R2.10 : Ingénierie des systèmes cyberphysiques
BUT 1 Génie Mécanique et Productique, semestre 2

Informatique et Programmation

version du 8 décembre 2025

BUT GMP

Université de Lille

IUT de Lille
Bd Paul Langevin - Cité Scientifique - BP 90179
59653 Villeneuve d'Ascq Cedex - France
Tél: 03 59 63 21 00 - iut@univ-lille.fr

MODULE R2.10 – Ingénierie des systèmes cyberphysiques

Objectifs

Utiliser de façon rationnelle un tableur et ses fonctions principales. Savoir traiter dans un langage structuré un problème simple. Nous utiliserons le langage orienté objet : Visual Basic Application.

Les points étudiés sont :

- Bases du tableur
- Types de données et opérateurs associés
- Manipulation des données (formulaire)
- Calcul simple
- Automatisation du traitement des données de tableur, autres logiciels métiers (macro)
- Bases de programmation
- Structure d'un programme
- Boucles et conditions

Emploi du temps et mode de contrôle des connaissances

- 5 séances de 4h de TP (contrôle de connaissance inclus)
- 1 contrôle de TP individuel lors du dernier TP

Cours MOODLE :
`Institut universitaire de technologie (IUT) > IUT - Site de Villeneuve d'Ascq > IUT - GMP > BUT1 - GMP > S2 > R2.10 Ingénierie des Systèmes Cyberphysiques > R2.10 Informatique`

clé d'inscription facile sur moodle : **cmhiht**

Vous devez venir en TP avec votre énoncé, sous peine d'1 point en moins sur la note finale par séance sans énoncé.

à la fin de chaque séance, enregistrer **tous les classeurs** sur lesquels vous avez travaillé dans la section moodle **Séance X** correspondante

au début de chaque séance, récupérer **le classeur non terminé** dans la section moodle **Séance X** de la séance précédente

La programmation structurée

C'est une méthode de réflexion permettant de résoudre un problème par informatique faisant appel à un théorème n'ayant jamais été démenti :

Tout problème à traiter par informatique peut être décrit à l'aide de trois structures nécessaires et suffisantes :

- ▷ *La séquence ;*
- ▷ *La condition ;*
- ▷ *La répétition.*

```
REM Sequence d'instructions
Sub macro
→   Dim variable1 As Integer
→   Dim variable2 As String
→   Instruction 1
→   Instruction 2
→   '...
End Sub
```

```
REM Instruction conditionnelle
If (condition) Then
→   Instruction 1
→   '...
Else
→   Instruction 2
→   '...
End If
```

```
REM Instruction Repeter ... Jusqu'a
Do
→   Instruction 1
→   Instruction 2
→   '...
Loop Until (condition)
```

```
REM Repeter pour i variant de 1 a 10
Dim i As Integer
For i = 1 To 10
→   Instruction 1
→   '...
Next i
```

Dans ces exemples, le symbole → représente une tabulation. La tabulation permet d'ajouter un décalage dans l'affichage qu'on appelle *indentation*. Le niveau d'indentation augmente quand on quitte une séquence pour entrer dans une condition ou une boucle de répétition. À l'inverse, le niveau d'indentation diminue quand on quitte la condition ou la boucle pour revenir au niveau de séquence précédent.

En langage Basic, l'indentation n'est pas nécessaire pour qu'un programme soit correctement exécuté. Par contre, dans d'autres langages comme le Fortran ou Python, l'indentation fait partie de la technique de programmation. Ceci étant dit, l'indentation est très utile pour le programmeur car elle permet de facilement repérer les blocs de code correspondant à des séquences situées à différents niveaux de condition ou de répétition.

Dans tous les TP, vous ferez très attention à respecter les règles d'indentation, notamment dans les programmes rendus lors de l'examen final, où ce critère sera l'un de ceux utilisés pour évaluer votre travail

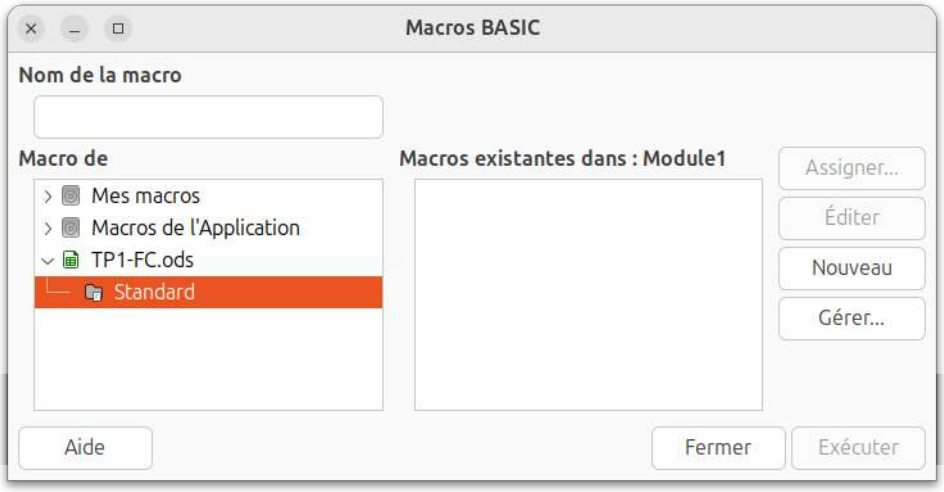
TP 1 : Mise en place d'une macro

1/ Ouvrir un Classeur dans le tableur LibreOffice Calc ;
2/ Double-cliquer avec le bouton gauche de la souris sur Feuille1 (en bas à gauche), la renommer F1. C'est dans cette feuille que nous allons travailler ;
3/ Enregistrer le classeur vide sur votre bureau `Fichier > Enregistrer-sous...` au format par défaut .ODS sous le nom TP1 suivi de vos initiales, par exemple : TP1-XX.

1.1 Écriture d'un programme Basic associé à votre fichier

> Suivre exactement les instructions suivantes afin d'associer votre programme à votre classeur et non pas à l'ordinateur

1/ Sélectionner `Outils > Macros > Gérer les Macros > Basic...` pour afficher la boîte de dialogue suivante



puis dans le cadre de gauche Macro de, développer votre classeur TP1-XX.ods. Ensuite, sélectionner Standard puis cliquer Nouveau Une autre boîte de dialogue s'affiche, proposant de créer un module Basic appelé Module1, valider en cliquant sur OK.

> Attention de NE PAS sélectionner les onglets Mes macros ou Macros LibreOffice, sinon votre programme ne sera pas enregistré avec votre fichier.

La fenêtre de programmation en Basic apparaît avec des lignes déjà écrites (`Sub Main ...`).

Effacer tout le texte avant de continuer.

2/ Saisir alors le premier sous-programme à tester. (sous-programme se traduit **Sub**Routine en Anglais). Il n'est pas nécessaire de recopier les commentaires en gris.

```
Option Explicit ' Obligatoire pour verifier la coherence des variables
Sub DisBonjour ' Nom du sous-programme
→   Dim nom As String ' Declaration de la variable nom en chaine de caracteres
→   nom = InputBox("Quel est votre nom ?") ' Demande de saisir une chaine
→   MsgBox("Alors, Bonjour " & nom) ' & : separateur de termes a afficher
End Sub
```

3/ Enregistrer la feuille avec `Fichier > Enregistrer` qui comprend également le programme que vous venez de saisir

4/ Faire exécuter le programme en appuyant sur la touche F5 ou en cliquant sur ▶. Corriger les erreurs si nécessaire et n'oubliez pas d'enregistrer votre fichier.

1.2 Création d'un Bouton de contrôle

Le programme étant mis au point, nous allons maintenant créer un bouton permettant de faire exécuter ce programme depuis la feuille F1.

