Dauphine | PSL
UNIVERSITÉ PARIS

2019 - 2020

Sciences des organisations
L2 - Semestre 4

UE Y06 - Sciences du digital (Informatique)

Exercices de TP

Lucie GALAND
Florian SIKORA

TP 1 - Excel

Avant de commencer
Créer un dossier intitulé *Info2* dans votre répertoire (*My_Work* ou *H:*, où votre nom apparaît).
Pour chaque TP, prendre l'habitude de créer un dossier dans *Info2*; pour cette séance, créer donc un dossier *TP1*.
Récupérer le fichier *FicheDonnees.xlsx* sur Mycourse et sauvegarder-le dans le dossier *TP1* que vous venez de créer. Ce fichier contient les données du problème traité lors de ce TP, vous travaillerez dans ce fichier.
Remarque : dans ce TP, tous les remplissages de cellules qui demandent des calculs doivent être faits à l'aide des formules Excel, jamais par saisie manuelle.

Le cas d'étude :
Un club cyclo décide d'organiser une journée « promotion du vélo » pendant laquelle les participants auront le choix entre différents types de parcours :
VTT1 : 10 km
VTT2 : 20 km
VTT3 : 30 km
ROUTE1 : 10 km
ROUTE2 : 20 km
ROUTE3 : 40 km
ROUTE4 : 100 km
L'inscription à la journée est payante. Le tarif dépend de la longueur du circuit choisi : 5€ le parcours découverte (longueur inférieure à 25 km) et 10€ le parcours sportif (longueur supérieure à 25 km). Ces données constituent les paramètres de notre problème, ils se trouvent dans la feuille *Parametres* (onglet *Parametres*) du fichier *FicheDonnees.xlsx*. La feuille *Donnees* (onglet *Donnees*) du fichier *FicheDonnees.xlsx* contient les données sur les performances des participants; on y enregistre le nom du participant, le type de parcours suivi, son heure de départ et son heure d'arrivée.

Q1 Nommage de cellules. Dans l'onglet *Parametres*, cliquer sur la cellule B17, *limite_km* apparaît alors dans la case en haut à gauche d'Excel (voir la figure 1). Cela signifie que la cellule B17 est nommée

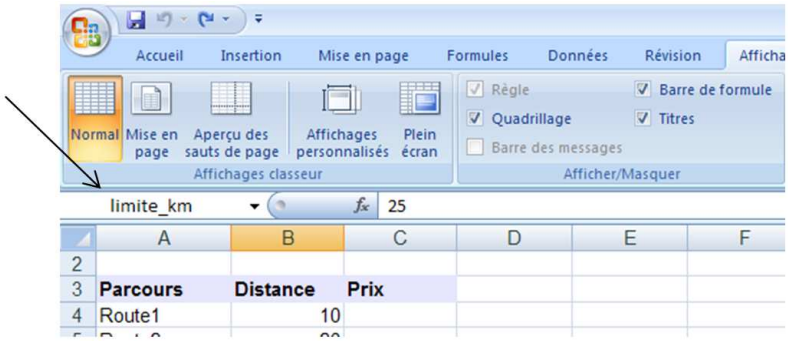

FIGURE 1 – Nom de cellules

limite_km et que ce nom peut être utilisé dans les formules Excel comme une référence absolue à la valeur contenue dans la cellule B17 (i.e. tout comme *B17*). Le renommage de cellule permet d'avoir des formules lisibles sans se soucier des $ pour les références absolues.
Vérifier aussi les cellules B14 et B15.

Q2 Fonction SI. Remplir la colonne *Prix* en utilisant la fonction *SI* d'Excel et les noms des cellules B14, B15 et B17.
Rappel : Insérer une fonction dans Excel : Onglet Formule/Insérer une fonction, ou saisie de "=SI" dans la cellule correspondante.

Q3 Nommage de plages de cellules. Se placer dans l'onglet *Donnees*, sélectionner les cellules E4-E18 et remarquer que l'on peut aussi nommer une plage de cellules. Vérifier aussi les autres colonnes de ce tableau. Nommer la plage B4-B18 *circuit*. Pour cela, il suffit de sélectionner la plage de cellules et d'écrire le nom dans la case en haut à gauche de l'écran.

Q4 Format de cellule. Remplir maintenant la colonne *duree* à l'aide d'une formule Excel et en utilisant les noms des colonnes. Pour que la durée soit affichée sous forme d'heure, sélectionner la plage de cellules, aller dans *Accueil/Cellules/Format/Format de cellule...* (ou clic droit *Format de cellule*) et choisir *Heure*. Ne pas hésiter à élargir la colonne si nécessaire.

Q5 Fonction RECHERCHE. Remplir les colonnes *tarif* et *distance* en utilisant la fonction Excel *RECHERCHE* (saisie de "=RECHERCHE" ou onglet *Formules/Insérer une fonction*, catégorie : Recherche & Matrices). Mettre le tarif au format monétaire (onglet *Accueil/Nombre*).

Q6 Fonctions HEURE, MINUTE, SECONDE. Remplir finalement la vitesse moyenne de chaque cycliste dans la colonne *vitesse moyenne*. Si le format de la colonne durée a bien été modifié en heure, il est possible d'utiliser les fonctions Excel *HEURE, MINUTE, SECONDE* pour faire ce calcul. Veiller à avoir un affichage au format *Nombre* dans la colonne *vitesse moyenne*.

Q7 Filtres automatiques. Le but de cette question est de créer des filtres automatiques sur les titres des tableaux. Pour cela, sélectionner les titres du tableau des données (A3-H3) et activer un filtre automatique (*Accueil/Édition/Trier et filtrer/Filtrer* ou *Données/Trier filtrer/Filtrer*). Les flèches qui apparaissent alors sur les titres permettent de filtrer les données et de n'afficher que celles qui satisfont un critère de sélection. Utiliser ces filtres pour n'afficher que les données correspondant

- au prix le moins élevé (désactiver ensuite le filtre) ;
- aux longues distances (supérieures à 35km). Pour cela, utiliser le *Filtre numérique*. Désactiver ensuite le filtre.

