

**Objectifs:** Écrire des fonctions qui utilisent des alternatives complexes et des boucles à compteur.

## Boucles

### Exercice 1 : Écho sans echo [REP-FOR-COUNT]

Concevez le programme `Echo` qui prend une chaîne de caractères `phrase` et un nombre `n` et affiche `n` fois la chaîne `phrase` avec un retour à la ligne entre chaque occurrence de `phrase`. Voici deux exemples d'exécutions attendues.

```
Chaîne à répéter : Hello
Nombre de fois : 2
Hello
Hello
```

```
Chaîne à répéter : Joyeux anniversaire
Nombre de fois : 3
Joyeux anniversaire
Joyeux anniversaire
Joyeux anniversaire
```

### Exercice 2 : Table de conversion d'euros en yens [REP-FOR-COUNT]

Concevez le programme `EuroToYen` affichant une table de conversion entre euro et yens japonais (en supposant que le taux est de 1 euro = 135.90 yens). Le programme demandera à l'utilisateur d'entrer le nombre de lignes qu'il désire et les affiche ensuite. Voici un exemple d'exécution.

```
Combien de lignes souhaitez-vous ? 11
1 euros = 135.9 yens.
2 euros = 271.8 yens.
3 euros = 407.70000000000005 yens.
4 euros = 543.6 yens.
5 euros = 679.5 yens.
6 euros = 815.4000000000001 yens.
7 euros = 951.3000000000001 yens.
8 euros = 1087.2 yens.
9 euros = 1223.1000000000001 yens.
10 euros = 1359.0 yens.
11 euros = 1494.9 yens.
```

### Exercice 3 : Factorielle [REP-ACC-NUM]

Concevez le programme `Factorielle` calculant la factorielle d'un nombre  $n$  (notée  $n!$ ), sachant que  $0! = 1$  et que  $n! = 1 \times 2 \times \dots \times n$ . Le programme prendra entrée un entier naturel  $n$  saisi par l'utilisateur et affiche la valeur de la factorielle de  $n$ . Voici quelques exemples à essayer, n'hésitez pas à en ajouter.

Saisie utilisateur	Affichage attendu
3	6
4	24
0	1
1	1

## Exercice 4 : Table HT vers TTC

Le but de cet exercice est d'écrire deux algorithmes qui affichent la conversion d'un prix HT en un prix TTC (en considérant une TVA de 19,6%). La première version repose uniquement sur le nombre de lignes devant être affichées car l'on commence toujours à partir de la valeur 1 euro. La seconde version nécessite en plus du nombre de lignes, la valeur de départ (10 dans l'exemple donné ci-dessous).

Combien de lignes ? 10

```
1 euros HT = 1.196 euros TTC.
2 euros HT = 2.392 euros TTC.
3 euros HT = 3.588 euros TTC.
4 euros HT = 4.784 euros TTC.
5 euros HT = 5.98 euros TTC.
6 euros HT = 7.176 euros TTC.
7 euros HT = 8.372 euros TTC.
8 euros HT = 9.568 euros TTC.
9 euros HT = 10.764 euros TTC.
10 euros HT = 11.96 euros TTC.
```

Combien de lignes ? 8

A partir de ? 10

```
10.0 euros HT = 11.96 euros TTC.
10.5 euros HT = 12.558 euros TTC.
11.0 euros HT = 13.156 euros TTC.
11.5 euros HT = 13.754 euros TTC.
12.0 euros HT = 14.352 euros TTC.
12.5 euros HT = 14.95 euros TTC.
13.0 euros HT = 15.548 euros TTC.
13.5 euros HT = 16.146 euros TTC.
```

1. Écrire un programme `HTversTTC1` qui demande à l'utilisateur le nombre de lignes qu'il souhaite et affiche le tableau de la première colonne ci-dessus.<sup>1</sup>
2. Définissez un deuxième programme `HTversTTC2` pour avoir l'affichage correspondant à la deuxième colonne, en supposant que l'utilisateur donne en plus du nombre de lignes la première valeur (ie. 10 dans cet exemple).

## Alternatives plus complexes

Dans les deux exercices qui suivent, vous devez écrire des programmes liés aux heures de la journée.

## Exercice 5 : Vous avez un moment ? [ALT-COMB]

Le programme `MomentJournee` indique si c'est la nuit, le matin, l'après midi ou la soirée, en fonction d'une heure reçue. On considère que le matin commence à 6h, l'après-midi commence à 12h, la soirée commence à 18h, et la nuit commence à 22h. Si l'heure donnée en paramètre n'est pas comprise dans l'intervalle  $[0, 23]$ , la fonction affichera erreur.

