FORMATION

Algorithmie

06/06/2023

Glodie Tshimini

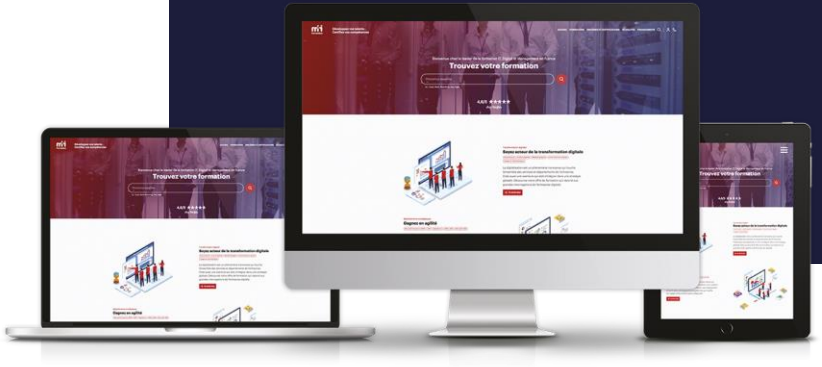
ALGORITHMIE

PLAN

Installation

I. Introduction
II. Variables et opérateurs
III. Structures de contrôle
IV. Fonction et tableaux
V. Tableaux multidimensionnels

INSTALLATION



m2iformation.fr

INSTALLATION

- Vscode ou un autre éditeur de code de votre choix.

Windows
Windows 8, 10, 11

User Installer	x64	x86	Arm64
System Installer	x64	x86	Arm64
.zip	x64	x86	Arm64
CLI	x64	x86	Arm64

.deb
Debian, Ubuntu

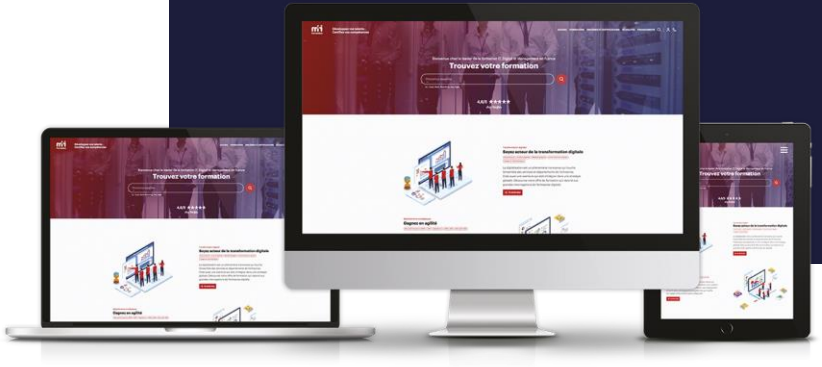
.rpm
Red Hat, Fedora, SUSE

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

Mac
macOS 10.11+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

I. INTRODUCTION



m2iformation.fr

QU'EST-CE QU'UN ALGORITHME ?

- **Suite d'opérations élémentaires permettant d'obtenir le résultat final déterminé à un problème**. *(source : Apprendre à programmer Christophe Dabancourt).*
- Décrit un traitement sans l'exécuter sur une machine.
- Fournit un résultat identique dans des conditions similaires.

- Exemples du quotidien
  - Recette de cuisine ;
  - Aller d'un point A à un point B .

QU'EST-CE QU'UN PROGRAMME ?

- **Exécution de l'algorithme sur une machine.**
- Pour que le programme puisse être exécuté, il faut utiliser un langage que la machine peut comprendre : un **langage de programmation**.
- Suite d'instructions qui sont évaluées par le processeur sur lequel tourne le programme.
- Les **instructions** utilisées dans le programme représente le **code source**.

DIFFÉRENCES ENTRE LANGAGES DE BAS ET HAUT NIVEAU

BAS NIVEAU

Un langage de bas niveau est un langage qui est considéré comme **plus proche du langage machine** (binaire) plutôt que du langage humain.