Q8 Fonctions SI imbriquées Ajouter maintenant une nouvelle colonne intitulé *rapidite* qui dira si le cycliste est lent (vitesse moyenne<=10), moyen ou rapide (vitesse moyenne>20). Remplir cette colonne à l'aide d'une formule Excel.

Q9 Formatage conditionnel. Colorier **d'une manière automatique** cette colonne (lent : gris, moyen : orange, rapide : rouge). Pour cela utiliser le formatage conditionnel (Accueil/Style/Mise en forme conditionnelle).

Q10 Fonction NB.SI. Remplir la colonne *Nombre de participants* en utilisant la fonction Excel *NB.SI*.

Q11 Fonctions SOMME et SOMME.SI. Préciser dans les cellules correspondantes les bénéfices du club en utilisant la fonction Excel *SOMME.SI* et *SOMME* (pour le bénéfice global).

Q12 Tableau croisé dynamique. Les tableaux croisés dynamiques (TCD) permettent de synthétiser des informations de manière dynamique, c'est-à-dire que les modifications (ajouts, suppressions, mises à jour) faites sur les données liées au tableau sont directement prises en compte dans le tableau de synthèse.

Le premier TCD à faire est simple, il permet de synthétiser les données par circuit : pour chaque circuit il faudrait afficher le nombre de participants et le bénéfice (somme des montants payés). Le TCD à obtenir est le suivant :

Étiquettes de lignes	Valeurs nombre de participants	Bénéfice
ROUTE1	1	5
ROUTE2	4	20
ROUTE3	1	10
ROUTE4	2	20
VTT1	3	15
VTT2	2	10
VTT3	2	20
Total général	**15**	**100**

FIGURE 2 – TCD simple

Pour cela, sélectionner les plages de cellules où se trouvent les données nécessaires (y compris les titres des colonnes qui seront utiles) et cliquer sur *Insertion/Tableaux/TblCroiséDynamique*. Une nouvelle fenêtre à droite de la feuille devrait apparaître (cf figure 3) Choisir les titres à mettre en lignes (*Circuit* ici) et

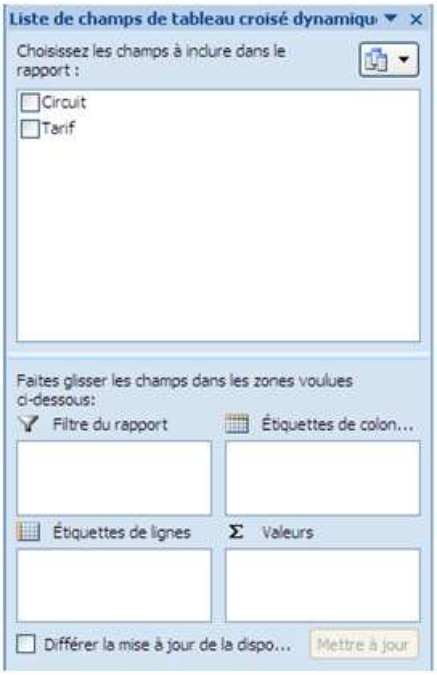

FIGURE 3 – Fenêtre Liste des Champs

glisser-les dans la case Étiquettes de lignes. Il faut maintenant définir les deux actions à effectuer sur ces données (somme des montants et nombre). Pour cela glisser *Tarif* dans la case Valeurs (case en bas à droite). La fonction proposée par défaut est la somme. Comme nous voulons mettre d'abord le nombre de participants, changer cette fonction en cliquant sur la petite flèche et en choisissant dans *Paramètres des champs de valeur* la fonction *Nombre*. Il est maintenant possible de mettre la somme en glissant à nouveau *Tarif* dans cette case.

Bien observer que chaque choix fait a tout de suite des effets sur le tableau en train d'être créé (ce qui peut aider à comprendre comment choisir les bons éléments).
Il est possible de faire des modifications sur le tableau résultant. Modifier les titres des colonnes pour obtenir le même tableau que celui de la figure 2. Pour cela, cliquer droit sur la cellule appropriée ou modifier directement sa valeur dans le champ en haut à droite.
Une fois le TCD fait, modifier le prix découverte et mettre 6€ dans la feuille *parametres*. Bien observer que cette modification est suivie dans les données de la page *Donnees* (colonne prix, bénéfice, etc.) et dans le TCD. Si ce n'est pas le cas, cliquer sur *Actualiser tout* dans *Données* de la barre d'outils.

Q13 TCD suite. Essayer d'obtenir maintenant le tableau de la figure 4.

Tarif	Distance	Circuit	Nom	vitesse_moyenne	Total
5,00 €	**10**	ROUTE1	**Toto**	9,23	9,230769231
		Moyenne ROUTE1			9,230769231
		VTT1	**Anna**	18,18	18,18181818
			Bob	10,00	10
			Jean	10,00	10
		Moyenne VTT1			12,72727273
	Max 10				**18,18181818**
	20	ROUTE2	**Dede**	20,16	20,15677492
			Dupont	18,20	18,20020222
			Le chien	8,00	8
			Zork	20,00	20
		Moyenne ROUTE2			16,58924429
		VTT2	**Paf**	12,00	12
			Zoe	12,00	12
		Moyenne VTT2			12
	Max 20				**20,15677492**
10,00 €	**30**	VTT3	**Fabrice**	11,61	11,61290323
			Titi	10,00	10
		Moyenne VTT3			10,80645161
	Max 30				**11,61290323**
	40	ROUTE3	**Joe**	26,67	26,66666667
		Moyenne ROUTE3			26,66666667
	Max 40				**26,66666667**
	100	ROUTE4	**Cecile**	19,93	19,93355482
			Jacques	33,33	33,33333333
		Moyenne ROUTE4			26,63344408
	Max 100				**33,33333333**
Total général					**15,95440151**

FIGURE 4 – Tableau croisé dynamique

TP 2 - Premiers pas en VBA

Exercice 1 : Environnement de travail

Le but de cet exercice est de configurer votre environnement de travail. Vous trouverez dans la *Fiche Technique* toutes les informations nécessaires à une configuration convenable de votre environnement de travail, qui sera valable pour tout le semestre.