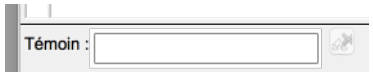
1/ Dans Fenêtre cliquer sur TP1-XX.ods, la feuille F1 apparaît ;
2/ Sélectionner Affichage > Barres d'outils > Contrôles de Formulaire. Dans la boîte d'outils qui apparaît, activer le mode Ebauche en cliquant sur l'icône ;
3/ Cliquer sur l'icône Bouton et dessiner un bouton sur la feuille F1 (il faut faire glisser le pointeur de la souris sur les icônes et attendre une seconde pour que le nom de l'icône apparaisse) ;
4/ Cliquer à l'intérieur du bouton avec le bouton droit, puis sélectionner Propriétés du contrôle et ensuite l'onglet Général ;
5/ Modifier le titre du bouton (étiquette), en le nommant BONJOUR, et agrandir la taille des caractères ;
6/ Sélectionner l'onglet Événements puis choisir le champ Éxécuter l'action ;

> *i* Il arrive parfois, sur les grands écrans, que le bouton ··· n'apparaisse pas. Dans ce cas, augmenter la largeur de la fenêtre

7/ Cliquer sur le bouton Macro puis sélectionner TP1-XX.ods > Standard > Module1. Le nom de la procédure `DisBonjour` apparaît alors dans la fenêtre de droite. Sélectionner `DisBonjour` puis cliquer sur OK ;
8/ Désactiver le mode conception avec le même icône ayant permis de l'activer et essayer le programme `DisBonjour` qui doit démarrer en cliquant sur votre bouton BONJOUR ;
9/ Copier / Coller le programme `DisBonjour` pour créer un autre programme `DisAge`. Ajouter quelques lignes afin de demander l'âge de l'utilisateur avec un test donnant un message d'erreur si l'âge est supérieur à 140 ans ;

```
Option Explicit ' Obligatoire pour verifier la coherence des variables
Sub DisAge ' Nom du sous-programme
→   Dim nom As String ' Declaration de la variable nom en chaine de caracteres
→   Dim age As Integer ' Declaration de la variable age en entier
→   nom = InputBox("Quel␣est␣votre␣nom␣?") ' Demande de saisir une chaine
→   age = InputBox("Quel␣est␣votre␣age␣?") ' Demande de saisir un entier
→   If (age > 140) Then
→   →   MsgBox("Erreur")
→   Else
→   →   MsgBox("L'age␣de␣" & nom & "␣est␣" & age)
→   End If
End Sub
```

10/ Tester le programme. Pour ce faire, ajouter la variable `age` dans la case Témoin :



puis exécuter le programme en mode pas à pas en tapant sur la touche F8 ou sur l'icône ;
11/ Modifier le programme avec un test donnant un message d'erreur si l'âge est négatif ou supérieur à 140 ans.

TP 2 : Algorithmie

Création du classeur TP2-XX et de la feuille de travail

1/ Ouvrir un Classeur dans le tableur LibreOffice Calc ;
2/ Double-cliquer avec le bouton gauche de la souris sur Feuille1, la renommer F1. Créer une seconde feuille nommée F2 ;
3/ Enregistrer le classeur vide sur votre bureau `Fichier > Enregistrer-sous...` au format par défaut .ODS sous le nom TP2-XX.

> Il est conseillé d'enregistrer votre classeur avant d'exécuter votre programme afin d'éviter de perdre votre travail si LibreOffice plante. Vous pouvez utiliser le raccourci clavier Ctrl+S

Exercice TP2_1 – Calcul élémentaire sur entiers

Écrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le cube de ce nombre. Une solution à compléter dans la ou les lignes noires :

```
Option Explicit
Sub TP2_1
→   REM Declaration des variables
→   Dim nombre As Integer
→   Dim cube As Integer

→   REM Programme
→   nombre = InputBox("Saisir un nombre")
→   cube = [redacted]

→   MsgBox("Le cube de " & nombre & " est "& cube)
End Sub
```

Créer un bouton de contrôle associé à ce programme.

Que se passe-t-il quand `nombre = 100` ? Modifier la déclaration de `cube` par `Dim cube As Long`. Refaire le test quand `nombre = 1000`.

Exercice TP2_1 – Affectation

Écrire un algorithme qui saisit deux entiers `a` et `b` et qui échange leurs valeurs. Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_2
→   Dim a As Integer, b As Integer, temp As Integer
→   a = InputBox ("Saisir la valeur de a")
→   b = InputBox ("Saisir la valeur de b")
→   MsgBox("Avant echange a = " & a & " b = " & b)
→   temp = a
→   [redacted]
→   MsgBox("Apres echange a = " & a & " b = " & b)
End Sub
```

Exercice TP2_3 – Conditions

Écrire un algorithme qui affiche `"Glace"` si la température saisie est inférieure ou égale à 0, `"Liquide"` si la température est comprise entre 1 et 100 et `"Vapeur"` si la température est supérieure à 100.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_3
→   Dim temp As Integer

→   temp = InputBox("Saisir la temperature")

→   If (temp <= 0) Then
→   →   MsgBox("Glace")
→   Else
→   →   If [redacted] Then
→   →   →   [redacted]
→   →   Else
→   →   →   [redacted]
→   →   End If
→   End If
→   MsgBox("Fin")
End Sub
```

Exercice TP2_4a – Conditions avec opérateurs logiques

Écrire un algorithme qui demande deux nombres entiers à l'utilisateur et l'informe ensuite si leur produit est nul, négatif ou positif (on ne laisse pas de côté le cas où le produit est nul).

on ne doit **pas calculer** le produit des deux nombres

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_4a
→   Dim a As Integer
→   Dim b As Integer

→   a = InputBox("Saisir un entier svp")
→   b = InputBox("Saisir un autre entier svp")

→   If ((a = 0) OR (b = 0)) Then
→   →   MsgBox("Produit nul")
→   Else
→   →   If [redacted] Then
→   →   →   MsgBox("Produit positif")
→   →   Else
→   →   →   [redacted]
→   →   End If
→   End If
→   MsgBox("Fin")
End Sub
```

Exercice TP2_4b – Conditions avec opérateurs logiques et variables booléennes

Comme précédemment, écrire un algorithme qui demande deux nombres entiers à l'utilisateur et l'informe ensuite si leur produit est nul, négatif ou positif. Cette fois, on va utiliser des variables *booléennes* pour stocker le résultat des comparaisons intermédiaires.

Une variable booléenne est définie en utilisant le type `Boolean` dans LibreOffice Basic. Une variable booléenne peut uniquement prendre les valeurs `True` et `False`.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_4b
→   Dim a As Integer, b As Integer
→   Dim unNul As Boolean, deuxPositifs As Boolean, deuxNegatifs As Boolean

→   a = InputBox("Saisir␣un␣entier␣svp")
→   b = InputBox("Saisir␣un␣autre␣entier␣svp")

→   unNul = (a = 0) OR (b = 0)
→   deuxPositifs = (a > 0) [redacted]
→   deuxNegatifs = [redacted]

→   If (unNul) Then
→   →   MsgBox("Produit␣nul")
→   Else
→   →   If [redacted] Then
→   →   →   MsgBox("Produit␣positif")
→   →   Else
→   →   →   [redacted]
→   →   End If
→   End If
→   MsgBox("Fin")
End Sub
```

Exercice TP2_5 – Conditions avec opérateurs logiques et ordre des valeurs

Écrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie : `"Poussin"` de 6 à 7 ans, `"Pupille"` de 8 à 9 ans, et `"Autre"` avant 6 ans ou après 9 ans.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_5
→   Dim age As Integer

→   age = InputBox("Saisir␣l'age␣de␣l'enfant")

→   If ((age < 6) OR (age > 9)) Then
→   →   MsgBox("Autre")
→   Else
→   →   [redacted]
→   →   →   MsgBox("Poussin")
→   →   [redacted]
→   →   →   MsgBox("Pupille")
→   →   [redacted]
→   End If
→   MsgBox("Fin")
End Sub
```

Exercice TP2_6 – Conditions sur caractères

Écrire un algorithme qui demande à l'utilisateur de saisir un nom d'animal. Puis le programme demande ensuite si c'est un poisson ou un mammifère (en saisissant `"P"` ou `"M"`).

Il écrit ensuite sur l'écran `"Le" xxxx "est un joli " yyyy`, où `yyyy` est soit `"poisson"`, soit `"mammifere"`, soit `"autre"`.

Par exemple, le programme affiche

- ▷ `"Le cheval est un joli mammifere"` si l'utilisateur a saisi `"M"` ;
- ▷ `"Le cheval est un joli autre"` si l'utilisateur n'a saisi ni `"M"` ni `"P"`.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_6
→   Dim animal As String
→   Dim categorie As String
→   Dim affiche As String

→   animal = InputBox("Saisir␣le␣nom␣d'un␣animal")
→   categorie = InputBox("Saisir␣P␣pour␣poisson␣ou␣M␣pour␣mammifere")

→   affiche = "autre"
→   If (categorie = "P") Then
→   →   affiche = "poisson"
→   Else
→   →   [redacted]
→   →   →
→   →   [redacted]
→   End If
→   MsgBox(animal & "␣est␣un␣joli␣" & affiche)
End Sub
```

