Chapitre 2
Variables - Constantes - Types de données

Durée : 1 heure 10

Mots-clés

déclaration, portée, durée de vie, type, affectation, argument, type de données VBA, type de données utilisateur, membre, conversion, variable de type objet

Objectifs

Maîtriser l'emploi des variables et des constantes pour l'écriture des procédures et la réalisation de programmes.

Prérequis

Pour valider les prérequis nécessaires, avant d'aborder le TP, répondez aux questions ci-après (certaines questions peuvent nécessiter plusieurs réponses) :

1. La déclaration des variables dans VBA :
   - **a.** est réalisée avec l'instruction `Option Explicit` dans la partie déclaration du module.
   - **b.** peut être étendue à l'ensemble des modules.
   - **c.** est obligatoire.
   - **d.** doit être suivie obligatoirement du type de données.
2. Un nom de variable :
   - **a.** peut contenir un espace.
   - **b.** doit commencer par une lettre.
   - **c.** doit être unique au sein d'une même portée.
3. Une variable est accessible uniquement par les procédures de son module quand elle est déclarée avec le mot-clé :
   - **a.** `Dim`, au sein de la procédure.
   - **b.** `Dim`, dans la partie déclaration du module.
   - **c.** `Private`.
   - **d.** `Public`.

4. Lorsqu'une variable perd sa portée :
   - **a.** elle devient accessible à toutes les procédures.
   - **b** elle n'a plus de valeur sauf si elle est déclarée statique.
   - **c** elle perd son type.
5. Les déclarations de variables suivantes sont correctes dans la partie déclaration du module :
   - **a.** `Public varTest1`
   - **b.** `Private dblTest2 As Double`
   - **c.** `dblTest3 As Double`
6. Dans l'instruction suivante `Dim strMot, strPhrase as String`, la variable `strMot` est de type :
   - **a.** `String`
   - **b.** `Variant`
   - **c.** `inconnu`
7. Pour déclarer la constante publique `Pi`, on écrit :
   - **a.** `Public Const Pi As Double = 3.1415926`
   - **b.** `Const Pi Public = 3.1415926 As Double`
   - **c.** `Const Pi = 3.1415926`
8. Le type de données par défaut des variables est :
   - **a.** `Byte`
   - **b.** `String`
   - **c.** `Variant`
9. La déclaration du type s'effectue avec le mot-clé :
   - **a.** `To`
   - **b.** `As`
   - **c.** `VarType`

10. Une variable peut être de type :
- **a.** tableau
- **b.** objet
- **c.** état Access

11. Une variable de type objet :
- **a.** contient une référence à l'objet.
- **b.** contient l'objet lui-même.
- **c.** contient la valeur de l'objet.

12. Le type de données numériques le plus précis est :
- **a.** `Single`
- **b.** `Currency`
- **c.** `Double`

13. La conversion des données est possible :
- **a.** oui
- **b.** non
- **c.** uniquement pour les chaînes de caractères.

14. La création de ses propres types de données est possible :
- **a.** oui
- **b.** non
- **c.** seulement pour les tableaux ou collections.

15. Pour connaître le type de données d'une variable, on utilise le mot réservé :
- **a.** `Is`
- **b.** `Cvar`
- **c.** `VarType`

Corrigé p. 175

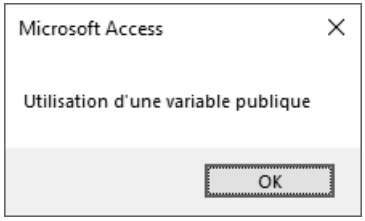
Énoncé 2.1 Déclarer et utiliser une variable

Durée estimative : 20 minutes

2.1.1 : Déclarer et utiliser une variable de niveau projet

Dans le module **Chapitre_02**, déclarez une variable publique nommée **strMessagePublic** de type `String`.

Créez une procédure nommée **Message** qui affiche le message suivant :

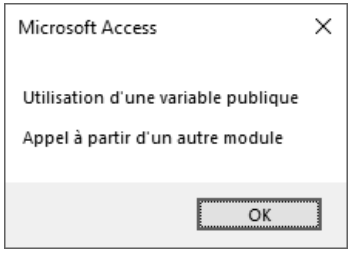


Indice

Pour afficher la boîte de dialogue, utilisez la fonction `MsgBox`.