Q1.1 Lancer Excel en suivant les instructions de la section 1.1 de la fiche technique.

Q1.2 En suivant les instructions de la section 1.2 de la fiche technique, configurer l'onglet *Développeur*, puis passer en type de référence *L1C1*.

Q1.3 Suivre ensuite les instructions de la section 1.3 pour configurer l'environnement de programmation : lancer et configurer l'éditeur VBE, puis créer un module.
Vous pouvez maintenant écrire une première macro (exercice suivant).

Exercice 2 : Première macro

Le but de cet exercice est d'écrire et d'exécuter une première macro. Vous trouverez dans la *Fiche Technique* toutes les informations nécessaires à l'exécution d'une macro.

Q2.1 Écrire dans un module la macro `bonjour` suivante :

```
Sub bonjour()
        MsgBox "bonjour"
End Sub
```

Q2.2 À l'aide de la section 2.2 de la fiche technique, enregistrer votre programme dans un fichier nommé *tp1.xlsm*.

Q2.3 Exécuter la macro `bonjour` (cf section 2.3 de la fiche technique).

Q2.4 À l'aide de la section 3 de la fiche technique, créer un bouton dans la feuille Excel pour pouvoir exécuter la macro `bonjour`. Vérifier son fonctionnement. Renommer le bouton "Bonjour".

Q2.5 Créer un deuxième bouton dans la feuille Excel, nommé "Au revoir", et affecter-lui une nouvelle macro qui affiche le message "Au revoir".

Exercice 3 : Variables VBA

Un étudiant de LSO2 a écrit la macro `macro2` suivante. Cette macro contient-elle des erreurs ?

```
Sub macro2()
Dim a As Integer, b As Double, c As Byte
a = 20: b = 3.14
MsgBox "a = " a ", b = " b
```

```
c = 512
MsgBox a fois c vaut & a*c
a = 37.2 + b
MsgBox "a = " a
b = "la valeur de pi est " & b
MsBox b
d = (a ^ 5) * c - 65
"d vaut " & d
End Sub
```

Recopier la macro et l'exécuter après avoir corrigé les éventuelles erreurs.

Exercice 4 : Comparaisons

Q4.1 Écrire une macro VBA qui demande à l'utilisateur de saisir trois entiers et qui affiche le plus petit des trois.

Q4.2 Écrire une macro VBA qui demande à l'utilisateur de saisir deux entiers et qui affiche celui des deux qui est le plus proche de 50. Par exemple, si l'utilisateur saisit 45 et 49 la macro doit afficher 49, et s'il saisit 25 et 65 la macro doit afficher 65.

Exercice 5 : Calculatrice

Écrire une macro VBA `calc` qui demande à l'utilisateur de saisir deux nombres `a` et `b` et un caractère `op` correspondant à un opérateur (`+`, `-`, `*`, `/`), et affiche le résultat du calcul de `a op b`. Sous Excel, créer un bouton nommé "Calculatrice" et affecter-lui la macro `calc`.

Exercice 6 : Chiffres identiques

Écrire une macro VBA qui demande à l'utilisateur de saisir un nombre compris entre 100 et 999 (inclus) et qui affiche "2 chiffres identiques" si le nombre saisi contient 2 chiffres identiques (comme par exemple 636 ou 224), "3 chiffres identiques" s'il en contient 3 (comme par exemple 111 ou 555), et "pas de chiffres identiques" sinon.

TP 3 - Boucles

Exercice 1 : Somme de termes

Q1.1 Écrire en VBA une macro qui demande à l'utilisateur de saisir deux entiers et qui affiche la somme des nombres compris entre ces deux nombres, les deux nombres inclus. Attention, le premier entier saisi par l'utilisateur peut être plus grand que le second.

Q1.2 Modifier la macro précédente pour qu'elle ne fasse la somme que des termes pairs. Il est demandé de n'effectuer qu'un seul test de parité dans toute la macro.

Exercice 2 : Sphinx

Écrire une macro VBA permettant de jouer au jeu suivant : un premier joueur choisit un nombre positif et l'indique à l'ordinateur à l'aide d'une boîte de dialogue (`Application.InputBox`). Un deuxième joueur essaie de deviner ce nombre en proposant des nombres (toujours avec `Application.InputBox`). A chaque tentative le programme indique si le nombre à trouver est plus petit ou plus grand que le nombre du deuxième joueur. La partie s'achève lorsque le deuxième joueur trouve la bonne solution. Le programme affiche alors :

Bravo ! Le nombre est bien ... Vous l'avez trouvé en ... coups.

Exercice 3 : Masterspirit

Le but de cet exercice est d'écrire un programme qui simule le jeu entre deux personnes suivant :

- une première personne retient un nombre compris entre 100 et 999. Le programme demande à cette personne son nombre par un `Application.InputBox`.
- La deuxième personne essaie de deviner ce nombre. Elle précise un nombre avec `Application.InputBox`.

Attention : pour simplifier l'exercice, on considère que les deux joueurs ne donnent que des nombres avec des chiffres différents (aucun d'eux ne va dire 252 par exemple) et on fait l'hypothèse que les joueurs respectent nos contraintes sur les nombres, donc pas besoin de vérification sur la validité du nombre.

Q3.1 Dans un premier temps, écrire la partie du programme qui demande aux deux joueurs leur nombre.

