Séquence 4

Les structures itératives

Partie 1

La « boucle » Pour…FinPour

Exercice 31

Écrire l'algo de calcul et d'affichage de la moyenne de x élèves à un devoir, x étant saisi.
La correction de cet exo est dans le cours.

Exercice 32

Écrire l'algo qui calcule et affiche la moyenne de x élèves à un devoir ainsi que la note la plus haute et la note la plus basse, x étant saisi.

Tout résidait dans le secret des initialisations :

Lexique	Algo moyenne, exo 32

somme (entier) : somme des notes successives saisies.
max (entier, calculé) : maximum courant.
min (entier, calculé) : minimum courant.
NbEleves (entier, saisi) : nombre d'élèves.
compteur (entier, calculé) : compteur de boucles.
note (entier, saisi) : note courante.
moyenne (réel, calculé) : moyenne à calculer.
	Début
	somme ← 0
	// On initialise max de manière à être sûr que même
	// la plus mauvaise note lui est supérieure ou égale
	max ← 0
	// On fait la même chose pour min, mais en sens
	// inverse
	min ← 20
	Saisir (nbEleves)
	Pour compteur de 1 à nbEleves
	Faire Afficher ("Entrez la ", i, "ème note ")
	Saisir (note)
	// On compare la note courante au minimum
	//courant, si elle lui est inférieure, elle devient
	// le minimum courant
	Si note < min
	Alors min ← note
	Finsi
	// Même raisonnement pour max
	Si note > max
	Alors max ← note
	Finsi

	somme ← somme + note FinPour moyenne ← somme / nbEleves Afficher ("La moyenne est de :", moyenne) Afficher (" La meilleure note est : ", max) Afficher (" La moins bonne note est : ", min) Fin

Je voudrais que l'on s'arrête un instant sur la variable **somme** : cette variable est un cumul ; elle prend des valeurs successives calculées à chaque passage dans la boucle et son calcul réutilise sa précédente valeur.

L'instruction **somme ← somme + note** signifie « somme vaut maintenant ce qu'elle valait avant + la note qui vient d'être saisie ».

Le fait que le calcul de la somme réutilise la précédente valeur de la somme nous oblige à initialiser la somme **avant l'entrée dans la boucle**.

Si on oublie d'initialiser ce type de variables appelées à prendre des valeurs successives, on a de sacrées surprises puisque la première fois qu'on entre dans la boucle, on les recalcule mais à partir d'une valeur non initialisée, donc, indéfinie.

Je voudrais également revenir sur la partie suivante de l'algorithme :

```
Si      note < min
Alors   min ← note
Finsi
Si      note > max
Alors   max ← note
Finsi
```

Certain(e)s d'entre vous ont peut-être écrit quelque chose comme :

```
Si      note < min
Alors   min ← note
Sinon   max ← note
Finsi
```

Il ne faut surtout pas écrire ça, ce traitement est faux. En effet, il signifie que si **note** n'est pas la note minimale, alors c'est qu'elle est la note maximale !!!

D'autres parmi vous ont peut-être eu l'idée d'écrire :

```
Si      note < min
Alors   min ← note
Sinon   Si      note > max
        Alors   max ← note
        Finsi
Finsi
```

Là non plus, ça n'est pas une bonne idée car dans le cas où **nbEleves = 1**, la seule note saisie est à la fois la note minimale et la note maximale.

Si on écrit l'algorithme comme ci-dessus, la note saisie sera bien affectée à min mais pas à max, qui conservera son 0 (zéro) comme valeur. Pas glop !!

Exercice 33

Écrire l'algo qui calcule et affiche le factoriel d'un nombre saisi au clavier.

Je vous présente les 2 versions (en augmentant et en diminuant le compteur d'itérations).

Lexique	Algo factoriel (version « en augmentant »)
nb (entier, saisi) : nombre dont on veut calculer le factoriel. fact (entier, calculé) : factoriel du nombre saisi, calculé au fil des itérations. compteur (entier, saisi) : compteur d'itérations.	Début Saisir (nb) // Il faut initialiser fact avant l'entrée dans la boucle. fact ← 1 Pour compteur de 2 à nb (augmenter de 1) Faire fact ← fact * compteur FinPour Afficher ("le factoriel de ", nb, " vaut ", fact) FinSi Fin

Une remarque avant de vous présenter la version « en diminuant » : Si **nb** vaut **1**, alors, **Pour compteur de 2 à nb (augmenter de 1)** devient , **Pour compteur de 2 à 1 (augmenter de 1)**. On n'entre pas dans la boucle car la borne maximale est inférieure à la borne minimale.

