Initiation à la programmation avec VisualBasic

PEIP Polytech' Marseille

Progression :

1. Notion d'algorithme
2. Premiers pas en VisualBasic (environnement VBA, Entrées-Sorties simplifiées)
3. Constantes, variables, expressions
4. Instruction alternative si-alors-sinon
5. Instruction répétitive tant-que
6. Variables indexées (tableaux)
7. Procédures et fonctions

I. Notion d'algorithme

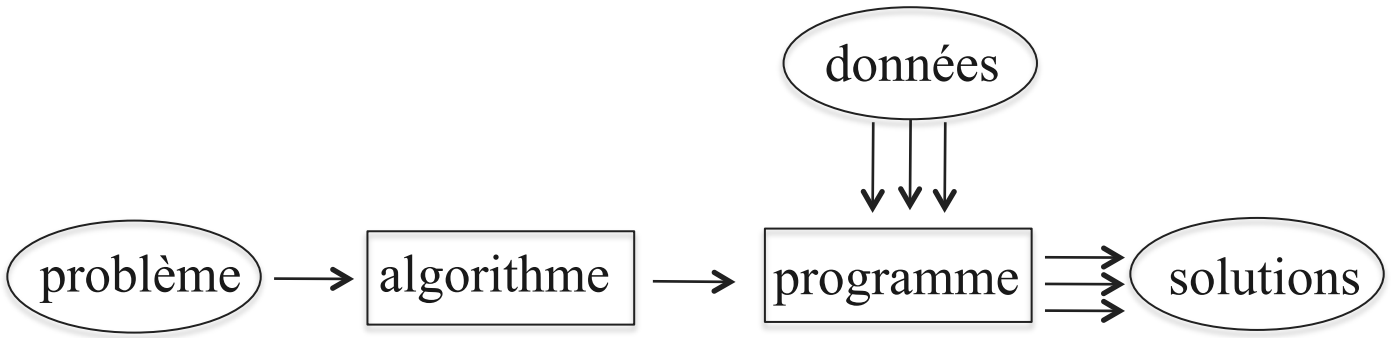
1. Qu'est-ce qu'un algorithme
2. Conception d'un algorithme : analyse hiérarchisée
3. Eléments de base pour écrire un algorithme

