# Algorithmique et programmation : les bases (VBA)

# Corrigé

**Résumé**

Ce document décrit l'écriture dans le langage VBA des éléments vus en algorithmique.

## Table des matières

# Liste des exercices

# 1 Pourquoi définir notre langage algorithmique ?

# 2 Structure d'un algorithme

## 2.1 Exemple d'algorithme : calculer le périmètre d'un cercle

Un exemple d'algorithme/programme est donné ci-dessous. Il décrit comment obtenir le périmètre d'un cercle à partir de son diamètre. Cet exemple est volontairement très simple.

Listing 1 – Programme VBA pour calculer le périmètre d'un cercle

```
1  Attribute VB_Name = "exo00_calculer_perimetre"
2  '*****************************************************************
3  ' Auteur : Claude Monteil <monteil@ensat.fr>
4  ' Version : 1.0
5  ' Titre : Déterminer le perimetre d'un cercle à partir de son rayon.
6  '*****************************************************************
7
8  Sub perimetre_cercle()
9      Const PI As Double = 3.14159
10     Dim rayon As Double ' le rayon du cercle lu au clavier
11     Dim perimetre As Double ' le perimetre du cercle
12
13     EffacerEcran "Périmètre d'un cercle"
14    '1.Saisir le rayon
15     Afficher "Rayon = "
16     Saisir rayon
17    '2.Calculer le perimetre
18     perimetre = 2 * PI * rayon    ' par definition
19    '3.Afficher le perimetre
20     Afficher "Le perimetre est : " & perimetre
21 End Sub
```

## 2.2 Structure de l'algorithme

La structure d'un programme Visual BASIC est proche de celle d'un algorithme. Dans l'environnement Excel, une fois lancé l'éditeur Visual BASIC (menu "Outils / Macro / Visual BASIC Editor", ou raccourci ALT-F11), il faut créer un module (menu "Insertion / Module") dans lequel on écrit le programme. On peut sauvegarder régulièrement le programme (inclus dans le fichier Excel) par le menu "Fichier / Enregistrer" (ou icône correspondante, ou raccourci Ctrl-S, le S étant l'initiale de Save ou Sauvegarder). On peut aussi sauvegarder un module de manière séparée en l'exportant (menu "Fichier / Exporter un fichier") : cela crée un fichier d'extension .bas (pour Basic), qui peut ensuite être réimporté dans d'autres classeurs Excel, ou d'autres environnements utilisant Visual Basic. Le module commence classiquement par un cartouche faisant apparaître le nom des auteurs du programme, la version ou la date de réalisation et l'objectif du programme. Ces éléments sont mis dans des commentaires (cf. plus bas) et seront donc ignorés par l'interpréteur. Les déclarations et instructions sont regroupées entre **Sub** `NomDuProgramme` et

**End Sub**. La constante PI est définie par le mot-clé **Const**. Il correspond donc à une définition. Notons qu'en Visual BASIC les constantes sont typées. Les instructions sont les mêmes que celles présentées en langage algorithmique même si elles ont une forme un peu différente.
**Remarque :** Dans le cadre de l'application de cet enseignement de l'algorithmique en Visual BASIC, nous utiliserons comme première instruction de tous les exemples l'instruction `EffacerEcran "Nom␣de␣l'exemple"` qui a pour effet d'effacer la feuille de calcul courante (utilisée comme écran d'affichage) et d'afficher en tête de la feuille le message mis entre guillemets (à adapter à chaque exemple, bien évidemment). Ceci permettra que les affichages réalisés lors de l'exécution-test d'un programme en cours de mise au point ne soient pas mélangés avec les affichages réalisés lors des précédentes exécutions.

## 2.3 Identificateurs

Un identificateur est un mot de la forme : une lettre (y compris le souligné) suivie d'un nombre quelconque de lettres et de chiffres ou du caractères souligné (même touche de clavier que le 8).
**Remarque :** Bien qu'il soit possible en VBA d'utiliser des lettres accentuées, cette pratique est à bannir pour des raisons de compatibilité !

## 2.4 Commentaires

Un commentaire commence par ' (caractère "quote" situé sur la touche 4) et se termine à la fin de la ligne.