Cela ne pose aucun problème car on a initialisé **fact** à **1**.

Dans le cas où **nb** vaut **1**, on aura donc bien affichage du message **Le factoriel de 1 vaut 1**.

Il en est de même dans le cas où **nb** vaut **0.**

Lexique	Algo factoriel (version « en diminuant »)
nb (entier, saisi) : nombre dont on veut calculer le factoriel. fact (entier, calculé) : factoriel du nombre saisi, calculé au fil des itérations. compteur (entier, saisi) : compteur d'itérations.	Début Saisir (nb) fact ← 1 Pour compteur décroissant de nb à 2 (diminuer de 1) Faire fact ← fact * compteur FinPour Afficher ("Le factoriel de ", nb, " vaut ", fact) FinSi Fin

Ici, on n'entrera dans la boucle que si **nb** vaut au moins **2**.

Exercice 34

Écrire l'algorithme qui calcule et affiche le nom et le salaire net de chacun des employés d'une entreprise et qui calcule et affiche également le salaire net moyen des employés.

Dans cette entreprise, les employés sont payés à l'heure (ils ont tous le même salaire horaire) et bénéficient d'une prime d'ancienneté.

Le nombre n d'employés est saisi, ainsi que le salaire horaire et le taux de retenue.

Pour chaque employé, le nombre d'heures travaillées le montant de la prime et le nom de l'employé sont également saisis au clavier.

La règle de calcul d'un salaire net est la suivante : ((nombre d'heures travaillées * salaire horaire) + prime) * (1 – taux de retenue).

Pour écrire cet algo, qui est une extension de l'exercice 9 de la séquence 2, on va repartir de la solution fournie dans le corrigé des exercices de la séquence 2 et la modifier petit à petit.

Voici ci-dessous l'algorithme de l'exercice 9.

Lexique	Algo exo 9 – Calcul du salaire net
nom (chaîne, saisie) : nom de l'employé. salHor (réel, saisi) : salaire horaire. nbH (entier, saisi) : nombre d'heures travaillées. tauxRet (réel, saisi) : Taux de retenue. prime (réel, saisi) : prime.	Début Afficher ("Saisissez le nom de l'employé") Saisir (nom) Afficher ("Saisissez le salaire horaire") Saisir (salHor) Afficher ("Saisissez le nombre d'heures travaillées") Saisir (nbH) Afficher ("Saisissez le taux de retenue") Saisir (tauxRet) Afficher ("Saisissez le montant de la prime") Saisir (prime) Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", (nbH * salHor + prime)*(1 – tauxRet), "euros.") Fin

Commençons par trier en 2 colonnes ce qui concerne un seul employé et ce qui concerne tous les employés dans cet exo 9 :

Ce qui concerne un seul employé	Ce qui est commun à tous les employés

Début
Afficher ("Saisissez le nom de l'employé")
Saisir (nom)
Afficher ("Saisissez le nombre d'heures travaillées")
Saisir (nbH)
Afficher ("Saisissez le montant de la prime")
Saisir (prime)
Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", (nbh * salHor + prime)*(1 – tauxRet), "euros.")
	Afficher ("Saisissez le salaire horaire")
	Saisir (salHor)
	// Le salaire horaire est le même pour tous les
	// employés, il est donc inutile de le ressaisir à
	// chaque calcul de salaire net d'un employé, il ne
	// faut donc pas le mettre dans la boucle, mais le
	// saisir une seule fois, avant la boucle.
	Afficher ("Saisissez le taux de retenue")
	Saisir (tauxRet)
	// De même, le taux de retenue est le même pour
	// tous et n'a donc pas besoin d'être ressaisi à
	// chaque tour de boucle.