Il est en général plus difficile à apprendre et à utiliser mais **offre plus de possibilité d'interactions** avec le hardware de la machine.

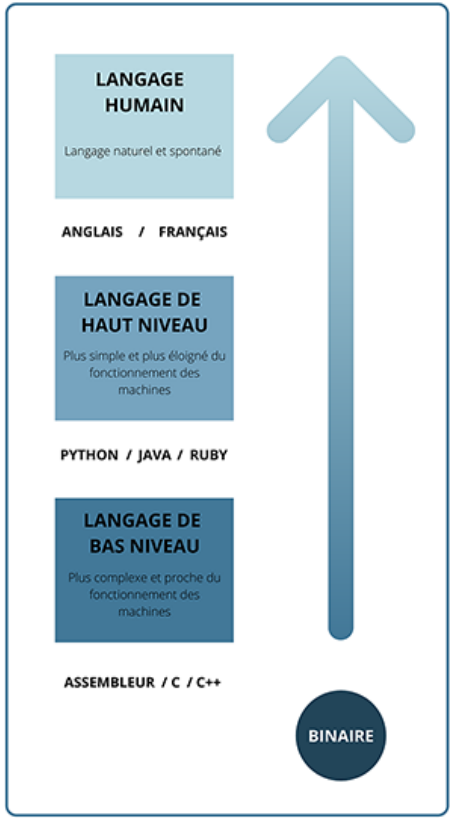
HAUT NIVEAU

Un langage de haut niveau est le contraire, il **se rapproche plus du langage humain** et est par conséquent **plus facile à appréhender**.

Cependant, les **interactions sont limitées** aux fonctionnalités que le langage met à disposition.

ILLUSTRATION

Source image Machine Learning et Deep Learning - ENI



DIFFÉRENCES COMPILATION ET INTERPRETATION

COMPILATION

La compilation d'un programme consiste à **transformer toutes les instructions en langage machine** avant que le programme puisse être exécuté.

Par conséquent il sera nécessaire de refaire la compilation après chaque modification du code source.

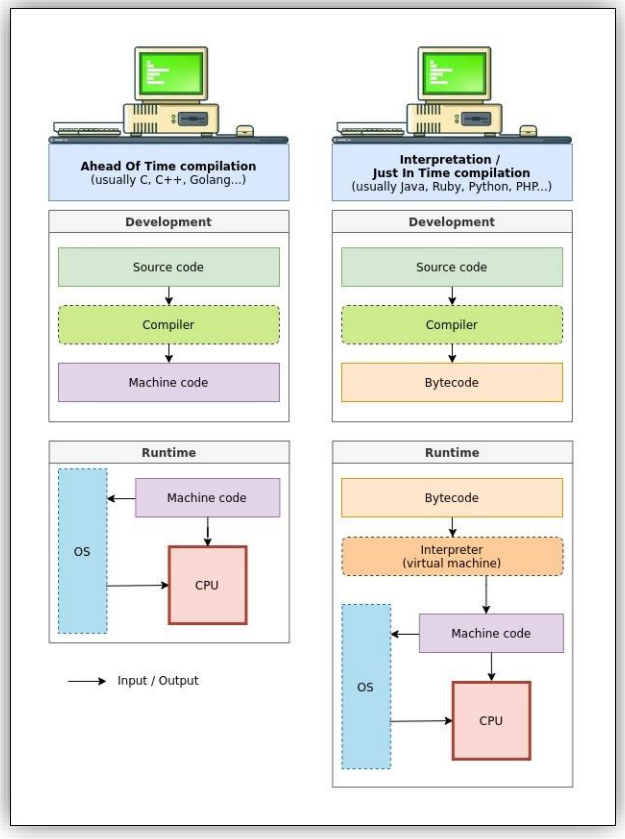
INTERPRETATION

L'interprétation d'un programme consiste à **traduire les instructions en temps réel** (on run time).