Q3.2 Le programme va dire maintenant avec `MsgBox` au deuxième joueur combien de chiffres sont bien placés et combien de chiffres existent dans le premier nombre mais ne sont pas à la bonne place. Par exemple :
le nombre du premier joueur est 123
le nombre du deuxième joueur est 152 : « il y a 1 chiffre bien placé et 1 chiffre mal placé » (car 1 est au bon endroit et 2 n'est pas à la bonne position).
Écrire la suite du programme qui indique au deuxième joueur combien de chiffres sont bien placés et combien de chiffres existent dans le premier mais sont en mauvaise position. Pour l'instant le deuxième joueur ne fait qu'un seul essai.

Q3.3 Compléter maintenant le programme pour qu'il demande au deuxième joueur de donner un nombre jusqu'à ce qu'il trouve le nombre du premier joueur. C'est à dire :

- si les 3 chiffres sont au bon endroit le programme s'arrête en affichant « bravo vous avez trouvé, le nombre est ... »
- sinon le programme demande un autre nombre au deuxième joueur jusqu'à ce qu'il trouve le bon nombre.

Exemple : le nombre du premier joueur est 123
Essai 1 du deuxième joueur : 456 : « il y a 0 chiffre bien placé et 0 chiffre mal placé »
Essai 2 du deuxième joueur : 912 : « il y a 0 chiffre bien placé et 2 chiffres mal placés »
Essai 3 du deuxième joueur : 152 : « il y a 1 chiffre bien placé et 1 chiffre mal placé »
Essai 4 du deuxième joueur : 129 : « il y a 2 chiffres bien placés et 0 chiffre mal placé »
Essai 5 du deuxième joueur : 123 : « il y a 3 chiffres bien placés et 0 chiffre mal placé. Bravo vous avez trouvé, le nombre est 123 ».

TP 4 - Fonctions et programmes VBA

Exercice 1 : Âge

Écrire une macro VBA qui demande à l'utilisateur de saisir son nom, son année de naissance et l'année en cours. Puis, à l'aide des boutons de `Msgbox`, la macro demande à l'utilisateur si son anniversaire est déjà passé cette année. Enfin, la macro affiche « Bonjour ..., vous avez ... ans » en indiquant le nom et l'âge de l'utilisateur à la place des points. La macro doit vérifier que le type des données saisies par l'utilisateur sont corrects (chaîne de caractères pour le nom, entiers pour les années).

Exercice 2 : Volume d'une sphère

Q2.1 Écrire une fonction VBA qui détermine le volume d'une sphère à partir de son rayon.

Pi= 3.14 sera declarée en constante. Pour déterminer le volume d'une sphère, on peut passer par le calcul de sa surface ($s = 4\Pi r^2$), le volume étant egal à $rs/3$.

Q2.2 Écrire une macro de test de cette fonction qui demande à l'utilisateur de saisir un rayon. Donner un titre approprié à la boîte de dialogue et placer une valeur par défaut dans la zone de texte.

Q2.3 Lorsqu'une fonction est définie en VBA, elle peut aussi être utilisée comme fonction sous Excel. Écrire la formule Excel d'appel de cette fonction dans la cellule L1C2 pour le calcul du volume d'une sphère dont le rayon est indiqué dans la cellule L1C1.

Exercice 3 : Conjecture de Syracuse

L'algorithme de Syracuse consiste à itérer l'opération suivante : à un nombre entier n, on associe $n/2$ si n est pair et $3n + 1$ si n est impair. On conjecture que quelque soit l'entier considéré initialement dans cet algorithme, on arrive toujours à 1 après un certain nombre d'itérations.

Q3.1 Écrire une fonction VBA `svt` qui étant donné un entier `n`, retourne le nombre associé à `n` dans l'algorithme de Syracuse. Par exemple, `svt(5)` retourne 16 et `svt(10)` retourne 5.

Q3.2 Écrire une procédure VBA qui prend en paramètre un entier `n` et qui affiche au fur et à mesure les nombres obtenus par l'application de l'algorithme de Syracuse jusqu'au premier 1, puis affiche le nombre d'itérations effectuées jusqu'à l'obtention du premier 1.

Q3.3 Écrire une macro VBA qui demande à l'utilisateur un entier `n`, puis appelle la procédure précédente. Donner un titre approprié à la boîte de dialogue et vérifier que la valeur saisie par l'utilisateur est bien de type numérique. Tester sur différentes valeurs (par exemple 7).

Exercice 4 : Conjecture de Goldbach

La conjecture de Goldbach est la suivante : « tout nombre pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers ». Par exemple, 8 = 3 + 5, 12 = 5 + 7, 18 = 5 + 13 (ou 7 + 11), 24

$= 5 + 19$ (ou $7 + 17$ ou $11 + 13$).

Q4.1 Écrire en VBA une fonction qui prend un entier `n` en paramètre et retourne vrai si `n` est premier, et faux sinon.

Q4.2 Écrire en VBA une macro qui demande à l'utilisateur de saisir un nombre pair supérieur à 2, puis affiche deux nombres premiers dont la somme vaut ce nombre. Tant que le nombre saisi par l'utilisateur n'est pas correct (pair et supérieur à 2), la macro redemande à l'utilisateur de saisir un nombre. Sous Excel, associer un bouton à cette macro.

Exercice 5 : Jeu des 17 allumettes

Ce jeu se joue à deux. Les joueurs sont devant 17 allumettes. Chaque joueur prend à tour de rôle 1, 2 ou 3 allumettes. Celui qui prend la dernière allumette a perdu.

