

Algorithmique

CM1-1 : Initiation à la programmation

Mickaël Martin Nevot

V2.0.1



Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

Algorithmique

- I. Présentation du cours
- II. Initiation à la programmation
- III. Algorithmique
- IV. Bonus

Programme (informatique)

- Enchaînement d'**instructions**
- Écrit dans un **langage de programmation**
- Exécuté par des appareils informatiques
- Pour effectuer une tâche donnée
- **Cycle de vie** :
 - Construction
 - Distribution
 - Utilisation (installation ?)
 - Abandon (évolution rapide du marché, des modes, etc.)

Programmation informatique

- Conception :
 - Données à traiter
 - Méthode employée (algorithme)
 - Résultat (données en sortie)



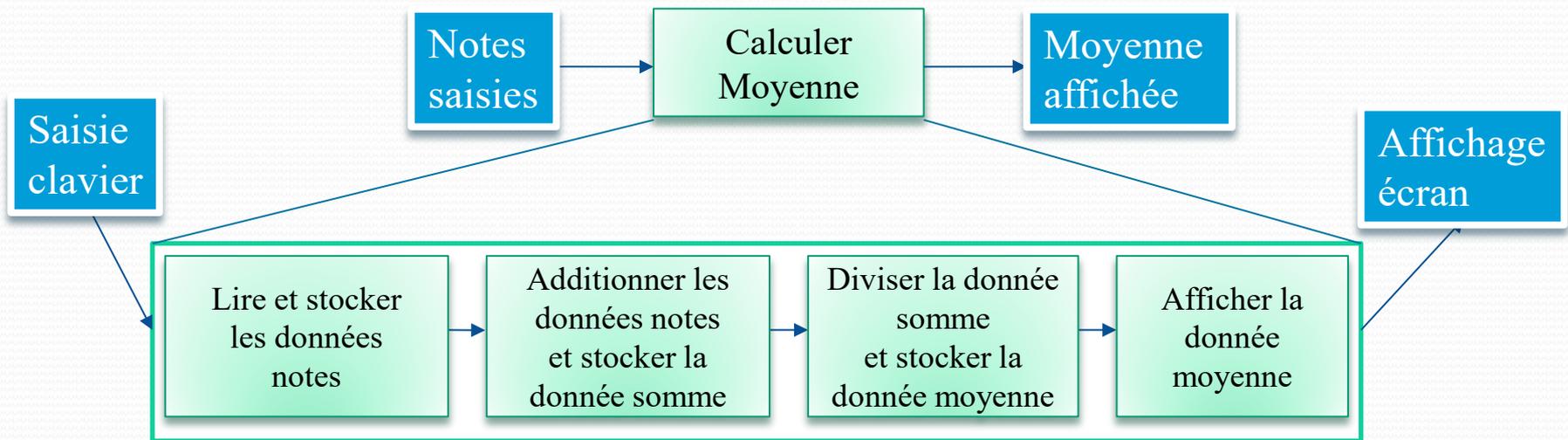
Programmation informatique

- Codage :
 - Contraintes d'architecture (choix du langage ?)
 - Langage de programmation
- Transformation du code source :
 - **Compilation** ou interprétation
- Test (important !)

```
End Sub  
Private Sub tbToolBar_ButtonClicked  
On Error Resume Next  
timTimer.Enabled = True  
Select Case Button.Key  
Case "Back"  
    brwWebBrowser.GoBack  
Case "Forward"  
    brwWebBrowser.GoForward  
Case "Refresh"  
    brwWebBrowser.Refresh  
Case "Home"  
    brwWebBrowser.Home  
End Case  
End Sub
```

Programmation informatique

- Conception
- Codage
- Transformation du code source
- Test



Historique

- **1842** : Ada Lovelace et Charles Babbage
- **1936** : machine de Turing / invention de la programmation
- **1950 - 1980** : cartes perforées
- **1954** : Fortran et système d'exploitation
- **1970** : programmation structurée
- **1980 – 1990** : programmation orientée objet (POO)