Voici donc une première ébauche de notre algo :

Lexique	Algo calcul itératif du salaire net – première ébauche
nom (chaîne, saisie) : nom de l'employé. salHor (réel, saisi) : salaire horaire. nbH (entier, saisi) : nombre d'heures travaillées. tauxRet (réel, saisi) : taux de retenue. prime (réel, saisi) : prime.	Début Afficher ("Saisissez le salaire horaire") Saisir (salHor) Afficher ("Saisissez le taux de retenue") Saisir (tauxRet) Pour i de ? à ? Faire Afficher ("Saisissez le nom de l'employé") Saisir (nom) Afficher ("Saisissez le nombre d'heures travaillées") Saisir (nbH) Afficher ("Saisissez le montant de la prime") Saisir (prime) Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", (nbH * salHor + prime)* (1 – tauxRet), "euros.") FinPour Fin

Réfléchissons encore un peu : Le traitement de calcul de salaire doit s'effectuer autant de fois qu'il y a d'employés et le nombre d'employés est saisi : il faut donc rajouter l'instruction **Saisir (NbEmp)**, **nbEmp** étant le nombre d'employés.

L'instruction **Pour** va utiliser cette valeur pour limiter le nombre d'itérations : on veut faire le traitement **nbEmp** fois, on va donc le faire pour des valeurs du compteur allant de 1 à nbEmp (on pourrait le faire pour le compteur allant de **2** à **nbEmp + 1**, mais ce serait se compliquer

inutilement la tâche, on peut aussi le faire pour le compteur décroissant de nbEmp à 1 mais ça n'a pas grand intérêt). On va donc rajouter **Pour i de 1 à nbEmp (augmenter de 1) Faire**.

Maintenant, il y a cette histoire de salaire moyen : comme pour la moyenne calculée dans l'exercice 31, il faut à chaque tour de boucle cumuler le salaire que l'on vient de calculer au cumul précédent, que l'on divisera ensuite par le nombre d'employés. Il ne faut pas oublier d'initialiser ce cumul avant d'entrer dans la boucle, de calculer ce cumul à chaque passage dans la boucle, et de calculer et afficher la moyenne après être sorti de la boucle (si on calcule et affiche la moyenne dans la boucle **pour**, on va calculer **nbEmp** moyennes mais une seule sera utile à calculer, la dernière, puisque c'est la seule qui est demandée).

Lexique	Algo calcul itératif du salaire net – version en voie d'amélioration

salHor (réel, saisi) : salaire horaire.
tauxRet (réel, saisi) : taux de retenue.
nbEmp (entier, saisi) : nombre d'employés.
totSal (réel, calculé) : valeurs successives du cumul des salaires nets.
nom (chaîne, saisie) : nom de l'employé courant (c'est-à-dire en train d'être traité).
prime (réel, saisi) : prime de l'employé courant.
nbH (entier, saisi) : nombre d'heures travaillées par l'employé courant.
i (entier, saisi) : compteur d'itérations.
	Début
	Afficher ("Saisissez le salaire horaire")
	Saisir (salHor)
	Afficher ("Saisissez le taux de retenue")
	Saisir (tauxRet)
	Afficher ("Saisissez le nombre d'employés")
	Saisir (nbEmp)
	// Initialisation du cumul des salaires
	totSal ← 0
	Pour i de 1 à nbEmp
	Faire Afficher ("Saisissez le nom de l'employé")
	Saisir (nom)
	Afficher ("Saisissez le nombre d'heures travaillées")
	Saisir (nbH)
	Afficher ("Saisissez le montant de la prime")
	Saisir (prime)
	// On ne laisse pas le calcul du salaire net dans
	// l'instruction d'affichage car on s'en sert 2 fois : une
	// fois pour calculer le cumul, et une fois pour l'affichage
	salNet ← (nbh * salHor + prime)* (1 – tauxRet)
	// Calcul du cumul
	totSal ← totSal + salNet
	Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", salNet, "euros.")
	FinPour
	// Calcul et affichage du salaire moyen
	Afficher ("Le salaire moyen est de ", totSal/nbEmp, "€.")
	Fin

Une dernière amélioration, et la version sera définitive : vous avez remarqué qu'à chaque passage dans la boucle, en calculant le salaire net de chaque employé, on recalcule l'expression **(1 – tauxRet)**.

Or, cette expression a la même valeur pour tous les employés.

Je vous propose donc de ne la calculer qu'une fois, avant l'entrée dans la boucle, de ranger le résultat du calcul dans une variable, et d'utiliser cette variable dans le calcul du salaire net.