2.1.2 : Utiliser une variable de niveau projet

Dans le module nommé **Module2**, créez une procédure nommée **AppelExtérieur** qui utilise la variable **strMessagePublic** du module **Chapitre_02** pour afficher le message suivant :



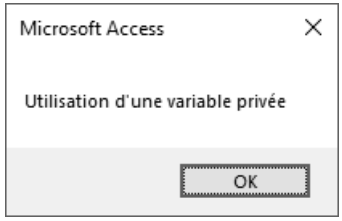
Indices

Pour forcer un saut de ligne : `vbCrLf`.

Utilisez l'opérateur de concaténation & pour composer le message.

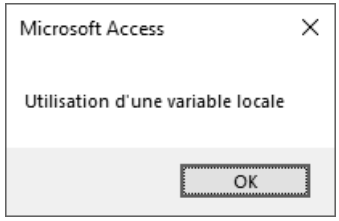
2.1.3 : Déclarer et utiliser une variable de niveau module

Dans le module **Chapitre_02**, déclarez une variable privée nommée **strMessagePrivé** de type `String`.
Créez une procédure nommée **MessageInterne** qui affiche le message suivant :



2.1.4 : Déclarer et utiliser une variable de niveau procédure

Dans le module **Chapitre_02**, créez une procédure nommée **MessageLocal**. Celle-ci utilise une variable locale nommée **strMessageLocal** de type `String`. Résultat :

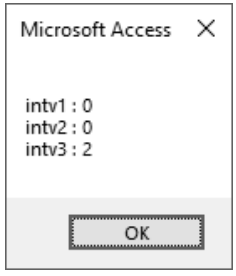


2.1.5 : Déclarer et utiliser plusieurs variables de même type

Dans le module **Chapitre_02**, créez une procédure nommée **VariablesLigne** dont voici le code :

```
Sub VariablesLigne()
 Dim intv1, intv2, intv3 As Integer
 MsgBox ("intv1 : " & VarType(intv1) & vbCrLf & "intv2 : " _
 & VarType(intv2) & vbCrLf & "intv3 : " & VarType(intv3))
End Sub
```

Résultat :



Corrigez la déclaration de sorte que les trois variables soient effectivement de type entier.

Corrigé p. 176

Énoncé 2.2 Déclarer et utiliser une constante

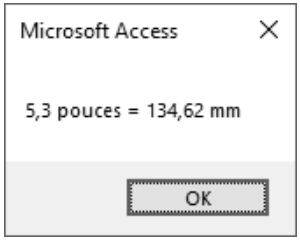
Durée estimative : 10 minutes

2.2.1 : Conversion

Dans le module **Chapitre_02**, déclarez une constante publique nommée **dblPouce** de type `Double`. Affectez-lui la valeur 25,4.

Créez une procédure nommée **ConversionPouceEnmm** qui convertit les pouces en millimètres.

Exemple :



Indice

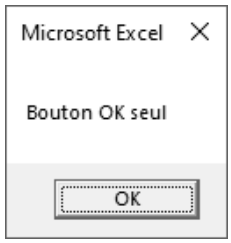
Faites attention au séparateur décimal.

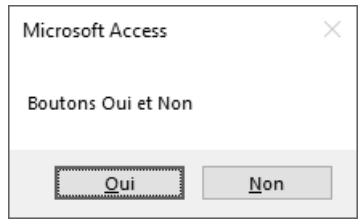
2.2.2 : Utiliser des constantes VBA pour le choix des boutons des boîtes de messages

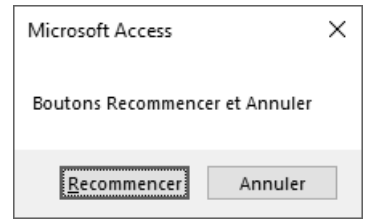
VBA possède de nombreuses constantes internes qui simplifient la programmation. Avec elles, il est possible par exemple de choisir facilement les boutons de la boîte de dialogue standard.