# 3 Variables

## 3.1 Qu'est ce qu'une variable ?

## 3.2 Définition d'une variable

En VBA, on utilise le mot-clé **Dim** suivi par le nom de la variable, le mot-clé As puis le type de la variable.

```
1 Dim prix_unitaire As Double    ' prix unitaire d'un article (en euros)
2 Dim quantite As Integer    ' quantité d'articles commandés
3 Dim nom As String    ' nom de l'article
```

Les types et leur signification seront présentés dans la suite du cours. On peut déclarer plusieurs variables sur une même ligne à condition de séparer chaque définition (nom de variable puis As puis le type) par une virgule. Ce raccourci est usuellement utilisé si le même commentaire s'applique à toutes les variables.

```
1 Dim a As Integer, b As Integer, c As Integer    ' trois entiers
```

**Attention :** Le type doit impérativement être répété pour chaque variable (il serait incorrect d'écrire `Dim a, b, c As Integer`, les 2 premières variable a et b se voyant affecter un type par défaut qui n'est pas `Integer`).

# 4 Types fondamentaux

Les opérateurs de comparaison entre les types fondamentaux se notent : <, >, <=, >=, = et <>.

## 4.1 Les entiers

Le type entier se note `Integer`. La division entière s'effectue avec l'opérateur \, et le reste de la division entière s'obtient avec l'opérateur `Mod`.

```
20 \ 3   ' 6 (le quotient de la division entière de 20 par 3)
20 mod 3   ' 2 (le reste de la division entière de 20 par 3)
```

**Remarque :** Les débordements de capacité sur les opérations entières provoquent une erreur à l'exécution. Ainsi, le type `Integer` est limité à des valeurs entre -32678 et +32767. Pour des capacités plus grandes, on peut utiliser le type `Long`, permettant des valeurs entre + ou - 2 milliards.

## 4.2 Les réels

Il existe deux types réels, les réels en simple précision (`Single`), autorisant 7 chiffres significatifs en virgule flottante, et les réels en double précision (`Double`), autorisant 15 chiffres significatifs. La valeur absolue s'obtient par la fonction prédéfinie `Abs`. La partie entière d'un réel s'obtient par la fonction `Int` ou la fonction `Fix` : ces 2 fonctions ont un comportement identique pour les réels positifs (par exemple `Int(3.14)` correspond à 3) mais se différencient pour les réels négatifs (`Int(-3.14)` correspond à -4, tandis que `Fix(-3.14)` correspond à -3). On peut également arrondir un réel en un entier avec la fonction `Round` : `Round(3.14)` vaut 3, et `Round(3.5)` vaut 4.

## 4.3 Les booléens

Le type booléen est `Boolean`. Les deux valeurs que peuvent prendre des variables booléennes se notent `True` et `False`. Les opérateurs logiques se notent `And` pour `Et`, `Or` pour `Ou` et `Not` pour `Non`.

**Remarque :** Les expressions booléennes sont toujours évaluées totalement, même si un résultat partiel permet de connaître le résultat total. Par exemple, `True Or expression` calculera toujours l'expression, même si on sait que le résultat sera forcément `True` quelle que soit la valeur de l'expression.

### 4.4 Les caractères

Le type caractère n'existe pas en tant que tel en VBA. Ce n'est qu'un cas particulier des chaînes de caractères (cf. paragraphe suivant). Les fonctions algorithmiques `Chr` et `Ord` se notent respectivement **`Chr`**`(i)` et **`Asc`**`(c)`.

```
1 c = "A"    ' la valeur de c est "A"
2 i = Asc(c)    ' la valeur de i est 65, code ASCII de "A"
```

### 4.5 Les chaînes de caractères

Une chaîne se déclare avec le mot-clé String. Une chaîne peut contenir jusqu'à 64 milliers de caractères. Il n'est pas utile de spécifier de taille maximale dans la déclaration d'une variable chaîne.

```
1 Dim chaine As String
2 chaine = "bonjour"
```

## 5 Constantes

Les constantes sont définies en utilisant le mot-clé **`Const`** à la place de **`Dim`**, en faisant suivre le type de la variable par sa valeur derrière le symbole = :

```
1 Cons PI As Single = 3.1415    ' Valeur de PI
2 Const MAJORITE As Integer = 18    ' Age correspondant à la majorité
3 Const TVA As Single = 19.6    ' Taux de TVA en %
4 Const CAPACITE as Integer = 160    ' Nombre maximum d'étudiants dans une promotion
5 Const INTITULE As String = "Algorithmique et programmation"    ' par exemple
```