Dans ce cas le code source est lu à chaque exécution et par conséquent les changements apportés au code seront pris en compte immédiatement.

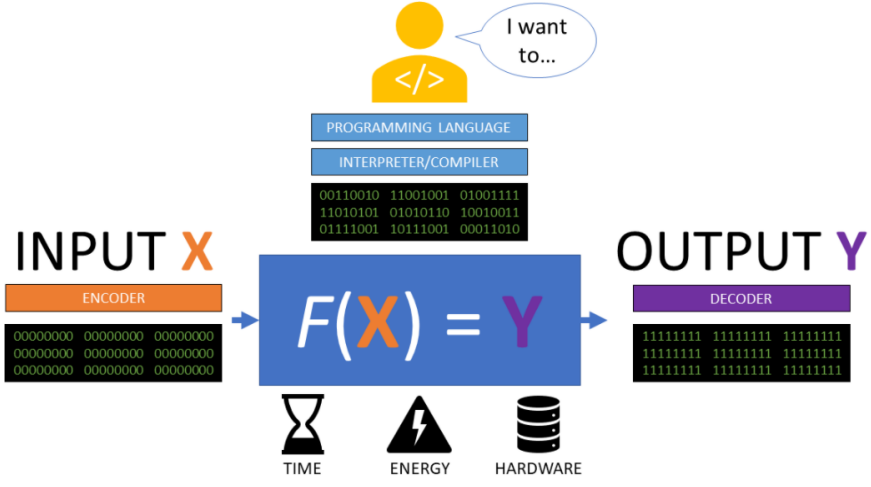
ILLUSTRATION

Source image thevaluable.dev



RESUMÉ

Source image edutechwiki

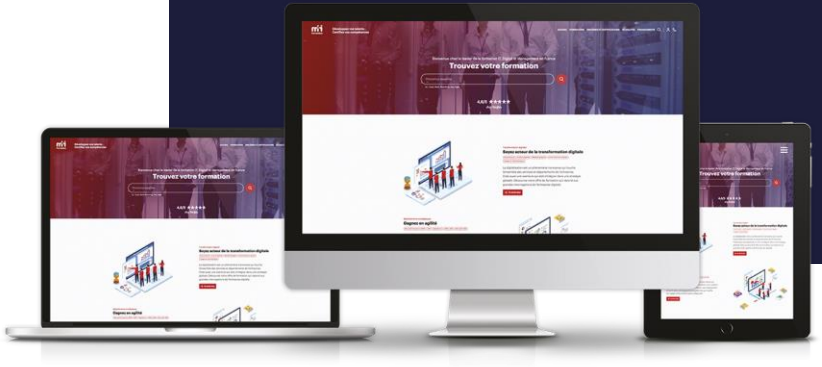


- **F(X) = Y :**
  - *X* représente l'*input* (**entrée**)
  - *Y* représente l'*ouput* (**sortie**)
  - *F()* représente la ***fonction*** qui permet de transformer X en Y

STRUCTURES DE PROGRAMMATION

- Les éléments fondamentaux :
  - Les variables ;
  - Les opérateurs ;
  - Les structures de contrôle
  - Les fonctions ;
  - Les tableaux (ou array) ;
  - Les objets.