Exercice TP2_7 – Itération faire jusqu'à : Do … Loop Until ()

Copier / coller le programme TP2_6, et l'améliorer en empêchant l'utilisateur de saisir autre chose que "p", "P", "m" ou "M". Le PC repose indéfiniment la question jusqu'à ce que la catégorie soit conforme.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_7
→   Dim animal As String
→   Dim categorie As String
→   Dim affiche As [redacted]

→   animal = InputBox("Saisir le nom d'un animal")
→   Do
→   →   categorie = InputBox("Saisir P pour poisson ou M pour mammifere")
→   →   affiche = "autre"
→   →   If (categorie = "P" OR categorie = "p") Then
→   →   →   affiche = "poisson"
→   →   Else
→   →   →   [redacted]
→   →   →   →   [redacted]
→   →   →   [redacted]
→   →   End If
→   Loop Until ([redacted])
→   MsgBox(animal & " est un joli " & affiche)
End Sub
```

Exercice TP2_8 – Itération sur nombre avec Do … Loop Until ()

Écrire un algorithme qui demande un nombre de départ et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur saisit le nombre 17, le programme affichera les nombres de 18 à 27.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_8
→   Dim depart As Integer
→   Dim nombre As Integer
→   Dim indice As Integer

→   depart = InputBox("Saisir un nombre de depart")
→   indice = 1
→   Do
→   →   [redacted]
→   →   MsgBox(nombre)
→   →   [redacted]
→   Loop Until (indice > 10)
→   MsgBox("Fin")
End Sub
```

Exercice TP2_9 – Itération sur nombre avec For … Next

Écrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur saisit le nombre 17, le programme affichera les nombres de 18 à 27.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_9
→   Dim depart As Integer
→   Dim nombre As Integer
→   Dim indice As Integer

→   depart = InputBox("Saisir un nombre de depart")
→   For indice = 1 To 10
→   →   [redacted]
→   →   MsgBox(nombre)
→   Next indice
→   MsgBox("Fin")
End Sub
```

Exercice TP2_10 – Test itératif de conformité avec Do … Loop Until ()

Écrire un algorithme qui demande de saisir un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître le message `"Plus petit !"`, et inversement `"Plus grand !"` si le nombre est inférieur à 10.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_10
→   Dim nombre As Integer

→   Do
→   →   nombre = InputBox("Saisir un nombre compris entre 10 et 20")
→   →   If (nombre < 10) Then
→   →   →   [redacted]
→   →   Else
→   →   →   If [redacted] Then
→   →   →   →   [redacted]
→   →   →   End If
→   →   End If
→   Loop Until [redacted]
→   MsgBox("Fin du programme")
End Sub
```

Exercice TP2_11 – Statistiques en ligne avec Do … Loop Until ()

Écrire un algorithme qui permet de saisir successivement une liste d'entiers. La boucle s'arrête quand l'entier saisi est négatif. L'algorithme doit retourner la valeur maximale dans la liste des entiers saisis ainsi que leur moyenne. La dernière valeur saisie qui est négative ne doit pas être prise en considération dans le calcul de ces statistiques.

Exemple, pour les entiers saisis : 2, 8, 3, 3, -1, l'algorithme affiche un maximum de 8 et une moyenne de 16/4 = 4.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_11
→   Dim nombre As Integer
→   Dim maximum As Integer
→   Dim somme As Integer
→   Dim nbSaisis As Integer
→   Dim moyenne As Double

→   somme = 0
→   maximum = -1
→   nbSaisis = 0
→   Do
→   →   nombre = InputBox("Nouvel␣entier")
→   →   If (nombre > maximum) Then
→   →   →   [redacted]
→   →   End If
→   →   If (nombre >= 0) Then
→   →   →   [redacted]
→   →   →   [redacted]
→   →   End If
→   Loop Until [redacted]
→   If (nbSaisis > 0) Then
→   →   [redacted]
→   →   MsgBox("Maximum␣=␣" & maximum & ",␣moyenne␣=␣" & moyenne)
→   Else
→   →   MsgBox("Pas␣de␣nombre␣saisi")
→   End If
End Sub
```

Exercice TP2_12 – Statistiques sur un tableau

Écrire un algorithme qui calcule la valeur moyenne de 3 notes saisies et stockées dans un tableau. Lors de la saisie d'une note il faut vérifier qu'elle est comprise entre 0 et 20.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_12
→   Dim notes(1 To 3) As Integer
→   Dim indice As Integer
→   Dim somme As Integer
→   Dim moyenne As Single

→   REM Saisie des notes
→   For indice = 1 To 3
→   →   Do
→   →   →   notes(indice) = InputBox("Saisir la note numero " & indice)
→   →   Loop Until ((notes(indice) >= 0) AND [redacted])
→   Next indice

→   REM Calcul de la moyenne
→   somme = 0
→   For indice = 1 To 3
→   →   somme = notes(indice)[redacted]
→   [redacted]
→   [redacted]
→   MsgBox("La moyenne vaut " & moyenne)
End Sub
```

Exercice TP2_13 – Indice du tableau

Le salaire horaire d'un employé dépend de sa catégorie d'après le tableau suivant :

Catégorie	Salaire horaire
1	5,5
2	7
3	8,2
4	10

Écrire un programme qui calcule le salaire d'un employé après avoir saisi sa catégorie et le nombre d'heures travaillées. Par exemple pour un employé de catégorie 1 qui a travaillé 20 heures, le salaire sera égal à 110 euros.

Créer un bouton de contrôle associé à ce programme.

```
Option Explicit
Sub TP2_13
→   Dim horaire(1 To 4) As Single
→   Dim salaire As Double
→   Dim heures As Integer
→   Dim categorie As Integer

→   REM Saisie des taux horaires
→   horaire(1) = 5.5
→   [redacted]
→   [redacted]
→   horaire(4) = 10

→   REM Calcul du salaire
→   Do
→   →   categorie = InputBox("Saisir la categorie entre 1 et 4")
→   Loop Until ((categorie >= 1) AND [redacted])
→   Do
→   →   heures = InputBox("Saisir le nombre d'heures travaillees")
→   Loop Until ([redacted])
→   salaire = [redacted] * horaire(categorie)
→   MsgBox("Pour la categorie " & categorie & " et " & heures & " heures")
→   MsgBox("Le salaire est de " & salaire & " euros")
End Sub
```

Exercice TP2_14 – A programmer sans compléter un code existant

Dans cette partie qui conclut le TP, vous allez écrire deux sous-programmes sans pouvoir compléter un code déjà existant, c'est-à-dire en créant vous-même les procédures Basic. Créer un bouton de contrôle associé à chacun de ces programmes.

2.0.1 Premier exercice

Écrire un programme Visual Basic nommé EXO1 effectuant les opérations suivantes :

- ▷ Le PC demande de saisir le nom d'une personne ;
- ▷ Il demande si cette personne est un européen ou un américain, en saisissant `"E"` ou `"A"` ;
- ▷ Il affiche enfin le message `XXXX "est un super" YYYY` en fonction des données saisies.

Ajouter un bouton EXO1 sur la première feuille du classeur afin d'appeler facilement votre programme. Par exemple le programme peut afficher `"Jules est un super europeen"` ou `"Maria est un super americain"`, ou encore `"Tom est un super autre"`. Il ne faut pas s'occuper des accords grammaticaux.

Deuxième exercice

Écrire un programme Visual Basic nommé EXO2 améliorant le fonctionnement du programme précédent :

- ▷ Le PC demande de saisir le nom d'une personne ;
- ▷ Il demande si cette personne est un européen ou un américain, en saisissant `"E"`, `"e"`, `"A"` ou `"a"`. Le programme repose indéfiniment la question tant que la réponse n'est pas satisfaisante ;
- ▷ Il affiche enfin le message `XXXX "est un super " YYYY` en fonction des données saisies.

Ajouter un bouton EXO2 sur la première feuille du classeur afin d'appeler facilement ce deuxième programme.

