TS TP 3 : BOUCLE ET ITÉRATEUR EN ALGORITHMIQUE.

I) PROGRAMMATION D'UNE BOUCLE, LE NOMBRE D'ITÉRATIONS ÉTANT DONNÉ :

1) La structure itérative ou boucle avec un nombre d'itérations donné:

Dans un algorithme, on peut répéter un calcul un nombre déterminé de fois.
Ce type de **boucle** est utilisé quand on veut répéter la même instruction un certain nombre de fois, ce nombre de répétitions étant connu.

Pour n variant **de** N_0 **à** N
  Faire
  | Traitement(s) : instruction élémentaire ou succession d'instructions élémentaires
Fin Pour

La variable n (ou toute autre lettre) sert de compteur et varie à chaque itération avec un pas de 1. Cela signifie que pour $n = N_0$, les instructions s'exécutent, puis pour $n = N_0 + 1$, elles s'exécutent à nouveau, en boucle, jusqu'à la valeur de $n = N$, où elles s'exécutent une dernière fois. La boucle est alors terminée et les instructions suivantes s'appliquent.

2) Exemple détaillé sur AlgoBox :

Dans AlgoBox, on utilise la commande POUR … DE … A que l'on peut insérer à l'aide du bouton Ajouter POUR … DE … A

Création d'un algorithme qui permet de calculer la somme des entiers de 1 à 10 :
Étape (a) Créer deux variables du type nombre **nommées n et somme.**
Étape (b)

- Se placer sur la ligne DEBUT ALGORITHME et cliquer sur le bouton Nouvelle Ligne.
- Cliquer ensuite sur le bouton Ajouter POUR … DE … A.
- Dans les champs POUR la variable :, sélectionner n ; ALLANT DE taper 1 ; A taper 10. Cliquer sur OK.

Étape (c)

- En étant bien positionné sur la ligne vide entre la ligne DEBUT POUR et la ligne FIN POUR, on clique alors sur le bouton AFFECTER valeur à variable
- Dans le champ La variable, sélectionner somme et dans le champ prend la valeur, taper somme + n. Cliquer sur OK.

Remarque : il n'est pas nécessaire ici d'initialiser la variable somme, car dans AlgoBox, la valeur affectée est 0 par défaut.

Étape (d) :

- Se placer sur la ligne FIN POUR, puis cliquer sur le bouton Nouvelle Ligne.
- Cliquer sur le bouton Ajouter AFFICHER Variable, puis sélectionner somme dans la ligne AFFICHER la variable et cliquer sur OK.

Description de l'algorithme pas à pas :
Étape (a) : La variable n sert de compteur et permet d'indiquer le nombre de fois que l'on veut répéter le calcul. Dans AlgoBox, cette variable est automatiquement incrémentée de 1 à chaque traitement.

Étapes (b) et (c) : C'est la boucle « POUR n allant de 1 à 10 » qui permet de répéter le calcul. La variable n compte le nombre d'itérations : ici, la valeur initiale de n est 1 et sa valeur finale est 10. Au premier passage, le programme affecte la valeur 1 à n, puis il effectue les instructions comprises entre DEBUT POUR et FIN POUR avec n valant 1. La variable somme vaut alors 0 + 1, soit 1.

Au deuxième passage, le programme incrémente n de 1, qui vaut maintenant 2. Il effectue à nouveau les instructions comprises entre DEBUT POUR et FIN POUR avec n valant 2 cette fois-ci. La variable somme vaut alors 1 + 2, soit 3. Puis, il continue jusqu'à la valeur finale de n, qui vaut 10.

Étape (d) : On sort de la boucle une fois que le nombre de répétitions, 10 ici, est atteint. Le programme affiche alors le résultat souhaité : la variable somme donne le résultat de la somme des entiers de 1 à 10.