Voici une série de valeurs avec lesquelles mettre à l'épreuve votre programme :

Saisie utilisateur	Affichage attendu
0	nuit
4	nuit
6	matinée
11	matinée
14	après-midi
19	soirée
23	nuit
24	erreur
-3	erreur

## Exercice 6 : À la bonne heure ! [ALT-COMB]

- Alors qu'en France l'heure de la journée est généralement affichée sur 24h, aux États-Unis, elle l'est sur 12h. Ainsi,
- en France, l'heure peut prendre une valeur entre 0 et 23.
  - aux États-Unis (mais pas seulement), l'heure prend une valeur entre 1 et 12 compris. Les heures avant midi sont représentées par l'heure, suivie de "AM", sauf 0h qui s'écrirait 12h00 AM. Les heures de l'après midi sont représentées par un nombre entre 1 et 12, suivi de "PM".

---

1. L'affichage des valeurs de type `double` peut être légèrement différent de ce que vous voyez ci-dessus.

1. On souhaite ici écrire le programme `HeuresUSVersFrance` permettant de passer de la manière d'écrire l'heure de la journée à l'américaine (sur 12h) à la manière française (sur 24h). On supposera dans tout cet exercice que l'heure et les minutes saisies sont toujours correctes (i.e. pas de nombres négatifs, pas de minutes  $\geq 60$ ).

Voici une série de valeurs avec lesquelles mettre à l'épreuve votre programme :

Saisies utilisateur			Affichage attendu
5	20	AM	5:20
5	20	PM	17:20
10	49	PM	22:49
12	10	AM	0:10
12	10	PM	12:10
1	05	AM	1:05

Le dernier scénario est optionnel, il présente une difficulté supplémentaire : prendre en compte l'éventuel 0 des dizaines sur les minutes dans le formatage

2. On souhaite désormais écrire le programme `HeuresFranceVersUS` qui fait l'opération inverse : pour une heure française saisie, il nous affichera l'heure dans son format US.

Saisie utilisateur		Affichage attendu
5	20	5:20AM
17	20	5:20PM
22	49	10:49PM
0	10	12:10AM
12	10	12:10PM

## Structures de contrôles imbriquées

### Exercice 7 : Remplacement d'un caractère [REP-FILTRE, REP-ACC-STR]

Cet exercice consiste à manipuler des chaînes de caractères.

1. Que fait le début de programme qui suit ?

```
class Remplacement extends Program{
    void algorithm() {
        String txt;
        char ancien, nouveau;
        println("Veuillez_saisir_votre_texte_");
        txt = readString();
        print("Caractère_à_replacer_");
        ancien = readChar();
        print("Caractère_de_replacement_");
        nouveau = readChar();
    }
}
```

2. Écrivez la suite d'instructions à ajouter à la fin du programme ci-dessus afin qu'il construise et affiche une nouvelle version du texte saisi, dans laquelle on a remplacé toutes les occurrences du caractère par celui utilisé pour le remplacer. Voici quelques scénarios à vérifier :

Saisies utilisateur			Affichage attendu
HELLO	o	O	HELLO
Blabla	i	u	Blabla
***TITRE***	*	x	xxxTITRExxx

## Prolongements

### Exercice 8 : Ordonner trois caractères [ALT-COMB]

Concevez le programme `Tri` permettant d'ordonner, selon l'ordre ASCII, trois caractères quelconques en complétant le squelette suivant.

```

class Tri extends Program {
    void algorithm() {
        char c1, c2, c3;
        c1 = readChar();
        c2 = readChar();
        c3 = readChar();

        // à compléter pour que c1, c2 et c3 contiennent les caractères saisis dans l'ordre
        ASCII

        println("'" + c1 + c2 + c3); //à garder tel quel
    }
}

```

Voici une série de valeurs avec lesquelles mettre à l'épreuve votre programme :

Saisies utilisateur			Affichage attendu
a	b	c	abc
b	a	c	abc
c	b	a	abc
b	c	a	abc
a	b	c	abc
c	a	b	abc
e	d	b	bde

Question subsidiaire : combien de comparaisons de nombres contient votre fonction ?


Il est possible de résoudre ce problème en n'utilisant que trois fois un symbole de comparaison. Trouvez cette solution.

### Exercice 9 : Nombre de majuscules [REP-FOR-STR, REP-FILTRE, REP-ACC-NUM]

Écrire un programme `NombreMajuscules` qui compte et affiche le nombre de majuscules contenues dans un texte saisi par un utilisateur. On ne s'occupera pas des majuscules accentuées.

### Exercice 10 : Nombre de mots

On suppose que l'utilisateur donne en entrée une phrase représentée sous la forme d'une chaîne de caractères et l'on souhaite calculer et afficher le nombre de mots contenus dans cette phrase.

Voici quelques exemples de comportement attendus (où  indique une saisie vide et `_` indique la saisie d'un espace):

Saisie utilisateur	Affichage attendu
test	1
	0
_	0
test	1
test test	2
À demain !	3

1. Écrivez le programme `CompterMots` qui correspond à la description ci-dessus.
2. Que faut-il modifier dans votre algorithme pour ne pas compter les signes de ponctuation (", " "; " ." "?" " !") comme des mots ?