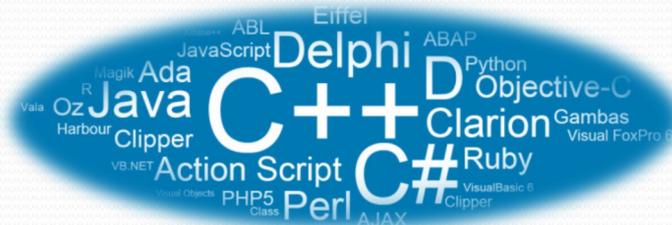
Environnement de programmation

- Éditeur
- Compilateur :
 - Vérifier la syntaxe
 - Traduire en langage machine



Programmation (paradigmes)

- Impérative : ← Tous les langages procéduraux et orientés objet
 - Séquences d'instructions qui modifie l'état du programme
- **Procédurale** : ← C, Pascal, Perl, BASIC, etc.
 - Programmation par appel procédural
- Fonctionnelle : ← Lisp, Haskell, Erlang, etc.
 - Appel de fonction sans changement d'état du programme
- **Orientée objet** : ← Java, C++, PHP, Ada, etc.
 - Définition et interaction d'entités avec comportement et état



Principes fondamentaux

- Données
- Opérateurs et instruction
- Affectation
- Condition
- Itération
- Fonction

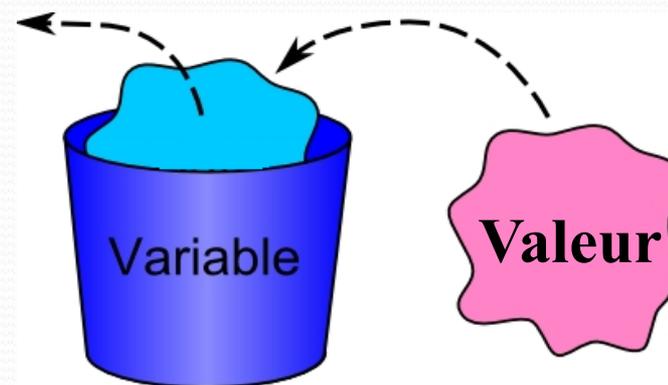


```
#include <stdio.h>

int main()
{
    printf("Hello World");
    return 42;
}
```

Données

- Variable :
 - **Nom** (symbole) : identificateur
 - **Valeur**
 - **Type** (parfois optionnel) :
 - Booléen
 - Entier
 - Nombre décimal
 - Chaîne de caractères
 - Composé (tableau, etc.)
 - Etc.
- **Constante** : variable dont la valeur ne varie pas
- **Valeurs littérales** : "hello!", 3, 2.75, true



Opérateurs et instruction

- Opérateur :

- Permet d'effectuer une opération (numérique, comparative, logique, d'affectation, etc.)

$a + 1$

+ : opérateur

- Expression :

- **Combinaisons** éventuellement « parenthésées » de **variables**, constantes et valeurs littérales avec des **opérateurs** produisant une **nouvelle valeur**

$((a + 1) / (b - 9))$

Expression

- Instruction :

- Étape d'un programme (composée d'expressions) dictant à l'ordinateur l'action à effectuer avant de passer à la suivante

$a \leftarrow ((a + 1) / (b - 9))$

Instruction

Affectation

- **Opération** qui permet, grâce à l'opérateur d'affectation, d'attribuer une valeur à une variable

- Exemples :