II. VARIABLES ET OPÉRATEURS

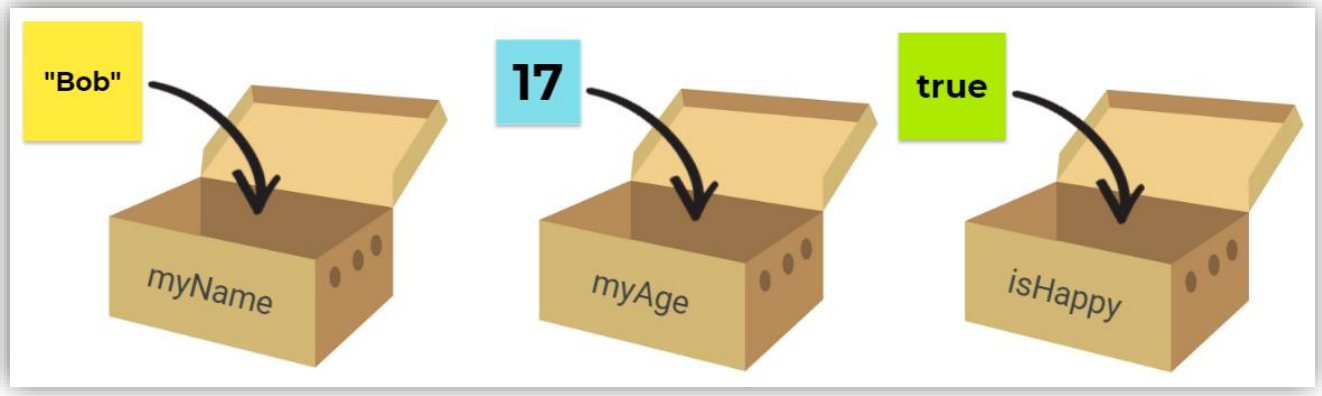


VARIABLE

- **Emplacement mémoire qui permet de stocker une information.**
- Définie par :
  - Un ***nom unique*** ;
  - Un ***type de données*** ;
  - Une ***valeur*** (information) qui peut être modifié au cours du déroulement de l'algorithme .
- Utilité des variables :
  - **Manipuler et stocker temporairement des données (informations) .**

VARIABLE

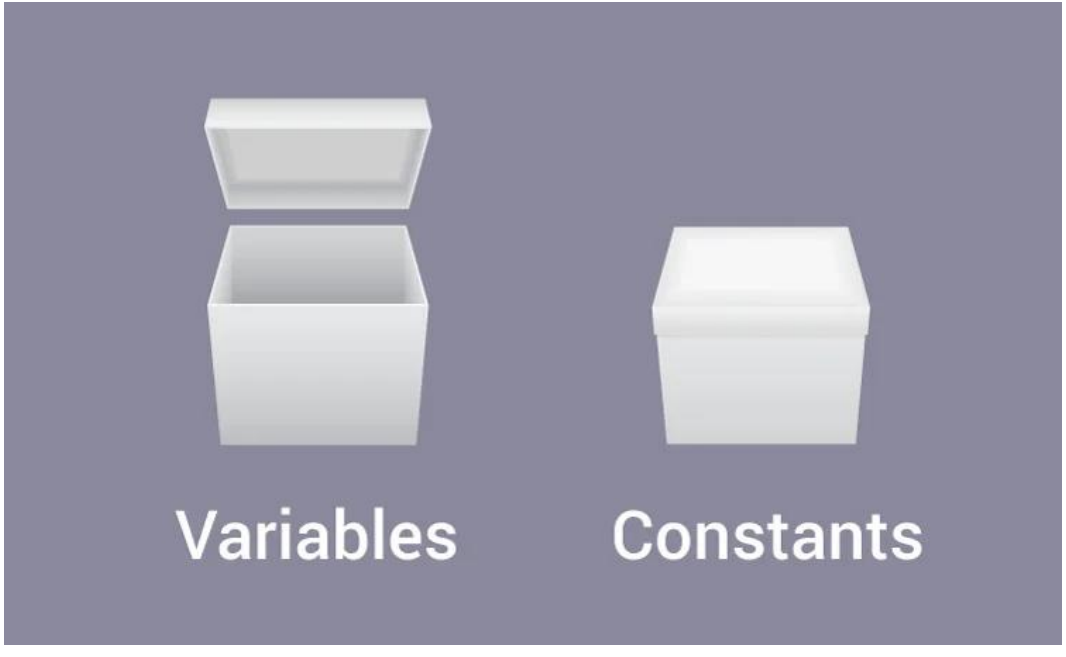
- On peut imaginer une **boîte** avec une **étiquette**.
- Une boite peut contenir un ou plusieurs éléments.
- L'étiquette sert en tant que **référence** de cette boîte.
- Il est possible de mettre des boîtes dans des boîtes.