TP 3 : Nouvelles fonctions

3.1 Création du classeur TP3-XX et de la feuille de travail

1/ Ouvrir un Classeur dans le tableur LibreOffice Calc ;

2/ Double-cliquer avec le bouton gauche de la souris sur Feuille1, la renommer F1. Créer une seconde feuille nommée F2 ;

3/ Enregistrer le classeur vide sur votre bureau **Fichier > Enregistrer-sous...** au format par défaut .ODS sous le nom TP3-XX.

Création d'une fonction

Nous allons créer une fonction permettant de calculer le volume d'un cône à partir du rayon de la base (R) et de sa hauteur (H). Suivre les instructions décrites dans le paragraphe 1/ du paragraphe 1.1.

Dans la fenêtre de programmation Basic, taper la fonction suivante :

```
Function VolCone(R as single, H as single) as single
→   VolCone=Pi*R^2*H/3
End Function
```

Dans la feuille F1, nous allons créer une colonne intitulée Rayon variant de zéro à 3 par pas de 0,1, une seconde colonne avec la hauteur h=2 constante et une troisième colonne qui utilise la fonction `VolCone` pour afficher le volume.

La feuille `F1` contient trois zones, `Rayon` sur fond jaune, `Hauteur` sur fond jaune également et `Volume Cone` sur fond bleu, qui fait appel à la fonction VolCone comme représenté sur la figure suivante :

	A	B	C	D	E
1	Rayon	Hauteur	Volume Cone		
2	0	2	0		
3					
4	0.1	2	0.02		
5					
6	0.6	2	0.75		
7					
8	0.5	3	0.78		
9					

Pour parvenir à ce résultat effectuer les opérations suivantes :

1. Entrer les titres des trois colonnes et fixer le format des nombres à 4 décimales (Format-Cellules)
2. Dans la colonne A Rayon, taper 0 puis 0,1 en dessous , sélectionner ces deux cellules avec la souris, saisir ensuite la poignée en bas à droite et la tirer vers le bas. Cette opération permet de remplir automatiquement la colonne avec des valeurs croissantes suivant l'incrément indiqué.
3. Effectuer le même traitement dans la colonne Hauteur en tapant au départ deux fois le même nombre (1) .
4. Dans la colonne Volume Cône se positionner sur la première case à remplir (C2) et taper = VolCone(puis cliquer sur la cellule correspondant au rayon , frapper la touche point-virgule cliquer ensuite sur la cellule correspondant à la hauteur . Enfin fermer la parenthèse .

   On doit ainsi obtenir =VOLCONE(A2;B2) et la valeur zéro s'affiche après validation.
5. Afin de recopier automatiquement cette fonction dans le reste de la colonne, cliquer sur la cellule, saisir la poignée et tirer vers le bas. On doit alors obtenir les valeurs données sur la figure précédente.

3.2 Travail à faire

3.2.1 Volume d'une sphère

Dans la même feuille ajouter une colonne (D) en orange intitulée Volume Sphère et écrire une fonction intitulée `Volsphere` permettant de calculer et afficher dans cette colonne le volume d'une sphère de rayon correspondant aux valeurs indiquées dans la colonne A.

	A	B	C	D	E
1	Rayon	Hauteur	Volume Cone	Volume Sphere	
2					
3	0	2	0	0	
4					
5	0.1	2	0.02	0.001	
6					
7	0.6	2	0.75	0.22	
8					
9	0.5	3	0.78	0.13	
10					

3.2.2 Fonction total Hors Taxes

Renommer la feuille 2 en F2 et ouvrir cette feuille. Créer une fonction en basic intitulée `TotHt` acceptant deux données en entrée : le nombre d'articles (Nart) et le prix unitaire hors taxe (Punit) et retournant en sortie le prix total hors taxes (TotHT). Ceci donne pour en-tête de la fonction :

Function `TotHT`(Nart As Integer, Punit As Single) As Single

Une réduction de 5% doit être appliquée si le nombre d'articles est supérieur à 10 . Essayer la fonction en créant un tableau comme ci-dessous.

	A	B	C	D	E
1	Nombre d'articles (Nart)	prix unitaire HT (Punit)	Total HT (TotHT)		
2	2	10	20		
3					
4	3	15	45		
5					
6	12	10	114		
7					
8	100	2	190		
9					

3.2.3 Fonction total toutes taxes

Créer une fonction en basic intitulée `PrixTTC` acceptant trois données en entrée : le nombre d'articles (Nart), le prix unitaire hors taxes (Punit), le taux de TVA (TVA) et retournant en sortie le prix total avec Taxes. Une réduction de 10% sur le prix total TTC sera appliquée si le nombre d'articles est supérieur à 5 .

NB : Le prix TTC = Prix HT * (1+ TVA/100)

	A	B	C	D	E
1	Nombre d'articles (Nart)	prix unitaire HT (Punit)	Taux TVA (TVA)	Prix TTC	
2	2	10	19.6		
3					
4	3	15	19.6		
5					
6	12	10	5.5		
7					
8	10	10	20		
9					
10	20	10	20		
11					

TP 4 : Notions d'Objet en Visual Basic

Création du classeur TP4-XX et de la feuille de travail

1/ Ouvrir un Classeur dans le tableur LibreOffice Calc ;
2/ Double-cliquer avec le bouton gauche de la souris sur Feuille1, la renommer F1. Créer une seconde feuille nommée F2 ;
3/ Enregistrer le classeur vide sur votre bureau **Fichier > Enregistrer-sous...** au format par défaut .ODS sous le nom TP4-XX.

4.1 Lecture / écriture de données dans la feuille F1

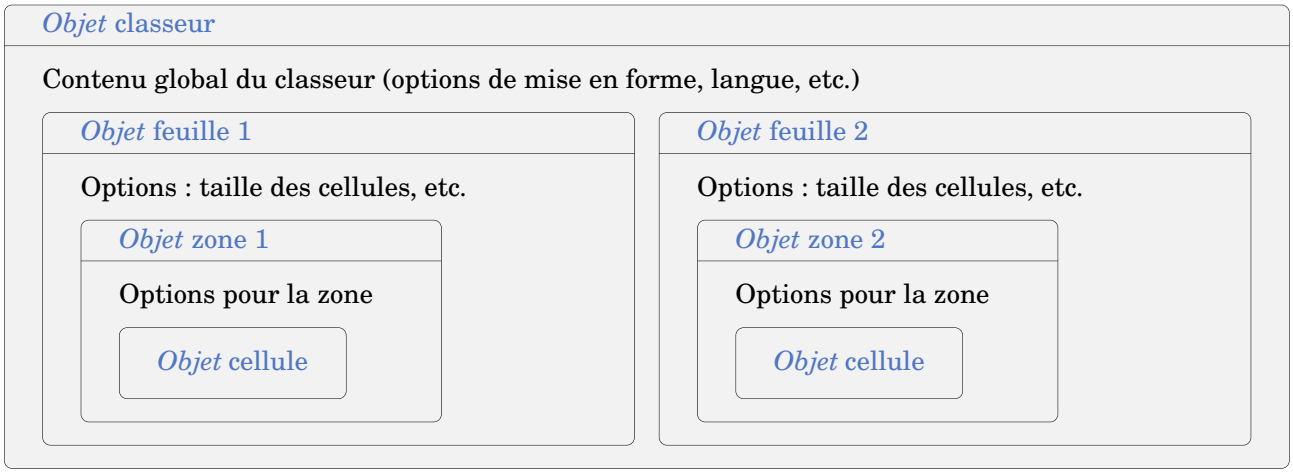
En LibreOffice Basic, un classeur est un *objet* qui contient plusieurs feuilles identifiées par leur nom.

Chaque feuille est également un objet dans lequel on peut définir des zones rectangulaires identifiées par l'intersection d'une plage de colonnes et d'une plage de lignes, par exemple `A1:D10`.

Ensuite, chaque zone est également un objet dans lequel on peut identifier chaque cellule par ses coordonnées `(colonne, ligne)` débutant à zéro, par exemple `(1,6)` qui localise la cellule `B7` dans la zone présentée comme exemple auparavant.

Enfin, une cellule est aussi un objet dont on peut obtenir ou définir des propriétés, par exemple son contenu (chaîne de caractères ou valeur numérique).