## 6 Expressions

## 7 Instructions d'entrée/sorties

### 7.1 Opération d'entrée

En standard, le VBA utilise l'instruction **`InputBox`** qui permet d'effectuer une saisie dans une boîte-message fugitive. Dans le cadre de ce module d'algorithmique, nous utiliserons l'instruction `Saisir` qui permet de saisir au clavier la valeur d'une variable et de garder trace dans la feuille de calcul courante de ce qui a été saisi :

```
1 Saisir Variable1
```

**Attention :** Il ne faut pas utiliser de parenthèses autour de la variable, contrairement à l'instruction algorithmique **`Lire`**`(Variable1)`. Une variante de l'instruction de saisie permet de proposer à l'utilisateur une valeur par défaut qui sera affectée à la variable s'il se contente de valider la saisie sans entrer de valeur particulière :

```
1 Saisir Duree, 5   ' la valeur 5 est affichée en présélection dans la boîte-message :
2     ' si l'usager valide directement par la touche 'Retour-chariot', la valeur 5 sera a
```

## 7.2 Opération de sortie

En standard, le VBA utilise l'instruction **`MsgBox`** qui permet d'afficher des informations dans une boîte-message fugitive. Dans le cadre de ce module d'algorithmique, nous utiliserons l'instruction `Afficher` qui permet d'afficher les informations dans la feuille de calcul courante.

```
1 Afficher "La durée vaut " & Duree
```

Noter l'utilisation de l'opérateur de concaténation `&` pour composer un message unique rassemblant plusieurs informations mises bout à bout (ici, une chaîne littérale et le contenu d'une variable).

**Attention :** Tout comme pour l'instruction de saisie, il ne faut pas utiliser de parenthèses autour des informations à afficher, contrairement à l'instruction algorithmique **`Ecrire`**`("La durée vaut", Duree)`.
Il est également possible d'utiliser la forme suivante utilisant la virgule comme séparateur :

```
1 Afficher "La durée vaut ", Duree
```

Dans ce cas, la valeur de la variable `Duree` sera placée dans la colonne suivante de la feuille de calcul. Cette forme sera utile si on veut afficher des informations tabulées. Noter que, si le premier élément `"La durée vaut "` ne rentre pas dans la première colonne, la partie qui dépasse sera tronquée à l'affichage et n'apparaîtra donc pas. L'instruction `Afficher ""` utilisée avec un chaîne vide (symbole " redoublé, et non pas 4 fois la quote simple) permet de sauter une ligne.

**Exercice 1 : Cube d'un réel**
Écrire un programme qui affiche le cube d'un nombre réel saisi au clavier.
**Solution :**

```
1  R0 : Afficher le cube d'un nombre réel
2
3  Tests :
4      0 -> 0
5      1 -> 1
6      2 -> 8
7      -2 -> -8
8      1.1 -> 1,331
9
10 R1 : Raffinage De « Afficher le cube d'un nombre réel »
11   | Saisir un nombre réel       x: out Réel
12   | Afficher le cube de x       x: in Réel
13
14 R2 : Raffinage De « Afficher le cube de x »
15   | Écrire(x * x * x)
```

# 8 Affectation

L'affectation se note avec un signe =.

**Attention :** Il ne faut pas confondre l'affectation et le test d'égalité, même s'ils utilisent le même opérateur =. Le contexte permet de les différencier sans ambiguïté : un test d'égalité apparaît dans une condition placée derrière **`If`**, **`While`** ou **`Until`** (cf. chapitre suivant sur les structures de contrôle).

```
i = 10    ' affectation
If i=10 Then    ' test d'égalité
```

**Exercice 2 : Permuter deux caractères**
Écrire un programme qui permute la valeur de deux variables `c1` et `c2` de type caractère.
**Solution :** Le principe est d'utiliser une variable intermédiaire (tout comme on utilise un récipient intermédiaire si l'on veut échanger le contenu de deux bouteilles).

```
Attribute VB_Name = "exo02_permuter"
'*********************************************************************
'* Auteur : Claude Monteil <monteil@ensat.fr>
'* Version : 1.0
'* Objectif : permuter deux caracteres
'*********************************************************************

Sub permuter_caracteres()
    Dim c1 As String, c2 As String  ' les deux caracteres a permuter
    Dim tmp As String ' variable intermediaire

    EffacerEcran "Permitation de caractères"
    '1.initialiser c1 et c2
    c1 = "A"
    c2 = "Z"
    Afficher "Avant : c1 = " & c1 & " et c2 = " & c2
    '2.permuter c1 et c2
    tmp = c1
    c1 = c2
    c2 = tmp
    '3.afficher pour verifier
    Afficher "Après : c1 = " & c1 & " et c2 = " & c2
End Sub
```