Source image Romanan Rumapea kotlin data types

CONSTANTE

- Une variable dont la valeur n'évolue pas au cours de l'exécution d'un programme.



Source image Lata Singh

TYPE OU DOMAINE DE DÉFINITION

- **Ensemble des valeurs que peut prendre une variable**
- **Primitifs** :
  - Un caractère alphanumérique (**char**) par extension une suite de caractère alphanumérique (**string**) : utilisées pour représenter du texte. *Exemples : "a" ,"hello world", "5".*
  - Chiffres(**number**), nombre entier, à virgule flottante, etc. : utilisés surtout avec des opérateurs mathématiques. *Exemples : 10 , 12.5* .
  - Valeurs booléennes (**booleans**) : valeurs dichotomiques (soit vrai, soit faux). *Exemples : **true, false.***

OPÉRATEURS

- Opérateur d'**affectation** ou d'**assignation** : = ou <-
- Les opérateurs **mathématiques** : +, -, /, DIV et MOD (%)
- Les opérateurs de **comparaison** : égalité (==), différence (!=), majeur (>) ou mineur (<)
- Les opérateurs **logiques** : AND (&) , OR (||) , NOT (!) .

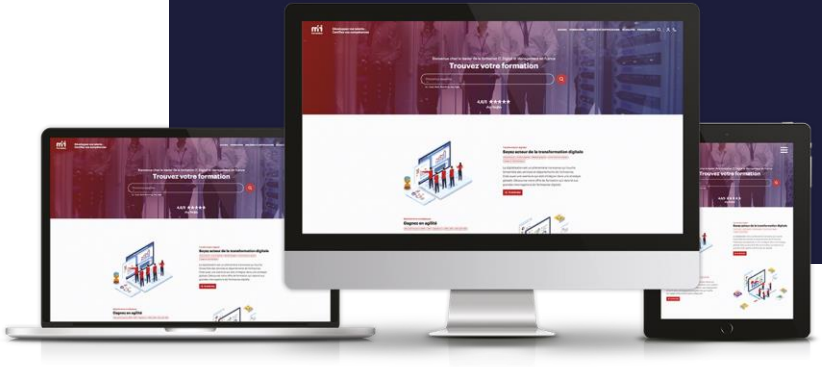
- Opérateur de **concaténation** : +
- Opérateurs d'**incrémentation** : ++
- Opérateurs de **décrémentation** : --