Cette hiérarchie d'objets est représentée sur la figure ci-dessous.



4.1.1 Déclaration de variables globales

Ouvrir la fenêtre de programmation Basic et recopier les lignes suivantes afin de définir des variables *globales*, c'est-à-dire qui seront accessibles dans toutes les procédures et fonctions Basic du module.

```
Option Explicit ' Obligatoire pour verifier la coherence des variables
Global classeur As Object ' Classeur complet
Global feuille1 As Object ' Une premiere feuille du classeur
Global zone1 As Object ' Une premiere zone de cellules
```

4.1.2 Affectations pour l'accès aux objets

Saisir le sous-programme `DefEspaceTravail` à la suite de la définition des variables globales. En appelant ce sous-programme dans toutes vos procédures et fonctions, vous pourrez facilement définir les objets auxquels vous voulez accéder dans le classeur.

```
Sub DefEspaceTravail
→   classeur = ThisComponent ' Le classeur actuellement ouvert
→   feuille1 = classeur.Sheets.getByName("F1") ' La feuille nommee F1
→   zone1 = feuille1.getCellRangeByName("A1:P30") ' La zone A1:P30 de F1
End Sub
```

4.1.3 Accès aux cellules

Enfin, saisir le programme suivant qui écrit d'abord le mot `"Nom"` dans la cellule `A1`, puis demande de saisir un nom avec `InputBox` et écrit ce nom dans la cellule `B1`.

```
Sub EcrireDansCellule
→   Dim cellule As Object ' Une cellule de la feuille
→   Dim nom As String ' Une chaine de caracteres
→   DefEspaceTravail ' Appeler le sous-programme de definition des objets

→   cellule = zone1.getCellByPosition(0, 0) ' Position (0,0) -> A1
→   cellule.String = "NOM" ' Ecrire la chaine "NOM" dans A1

→   nom = InputBox("Saisir␣un␣nom") ' Saisir une chaine
→   cellule = zone1.getCellByPosition(1, 0) ' Position (1,0) -> B1
→   cellule.String = nom ' Recopier la chaine dans B1
End Sub
```

Pour accéder au contenu d'une cellule, il faut d'abord accéder à l'objet cellule lui-même en le stockant dans la variable `cellule`, puis au contenu de cet objet en utilisant la *méthode* `.String`. Comme son nom l'indique, la méthode `.String` permet de lire ou de définir une chaîne de caractères contenue dans la cellule.

Pour tester facilement votre programme, vous devez définir un bouton Saisir NOM sur la feuille de travail et associer ce bouton à la macro `EcrireDansCellule`.

Modifier ensuite votre programme pour qu'il écrive également le nom qui a été saisi dans les cellules `A3`, `B1` et `B4`.

4.1.4 Génération automatique d'une colonne de nombres

Saisir le sous-programme suivant. Il permet de remplir automatiquement la colonne `C`, de la ligne 1 à la ligne 10.

```
Sub RemplirColonneC
→   Dim cellule As Object ' Une cellule de la feuille
→   Dim ligne As Integer ' Un numero de ligne
→   DefEspaceTravail ' Appeler le sous-programme de definition d'une zone

→   For ligne = 0 To 9
→   →   cellule = zone1.getCellByPosition(2, ligne) ' Dans colonne C
→   →   cellule.Value = 2 * ligne ' Definir la valeur
→   Next ligne
End Sub
```

Pour tester facilement ce programme, créer un bouton Remplir sur la feuille de travail et l'associer à la macro.

Modifier ensuite ce programme pour remplir la ligne 3 de la colonne `A` jusqu'à la colonne `K`.

4.1.5 Copie de cellules

Le sous-programme suivant permet de recopier la valeur numérique contenue de la cellule `B5` dans la cellule `B6`.

```
Sub CopierCellule
→   Dim cellule1 As Object ' Une cellule de la feuille
→   Dim cellule2 As Object ' Une autre cellule de la feuille
→   DefEspaceTravail ' Appeler le sous-programme de definition d'une zone

→   cellule1 = zone1.getCellByPosition(1, 4) ' Cellule B5
→   cellule2 = zone1.getCellByPosition(1, 5) ' Cellule B6
→   cellule2.Value = cellule1.Value ' Recopier une cellule dans l'autre
End Sub
```

Pour recopier une chaîne de caractères plutôt qu'une valeur numérique, on aurait utilisé la méthode `.String` plutôt que la méthode `.Value`

Recopier ce sous-programme dans vos macros Basic et créer un bouton Copier Cellule pour tester son fonctionnement. Que se passe-t-il quand la cellule `B5` contient une chaîne de caractères ?

4.1.6 Autres méthodes d'accès aux propriétés d'une cellule

Il existe de nombreuses méthodes qui permettent de lire ou de fixer les propriétés d'une cellule. Par exemple, la méthode `.CharColor` permet de définir la couleur des caractères affichés dans une cellule, la méthode `.CharHeight` la taille de ces caractères et la méthode `.CellBackColor` la couleur de fond de la cellule. Le code suivant permet de modifier les propriétés de la cellule `A1`, pour afficher son contenu en bleu sur fond jaune avec des caractères de hauteur 16 :

```
Sub ColorierA1
→   Dim cellule As Object ' Une cellule de la feuille
→   DefEspaceTravail ' Appeler le sous-programme de definition d'une zone

→   cellule = zone1.getCellByPosition(0, 0) ' Cellule A1
→   REM On utilise la fonction RGB(rouge, vert, bleu) pour calculer une couleur
→   cellule.Charcolor = RGB(0, 0, 255) ' Pas de rouge, pas de vert, que du bleu
→   cellule.CellBackColor = RGB(255, 255, 0) ' rouge + vert = jaune
→   cellule.CharHeight = 16 ' Caracteres de hauteur 16
End Sub
```

Recopier ce sous-programme dans vos macros Basic et créer un bouton Colorier A1 pour tester son fonctionnement.

4.2 Travail en autonomie

4.2.1 Accès à la cellule courante

La cellule *courante* est celle qui est actuellement sélectionnée, soit à l'aide de la souris, soit avec les touches de déplacement (flèches, pages haut / bas, etc). Elle est indiquée par un cadre coloré ou grisé particulier, de même que la colonne et la ligne qui la définissent. Il ne peut y avoir qu'une seule cellule courante dans le tableur actif. Cette cellule est donc décrite par un objet attaché au classeur lui-même et non pas à une feuille ou une zone.

Pour accéder à cette cellule en Basic, il suffit de définir une variable objet et de récupérer la cellule courante de la façon suivante :

```
Dim celluleCourante As Object ' Definition de la variable
celluleCourante = ThisComponent.getCurrentSelection ' Cellule courante du classeur
```

Pour tester, modifier le code de la procédure `CopierCellule` pour recopier la valeur numérique contenue dans la cellule courante vers la cellule `A15` quand on appuie sur le bouton Copier Cellule.

Modifier les propriétés de la cellule `A15` pour améliorer sa visibilité (police 24, texte rouge).

4.2.2 Copie entre deux feuilles du même classeur

Modifier le programme précédent afin qu'il copie le contenu de la case courante de la feuille `F1` vers la cellule `A1` de la feuille F2.

Pour cela il faudra déclarer deux nouvelles variables objet `feuille2` et `zone2` au même endroit que la déclaration des variables `feuille1` et `zone1`. Puis il faudra compléter la procédure `DefEspaceTravail` en ajoutant les lignes :

```
feuille2 = classeur.Sheets.getByName("F2") ' La feuille nommee F2
zone2 = feuille2.getCellRangeByName("A1:P30") ' La zone A1:P30 de F2
```

Modifier encore ce programme afin qu'il copie une liste de noms se trouvant dans les cases `B2` à `B11` de la feuille F1 dans les mêmes cases de la feuille F2 (on pourra s'inspirer du programme écrivant une suite de nombres).

4.2.3 Type de contenu d'une cellule

Pour connaître le type de contenu d'une cellule, on peut utiliser la méthode `.CellContentType`. La valeur retournée indique le type de contenu :

0	cellule vide
1	valeur numérique
2	chaîne de caractères

Écrire un programme qui affiche le type de contenu de la cellule courante avec la fonction `MsgBox`. Par exemple, pour une cellule vide le message affiché est `"Cette cellule est vide"`.