- `a ← 1`

Affectation d'une valeur littérale

- `b ← 2 + 3`

Affectation d'une expression

- `a ← b`

Affectation d'une variable

- `a ← true`

Affectation d'un booléen

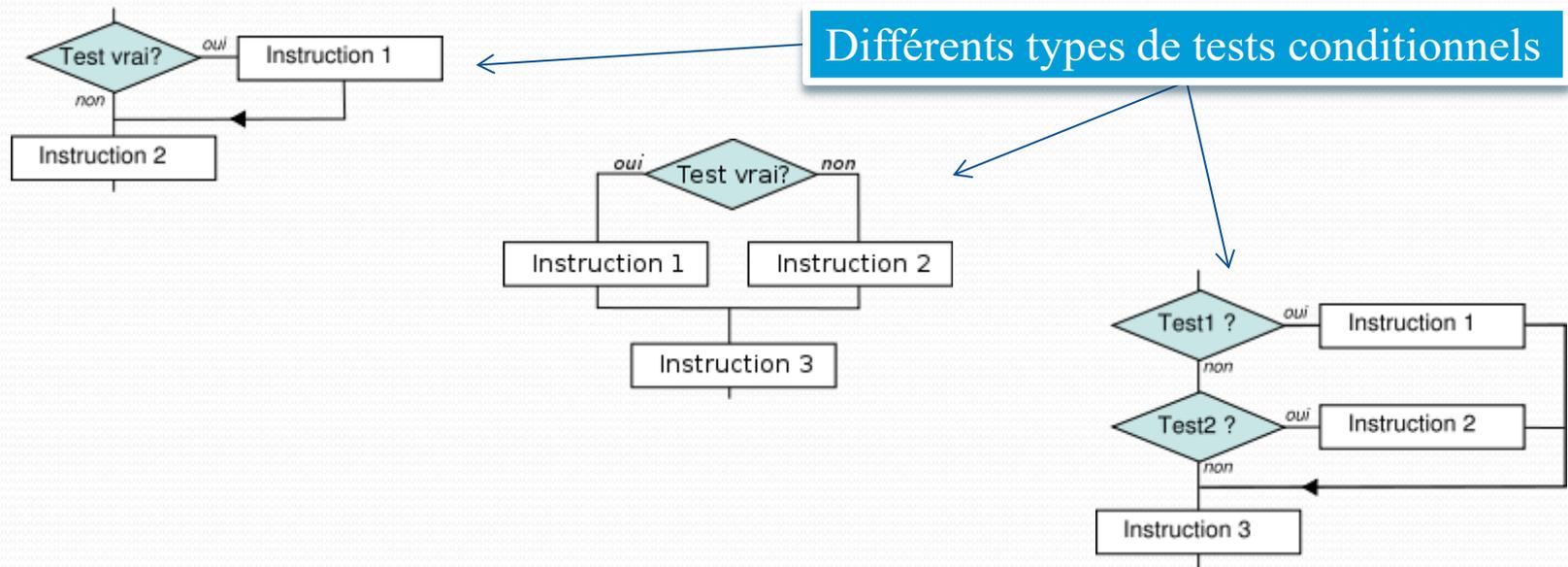
- `a ← "hello"`

Affectation d'une chaîne de caractères

Exemple d'opérateur d'affectation

Condition

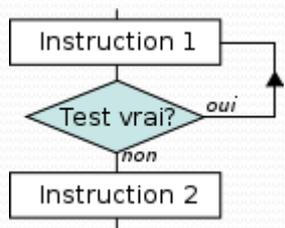
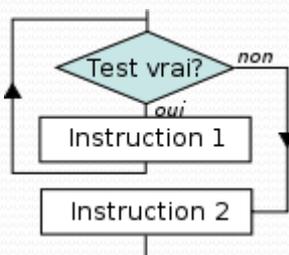
- Une condition est une **expression** qui ne peut être que **vrai** (*true*) ou **faux** (*false*)
- Utilisée pour les tests conditionnels



Également utilisée comme condition de continuation ou d'arrêt d'une itération

Itération

- Une itération est une **répétition**, potentiellement infinie, **d'instructions**
- Souvent couplée avec l'utilisation de **tableaux**



Différents types d'itérations

Valeur	45	154	58	78	31	5	74
Index	0	1	2	3	4	5	6

Les notions de condition et d'itération sont utilisées par les **structures de contrôle** (commandes qui contrôlent l'ordre d'exécution des instructions)

Fonction

- **Fonction** : séquence d'instructions indépendantes renvoyant une **valeur**
 - Procédure : fonction qui ne retourne aucun résultat
- Défini par :
 - **Nom** (symbole) : identificateur
 - **Paramètres** (liste de variables)
 - Un type de sa valeur de retour (parfois optionnel)
 - Un **bloc d'instructions** (contenant une instruction de retour)

```
sum(a, b)
  return a + b
```

Ici, a et b sont des **paramètres** (formels)

```
...
a ← sum(2, 3)
```

2 et 3 sont des **arguments** (ou paramètres effectifs)

Ici, a n'est pas un paramètre

Un appel de fonction est remplacé par sa valeur de retour

Crédits

Auteur

Mickaël Martin Nevot

mmartin.nevot@gmail.com



Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com

