «résultats »et les variables de travail ;**
- n'hésitez pas à faire une ébauche de résolution en français avant d'élaborer l'algorithme définitif pseudo-codé.
- Écrivez la partie algorithmique **AVANT** de vous lancer dans la programmation en Java.

Écrivez les algorithmes et codez les programmes Java correspondant qui

1. étant donné deux nombres quelconques, recherche et affiche le plus grand des deux. Attention ! On ne veut pas savoir si c'est le premier ou le deuxième qui est le plus grand mais bien quelle est cette plus grande valeur. Le problème est donc bien défini même si les deux nombres sont identiques.
2. étant donné trois nombres quelconques, recherche et affiche le plus grand des trois.
3. affiche un message indiquant si un entier est strictement négatif, nul ou strictement positif.
4. étant donné trois nombres, recherche et affiche si le premier des trois appartient à l'intervalle donné par le plus petit et le plus grand des deux autres (bornes exclues). Qu'est-ce qui change si on inclut les bornes ?
5. étant donné une équation du deuxième degré, déterminée par le coefficient de x^2, le coefficient de x et le terme indépendant, recherche et affiche la (ou les) racine(s) de l'équation (ou un message adéquat s'il n'existe pas de racine réelle).
6. à partir d'un moment exprimé par 2 entiers, heure et minute, affiche le moment qu'il sera une minute plus tard.
7. vérifie si une année est bissextile. Pour rappel, les années bissextiles sont les années multiples de 4. Font exception, les multiples de 100 (sauf les multiples de 400 qui sont bien bissextiles). Ainsi 2012 et 2400 sont bissextile mais pas 2010 ni 2100.

Stationnement alternatif

Dans une rue où se pratique le stationnement alternatif, du 1 au 15 du mois, on se gare du côté des maisons ayant un numéro impair, et le reste du mois, on se gare de l'autre côté. Écrivez un algorithme et le code java correspondant qui, sur base de la date du jour et du numéro de maison devant laquelle vous vous êtes arrêté, indique si vous êtes bien stationné ou non.

La fièvre monte

Chez l'humain la température corporelle normale moyenne est de 37 °C (entre 36,5 °C et 37,5 °C selon les individus). La fièvre est définie par une température rectale au repos supérieure ou égale à 38,0 °C. Une fièvre au-delà de 40 °C est considérée comme un risque de santé majeur et immédiat. Lorsque la fièvre est modérée (de 37,7 °C à 37,9 °C), on parle de fébricule.

[Wikipedia]

Écrivez un module fièvre qui lit une température au clavier et qui affiche si le patient a de la température (supérieure ou égale à 38,0°C) ET si cette cette fièvre est modérée (entre 38,0°C et 40,0°C) ou à risque (strictement supérieur à 40,0°C). Rien ne doit être affiché si le patient n'a pas de fièvre.

Écrivez le code java correspondant.

Taxes communales

Dans ma commune, les taxes communales des enlèvements des immondices s'élèvent à

- 80€ pour une personne isolée ;
- 135€ pour une famille de 2 ou 3 personnes ;
- 175€ pour une famille de 4 personnes ou plus.

Écrivez un module qui lit le nombre de personnes composant la famille et qui affiche le prix de la taxe à payer.

Écrivez le code java correspondant.

Au cinéma

À Bruxelles, lors de chaque projection cinématographique, une taxe de 0,5€ est prélevée sur le prix du billet de chaque spectateur.

- Écrivez un module qui lit le nombre de spectateurs et qui affiche le prix de la taxe à payer.
- Écrivez le code java correspondant.

— Si le film projeté est un documentaire, aucune taxe n'est prélevée. Écrivez un module qui lit le nombre de spectateurs et un booléen (à vrai si le film est un documentaire et faux sinon) et qui affiche le prix de la taxe à payer.
— Écrivez le code java correspondant.