Visualisation sous forme d'un tableau :

Étapes	*valeur de n*	*Valeur de somme*
Initiale		*0*
Fin de la 1ère boucle	*1*	*1*
Fin de la 2ème boucle	*2*	*3*
Fin de la 3ème boucle	*3*	*6*
…	…	…

Programmer cet algorithme sur AlgoBox et le tester !
Enregistrer le dans vos documents sous le nom TP3ExI)2).

3) Exercices :

Exercice 1 : On donne l'algorithme suivant :

```
Variables entières : N ; i et produit (N > 0)
Début de l'algorithme
Choisir N
produit prend la valeur 1
Pour i variant de 1 à N
        produit prend la valeur produit × i
Fin Pour
Écrire produit
Fin de l'algorithme
```

a) Faire fonctionner cet algorithme « à la main » pour $N = 3$; puis $N = 5$.
b) Quel résultat fournit cet algorithme ?
c) Programmer cet algorithme sur AlgoBox et vérifier les résultats de la question a).
Enregistrer le dans vos documents sous le nom TP3Ex1.

Attention à initialiser produit, sinon par défaut, la valeur attribuée sera 0.
Remarque : Le résultat produit obtenu précédemment s'appelle factorielle N et se note N !
(Notation au Programme de Terminale S)

Exercice 2 : On reprend l'algorithme du paragraphe I) 2) :

```
Variables : n et somme (entiers)
Début de l'algorithme
Pour n variant de 1 à 10
        somme prend la valeur somme + n
Fin Pour
Écrire somme
Fin de l'algorithme
```

```
Version modifiée :
Variables :
Début de l'algorithme




Fin Pour
Afficher somme
Fin de l'algorithme
```

a) Modifier cet algorithme pour que l'on demande à l'utilisateur la valeur de *n*, c'est à dire pour que l'algorithme donne le résultat de la somme des *n* premiers entiers, ce nombre *n* étant choisi par l'utilisateur.
Indication : introduire une nouvelle variable (i par exemple) qui servira de compteur et insérer dans l'algorithme un code permettant de saisir la valeur de n souhaitée.
b) Programmer cet algorithme sur AlgoBox et le tester.
Enregistrer le dans vos documents sous le nom TP3Ex2.

II) PROGRAMMATION D'UNE BOUCLE, AVEC FIN CONDITIONNELLE :

1) La structure itérative avec fin de boucle conditionnelle :

Dans le paragraphe I), on a vu comment faire répéter une série d'instructions à l'ordinateur quand on connaissait le nombre d'itérations.
Si ce nombre n'est pas connu, on peut néanmoins faire répéter une série d'instructions tant qu'une (ou plusieurs) condition(s) est (ou sont) vérifiée(s).
Ce type de **boucle** est utilisé quand on veut répéter la même instruction un certain nombre de fois, sans que l'on sache à l'avance combien d'itérations seront nécessaires. (Sinon, on utilise la boucle POUR … DE … A …).
On teste alors une condition en début de boucle et le traitement dans la boucle n'est réalisé que si la condition est vérifiée. Attention, si la condition n'est pas vérifiée au départ, la boucle n'est jamais exécutée …

Tant que (condition(s))
Faire
| Traitement(s) : instruction élémentaire ou succession d'instructions élémentaires
Fin Tant Que

2) Exemple détaillé sur AlgoBox *(Inspiré du document d'accompagnement 2nde « Algorithmique »)* :

Dans AlgoBox, on utilise la commande TANT QUE … que l'on peut insérer à l'aide du bouton
Ajouter TANT QUE …

Création d'un algorithme qui permet de calculer la somme *S* détenue à la banque après un nombre *N* de semestres.
On suppose que la somme placée au départ est 5000 € et qu'elle est placée à un taux d'intérêts composés de 2 % par semestre. On cherche le nombre de semestres nécessaires pour que cette somme dépasse 8000 €.

