R1.01 - Initiation au développement
BUT Informatique
Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr
2024–2025

TD n°2

Objectifs : Utiliser des **conditions** (expressions booléennes) et des **alternatives simples** (structures de contrôles).

Remarque : Dans les énoncés, un ou plusieurs exemples d'exécutions attendues du programme sont souvent indiqués sous la forme suivante :

```
En épaisseur standard, le texte affiché par le programme
En gras, le texte entré par l'utilisateur
```

Attention, les programmes que vous réalisez doivent bien fonctionner pour ces exemples mais aussi dans l'ensemble des autres cas qui pourraient se présenter.

Exercice 1 : Conditions [EXPR-BOOL*]

Dans cet exercice, il vous sera demandé d'écrire différentes conditions.

1. Une plante a besoin d'une température comprise entre 8 et 16 degrés et d'une humidité de plus de 50%. En supposant que vous avez des variables `temperature` et `tauxHumidite`, exprimez la condition permettant à la plante de survivre.
2. Pour valider un semestre de DUT (le diplôme précédent), il était nécessaire d'avoir une moyenne générale supérieure ou égale à 10 et aucune moyenne d'unité d'enseignement (UE) en dessous de 8. En supposant qu'il y a deux unités d'enseignement et qu'on dispose des trois variables suivantes, `moyenneGenerale`, `moyenneUE1` et `moyenneUE2`:
   — Quel est le type de ces variables ?
   — Donnez la condition exprimant la validation d'un semestre à l'aide de ces variables.
3. Une bande de copains à la fin d'un repas se demandent s'ils vont faire une partie de bowling ou aller au cinéma. En supposant que vous avez deux variables `bowling` indiquant si le bowling a été choisi, et `cinema` indiquant si le cinéma a été choisi:
   — Donnez le type de ces variables.
   — Donnez la condition exprimant le fait que le choix fait est raisonnable (une seule activité a été choisie).
4. Pour partir en stage, un étudiant doit avoir un sujet de stage validé, être accepté par l'entreprise et avoir une convention de stage signée ou en cours de signature. En supposant que vous avez les variables `stageValide`, `accordEntreprise` , `conventionSignee` et `enCours`:
   — Quel est le type de chacune des variables ?
   — Donnez la condition exprimant la possibilité de départ en stage.

Exercice 2 : Chanter sous la pluie [ALT-SMPL*]

Les gens qui habitent dans le Nord, lorsqu'il pleut et qu'ils sont de bonne humeur, ils chantent (sous la pluie ;), sinon ils rient. On suppose dans cet exercice que l'on dispose des variables `pluie` et `bonneHumeur` et que l'on affiche juste les messages `"I'm singing in the rain ..."` et `"Ah ah ah ah !"`.

1. Donnez la condition qui est vérifiée lorsque les ch'ti chantent.
2. Donnez le code de l'alternative affichant le chant ou le rire en fonction des variables données ci-dessus.

Exercice 3 : Premier programme avec une alternative [ALT-SMPL]

On souhaite disposer d'un programme iJava qui reçoit comme entrée utilisateur une note (type `double`) et affiche si la moyenne est atteinte ou non en fonction de la note. On considère que l'utilisateur entre bien une note entre 0 et 20 inclus (on ne gérera pas les autres situations). Exemples d'exécutions attendues :

```
Entrez une note sur 20 : 14,5
La moyenne est atteinte
```

```
Entrez une note sur 20 : 9,89
La moyenne n'est pas atteinte
```

1. Compléter le programme qui suit.

```
1  class CommentaireMoyenne extends Program {
2    void algorithm() {
3      println("Entrez une note sur 20 : ");
4      double note = readDouble();
5      ...
6      ...
7      ...
8      ...
9      ...
10     ...
11   }
12 }
```

2. Dessiner l'organigramme de votre programme.

Exercice 4 : Déterminer la nature d'un caractère [ALT-CASC*]

Concevez un programme `Nature` indiquant la nature d'un caractère saisi, c'est-à-dire le classifiant en numérique, alphabétique ou autre.

RAPPEL: Les caractères numériques sont à la suite dans la table ASCII, ils sont donc compris entre `'0'` et `'9'`. De la même manière, les lettres de l'alphabet en majuscules sont comprises entre `'A'` et `'Z'` et celles minuscules entre `'a'` et `'z'`.

Votre programme attend un caractère de l'utilisateur et affiche `"NUMÉRIQUE"`, `"ALPHABÉTIQUE"` ou `"AUTRE"` (tous les autres types de caractères tels que `'='`, `'è'`, `'.'` ...).