**Exercice 3 : Cube d'un réel (avec une variable)**
Reprenons l'exercice 1.

**3.1** Utiliser une variable intermédiaire pour le résoudre.
**Solution :** On reprend le même R0 et les mêmes tests. En fait, seule la manière de résoudre le problème change.

```
R1 : Raffinage De « Afficher le cube d'un nombre réel »
  | Saisir un nombre réel       x: out Réel
  | Calculer le cube de x       x: in Réel ; cube: out Réel
  | Afficher le cube

R2 : Raffinage De « Afficher le cube de x »
  | cube <- x * x * x
```

```
1  Attribute VB_Name = "exo03_cube_variable"
2  '*********************************************************************
3  '* Auteur : Claude Monteil <monteil@ensat.fr>
4  '* Version : 1.0
5  '* Objectif : afficher le cube d'un nombre reel (usage d'une variable)
6  '*********************************************************************
7
8  Sub cube_var()
9       Dim x As Double      ' un nombre saisi par l'utilisateur
10      Dim cube As Double   ' le cube de x
11
12      EffacerEcran "Cube d'un nombre (avec variable)"
13     '1.Saisir un nombre reel
14      Afficher "Nombre = "
15      Saisir x
16     '2.Calculer le cube de x
17      cube = x * x * x
18     '3.Afficher le cube de x
19      Afficher "Son cube est : " & cube
20 End Sub
```

**3.2** Quel est l'intérêt d'utiliser une telle variable ?

**Solution :** L'intérêt d'utiliser une variable intermédiaire est d'améliorer la lisibilité du programme car elle permet de mettre un nom sur une donnée manipulée. Ici on nomme `cube` la donnée `x * x * x`.

De plus, ceci nous a permis, au niveau du raffinage, de découpler le calcul du cube de son affichage. Il est toujours souhaitable de séparer calcul des opérations d'entrées/sorties car l'interface avec l'utilisateur est la partie d'une application qui a le plus de risque d'évoluer.

**3.3** Exécuter à la main l'algorithme ainsi écrit.

**Solution :** À faire soi-même !

# 9 Structures de contrôle

## 9.1 Enchaînement séquentiel

La séquence s'exprime comme en algorithmique.

## 9.2 Instructions conditionnelles

### 9.2.1 Conditionnelle `Si ... Alors ... FinSi`

Deux variantes sont utilisables selon que la ou les instructions conditionnées s'écrivent de manière courte ou longue. Forme courte :

```
1  If condition Then une ou plusieurs instructions tenant sur cette unique ligne
```

S'il y a plusieurs instructions courtes qui tiennent sur la ligne, il faut utiliser `:` pour séparer chaque instruction. Forme longue (la plus générale) :

```
If condition Then
    instruction
    ...
    instruction
End If
```

Bien noter que, pour la forme courte, la fin de l'unique ligne tient lieu de **End If** implicite. Il est tout à fait possible d'utiliser des parenthèses autour de la condition si on préfère mieux la mettre en valeur.

**Exercice 4 : Une valeur entière est-elle paire ?**
Écrire un algorithme qui lit une valeur entière au clavier et affiche « paire » si elle est paire.
**Solution :**

```
R0 : Afficher « paire » si une valeur entière saisie au clavier est paire

tests :
    2 -> paire
    5 -> -----
    0 -> paire

R1 : Raffinage De « Afficher ... »
  | Saisir la valeur entière n
  | Afficher le verdict de parité

R2 : Raffinage De « Afficher le verdict de parité »
  | Si n est paire Alors
  |  | Écrire("paire")
  | FinSi

R3 : Raffinage De « n est paire »
  | Résultat <- n Mod 2 = 0
```

Dans le raffinage précédent un point est à noter. Il s'agit du raffinage R2 qui décompose « Afficher le verdict de parité ». Nous n'avons pas directement mis la formule « `n Mod 2 = 0` ». L'intérêt est que la formulation « n est paire » est plus facile à comprendre. Avec la formule, il faut d'abord comprendre la formule, puis en déduire sa signification. « n est paire » nous indique ce qui nous intéresse comme information (facile à lire et comprendre) et son raffinage (R3) explique comment on détermine si n est paire. Le lecteur peut alors vérifier la formule en sachant ce qu'elle est sensée représenter.