Q5.1 Écrire en VBA une procédure `jeu17` qui prend en paramètre deux chaînes de caractères correspondant aux noms des deux joueurs et qui simule le jeu des 17 allumettes. La procédure affiche le nom du vainqueur à la fin de la partie. On suppose que les joueurs sont honnêtes (ils prennent bien à chaque fois 1, 2 ou 3 allumettes) et qu'ils veulent gagner, ainsi s'il reste 2, 3 ou 4 allumettes le joueur dont c'est le tour va en prendre 1, 2 ou 3 (respectivement) afin de gagner.

Q5.2 Écrire une macro de test de la procédure `jeu17` dans laquelle on demande le nom des deux joueurs.

Q5.3 Modifier le programme pour déterminer aussi le nombre de coups (prises d'allumettes) du vainqueur.

TP 5 - Cellules Excel

Exercice 1 : Échiquier

Le but de cet exercice est de dessiner un échiquier noir-rouge à partir d'une cellule dont les coordonnées sont données en paramètres d'entrée.

Q1.1 Écrire une procédure, `Echiquier10x10`, qui prend en paramètres d'entrée deux entiers `x` et `y`, et qui dessine un échiquier noir-rouge de taille 10×10 à partir de la case (`x`, `y`) (coin supérieur gauche de l'échiquier). On appliquera la règle de coloriage suivante : la première case est noire et la couleur d'une case (i, j) ne dépend que de la parité de $i + j$.

Remarque : pour colorier la cellule (i, j) la commande est :

```
Cells(i, j).Interior.Color = RGB(0, 0, 0) 'Noir
Cells(i, j).Interior.Color = RGB(200, 0, 0) 'Rouge
```

Écrire une macro de test de cette procédure.

Q1.2 Écrire maintenant une nouvelle procédure `echiquierNxN` qui est similaire à la précédente sauf que le nombre de cases est aussi entré en paramètre.

Exercice 2 : Dessin d'une pyramide

Q2.1 Écrire une procédure `colorierSegment(x As Integer, l As Integer, m As Integer)` qui colorie un segment horizontal de taille `x` à partir de la cellule `(l,m)`.

Q2.2 En utilisant `colorierSegment`, écrire une procédure `colorierPyramide(lSommet as Integer, cSommet as Integer, n as Integer)` qui dessine une pyramide de hauteur `n` dont le sommet est la cellule `(lSommet, cSommet)`.

TP6 - Indice de Gini

Un moyen d'évaluer le degré d'inégalité de la répartition des revenus (ou autres) au sein d'une société est le calcul de l'indice de Gini. Cet indice permet de mesurer l'écart à une distribution strictement égalitaire des revenus pour une répartition donnée. L'indice donne une valeur entre 0 (répartition égalitaire) et 1 (répartition la plus inégalitaire). Pour ce faire on compare les courbes représentant les pourcentages de revenus cumulés détenus par les individus considérés (courbes de Lorenz). Techniquement il s'agit de comparer l'aire comprise entre ces courbes. On va ici considérer des groupes d'individus (les quintiles), et ainsi approximer l'indice de Gini pour ces données. Le but de ce TP est de s'exercer à utiliser des fonctions et des procédures de VBA, l'indice que l'on calcule reste une approximation de l'indice de Gini. Prenons un exemple avec les données suivantes d'un pays imaginaire :

$$L = \{0.042, 0.111, 0.175, 0.244, 0.428\}$$

Ces données signifient que par exemple les 20% plus pauvres du pays considéré n'ont que 4.2% des revenus totaux tandis que les 20% plus riches ont 42.8% des revenus totaux. On voit bien que les pourcentages sont non-décroissants et leur somme fait bien 1. Si on travaille avec des pourcentages cumulés on obtiendra :

$$Lor = \{0.042, 0.153, 0.328, 0.572, 1\}$$

Dans un pays égalitaire où les revenus sont distribués d'une manière égale entre tous les individus on aura la distribution cumulée suivante :

$$Lor\_égalitaire = \{0.2, 0.4, 0.6, 0.8, 1\}$$

L'indice de Gini est calculé comme étant le double de la différence d'aire entre les courbes de *Lor* et de *Lor_égalitaire*. L'approximation de cet indice sera faite de la manière suivante : on estime d'abord l'aire comprise sous la courbe définie par les données comme la moyenne d'une sous-estimation et d'une sur-estimation de cette aire déterminée par la formule :

$$Aire\_estimee = \frac{\sum_{i=1}^{4} 0.2 \times Lor(i) + \sum_{i=1}^{5} 0.2 \times Lor(i)}{2}$$

Puis on compare **le double** de cette aire estimée à la valeur 1 (double de l'aire sous la courbe de *Lor_égalitaire*). Pour le pays imaginaire donné en exemple, on obtient la valeur 0.36 comme approximation de l'indice de Gini. Pour des données réelles, on pourra consulter l'adresse suivante : `https://fr.wikipedia.org/wiki/Liste_des_pays_par_%C3%A9galit%C3%A9_de_revenus`. Une vidéo de vulgarisation sur le sujet (à consulter en dehors de la séance de TP) : `https://youtu.be/3QqR3AQe-SU`

Pour ce TP vous allez utiliser le fichier IndiceGini.xlsm où vous avez quelques données sur quelques pays (imaginaires ou non). Vous trouverez le fichier sur MyCourse. Copiez-le dans votre répertoire de travail afin de pouvoir travailler avec ce fichier.

Q1 Écrire en VBA une fonction `LignePays` qui prend en entrée le nom d'un pays et qui renvoie le numéro de ligne du pays concerné dans le fichier. La fonction `LignePays` retourne 0 si le pays n'apparaît pas dans le fichier Excel.

Q2 Écrire en VBA une fonction `Verification` qui prend en entrée le nom d'un pays et qui renvoie vrai si les pourcentages des revenus de ce pays sont cohérents (dans l'ordre non décroissant et dont la somme est 1) et faux sinon. On pourra utiliser la fonction `LignePays`.