1. Qu'est-ce qu'un algorithme

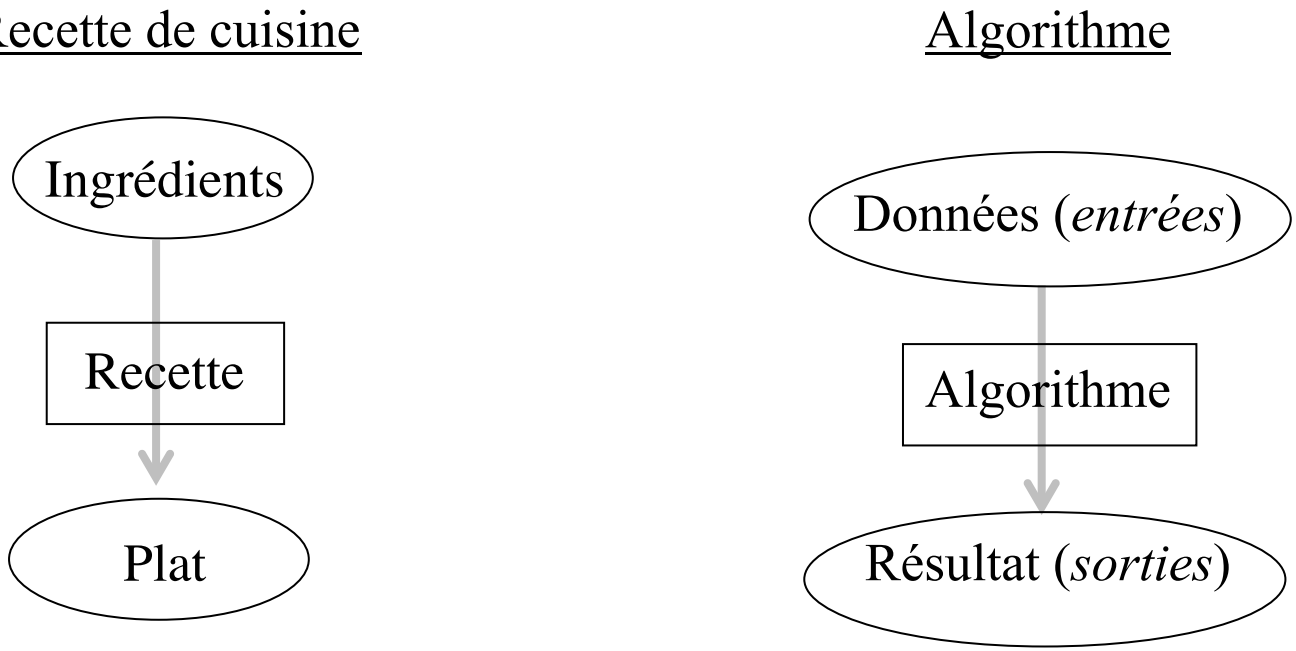
a) Définition

- Suite **finie** d'instructions **non ambiguës**
- dans un langage pseudo-naturel simple
- permettant de décrire une **méthode** pour répondre à un problème en un temps fini
- en s'affranchissant des aspects matériels de mise en œuvre (type de machine, langage de programmation utilisé, …)

L'étape suivante sera de traduire cet algorithme dans un langage de programmation « compréhensible » et exécutable par une machine pour un jeu de données connues.



b) Un parallèle culinaire



Dans le cas d'un programme informatique, les données manipulées peuvent être classées suivant 3 grandes catégories :

- les données numériques (nombres entiers, réels, …)
- les données alphanumériques (lettres, chiffres, caractères spéciaux, …)
- les données logiques (vrai, faux)

2. Conception d'un algorithme : analyse hiérarchisée

Premiers points à clairement définir avant de chercher à résoudre un problème :

- Quelles sont les données initiales (nature, domaine) ?
- Quel est le résultat souhaité ?

a) Définir un premier algorithme très général

- ensemble d'actions de haut niveau
- à réaliser de manière séquentielle

Recette des crêpes au nutella :

I. Faire une pâte à crêpe
II. Cuire une crêpe dans une poêle
III. Étaler du nutella sur la crêpe

b) Raffinements successifs : *Conception **structurée***

Recette des crêpes au nutella :

I. Faire une pâte à crêpe
   1. Mettre 350g de farine dans un bol
   2. Ajouter ½ litre de lait
   3. Mélanger
   4. Ajouter 3 œufs
   5. Mélanger

II. Cuire une crêpe dans une poêle

III. Étaler du nutella sur la crêpe

Remarques : certaines actions sont soumises à condition. Exemple Action "Mélanger".

- **Tant que** le mélange n'est pas homogène
  - tourner la pâte avec une fourchette
- **Fin tant-que**

c) De quoi a t-on besoin ?

Recette de cuisine		Algorithme
• *ingrédients (farine, beurre, ...)*	→	données
• *récipients (plat, bol, ...)*	→	variables
• *outils (couteaux, ...)*	→	opérateurs
• *instructions simples (mettre dans, ...)*	→	`affectation, ...`
• *instructions conditionnelles*	→	`si-alors-sinon`
• *instructions répétitives*	→	`tant-que`

3. Eléments de base pour écrire un algorithme

On dispose d'une certaine liberté de langage pour décrire un algorithme.
On utilisera toutefois les notions suivantes :

a) Déclaration de variables (contenants)
b) Lecture/écriture (communication d'information)
c) Expressions arithmétiques et logiques
d) Affectation
e) Instruction conditionnelle
f) Répétition d'une action

a) Déclarer une variable

Une variable : zone mémoire désignée par un nom pour stocker une valeur susceptible de changer dans le temps.

Déclarer des variables :

⇨ annoncer au début toutes les variables utilisées (contenants)

⇨ en précisant la nature du contenu (entier, réel, caractère, chaîne de caractères…)

ex : `soient a, b, c trois réels`

b) Lecture/écriture

⇨ pour représenter les échanges homme-machine (du point de vue de la machine)

- **`lire(`***`variable`***`)`** *homme => machine*
- **`écrire(`***`variable`***`)`** *homme <= machine*

c) Expressions

Elles sont construites à l'aide d'opérateurs

⇨ Expressions arithmétiques avec + , - , x , / , ( )

ex : b^2-4ac

⇨ Expressions logiques (seront vues plus tard) : résultat en `Vrai/Faux`

d) Affectation :

- Une variable ne peut contenir qu'une seule information à un instant t donné

  syntaxe : *variable* **<-** *expression*

  exemple : `d` **<-** b^2-4ac

Remarques :