Étape (a) Créer deux variables du type nombre **nommées N et S.**

Étape (b)
- Se placer sur la ligne DEBUT ALGORITHME et cliquer sur le bouton Nouvelle Ligne.
- Cliquer alors sur le bouton AFFECTER valeur à variable
- Dans le champ La variable, sélectionner S et dans le champ prend la valeur, taper 5000
- Se placer sur la ligne S PREND LA VALEUR 5000 et cliquer sur le bouton Nouvelle Ligne.
- Cliquer ensuite sur le bouton Ajouter TANT QUE …
- Dans le champ TANT QUE la condition:, taper S<8000. Cliquer sur OK.

Étape (c)
- En étant bien positionné sur la ligne vide entre la ligne DEBUT TANT QUE et la ligne FIN TANT QUE, on clique alors sur le bouton AFFECTER valeur à variable
- Dans le champ La variable, sélectionner S et dans le champ prend la valeur, taper

1.02*S .(S est placée à 2 % d'intérêts composés, donc chaque semestre, la nouvelle somme vaut l'ancienne somme + 2 % de l'ancienne somme, soit 1,02 × l'ancienne somme)

- Cliquer sur le bouton Nouvelle Ligne, puis sur le bouton AFFECTER valeur à variable. Dans le champ La variable, sélectionner N et dans le champ prend la valeur, taper N+1 (on incrémente, à chaque boucle, le nombre de semestres nécessaires)

Étape (d) :

- Se placer sur la ligne FIN TANT QUE, puis cliquer sur le bouton Nouvelle Ligne.
- Cliquer sur le bouton Ajouter AFFICHER Variable, puis sélectionner N dans la ligne AFFICHER la variable, cocher la case Ajouter un retour à la ligne et cliquer sur OK. Faire de même pour la variable S.

Description de l'algorithme :
Tant que la somme S est inférieure à 8000 €, la boucle se réalise et le nombre de semestres N augmente d'autant.
On sort de la boucle une fois que la condition est réalisée. L'instruction entre DEBUT TANT QUE et FIN TANT QUE n'est alors plus réalisée et le programme affiche alors le résultat souhaité : la variable N donne le nombre de semestres nécessaires et la variable S donne la somme sur le compte.

Visualisation sous forme d'un tableau :

Étapes	*valeur de N*	*Valeur de S*
Initiale		*5000*
Fin de la 1ère boucle	*1*	*5100*
Fin de la 2ème boucle	*2*	*5202*
Fin de la 3ème boucle	*3*	*5306.04*
…	…	…

Programmer cet algorithme sur AlgoBox et le tester !
Enregistrer le dans vos documents sous le nom TP3ExII)2).

3) Exercices :

Exercice 3 : On donne l'algorithme suivant :

```
Variables entières : n et r
Début de l'algorithme
Entrer n
r prend la valeur n
Tant que r ≥ 11
        r prend la valeur r - 11
Fin Tant que
Afficher r
Fin de l'algorithme
```

a) Faire fonctionner cet algorithme « à la main » pour $n = 3$; puis $n = 65$ et enfin pour $n = 121$.
b) Quel résultat fournit cet algorithme ?
c) Programmer cet algorithme sur AlgoBox et vérifier les résultats de la question a).
Enregistrer le dans vos documents sous le nom TP3Ex3.
Attention à initialiser la variable r sinon, par défaut, la valeur attribuée sera 0 et la boucle ne sera jamais réalisée.

Exercice 4 : On reprend l'algorithme du paragraphe II) 2) :

Variables : *N* et *S* Début de l'algorithme *S* vaut 5000 **Tant que** *S* est strictement inférieure à 8000 *S* prend la valeur *S*×1,02 *N* prend la valeur *N* + 1 **Fin Tant que** Afficher *N* et *S* Fin de l'algorithme	*Version modifiée :* Variables : Début de l'algorithme **Fin Tant que** Afficher *N* et *C* Fin de l'algorithme

a) Modifier cet algorithme pour que l'on demande à l'utilisateur la valeur de *S* et que l'ordinateur trouve la valeur de *N* à partir de laquelle, la somme *S* a doublé.
Indication : Nommer (par exemple) C le capital obtenu après intérêts.
b) Programmer cet algorithme sur AlgoBox et le tester.
Enregistrer le dans vos documents sous le nom TP3Ex4.