Q3 Écrire en VBA une procédure `Remplir_Verification` qui prend en entrée le nom d'un pays et qui remplit la case de vérification de ce pays (1 si la vérification est bonne et 0 sinon) en utilisant la fonction `Verification`. On suppose ici que le pays passé en paramètre apparaît bien dans le fichier Excel.

Q4 Écrire en VBA une procédure `Remplir_Lorenz` qui prend en entrée le nom d'un pays et qui remplit les cellules correspondant aux termes de la courbe de Lorenz (pourcentages cumulés). On suppose ici que le pays passé en paramètre apparaît bien dans le fichier Excel.

Q5 Écrire en VBA une fonction `indice_Gini` qui prend en entrée le nom d'un pays et qui renvoie la valeur de l'indice de Gini de ce pays (ici on fait l'hypothèse que les termes de Lorenz sont déjà calculés).

Q6 Écrire en VBA une procédure `remplir_Gini` qui prend en paramètre un numéro de ligne `i` et qui remplit la cellule LiC13 **par une formule Excel** faisant appel à la fonction `Calcul_Gini`. On demande bien ici de remplir la cellule avec une **formule Excel** (utiliser `FormulaR1C1Local`) et non une valeur. Ainsi, une fois la formule Excel présente en LiC13, la valeur figurant en LiC13 sera automatiquement mise à jour lorsque les données de la ligne `i` changent. Vérifier (si besoin, double-cliquer sur la cellule LiC13, puis taper sur `entrée`).

Q7 Écrire finalement une macro VBA qui demande le nom d'un pays et qui fait la suite des actions suivantes avec les procédures et fonctions créées précédemment :

- demander à l'utilisateur le nom du pays. Si le pays n'existe pas la macro doit afficher un message d'erreur et redemander un autre pays jusqu'à ce que l'utilisateur entre un pays existant. Le message d'erreur doit être le suivant :

  « *ce pays n'est pas répertorié, essayer un autre pays* »
- faire la vérification,
- remplir la case de la vérification,
- si la vérification n'est pas bonne, arrêter la procédure en affichant le message :

  « *Les pourcentages de revenus de ce pays sont erronés, veuillez les modifier.* »
- si la vérification est bonne, calculer les termes de Lorenz et remplir les cellules correspondantes, et calculer l'indice de Gini et remplir la cellule correspondante.

Modifier ensuite cette macro pour qu'elle demande à l'utilisateur s'il veut continuer ou non à la fin des calculs d'un pays avec le message suivant :

« *voulez vous continuez ? (oui/non)* »

et pour qu'elle reprenne tout depuis le début si la réponse est « oui ». Utiliser pour cela les boutons de `MsgBox`.

TP7 - Représentation binaire des nombres

Exercice 1 : Conversion d'un entier positif en base 2

Dans cet exercice, on suppose qu'un nombre binaire est stocké sur la première ligne de la feuille Excel en mettant un chiffre par cellule sur 16 bits. Ainsi le nombre binaire 10010 par exemple est stocké dans la feuille Excel en mettant 1 dans les cellules L1C12 et L1C15, 0 dans les cellules L1C13, L1C14 et L1C16, et 0 ou rien des les cellules de la plage L1C1 :L1C11.

Q1.1 Écrire une procédure `decToBin` convertissant un décimal en binaire. Cette procédure prend en paramètre le nombre entier à convertir `n` et écrit le nombre binaire obtenu dans la feuille Excel sur la première ligne de la colonne 1 à la colonne 16. Tester la procédure.

Q1.2 Écrire une fonction `BinToDec` qui retourne la valeur décimale du nombre binaire représenté à la première ligne de la feuille Excel (colonnes 1 à 16). Tester la fonction.

Q1.3 Écrire une macro `verification` qui demande à l'utilisateur de saisir un nombre compris entre 0 et $2^{16} - 1$, convertit ce nombre en binaire en le stockant dans la feuille de calcul, convertit le nombre binaire obtenu en décimal, et enfin vérifie que l'entier décimal obtenu est bien le même que celui saisi par l'utilisateur.

Exercice 2 : Calcul binaire sur des entiers positifs

On suppose dans cet exercice que plusieurs nombres binaires sont représentés sur 16 bits à raison d'un chiffre par colonne (comme dans l'exercice précédent) sur les premières lignes de la feuille Excel (un nombre par ligne).

Q2.1 Écrire une procédure en VBA qui prend en paramètre un numéro de ligne et qui multiplie par 2 le nombre binaire situé à cette ligne. Cette multiplication doit être effectuée sans passer par une conversion en base 10.

Q2.2 Tester la procédure précédente en vérifiant le calcul à l'aide de la fonction `BinToDec` écrite précédemment.

Q2.3 Écrire une procédure en VBA qui prend en paramètre un numéro de ligne `li` et qui multiplie par 8 le nombre binaire situé à cette ligne. Cette multiplication doit être effectuée sans passer par une conversion en base 10. Tester la procédure

Q2.4 Écrire une procédure `addition` qui prend en paramètre trois numéros de ligne `l1`, `l2` et `l3` et place à la ligne `l3` le résultat de l'addition du nombre représenté à la ligne `l1` avec celui représenté à la ligne `l2`. Si le résultat nécessite plus de 16 bits, alors `addition` affiche un message d'erreur.

Exercice 3 : Base 5

Reprendre le premier exercice en considérant que les nombres de la feuille Excel sont maintenant représentés en base 5 et non plus en base 2, toujours avec 16 chiffres au maximum.

TP9 - Recherche d'un élément

Dans ce TP, on s'intéresse à la recherche d'un mot dans une liste de mots stockés dans une feuille Excel. On utilisera pour cela le fichier **listeMots.xlsm**, disponible sur MyCourse. Ce fichier est un extrait d'un des dictionnaires proposés sur le site `http://infolingu.univ-mlv.fr/`. Il contient une liste de mots en ordre alphabétique dans la première colonne de la feuille Excel, de la ligne 1 à la ligne 758983. Copiez le fichier dans votre répertoire de travail avant de l'utiliser.