EXERCICE 1 : variables et types

EXERCICE 1

2-exercices/exercice1.md

III. Structures de contrôle

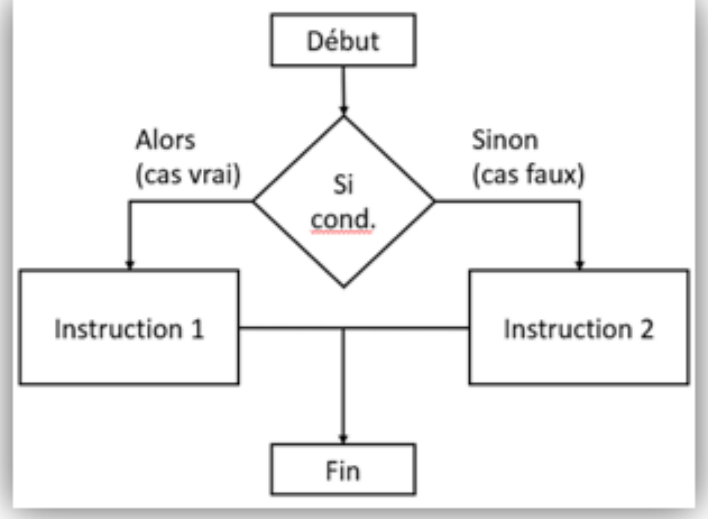


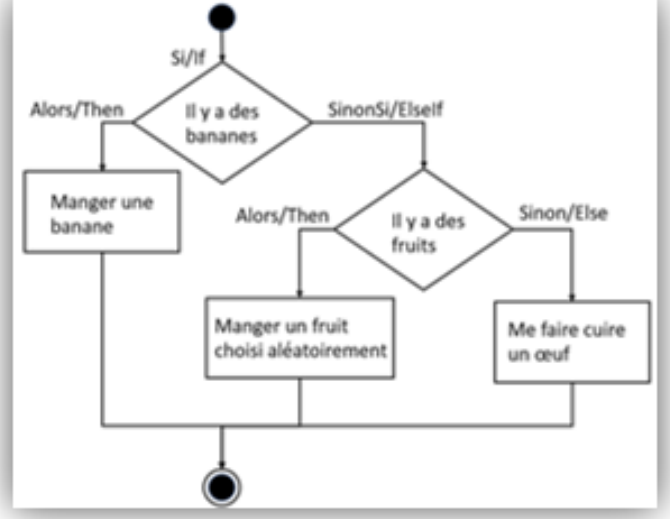
STRUCTURE CONDITIONNELLE

Les structures de contrôle permettent **d'exécuter seulement certaines instructions d'un programme selon la vérification d'une ou plusieurs conditions**.

La version sémantique la plus répandue des structures de contrôle est « **si… alors…sinon** ».

[Source images Pro du code]





EXERCICES 2 à 4 : structures de contrôle

EXERCICES 2 à 4

2-exercices/exercice2.md

2-exercices/exercice3.md

2-exercices/exercice4.md

Structures itératives (boucles)

- Les boucles sont à la base d'un concept très utile en programmation : l'**itération**.
- L'itération permet d'exécuter de manière récursive **(plusieurs fois) des instructions**.

Source image Forum cheat gam3

BOUCLE TANT QUE

- **S'exécute tant qu'une condition est respectée.**
- Utilise un **opérateur d'incrémentation** pour **éviter une boucle infinie**.
- Exemples d'utilisation :

Tant que la liste n'est pas totalement parcouru :

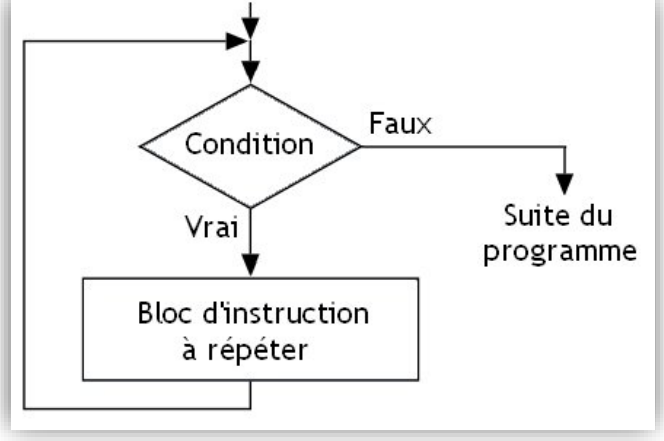
Affiche l'élément de la liste

Fin de Tant que

Tant que ce chiffre ne dépasse pas 16 :