- une variable à droite de **<-** : désigne le contenu
- la variable à gauche de **<-** : désigne le contenant

On distingue :

- l'affectation qui **initialise**

  ```
  x <- 1
  s <- x+1
  ```

- l'affectation qui **modifie**

  ```
  s <- s+x
  ```

Remarque : **lire**(*variable*) est une forme d'initialisation

Dans les prochains chapitres :

e) *Instruction conditionnelle*

f) *Répétition d'une action*

II. Premiers pas en Visual Basic

1. Présentation générale
2. Mon premier programme VBA
3. Les constantes
4. Les Variables
5. Les Expressions arithmétiques
6. Précisions sur le type Variant

1. Présentation générale

Le Basic (***B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode*) :

- ✓ « vieux » langage à l'échelle de l'informatique (1963 !)
- ✓ permet de traduire un algorithme en instructions interprétables par une machine
- ✓ facile d'accès pour traiter de petites applications
  (mais trop permissif pour développer de grosses applications…)
- ✓ a heureusement fortement évolué depuis sa création **:o)**
- ✓ plusieurs variantes (-> portabilité médiocre ! ! !)

Visual **B**asic pour **A**pplications (produit Microsoft) :

- ✓ intégré dans différents logiciels (Excel, …)
- ✓ largement diffusé
- ✓ version modernisée de Basic qui introduit notamment :
  - la **programmation objet** (non abordée dans ce cours)
  - la gestion d'**Interfaces Homme-Machine** (boites de dialogue, boutons, …)

2. Mon premier programme VBA

VBA est disponible dans l'environnement Excel

- Lancer Excel (accepter les macros !)
- Sous Mac : *Outils –> Macro –> VisualBasicEditor*
- Sous PC : *Développeur –> VisualBasic*

➔ lance l'environnement VBA

Pour écrire un programme VBA

- *Insertion –> Module*

➔ ouvre une fenêtre d'édition


```
Sub maProcedure()
   'ce programme vous dit bonjour
   Dim var As String
   var = InputBox("donnez votre nom :")
   MsgBox    "Hello " & var
End Sub
```

a) L'instruction Sub

- permet de déclarer une procédure avec un nom et éventuellement des arguments
- toutes les instructions entre **Sub** et **End Sub** constituent le **corps** de la procédure
- règle de rédaction : on indentera le corps de la procédure

Syntaxe simplifiée de l'instruction Sub :

```
Sub nom([listeArguments])
    [déclaration]
    [instruction]
End Sub
```

```
Sub maProcedure()
    'ce programme vous dit bonjour
    Dim var As String
    var = InputBox("donnez votre nom :")
    MsgBox  "Hello  " & var
End Sub
```

b) Identificateurs

Ce sont les noms que l'on va donner aux variables et aux procédures.

✓ Exemples dans le programme précédent :

`maProcedure` : nom de la procédure

`var` : nom d'une variable

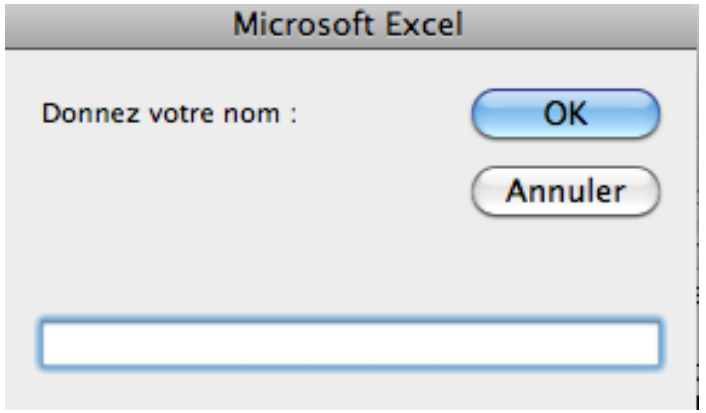
✓ Syntaxe : *lettre*[*lettre ou chiffre*]

✓ Exemple : x x1 compteur Peip1

✓ Remarques :

- les majuscules sont distinctives des minuscules
- les mots réservés de VBA commencent par une majuscule
- on choisira des identificateurs **mnémoniques**

c) Les entrées/sorties « simplifiées »



- Lecture d'une donnée (homme => machine)

```
variable = InputBox("donnez votre nom")
```

- Affichage d'un message (machine => homme)

```
MsgBox message
```

où *message* est soit :

- une chaîne de caractère (ex: "Hello ")
- une variable
- une expression arithmétique

- Plusieurs messages peuvent être **concaténés** en un seul avec l'opérateur **&**

  Exemple : `MsgBox "le double de 5 est " & 2*5`

3. Les constantes

Ce sont les valeurs explicites nécessaires au programme.