Q1 Écrire une fonction `RechercheSq` qui prend en paramètre un mot (chaînes de caractères) et qui le cherche dans la liste des mots du fichier **listeMots.xlsm**. La recherche sera séquentielle, c'est-à-dire que le mot recherché va être comparé au premier mot de la liste (celui à la ligne 1), si ce n'est pas le même, il sera comparé au suivant (celui à la ligne 2) et ainsi de suite. La fonction `RechercheSq` retourne le numéro de la ligne à laquelle se trouve le mot recherché, ou 0 si le mot n'est pas dans la liste.

Q2 Tester la fontion `RechercheSq` avec les mots : "bonjour", "informatique", "train" et "vba".

Q3 Écrire une fonction `RechercheDicho` qui recherche elle aussi un mot `m` passé en paramètre, mais en procédant différemment. Puisque les mots du fichier Excel sont triés dans l'ordre alphabétique, elle va d'abord chercher `m` au milieu de la plage de cellules (c.à.d à la ligne 379491 car 758983/2 = 379491.5), si `m` est différent du mot à la ligne 379491, et s'il est plus grand (resp. plus petit), elle va chercher `m` dans la plage de cellules L379492C1:L758983C1 (resp. L1C1:L379490C1). Pour cela, `m` est comparé au mot situé au milieu de la sous-plage concernée et ainsi de suite...

Q4 Tester la fonction `RechercheDicho` avec les mêmes mots que précédemment. Y a-t-il une différence dans les temps d'exécution des deux fonctions ? Si oui, laquelle est plus rapide et pourquoi ?
Pour s'en assurer, on peut écrire une procédure `test` qui exécute successivement chacune des deux méthodes pour la recherche d'un mot passé en paramètre, en mesurant leur temps d'exécution, puis affiche les deux temps d'exécution obtenus. Pour mesurer le temps d'exécution des méthodes, on pourra utiliser la fonction `Timer` de VBA qui retourne le temps système courant. Ainsi pour mesurer le temps d'exécution d'une méthode `execMethode`, il suffit d'écrire les instructions suivantes :

```
Dim debut As Single, fin As Single
debut = Timer
execMethode
fin = Timer
```

Le calcul `fin - debut` permet de connaître le temps d'exécution de `execMethode`.

TP10 - Tri d'éléments

Dans ce TP, nous allons comparer différentes méthodes de tri. Pour cela, nous allons considérer le problème de tri des données situées dans la première colonne de la feuille de calcul Excel.

Q1 Écrire la macro `generation` qui remplit les cellules de la colonne L1C1:L500C1 par 500 nombres compris entre 1 et 10000 générés aléatoirement. On pourra utiliser pour cela la fonction `Rnd` de VBA qui génère un nombre aléatoire compris entre 0 et 1. Pour générer un entier compris entre a et b il suffit donc d'exécuter l'instruction :

```
Int((b - a + 1)* Rnd() + a)
```

Exécuter la macro `generation` pour remplir la colonne 1.

Q2 Écrire la macro `TriSelect` qui place les éléments de la première colonne de la feuille de calcul dans la seconde colonne dans l'ordre croissant. Pour cela, `TriSelect` copie les éléments de L1C1:L500C1 dans L1C2:L500C2, puis applique l'algorithme de tri par sélection pour trier les éléments de la deuxième colonne.

Q3 Écrire une fonction `verifTri` qui prend en paramètre un numéro de colonne `j` et retourne vrai si les éléments de la colonne `j`, de la ligne 1 à la ligne 500, sont triés en ordre croissant, et faux sinon. Puis, écrire une macro qui vérifie que le tri réalisé par `TriSelect` est correct en utilisant `verifTri`.

Q4 Écrire la macro `TriBulle` qui place les éléments de la première colonne de la feuille de calcul dans la troisième colonne dans l'ordre croissant. Pour cela, `TriBulle` copie les éléments de L1C1:L500C1 dans L1C3:L500C3, puis applique l'algorithme de tri à bulles pour trier les éléments de la troisième colonne.

Q5 Écrire la macro `verifieTris` qui affiche un message d'erreur si le résultat de l'exécution de `TriSelect` et `TriBulle` n'est pas le même ou si les éléments ne sont pas triés en ordre croissant. Tester la macro `verifieTris`, et corriger si nécessaire.

Q6 On s'intéresse maintenant à la comparaison du temps d'exécution des ces deux méthodes de tri. Pour cela écrire une macro `Test` qui exécute successivement chacune des deux méthodes en mesurant leur temps d'exécution. La macro `Test` affiche ensuite les deux temps d'exécution obtenus. Pour mesurer le temps d'exécution des méthodes, on pourra utiliser la fonction `Timer` de VBA qui retourne le temps système courant. Ainsi pour mesurer le temps d'exécution d'une méthode `execMethode`, il suffit d'écrire les instructions suivantes :

```
Dim debut As Single, fin As Single
debut = Timer
execMethode
fin = Timer
```

Le calcul `fin - debut` permet de connaître le temps d'exécution de `execMethode`. Quelle est la méthode de tri la plus efficace ? Pourriez-vous expliquer pourquoi ? Tester aussi avec des données initialement triées ou quasiment triées, et des données initialement triées en ordre inverse.

TP11 - Récursivité

Exercice 1 : Affichage ordonné

Q1.1 Que fait la procédure `affichage` ? Vérifier en écrivant une macro VBA qui l'appelle.

```
Sub affichage(n As Integer)
    If n >= 0 Then
        MsgBox n
        affichage n - 1
    End If
End Sub
```

Q1.2 Modifier la procédure `affichage` pour qu'elle effectue les mêmes affichages successifs mais dans l'ordre inverse.