Affichage seulement du bouton **OK** :

```
Sub ConstanteVBA()
 Dim intRésultat As Integer
 intRésultat = MsgBox("Bouton OK seul", vbOKOnly)
End Sub
```







Complétez le code de la procédure **ConstanteVBA** pour obtenir l'affichage des deux autres boîtes de dialogue ci-dessus : la boîte avec les boutons **Oui** et **Non** et la boîte avec les boutons **Recommencer** et **Annuler**.

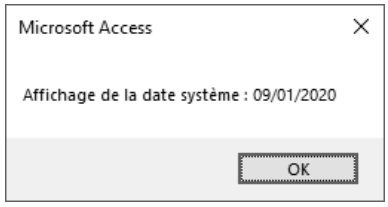
Indice

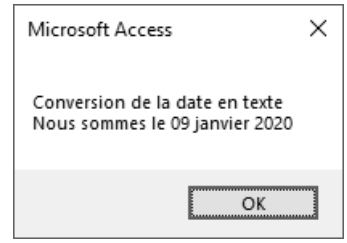
La liste complète des constantes VBA disponibles pour la gestion des boîtes de dialogue MsgBox est accessible par le lien suivant :
https://docs.microsoft.com/fr-fr/office/vba/language/reference/user-interface-help/msgbox-constants

Corrigé p. 177

Énoncé 2.3 Utiliser la date système

Durée estimative : 10 minutes
Créez la procédure **ConversionDate** qui effectue une conversion de la date système en type `String`. Procédez aux deux affichages suivants.
Exemple :





Indices

Utilisez la fonction VBA `Date` *pour avoir la date système.*

Utilisez la fonction VBA `CStr` *pour la conversion.*

Syntaxe : `CStr(expression)`

Utilisez la fonction VBA `Format` *pour la présentation de la date après conversion en texte.*

Syntaxe simplifiée : `Format(expression)`

Corrigé p. 178

Énoncé 2.4 Créer un type de données "PoissonTropical" défini par l'utilisateur

Durée estimative : 5 minutes
Créez dans le module **Chapitre_02** un type de données public nommé **PoissonTropical**. Les membres de ce nouveau type sont : nomPoisson, couleur, poids et lieu d'habitat.

Indice

```
Public . . . PoissonTropical
 NomPoisson As . . .
 Couleur . . .
 Poids . . .
 Lieu . . .
End Type
```

Corrigé p. 179

Énoncé 2.5 Utiliser le type "PoissonTropical"

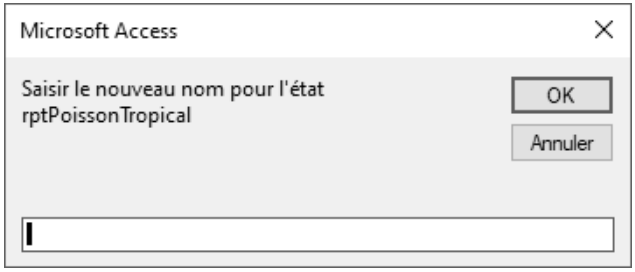
Durée estimative : 15 minutes
Complétez la procédure **NouveauPoissonTropical** ci-après. Elle permet de saisir un nouveau poisson tropical et de l'ajouter comme enregistrement de la table **tblPoissonTropical**.

```
Sub NouveauPoissonTropical()
 Dim rs As DAO.Recordset
 Dim NouveauPoisson As PoissonTropical
 'On Error GoTo Sortie_Sur_Erreur

 ' Ouvrir la table en lecture/écriture
 Set rs = CurrentDb.OpenRecordset("tblPoissonTropical", dbOpenDynaset)
 rs.AddNew

 NouveauPoisson.NomPoisson = ....
 rs("NomPoisson").Value = ...
        .
        .
         .
 rs.Update
 rs.Close
 Set rs = Nothing
Sortie_Sur_Erreur:
End Sub
```

Corrigé p. 179

Énoncé 2.6 Renommer un état Access

Durée estimative : 10 minutes
Créez, dans le module **Chapitre_02**, la procédure **RenommerEtat** qui renomme l'état **lstPoissonTropical** en un nouveau nom saisi par l'utilisateur.



Indice

Utilisez l'instruction `DoCmd.Rename`.

Corrigé p. 180