Réalise un calcul

Fin de Tant que



Source image zeste de savoir

BOUCLE FAIRE TANT QUE

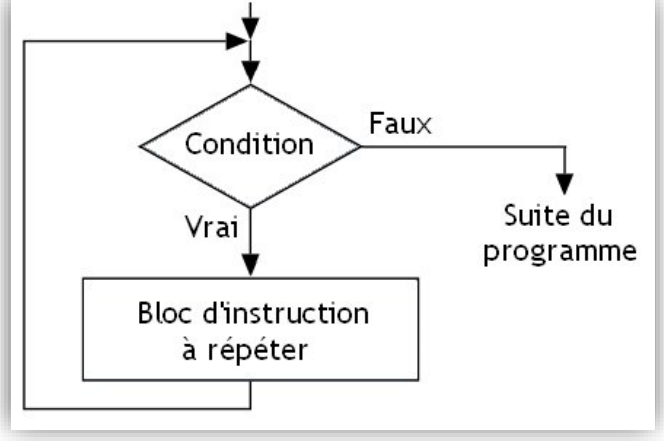
- La boucle **Faire…tant que** aussi appelée **Répéter…tant que**
  - Similaire à la boucle Tant Que.
  - **La condition est évalué à la fin.**
  - **S'exécute au moins une fois.**
- Exemple :

Faire :

Ecrire 'Entrez un nombre >= à 10'

Lire nombre

Tant que (nombre < 10)



Source image zeste de savoir

BOUCLE POUR

1. **Un compteur initialise le début de la boucle.**
2. **Une condition de sortie (évaluation de la condition).**
3. **Incrémentation du compteur** (changement de la valeur du compteur).

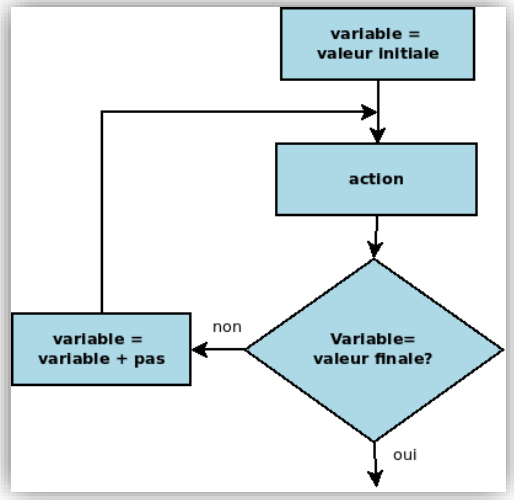
- Exemple :

DebutPour

Pour compteur de 1 à 10, pas de 1 :

*Affiche 5 * compteur*

FinPour



Source image zeste STI 2D

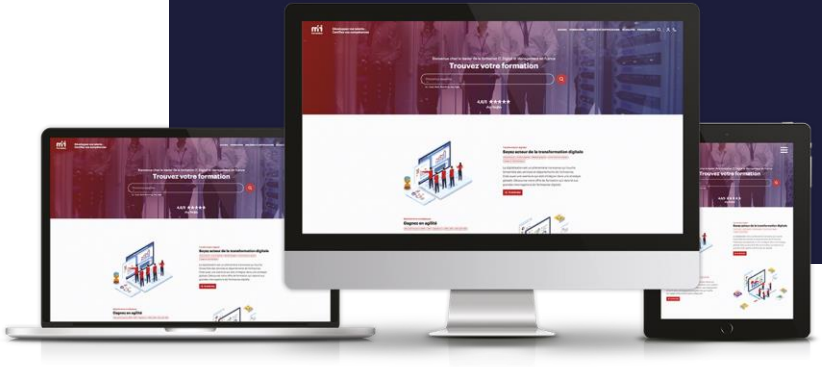
EXERCICES 5 et 6 : structures itératives

EXERCICES 5 et 6

2-exercices/exercice5.md

2-exercices/exercice6.md

IV. Fonction et tableaux



FONCTION

- *"programme dans le programme"*
- On utilise des fonctions pour **regrouper des instructions et les appeler sur demande** : chaque fois qu'on a besoin de ces instructions, il suffira d'appeler la fonction au lieu de répéter toutes les instructions.
- Pour accomplir ce rôle, le **cycle de vie d'une fonction comprend 2 phases** :

1. Une phase unique dans laquelle la fonction est **déclarée**
   On définit à ce stade toutes les instructions qui doivent être groupées pour obtenir le résultat souhaité.