4.2.4 Table de Pythagore

Écrire un programme qui permet de générer dans la feuille F2 du tableur une table de Pythagore de 10 lignes et 10 colonnes (table de multiplication). La cellule `A1` doit contenir la lettre `X` majuscule, la première ligne et la première colonne doivent contenir les nombres de 1 à 10 :

×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

4.2.5 Gestion des notes d'un groupe d'étudiants

L'objectif de cette partie du TP est de permettre de faciliter la saisie des notes d'un groupe d'étudiants et de calculer la moyenne de ces notes. Un exemple de feuille de calcul de ce type est présenté sur la figure suivante :

	A	B	C	D	E
1	Numero	Nom	Note	4	
2	1	Romuald	11		
3	2	Lisa	14		
4	3	Frédéric	15		
5	4	Typhaine	13		
6	Moyenne		12,25		
7					

Écrire un premier programme `PreparerFeuille` qui permet d'effacer le contenu de la feuille de calcul F2 et de remplir la première ligne avec les chaînes de caractères `"Numero"`, `"Nom"` et `"Note"` dans les cellules `A1`, `B1` et `C1`. Pour effacer le contenu de toute une feuille, de toute une zone, ou d'une seule cellule, on peut utiliser la méthode `.clearContents("127")`, par exemple `zone2.clearContents("127")`. Créer un bouton Preparer Feuille et l'attacher à votre programme pour le tester.

Après cette préparation, l'utilisateur doit saisir une liste de noms dans la colonne `B`. Une fois cette liste saisie, un deuxième programme `CompterEtudiants`, associé à un bouton Compter Etudiants, permet de préparer la saisie des notes. Ce programme réalise successivement les actions suivantes :

1. il détermine tout d'abord le nombre d'étudiants à traiter en explorant la colonne `B` cellule par cellule avec la méthode `.CellContentType`. Tant que la cellule n'est pas vide le programme considère qu'il s'agit d'un nom d'étudiant et il passe à la cellule suivante ;
2. si le nombre d'étudiants est nul, le programme affiche un message d'erreur et il s'arrête. Sinon, il mémorise le nombre d'étudiants dans la cellule `D1`, il remplit la colonne `A` avec les numéros des étudiants, puis il ajoute en bas de cette colonne une cellule contenant la chaîne `"Moyenne"` ;

Enfin, l'utilisateur va saisir et traiter les notes des étudiants en utilisant un troisième programme `SaisirNotes`, associé à un bouton Saisir Notes. Ce programme réalise successivement les actions suivantes :

1. il lit le nombre d'étudiants dans la cellule `D1`. Si ce nombre est nul, il affiche un message d'erreur et il s'arrête ;
2. il parcourt la colonne `C` cellule par cellule en tenant compte du nombre d'étudiants. Si une cellule contient déjà une valeur numérique contenue entre 0 et 20, il passe à la cellule suivante. Sinon, il demande une note avec la fonction `InputBox`, jusqu'à ce que la note saisie soit comprise entre 0 et 20, puis il la stocke dans la cellule ;
3. enfin, le programme calcule la moyenne des notes et il la stocke dans la colonne `C` sur la même ligne que le texte `"Moyenne"`.

TP 5 : Applications au codage binaire

Depuis moodle, téléchargez le fichier `TP5-mot-binaire-etudiant.ods`. La feuille `F1` contient trois zones, représentant trois mots binaires codés sur 4 bits : `A` sur fond jaune, `B` sur fond vert et `R` sur fond bleu, comme représenté sur la figure suivante :

	A	B	C	D	E
1	a3	a2	a1	a0	A en base 10
2	1	1	1	0	14
3					
4	b3	b2	b1	b0	B en base 10
5	0	1	1	1	7
6					
7	r3	r2	r1	r0	R en base 10
8	1	0	0	0	8
9					

Le programme Basic définit d'abord des variables globales objet pour accéder aux trois zones contenant les bits des 3 mots. Une procédure `DefinirZones` est également présente pour initialiser ces objets. Elle est appelée au début de toutes les autres macros qui doivent accéder aux bits des mots binaires `A`, `B` et `R` :

```
Option Explicit
Global zoneA As Object ' 4 cellules du mot A
Global zoneB As Object ' 4 cellules du mot B
Global zoneR As Object ' 4 cellules du mot R

Sub DefinirZones
→   zoneA = ThisComponent.Sheets.getByName("F1").getCellRangeByName("A2:D2")
→   zoneB = ThisComponent.Sheets.getByName("F1").getCellRangeByName("A5:D5")
→   zoneR = ThisComponent.Sheets.getByName("F1").getCellRangeByName("A8:D8")
End Sub
```

Exécuter de la macro exemple `ComplementerA` pour vérifier ce qu'elle réalise sur le mot binaire `A` et donc sur sa valeur en décimal.

```
Sub ComplementerA
→   Dim cellule As Object
→   Dim bit As Integer
→   DefinirZones

→   If (TestBinaire) Then
→   →   For bit = 0 To 3
→   →   →   cellule = zoneA.getCellByPosition(3 - bit, 0)
→   →   →   cellule.Value = 1 - cellule.Value
→   →   Next bit
→   End If
End Sub
```

Modifier les cellules du mot `A` manuellement et vérifier que l'opération de complémentation vérifie d'abord que le mot `A` est valide.

Compléter le code de la fonction `TestBinaire` pour qu'elle teste également la validité des deux autres mots binaires.

Analyser en détail le code Basic déjà présent dans le fichier avant de passer aux questions suivantes!

5.1 Travail à réaliser

Vous devez ajouter des macros Basic dans le fichier `TP5-mot-binaire-etudiant.ods` et / ou compléter les macros présentes afin de réaliser les opérations décrites ci-après sur les mots binaires `A`, `B` et `R`. Pour faciliter le test de vos macros, vous devez également créer des boutons d'appel sur la feuille F1.

1. macro `SaisirA` qui demande un entier décimal avec `InputBox` et convertit cet entier binaire en le plaçant dans le mot `A`;
2. macro `SaisirB` qui fait le même pour le mot `B`;
3. macro `AOUB` qui calcule le `OU` bit-à-bit des mots `A` et `B` et stocke le résultat dans `R`;
4. macro `AETB` qui calcule le `ET` bit-à-bit des mots `A` et `B` et stocke le résultat dans `R`;
5. macro `AXORB` qui calcule le `OU` exclusif (`XOR`) bit-à-bit des mots `A` et `B` et stocke le résultat dans `R`;
6. macro `IncrementerA` qui incrémente la valeur du mot `A`;
7. macro `Addition` qui additionne les mots `A` et `B` et stocke le résultat dans `R`. S'il y a un débordement, c'est-à-dire si le résultat "ne tient pas" sur 4 bits, il faut afficher un message d'erreur.

Pour l'incrémentation et l'addition, vous devez proposer des algorithmes qui travaillent directement sur les bits, sans passer par une conversion en décimal

5.2 Rappel de logique et aide

5.2.1 Opérations binaires sur 2 bits

a	b	a OR b	a AND b	a XOR b
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

5.2.2 Table de conversion d'une base à une autre

décimal	binaire	octal	hexadécimal
0	00000	000	00
1	00001	001	01
2	00010	002	02
3	00011	003	03
4	00100	004	04
5	00101	005	05
6	00110	006	06
7	00111	007	07
8	01000	010	08
9	01001	011	09
10	01010	012	0A
11	01011	013	0B
12	01100	014	0C
13	01101	015	0D
14	01110	016	0E
15	01111	017	0F
16	10000	020	10
17	10001	021	11

5.2.3 Incrémentation : I = A + 1

A décimal	A binaire	I décimal	I binaire
0	00000	1	00001
1	00001	2	00010
2	00010	3	00011
3	00011	4	00100
4	00100	5	00101
5	00101	6	00110
6	00110	7	00111
7	00111	8	01000
8	01000	9	01001
9	01001	10	01010
10	01010	11	01011
11	01011	12	01100
12	01100	13	01101
13	01101	14	01110
14	01110	15	01111
15	01111	16	10000

5.3 Conversion valeur décimale vers mot binaire

Le programme suivant convertit un nombre en une chaîne binaire équivalente. Attention, dans votre code, les chiffres d'un mot binaire ne sont pas représentés par des caractères "0" ou "1", mais par des valeurs numériques binaires stockées dans des cellules.

```
Sub DecimalBinaire
→   Dim decimal As Integer
→   Dim binaire As String

→   decimal = InputBox("Saisir␣un␣nombre")
→   binaire = "" ' Initialiser une chaine vide
→   Do
→   →   If ((decimal MOD 2) = 1) Then ' Modulo 2
→   →   →   binaire = "1" & binaire
→   →   Else
→   →   →   binaire = "0" & binaire
→   →   End If
→   →   decimal = decimal \ 2 ' Reste de la division entiere
→   Loop Until (decimal = 0)
→   MsgBox("En␣binaire:␣" & binaire)
End Sub
```

5.4 Conversion mot binaire du tableau CellA vers valeur décimale

Le programme suivant convertit le mot binaire contenu dans cellA en un nombre équivalent.