Que remarque-t-on pour *N* ?
Prouvons cette dernière remarque :
c) Modifier l'algorithme précédent pour que l'ordinateur donne le plus petit entier *N* à partir duquel la somme aura doublé sans demander à l'utilisateur la somme de départ.
On remarquera qu'à chaque semestre, la somme étant multipliée par 1,02, au bout de N semestres, la somme est multipliée par $1{,}02^N$ …
Programmer cet algorithme sur AlgoBox et le tester.
Enregistrer le dans vos documents sous le nom TP3Ex4bis.

Exercice 5 : ***Algorithme de dichotomie***

Soit la fonction *f* définie sur $\mathbb{R}$ par $f(x) = x^3 + 2x - 2$.

Le but de l'exercice est de concevoir un algorithme permettant de trouver une valeur approchée de cette solution à 0,01 près.

a) Compléter le tableau de valeurs ci-dessous :

x	- 2	- 1	0	1	2	3
f(x)						

b) Prouver que l'équation $f(x) = 0$ admet une unique x_0 solution sur $\mathbb{R}$. Prouver que $x_0 \in [0 ; 1]$.

c) On se place désormais sur l'intervalle [0 ; 1]. On pose $a = 0$; $b = 1$ et $m = \dfrac{a + b}{2}$.

Comparer le signe de *f(a)* ; *f(b)* et *f(m)*. En déduire l'intervalle auquel appartient la solution x_0.
d) On se propose de concevoir un algorithme permettant d'obtenir une valeur approchée de x_0 à 0,01 près.
Le principe est le suivant :
* *m* prendra la valeur $\dfrac{a + b}{2}$
* Si *f(m)* et *f(a)* sont de même signe, alors *a* prendra la valeur de *m*, sinon *b* prendra la valeur de *m*.
Concevoir cet algorithme et le programmer sur le logiciel AlgoBox.
Le tester et donner une valeur approchée de x_0 à 0,01 près.
Indication : utiliser la boucle TANT QUE …conditionnée par l'instruction b - a > 0,01.
Remarque : on pourra utiliser la fonction Utiliser une fonction numérique *et après avoir coché le bouton* Utiliser une fonction*, on définira F1(x).*
Enregistrer le travail dans vos documents sous le nom TP3Ex5.

Exercice 6 : *La suite de Syracuse …*

Étape 1 : Étude d'une fonction auxiliaire.

f est la fonction qui à un entier naturel *n* associe l'entier *f(n)* défini de la façon suivante :

$$\begin{cases} \text{si } n \text{ est un entier pair, alors } f(n) = \dfrac{n}{2} ; \\ \text{si } n \text{ est un entier impair, alors } f(n) = 3n + 1. \end{cases}$$

a) Compléter le tableau de valeurs ci-dessous :

n	0	1	2	3	4	5	6	8	10	16
f(n)										

b) L'algorithme ci-dessous permet le calcul des valeurs prises par la fonction *f* :

Entrée
Saisir *n*
Traitement
Si *n* est pair **Alors**
y prend la valeur $\frac{n}{2}$
Sinon
y prend la valeur $3n + 1$
FinSi
Sortie
Afficher *y*

Écrire le programme sur le logiciel AlgoBox et l'enregistrer dans vos documents sous le nom TP3Ex6.

c) Vérifier les résultats de la question **a)**.

Étape 2 : La suite de Syracuse.

La suite de Syracuse est une suite de nombres entiers obtenue à partir d'un entier *n* non nul de la façon suivante :

- Si l'entier *n* est pair, l'entier qui le suit dans la suite sera $\frac{n}{2}$
- Si l'entier *n* est impair, l'entier qui le suit dans la suite sera égal à $3 \times n + 1$.