2. Une phase qui peut être répétée une ou plusieurs fois dans laquelle la fonction est **exécutée**
   On demande à la fonction de mener à bien toutes les instructions dont elle se compose à un moment donnée dans la logique de notre application.

EXEMPLES FONCTIONS

```
Fonction somme (nb1: entier, nb2: entier) : entier
Variable resultat : entier
Début
    resultat <- nb1 + nb2
    retourne resultat
Fin
```

somme(10, 20) // retourne le résultat 30 que l'on peut stocker dans une variable du programme principal
somme(152,265)

```
Fonction bonjour () : vide
Début
    écrire("Bonjour ")
Fin
```

bonjour() // affiche Bonjour sur le périphérique de sortie
bonjour()

```
Fonction presentation(nom: chaine, prenom: chaine, age: entier) : vide
Début
    écrire("Je m'appelle ", prenom, " ", nom,",  j'ai ", age, " ans.")
Fin
```

presentation("Tshimini", "Glodie", 30) // affiche Je m'appelle Glodie Tshimini, j'ai 30 ans.

TABLEAUX

- **Les tableaux (array) sont des listes indexées d'éléments appartenant au même domaine de définition (type).**
- Un exemple tout simple de tableau, une liste de courses :
  - Orange
  - Fraise
  - Pomme
- Attention : l'**index**, indice, case ou rang d'un tableau **commence à 0.**
  - L'élément "Orange" est dans la première case du tableau à l'index 0.
  - L'élément "Fraise" est au rang 1.
  - L'élément "Pomme" est au rang 2.
  - A l'indice 3, il y a aucun élément.

TABLEAUX

Source image Pier import

- **La dernière valeur,** se trouve à l'indice *taille du tableau - 1*
- Trouvez les valeurs suivantes pour les tableaux ci-après
  - numbers[1,2,3]
  - strings["a","b", "c", "d", "e"]

1. numbers[1] vaut ?
2. strings[0] vaut ?
3. strings[4] vaut ?
4. numbers[3] vaut ?

EXERCICE 7 : fonction et tableaux

EXERCICE 7

2-exercices/exercice7.md

TABLEAUX MULTIDIMENSIONNELS

Source image Amazon

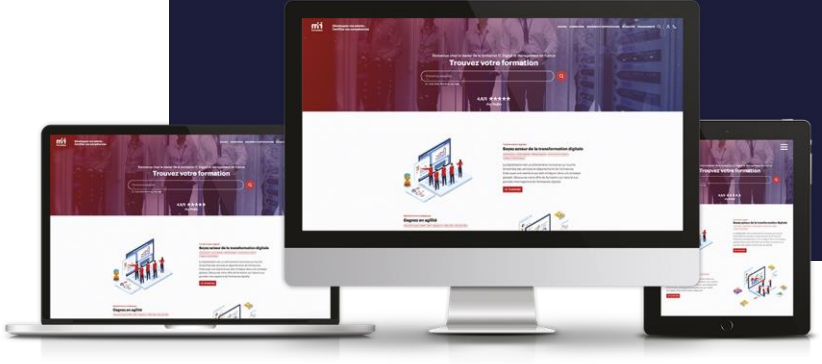
- **Un tableau dont les éléments peuvent être d'autres tableaux.**
- Il n'y a **pas de limite au niveau du nombre des dimensions**, cependant **au-delà de 2** dimensions, c'est **plus en plus difficile** pour les humains de **visualiser les informations**.
- On parle généralement de tableau à N dimensions, avec N le nombre des dimensions.

EXERCICE 8 : tableaux multidimensionnels

EXERCICE 8

2-exercices/exercice8.md

FIN



m2iformation.fr