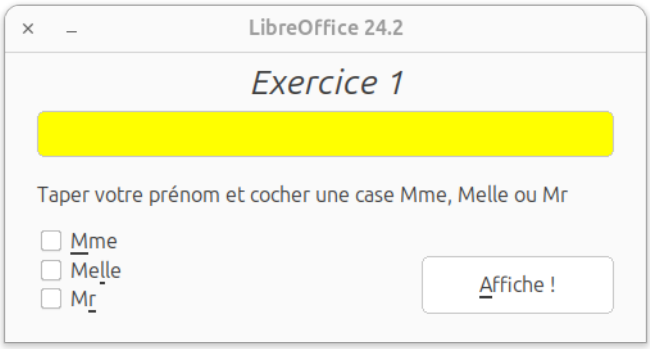
```
Sub BinaireDecimal
→   Dim valA10 As Integer
→   Dim i As Integer
→   Dim puis As Integer

→   valA10 = 0
→   →   puis = 1
→   →   i=0
→   →   do
→   →   →   valA10=valA10 + puis*CellA(i).value
→   →   →   puis = puis *2
→   →   →   i = i+1
→   →   loop until (i >3)

→   MsgBox("En␣decimal:␣" & valA10)
End Sub
```

TP 6 : Création et gestion de boîtes de dialogue

Création du classeur TP6-XX et de la feuille de travail

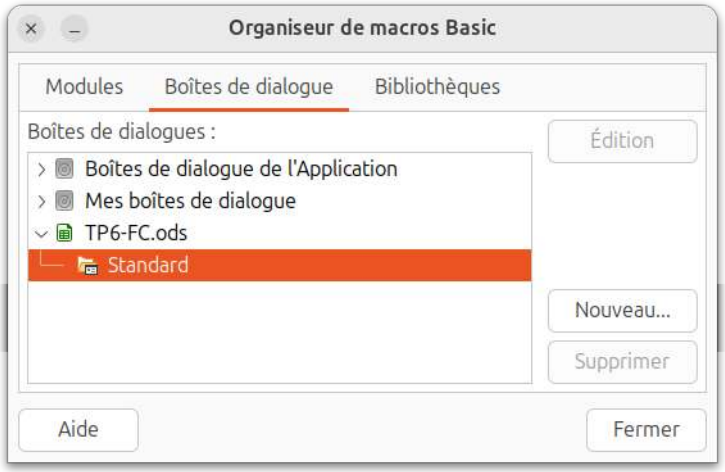
1/ Ouvrir un Classeur dans le tableur LibreOffice Calc ;
2/ Double-cliquer avec le bouton gauche de la souris sur Feuille1, la renommer F1. Créer une seconde feuille nommée F2 ;
3/ Enregistrer le classeur vide sur votre bureau `Fichier > Enregistrer-sous...` au format par défaut .ODS sous le nom TP6-XX.

6.1 Création d'une boîte de dialogue comme représentée ci-dessous



6.1.1 Dessin de la boîte de dialogue

1/ Sélectionner `Outils > Macros > Gérer les boîtes de dialogue...` pour afficher la boîte de dialogue suivante



2/ Puis dans Standard cliquer Nouveau...
3/ Nommer cette boîte Bonjour, cliquer sur Ok puis sur Edition La fenêtre LibreOffice Basic s'ouvre et présente une boîte de dialogue vide ainsi qu'une boite d'outils semblable aux outils déjà utilisés pour créer des boutons sur les feuilles de tableur dans les TP précédents. Si la boîte d'outils n'apparaît pas, cocher cette option dans Affichage.
4/ Choisir contrôle Champ d'étiquette AB et écrire le titre Exercice 1 en police 16 gras italique.
5/ Cliquer sur l'outil permettant de créer une Zone de Texte et dessiner cette zone dans la boîte de dialogue. Ouvrir la fenêtre propriétés à l'aide d'un clic-droit, nommer cette zone ZTexte1 puis choisir une police de caractères 14 points et une couleur d'arrière plan jaune.

6/ Sous cette boite insérer un Champ d'étiquette AB avec le texte d'accueil :
`Taper votre prénom et cocher la case Mme, Melle ou Mr`.

7/ Dessiner un Bouton OK, changer son nom en Bouton1 et changer son étiquette en Affiche! en caractères gras.

8/ Dessiner une Case à cocher, changer son nom en Case1 avec comme étiquette Mme. Créer deux autres cases Case2 (étiquette Melle) et Case3 (étiquette Mr).

Attention de bien respecter la casse des caractères (majuscules ou minuscules) dans le nom des contrôles

6.1.2 Écriture du programme de contrôle en Basic

1/ Ouvrir la feuille F1, puis sélectionner `Outils > Macros > Gérer les Macros > Basic...`

2/ Ensuite, sélectionner Standard puis cliquer Nouveau... Une autre boîte de dialogue s'affiche, proposant de créer un module Basic appelé Module1, valider en cliquant sur OK. Effacer le sous-programme Main.

3/ Saisir le programme suivant qui permettra d'afficher la boite de dialogue créée précédemment (ne pas saisir les commentaires en gris dans cet exemple) :

```
Option Explicit
Global boiteBonjour As Object ' Objet pour gerer la boite de dialogue
Global texte As Object ' Objet pour le champ de saisie de texte

Sub MontrerBonjour
→   DialogLibraries.LoadLibrary("Standard")
→   REM Ci-dessous faire attention a saisir Bonjour en respectant la casse
→   boiteBonjour = CreateUnoDialog(DialogLibraries.Standard.Bonjour)
→   REM Ci-dessous faire attention a saisir ZTexte1 en respectant la casse
→   texte = boiteBonjour.GetControl("ZTexte1")
→   boiteBonjour.Execute()
End Sub
```

- ▷ Les deux premières lignes du programme permettant de décrire le chemin permettant de trouver la boite et de l'affecter à la variable objet `boiteBonjour`.
- ▷ La troisième ligne permet de relier la variable objet `texte` à la zone de texte ZTexte1.
- ▷ La quatrième ligne `boiteBonjour.Execute()` correspond à une méthode de l'objet `boiteBonjour`, c'est-à-dire à une séquence de code exécutable propre à cet objet. Dans ce cas cette méthode fait apparaître la boîte de dialogue.

4/ Essayer ce programme, la boîte de dialogue intitulée Bonjour, que vous avez crée précédemment doit apparaître.

En cas d'erreur, vérifier l'orthographe des mots Bonjour et ZTexte1 qui doivent absolument être identiques dans les dessins de boites et votre programme

5/ Saisir le sous-programme suivant permettant d'écrire le prénom dans la Zone de texte et ensuite d'écrire Bonjour, suivi du prénom, lors d'un clic sur le Bouton1.

```
Sub Affiche
→   Dim prenom As String
→   prenom = texte.Text
→   texte.Text = "Bonjour␣" & prenom
End Sub
```

Ce module doit être appelé lors de l'appui sur le bouton nommé Bouton1. Pour cela, dans l'éditeur de boîte de dialogue, il faut ajouter la macro `Affiche` dans l'évènement Exécuter l'action associé au bouton Bouton1.