Exercice 2 : Conjecture de Syracuse

Soit la fonction s suivante :

$$s(n) = \begin{cases} 1 & \text{si } n = 0 \text{ ou } 1 \\ s(\frac{n}{2}) & \text{si } n \text{ est pair} \\ s(3n+1) & \text{sinon} \end{cases}$$

Si la conjecture de Syracuse est vérifiée, alors $s(n) = 1$ quel que soit n positif.

Q2.1 Écrire en VBA une fonction récursive `syra` qui prend un entier n en paramètre et qui retourne $s(n)$. On suppose que n est positif ou nul. Vérifier la conjecture de Syracuse.

Q2.2 Écrire une version itérative du calcul de $s(n)$ (cf exercice 3 du TP4). Vérifier que les résultats obtenus sont bien les mêmes pour la version itérative et pour la version récursive en affichant les valeurs intermédiaires.

Exercice 3 : Puissance

Q3.1 Écrire en VBA une fonction récursive `puiss` qui prend deux entiers x et n en paramètre et qui retourne x^n, sans utiliser l'opérateur $\wedge$. On suppose que n est positif ou nul.

Q3.2 Tester la fonction VBA `puiss` sous Excel, et comparer les résultats obtenus à ceux donnés par la fonction `PUISSANCE` d'Excel.

Q3.3 Écrire une macro VBA qui demande à l'utilisateur de saisir deux entiers (types numériques) x et n, et qui affiche la valeur x^n et le nombre d'appels récursifs effectué par `puiss`. Pour compter le nombre d'appels récursifs, on peut utiliser une variable globale, incrémentée à chaque appel, ou stocker un compteur dans une cellule de la feuille Excel, lui aussi incrémenté à chaque appel.

Q3.4 Sachant que $x^0 = 1, x^n = (x^2)^{n/2}$ si n est pair et $x^n = x \cdot (x^2)^{(n-1)/2}$ si n est impair, écrire une autre fonction récursive `puissMaligne` calculant x^n. Tester et comparer le nombre d'appels des deux fonctions

récursives.

Exercice 4 : Tri fusion

On reprend la comparaison des algorithmes de tri du TP précédent. Les questions de cet exercice sont donc à faire dans le fichier du TP10 sur le tri, à la suite de ce qui a été fait (écriture en VBA de la macro génération, des algorithmes de tri par sélection et de tri à bulles). On rappelle que l'on génère aléatoirement des entiers en colonne 1 de la feuille Excel, on recopie ces entiers en colonnes 2 et 3, puis on les trie par un tri par sélection en colonne 2, et par un tri à bulles en colonne 3. La colonne 4 va être utilisée pour y effectuer un tri fusion, après recopie des entiers non triés de la colonne 1.
Pour cela, on va utiliser la procédure récursive `triFusion` suivante :

```
Sub triFusion(col As Long, deb As Long, fin As Long)
        Dim milieu As Long
        If deb >= fin Then
                Exit sub
        End if
        'Copie des éléments de col à col+1 pour les trier
        Range(Cells(deb, col+1), Cells(fin, col + 1)).Value = _
              Range(Cells(deb,col),Cells(fin, col)).Value
        milieu = (deb + fin) \ 2
        'tri récursif de la première moitié dans col + 1
        ............
        'tri récursif de la seconde moitié dans col + 1
        ............
        'interclasser les deux moitiés précédentes dans col
        ............
        'effacer les éléments dans col + 1
        Range(Cells(deb, col + 1), Cells(fin, col + 1)).clearcontents
End Sub
```

`triFusion` trie les éléments de la colonne `col` de la ligne `deb` à la ligne `fin` en effectuant les opérations suivantes :

- copie des éléments dans la colonne suivante (`col`+1) ;
- tri par un algorithme de tri fusion de la première moitié des éléments de la colonne `col`+1 ;
- tri par un algorithme de tri fusion de la seconde moitié des éléments de la colonne `col`+1 ;
- fusion (interclassement) dans la colonne `col` des deux moitiés triées de la colonne `col`+1 en une collection triée ;
- suppression des éléments de la colonne `col`+1.

Q4.1 Écrire la procédure d'interclassement qui prend les numéros de colonne `c` et de ligne `d1`, `f1`, `d2` et `f2`, et qui place en colonne `c-1` à partir de la ligne `d1` le résultat de la fusion triée (interclassement) de deux collections triées d'éléments situées en colonne `c`, l'une de la ligne `d1` à la ligne `f1` et l'autre de la ligne `d2` à la ligne `f2`.
Pour interclasser, on utilise deux variables `pos1` et `pos2` indiquant la ligne du plus petit élément courant dans chacune des deux collections. On prend le plus petit de ces deux premiers éléments courants et on le place à la position courante dans la colonne `c`-1 (i-ème position pour la i-ème selection d'un élément). On met à jour la variable de la position du plus petit élément courant correspondante (si le plus petit élément

était dans la première collection par exemple, alors la variable `pos1` est incrémentée de 1). On réapplique le même procédé jusqu'à ce que les deux collections à interclasser ait été parcourues intégralement.

Q4.2 Recopier et compléter maintenant la procédure `triFusion` donnée ci-dessus.

Q4.3 Écrire la macro `triFus` qui place les éléments de la première colonne de la feuille de calcul dans la quatrième colonne dans l'ordre croissant. Pour cela, `TriFus` copie les éléments de L1C1 :L300C1 dans L1C4 :L300C4, puis applique l'algorithme de tri fusion pour trier les éléments de la quatrième colonne en faisant appel à la procédure récursive `triFusion`.

Q4.4 Modifier la macro `verifieTris` du TP précédent pour qu'elle teste aussi le résultat de l'exécution de `TriFusion`.

Q4.5 Comparer les temps d'exécution des trois algorithmes de tri. Quelle est la méthode de tri la plus efficace ? Pourriez-vous expliquer pourquoi ?