



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	2
1.1	si-alors-fin si	2
1.2	si-alors-sinon-fin si	3
2	selon que	4
2.1	selon que (version avec listes de valeurs)	4
3	Exercices	7
3.1	Compréhension d'algorithme	7
3.2	Compréhension de codes Java	8
3.3	À vous de jouer...	10



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**.

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

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**.

```
public static void negatif (int nombre1) {  
    if (nombre1 < 0) {  
        System.out.println (nombre1 + " est négatif");  
    }  
}
```

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 négatif  
algorithme signeNombre (nb : entier)  
    si nb < 0 alors  
        afficher "le nombre", nb, " est négatif"  
    sinon  
        afficher "le nombre", nb, " est positif ou nul"  
    fin si  
fin algorithme
```

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 ();  
        signe(nombre1);  
    }  
}
```

```

    }
    public static void signe(int nombre1) {
        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 apparaître 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 correspondant
algorithme jourSemaine (jour : entier)
    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 algorithme

```

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 précédentes
}

```

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);
    }
}

```

```
}  
}
```

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 retourner/afficher ?

```
— algorithme exerciceA (a, b : entiers) → entier  
    si a > b alors  
        a ← a+2*b  
    fin si  
    retourner a  
fin algorithme  
Si les nombres reçus sont respectivement 2 et 3 ?  
__ Si les nombres reçus sont respectivement 4 et 1 ?  
—  
— algorithme exerciceB()  
    a, b, c : entiers  
    demander b, a  
    si a > b alors  
        c ← a DIV b  
    sinon  
        c ← b MOD a  
    fin si  
    afficher c  
fin algorithme  
Si les nombres lus sont respectivement 2 et 3 ?  
__ Si les nombres lus sont respectivement 4 et 1 ?  
—  
— algorithme exerciceC ()  
    x1, x2 : entiers  
    ok : booléen  
    demander 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 algorithme

```

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 ?

```

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();
        System.out.println(max(nb1, nb2));
    }
    public static int max(int nb1, int nb2) {
        int max;
        if (nb1 < nb2){
            max= nb1;
        } else {
            max = nb2;
        }
        return max;
    }
}

```

```

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();
System.out.println(max(nb1, nb2, nb3));
}
public static int max(int nb1, int nb2, int nb3) {
    int max;
    if (nb1 > nb2 && nb1 > nb3){
        max = nb1;
    } else {
        if (nb2 > nb3){
            max = nb2;
        } else {
            max = nb3;
        }
    }
    return max;
}
}

```

```

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();
        System.out.println(trio(nb1));
    }
    public static void trio(int nb1) {
        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();
        System.out.println(trio(nb1));
    }
    public static void trio(int nb1) {
        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...

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 ;
- résolvez le problème via quelques exemples précis ;
- 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 ;
- déclarez ensuite les variables (et leur type) qui interviennent dans l'algorithme ; les noms des variables risquant de ne pas être suffisamment explicites.
- 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 retourne le plus grand des trois.
3. retourne un message indiquant si un entier est strictement négatif, nul ou strictement positif.
4. étant donné trois nombres, recherche et retourne 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, retourne 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 algorithme **fièvre** qui reçoit une température et qui affiche si le patient a de la température (supérieure ou égale à 38,0°C) ET si 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 algorithme qui reçoit le nombre de personnes composant la famille et qui retourne 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 algorithme qui reçoit le nombre de spectateurs et qui retourne 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 algorithme qui reçoit le nombre de spectateurs et un booléen (à vrai si le film est un documentaire et faux sinon) et qui retourne le prix de la taxe à payer.
- Écrivez le code java correspondant.