Raffiner est quelque chose de compliquer car on a souvent tendance à descendre trop vite dans les détails de la solution sans s'arrêter sur les étapes intermédiaires du raffinage alors que ce sont elles qui permettent d'expliquer et de donner du sens à la solution.

Dans cet exercice, vous vous êtes peut-être posé la question : « mais comment sait-on que n est paire ». Si vous avez trouvé la solution vous avez peut-êre donnée directement la formule alors que le point clé est la question. Il faut la conserver dans l'expression de votre algorithme ou programme, donc en faire une étape du raffinage.

Si vous arrivez sur une étape que vous avez du mal à décrire, ce sera toujours une indication d'une étape qui doit apparaître dans le raffinage. Cependant, même pour quelque chose de simple,

que vous savez faire directement, il faut être capable de donner les étapes intermédiaires qui conduisent vers et expliquent la solution proposée. Ceci fait partie de l'activité de construction d'un programme ou algorithme.

**Remarque :** Il est généralement conseillé d'éviter de mélanger traitement et entrées/sorties. C'est pourtant ce qui a été fait ci-dessus. On aurait pu écrire le premier niveau de raffinage différemment en faisant.

```
R1 : Raffinage De « Afficher ... »
  | Saisir la valeur entière          n: out Entier
  | Déterminer la parité de n        n: in ; paire: out Booléen
  | Afficher le verdict de parité    paire: in Booléen

R2 : Raffinage De « Déterminer la parité de n »
  | parité <- (n Mod 2) = 0

R2 : Raffinage De « Afficher le verdict de parité »
  | Si paire Alors
  |  | Écrire("paire")
  | FinSi
```

On constate ici que la variable intermédiaire « paire » permet d'avoir un programme plus lisible car on a donné un nom à la quantité `(n Mod 2) = 0`.

```
Attribute VB_Name = "exo04_parite"
'*********************************************************************
'* Auteur : Claude Monteil <monteil@ensat.fr>
'* Version : 1.0
'* Objectif : Afficher " paire " si une valeur entière est paire.
'*********************************************************************

Sub tester_parite()
     Dim n As Integer  ' valeur saisie au clavier

     EffacerEcran "Parité d'un nombre"
     '1.Saisir la valeur entiere n
     Afficher "Valeur = "
     Saisir n
     '2.Afficher le verdict de parite
     If (n Mod 2 = 0) Then ' n est paire
          Afficher "paire"
     End If
End Sub
```

### 9.2.2 Conditionnelle `Si ... Alors ... Sinon ... FinSi`

```
If condition Then
    instruction
    ...
Else
    instruction
```

```
    ...
End If
```

**Exercice 5 : Maximum de deux valeurs réelles**
Étant données deux valeurs réelles lues au clavier, afficher à l'écran la plus grande des deux.
**Solution :**

```
R0 : Afficher le plus grand de deux réels saisis au clavier

tests :
    1 et 2 -> 2
    2 et 1 -> 1
    3 et 3 -> 3

R1 : Raffinage De « Afficher le plus grand de deux réels ... »
  | Saisir les deux réels       x1, x2 : out Réel
  | Déterminer le maximum       x1, x2 : in ; max : out Réel
  | Afficher le maximum

R2 : Raffinage De « Déterminer le maximum »
  | Si x1 > x2 Alors
  |  | max <- x1
  | Sinon
  |  | max <- x2
  | FinSi
```

```
Attribute VB_Name = "exo05_max"
'*************************************************************************
'* Auteur : Claude Monteil <monteil@ensat.fr>
'* Version : 1.0
'* Objectif : Determiner le max de deux valeurs reelles
'*************************************************************************

Sub calculer_max()
    Dim x1 As Double, x2 As Double ' les deux reels saisis au clavier
    Dim max As Double              ' le plus grand de x1 et x2

    EffacerEcran "Maximum de 2 nombres"
    '1.Saisir les deux reels
    Afficher "Saisir un premier nombre reel : "
    Saisir x1
    Afficher "Saisir un second nombre reel : "
    Saisir x2
    '2.Déterminer le maximum
    If (x1 > x2) Then
        max = x1
    Else
        max = x2
    End If
    '3.Afficher le maximum
    Afficher "max(" & x1 & "," & x2 & ") =" & max
End Sub
```