On recommence le procédé avec l'entier obtenu après le processus et on obtient une suite d'entiers qui se terminera nécessairement par 1 quel que soit l'entier *n* non nul de départ.
Les nombres de la suite sont appelés les étapes du vol et le nombre d'étapes avant d'obtenir 1 est appelé la durée du vol.

EXEMPLE : En utilisant les résultats du tableau de l'étape 1, vérifier les suites ci-dessous :

- Nombre de départ : $n = 2$
  2 ; **1** — La durée du vol est 1 car il faut une étape pour obtenir 1.
- Nombre de départ : $n = 3$
  3 ; 10 ; 5 ; 16 ; 8 ; 4 ; 2 ; **1** — La durée du vol est 7.
- Nombre de départ : $n = 4$
  4 ; 2 ; **1** — La durée du vol est 2.
- Nombre de départ : $n = 5$
  5 ; 16 ; 8 ; 4 ; 2 ; **1** — La durée du vol est 5.
- Nombre de départ : $n = 6$
  6 ; 3 ; 10 ; 5 ; 16 ; 8 ; 4 ; 2 ; **1** — La durée du vol est 8.

a)
Modifier l'algorithme précédent afin qu'il affiche la suite de Syracuse pour n'importe quel entier *n* non nul saisi.
Le programmer sur le logiciel AlgoBox et le tester.
L'enregistrer dans vos documents sous le nom TP3Ex6bis.
Indication : On utilisera la boucle TANT QUE … conditionnée par y >1 où y est la variable qui contiendra les différents termes de la suite.

b)
Compléter l'algorithme ci-dessus pour qu'il affiche en plus la durée du vol.
Le Programmer le sur le logiciel AlgoBox et le tester.
L'enregistrer dans vos documents sous le nom TP3Ex6fin.

Exercice 7 : *Simulation du lancer d'une pièce équilibrée.*
On désire créer un algorithme simulant le lancer d'une pièce équilibrée : pour cela, on utilise les fonctions random() et floor(x) du logiciel AlgoBox.

La fonction random() *permet de générer un nombre pseudo-aléatoire compris entre 0 et 1 (1 exclu).*

La fonction floor(x) *permet d'obtenir la partie entière d'une variable x.*

Ces fonctions, correctement combinées, permettent de générer des entiers égaux à 0 ou 1. On décide (par exemple) que la valeur 0 simule la sortie d'un PILE et la valeur 1 simule celle d'un FACE.

a) Quelle formule faut-il taper pour obtenir l'obtention de 0 ou 1 à l'aide des fonctions random() et floor(x) du logiciel AlgoBox ?

b) Écrire un algorithme demandant à l'utilisateur le nombre *n* de lancers et affichant les résultats de la simulation du lancer d'une pièce équilibrée au cours de ces *n* lancers.
Indications :
*On utilisera la formule du **a)** et on insèrera dans l'algorithme un code permettant de saisir la valeur de n souhaitée.*

Remarque : La fonction ALGOBOX_ALEA_ENT(p,n) permet de générer un entier pseudo-aléatoire entre p et n.

c) Programmer cet algorithme sur AlgoBox et le tester.
Enregistrer le dans vos documents sous le nom TP3Ex7.

Exercice 8 :
a) Modifier l'algorithme précédent pour qu'il affiche le nombre *nb* de fois où l'on obtient PILE à l'issue de ces *n* lancers.
On ne demandera plus d'afficher la valeur de k.
b) Compléter l'algorithme ci-dessus pour que s'affiche également la fréquence *f* du PILE au cours de ces *n* lancers.
c) Programmer cet algorithme sur AlgoBox et le tester.
L'enregistrer dans vos documents sous le nom TP3Ex8.
d) Vers quelle valeur cette fréquence se rapproche-t-elle ? Était-ce prévisible ?
Remarque : Sur AlgoBox, le nombre de répétitions des boucles est limité à 200 000.