On distingue :

- les constantes numériques : `0` `1` `3.14` `314.E-2`
- les messages (ou chaînes de caractères) : `"Bonjour"` `"Donnez votre nom ?"`

4. Les Variables

On en déclare autant que nécessaire pour l'algorithme

a) Caractéristiques d'une variable

- porte un nom (identificateur)
- désigne un emplacement mémoire spécifique
- permet le stockage à court terme de valeurs
- son contenu peut changer dans le temps
- plusieurs **types** possibles (entier, réel, chaîne de caractères, …)

b) Déclaration des variables

- ✓ Réserve des emplacements mémoire
- ✓ Définit le codage de l'information (un entier n'est pas codé comme un réel)
- ✓ Non obligatoire sous VBA (déclaration implicite à la première occurrence)…
- ✓ mais fortement conseillé !!!

Il y a une douzaine de **types** pour déclarer des variables en VBA. Les plus courants :

- **`Integer`** pour stocker des valeurs entières entre [-32768, 32767]
- **`Long`** pour stocker des valeurs entières jusqu'à 10 chiffres
- **`Single`** pour stocker des valeurs décimales à 7 chiffres significatifs
- **`Double`** pour stocker des valeurs décimales à 15 chiffres significatifs
- **`String`** pour stocker des chaînes de caractères (message)
- **`Variant`** pour stocker indifféremment des réels, des entiers, des chaînes, …

Syntaxe de déclaration de variables :

```
Dim variable As type[, variable As type]
```

Exemples :

```
Dim nom As String, prenom As String
Dim age As Integer, salaire As Single
```

Conseils de bon programmeur :

- N'économisez pas sur le nombre de variables
- Chaque variable doit avoir un rôle bien défini et ne doit pas en changer
- Précisez ce rôle en commentaire pour les plus importantes
- Distinguez :
  - les variables recevant les données initiales
  - les variables nécessaires aux résultats intermédiaires
  - les variables qui recevront les résultats
- Donnez des noms qui "parlent" (identificateurs mnémoniques)

c) Affectation d'une variable

✓ Syntaxe : `variable = expression`

✓ Exécution de l'affectation :

1. évaluation de l'expression
2. recopie de la valeur dans la variable
3. en « écrasant » l'ancienne valeur contenue dans cette variable

✓ Rappel :

- l'affectation qui **initialise**

  ```
  x = 1
  s = x+1
  ```

- l'affectation qui **modifie**

  ```
  s = s+x
  ```
  (ce n'est pas une équation mathématique !!!)

Remarque : une variable doit toujours être initialisée avant sa première utilisation

5. Expressions arithmétiques

✓ Opérateurs arithmétiques :

- **+ - / * ( )**
- **^ Mod ** *(division entière)*

✓ Les opérateurs arithmétiques respectent les priorités admises en mathématiques

Remarque : attention aux types des variables (arrondi lors de l'affectation à un entier)

```
Sub test1()
  Dim a As Integer
  Dim b As Single
  a = 7
  b = a/3          ← b vaut 2.333333
  MsgBox "b = " & b
End Sub
```

```
Sub test2()
  Dim a As Integer
  Dim b As Integer
  a = 7
  b = a/3          ← b vaut 2
  MsgBox "b = " & b
  b = a/2          ← b vaut 4  (-4 si a =-7)
  MsgBox "b = " & b
End Sub
```

6. Précisions sur le type Variant

- ✓ Peut contenir indifféremment des nombres, des booléens, des chaînes ou des tableaux
- ✓ Assure une conversion automatique
- ✓ Accepte les opérations sur ces différents types
- ✓ Permet de construire un message de données hétérogènes pour affichage

```
Sub appliquerTVA()
  'calcul du prix TTC
  Dim prix As Variant
  prix = InputBox("donnez le prix HT :")
  prix = prix*1.2           'Opération arithmétique
  prix = prix & "€ TTC"     'Concaténation de messages
  MsgBox prix
End Sub
```

Conseil : limitez son utilisation au strict nécessaire