Exemples d'exécutions attendues :

```
Entrez un caractère : A
ALPHABÉTIQUE
```

```
Entrez un caractère : 3
NUMÉRIQUE
```

```
Entrez un caractère : z
ALPHABÉTIQUE
```

```
Entrez un caractère : -
AUTRE
```

1. Écrire le programme `Nature` respectant les spécifications ci-dessus.

2. Dessiner l'organigramme de votre programme

Exercice 5 : Conditions complexes [EXPR-BOOL]

Nous allons dans cet exercice nous entraîner un peu à l'expression de **conditions** et à leur utilisation au sein d'une instruction **alternative**. L'algorithme `aOuB` consiste à afficher la lettre `A` ou la lettre `B` en fonction de la valeur de deux variables entières `x` et `y` ne pouvant prendre que la valeur 0 ou 1.

Pour le premier exemple, la table à utiliser pour associer les valeurs numériques aux caractères `'A'` et `'B'` est la suivante:

	y = 0	y = 1
x = 0	'A'	'A'
x = 1	'A'	'B'

1. En supposant que l'on a en entrée deux entiers (compris entre 0 et 1 inclus), donnez le corps de la fonction `aOuB` permettant d'afficher la lettre `A` ou la lettre `B` selon les conditions du tableau donné ci-dessus. Vous veillerez à écrire un algorithme qui **utilise une seule alternative**.

```
1  class ConditionsComplexes extends Program {
2
3    void algorithm() {
4      int x,y;
5      char aOuB;
6      print("Entrez␣une␣valeur␣pour␣x␣:␣");
7      x = readInt();
8      print("Entrez␣une␣valeur␣pour␣y␣:␣");
9      y = readInt();
10     //À COMPLÉTER pour que la variable aOuB ait la bonne valeur à la fin
11     ...
12
13
14
15     println(aOuB);
16   }
17
18 }
```

Exemples d'exécutions attendues :

```
Entrez une valeur pour x : 0
Entrez une valeur pour y : 0
A
```

```
Entrez une valeur pour x : 1
Entrez une valeur pour y : 1
B
```

2. Quelle partie du code faut-il changer si l'on désire utiliser un autre tableau ? Entourez-la.
3. Proposez le code à mettre la place pour les tableaux suivants:

	y = 0	y = 1
x = 0	'B'	'A'
x = 1	'A'	'B'

	y = 0	y = 1
x = 0	'A'	'A'
x = 1	'B'	'B'

	y = 0	y = 1
x = 0	'A'	'B'
x = 1	'B'	'A'

Prolongement

Exercice 6 : Qui suis-je

Le programme `QuiSuisJe` qui sert de cadre à cet exercice a été volontairement conçu pour être difficilement compréhensible :) Le but n'est donc pas de chercher à comprendre ce qu'il fait, mais de réussir à effectuer quelques traces d'exécution pour observer les différents flux d'exécution.

1. Donnez la trace d'exécution de cet algorithme avec les entrées suivantes en utilisant un organigramme de la structure du programme.

Entrées	Sorties
0, 0, 0	
1, 0, 5	
2, 4, -6	
0, 0, 34	
2, 4, 2	

2. Question subsidiaire : à quoi correspond cet algorithme ?

```
1  class QuiSuisJe extends Program {
2    void algorithm(){
3      double nbRoues, prix, somme, deltaplane, kazu, saya;
4      nbRoues = readDouble();
5      prix = readDouble();
6      somme = readDouble();
7      if (nbRoues == 0) {
8        if (prix == 0) {
9          if (somme == 0) {
10           println("Tout␣est␣possible␣!");
11         } else {
12           println("Rien␣n'est␣possible␣!");
13         }
14       } else {
15         kazu = - somme / prix;
16         println("Seul␣" + kazu + "␣est␣admis.");
17       }
18     } else {
19       deltaplane = 2 * nbRoues * 2 * somme;
20       deltaplane = prix * prix - deltaplane;
21       if (deltaplane >= 0) {
22         if (deltaplane == 0) {
23           kazu = 2 * nbRoues;
24           kazu = (-1 * prix) / kazu;
25           println( kazu + "␣est␣admis␣deux␣fois.");
26         } else {
27           somme = 2 * nbRoues;
28           prix = -1 * prix;
29           kazu = (prix -1 * pow(deltaplane, 0.5)) / somme;
30           saya =  (prix + pow(deltaplane, 0.5)) / somme;
31           println( kazu + "␣et␣" + saya + "␣sont␣acceptés.");
32         }
33       } else {
34         println("Rien␣n'est␣possible␣!");
35       }
36     }
37   }
38 }
```