6/ Pour pouvoir utiliser plusieurs fois le bouton et effacer automatiquement la zone de texte, ajouter le sous-programme suivant et le faire exécuter lorsque le curseur de la souris est à l'intérieur de la zone de texte. Pour cela, il faut associer la macro `Efface` à l'évènement Souris à l'intérieur de la zone de texte ZTexte1.

```
Sub Efface
→   texte.Text = ""
End Sub
```

7/ Afin d'écrire une phrase de bienvenue complète, il faut tester l'état des cases à cocher. Pour la case Mme, il faut déclarer une variable objet en Global, par exemple :

```
Global mme As Object
```

puis définir cette variable dans la macro `MontrerBonjour` en ajoutant la ligne suivante au dessus de la ligne `boiteBonjour.Execute()` :

```
→   mme = boiteBonjour.GetControl("Case1")
```

et enfin, ajouter dans la macro `Affiche` la ligne suivante :

```
→   If mme.State = 1 Then : texte.Text = "Bonjour␣Madame␣" & Prenom : End If
```

Tester ce module et le bon fonctionnement de la case à cocher.

En cas d'erreur, vérifier l'orthographe du mot Case1 qui doit absolument être identique dans les dessins de boîtes et votre programme

8/ Ajouter les lignes de programme pour gérer les deux autres cases à cocher.

9/ En utilisant les propriétés des cases à cocher et des sous-programmes comme ci-dessous, faire en sorte que lorsqu'on coche une case les deux autres soient décochées.

```
Sub RazMme
→   If mme.State = 1 Then
→   →   melle.State = 0
→   →   mr.State = 0
→   End If
End Sub
```

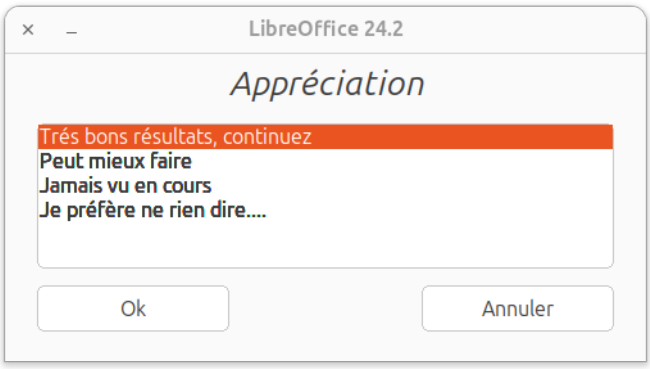
6.2 Travail à faire par vous-même

6.2.1 Compteur / décompteur comme celui représenté ci-dessous

- Nommer cette boite de dialogue Compteur, c'est plus explicite que Boite2 ;
- Le compteur démarre lors d'un appui sur le bouton Marche et s'arrête lors d'un appui sur le bouton Arret ;
- Le compteur n'est pas remis à zéro lorsqu'on passe de comptage à décomptage ;
- Le comptage est réalisé dans une macro `Marche`, qui incrémente ou décrémente le compteur tant que la variable booléenne globale `fin` est fausse (valeur `False`) ;
- L'arrêt du comptage est réalisé dans une macro `Arret` qui modifie la variable booléenne globale `fin` ;
- Pour éviter un comptage trop rapide, on introduit l'instruction `Wait 500` dans le programme pour attendre 500 millisecondes entre deux incrémentations ;
- La zone représentant la valeur numérique est obtenue par un contrôle de type NumericField.

6.2.2 Choix d'une appréciation avec une boîte de dialogue

Un enseignant en lycée souhaite automatiser l'écriture des appréciations dans les bulletins de notes. Pour cela il crée une boite de dialogue comportant une Zone de liste ou ListBox comme représenté ci-dessous.



Pour ajouter une appréciation dans une feuille du tableur contenant des noms d'élèves et des notes, l'enseignant sélectionne d'abord la cellule de destination. Ensuite, il affiche la boîte de dialogue en cliquant sur un bouton Appréciation. Il sélectionne la phrase qu'il souhaite dans la liste, puis il valide avec le bouton Ok. La phrase sélectionnée est recopiée dans la cellule courante du tableur.

Commencer par dessiner la boîte de dialogue :

- La nommer Appreciation, c'est plus parlant que Boite3 ;
- Ajouter une Zone de liste qui s'appelera Phrases ;
- Ajouter un bouton Ok qui servira à valider la saisie. Dans l'option Type de bouton sélectionner OK, ce qui permettra de fermer la boîte lors de l'appui sur ce bouton, en indiquant que la choix est

réalisé. Vous pouvez également indiquer que ce bouton est activé par défaut en sélectionnant Oui dans l'option Bouton par défaut.

- ▷ Ajouter un bouton Annuler qui servira à quitter la boîte sans valider la saisie. Dans l'option Type de bouton sélectionner Annuler, ce qui permettra de fermer la boîte lors de l'appui sur ce bouton, sans valider le choix.

Le sous-programme Basic associé à la boîte de dialogue, appelé par l'appui sur Appréciation est le suivant :

```
Option Explicit
Global boiteAppreciation As Object ' Objet pour gerer la boite de dialogue

Sub MontrerAppreciation
→   Dim listePhrases As Object ' Objet pour le choix de phrase
→   Dim cellule As Object ' La cellule actuellement selectionnee

→   DialogLibraries.LoadLibrary("Standard")
→   boiteAppreciation = CreateUnoDialog(DialogLibraries.Standard.Appreciation)
→   listePhrases = boiteAppreciation.GetControl("Phrases")
→   REM vous pouvez modifier les phrases suivantes selon votre inspiration
→   listePhrases.additem("Tres bons resultats, continuez", 0)
→   listePhrases.additem("Peut mieux faire", 1)
→   listePhrases.additem("Jamais vu en cours", 2)
→   listePhrases.additem("Je prefere ne rien dire....", 3)
→   If (boiteAppreciation.Execute()) Then
→   →   [redacted]
→   →
→   End If
End Sub
```

- ▷ La méthode `.Selecteditem` de l'objet `listePhrases` permet de récupérer la chaîne de caractères sélectionnée dans la liste ;
- ▷ Compléter le code Basic et créer le bouton Appréciation ;
- ▷ Vérifier le bon fonctionnement de cette méthode de saisie de contenu.

TP 7 : Empreinte carbone (optionnel)

Ce dernier TP utilise uniquement des fonctions qui ont été étudiées dans les séances de travaux pratiques précédentes. Si vous avez fait l'effort d'analyser et comprendre tous les exercices, vous avez acquis suffisamment d'autonomie pour écrire le programme proposé dans ce TP sans l'aide de l'enseignant.

Le thème de cette séance de TP est le calcul de l'empreinte carbone générée par le transport de marchandises. Il est basé sur le tableau donné à la page suivante.

Dans le cadre de ce TP, on s'intéressera uniquement à la masse équivalente de carbone produite par le transport d'une certaine quantité de marchandises sur une distance donnée.

7.1 Etape 1 Bilan carbone par train

La macro 1 doit permettre à l'utilisateur de bâtir le parcours d'une masse M de marchandises à l'aide du train en spécifiant le nombre de kilomètres parcourus. Il faudra construire une boite de dialogue permettant de demander la masse de marchandise à transporter et le nombre de kilomètres à parcourir.

7.2 Etape 2 Bilan carbone par camion

La macro 2 doit permettre à l'utilisateur de bâtir le parcours d'une masse M de marchandises à l'aide du camion en spécifiant le nombre de kilomètres parcourus et le type de camion. Il faudra construire une boite de dialogue permettant de demander la masse de marchandise à transporter, le nombre de kilomètres à parcourir et le type de camion selon un menu déroulant.

7.3 Etape 3 Bilan carbone par avion

La macro 3 doit permettre à l'utilisateur de bâtir le parcours d'une masse M de marchandises à l'aide d'un avion en spécifiant le nombre de kilomètres parcourus et le type d'avion. Il faudra construire une boite de dialogue permettant de demander la masse de marchandise à transporter, le nombre de kilomètres à parcourir et le type d'avion selon un menu déroulant.

7.4 Etape 4 Synthèse

Le détail du parcours sera récapitulé dans la feuille de calcul sous forme de plusieurs lignes indiquant le moyen de transport utilisé , le nombre de km parcourus et le nombre de kg équivalent carbone produit. La dernière ligne donnera le total de kg équivalent carbone produit.

Exemple :

Moyen Transport	Distance (km)	Équivalent carbone (kg)
Camion de 1,5 à 2,5 tonnes essence	105	101.5
Avion AirBus A319	1600	2176.1
Train , Allemagne	250	27.8
Camion de 1,5 à 2,5 tonnes diesel	15	13.7
Total	1970	2119.1

Vous pouvez laisser libre cours à votre imagination pour la présentation de ce programme.