Lexique	Algo calcul du salaire net – version définitive
salHor (réel, saisi) : salaire horaire. tauxRet (réel, saisi) : taux de retenue. nbEmp (entier, saisi) : nombre d'employés. totSal (réel, calculé) : valeurs successives du cumul des salaires nets. nom(chaîne, saisie) : nom de l'employé courant (c'est-à-dire en train d'être traité). prime (réel, saisi) : prime de l'employé courant. nbH (entier, saisi) : nombre d'heures travaillées par l'employé courant. i (entier, saisi) : compteur d'itérations. unMoinsTauxRet (réel, saisi) : variable valant 1 moins le taux de retenue.	Début Afficher ("Saisissez le salaire horaire") Saisir (salHor) Afficher ("Saisissez le taux de retenue") Saisir (tauxRet) unMoinsTauxRet ← 1 - tauxRet Afficher ("Saisissez le nombre d'employés") Saisir (nbEmp) totSal ← 0 Pour i de 1 à nbEmp Faire Afficher ("Saisissez le nom de l'employé") Saisir (nom) Afficher ("Saisissez le nombre d'heures travaillées") Saisir (nbH) Afficher ("Saisissez le montant de la prime") Saisir (prime) salNet ← (nbH * salHor + prime) * unMoinsTauxRet totSal ← totSal + salNet Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", salNet, "euros.") FinPour Afficher ("Le salaire moyen est de ", totSal/nbEmp, "€.") Fin

Voilà.

Exercice 35

Un nombre est dit « parfait » s'il est égal à la somme de tous ses diviseurs sauf lui-même. Écrire l'algorithme qui indique si un nombre saisi au clavier est parfait ou non.

Alors ? À quelle valeur doit-on initialiser le cumul des diviseurs du nombre saisi ?

On peut l'initialiser à 0, ou mieux, à 1.

Si on l'initialise à 0, l'itération se fera de **1** à **(nb div 2)** et comme 1 est diviseur de tous les nombres, au premier passage dans la boucle, le cumul passera toujours à 1. Donc, autant économiser un tour de boucle en initialisant d'emblée le cumul à 1 et en faisant l'itération de **2** à **(nb div 2)**.

Ce à quoi nous devons réfléchir dans ces 2 choix d'initialisation, c'est aux cas où les nombres saisis sont 0 ou 1, qui ne sont ni l'un ni l'autre des nombres parfaits.

Voici d'abord l'algo avec initialisation du cumul à 0 :

Lexique	Algo Nombre parfait – initialisation du cumul à 0
nb (entier, saisi) : nombre à tester. cumul (entier, calculé) : cumul des diviseurs du nombre. Prend des valeurs successives au cours des itérations. cpteur (entier, calculé) : compteur d'itérations qui sert éventuellement de diviseur. DIV : opération de division entière. MOD : opération qui calcule le reste de la division entière.	Début Afficher ("Saisissez le nombre") Saisir (nb) cumul ← 0 Pour cpteur de 1 à (nb DIV 2) (augmenter de 1) Faire Si nb MOD cpteur = 0 Alors cumul ← cumul + cpteur FinSi FinPour Si cumul = nb Alors afficher (nb, " est un nombre parfait") Sinon afficher (nb, " n'est pas un nombre parfait") FinSi Fin

Dans cet algo, si **nb** vaut **0**, que se passe-t-il ?

On n'entre pas dans la boucle car **(nb div 2)** vaut **0**, la borne maximum est donc inférieure à la borne minimum de l'itération.

La variable **cumul** reste donc à **0**.

Donc, **cumul = nb**.

Donc, on aura un message nous indiquant que 0 est un nombre parfait, ce qui est faux. En effet, tous les nombres sont des diviseurs de 0, mais 0 n'est pas la somme de tous ces nombres.

Et si **nb** vaut **1** ?

On n'entre pas non plus dans la boucle. La variable cumul reste donc à 0, mais comme **cumul** est différent de **nb** (qui vaut 1), le message nous indiquera que 1 n'est pas un nombre parfait, ce qui est exact.

En ce qui concerne cet algo, on doit donc régler le problème du 0. On peut le faire très simplement en rajoutant une condition pour l'affichage du message :

```
Si      cumul = nb et nb <> 0
Alors   afficher (nb, " est un nombre parfait")
Sinon   afficher (nb, " n'est pas un nombre parfait")
FinSi
```

Voici maintenant la version où le cumul est initialisé à 1 :

Lexique	Algo Nombre parfait – Initialisation du cumul à 1
nb (entier, saisi) : nombre à tester. cumul (entier, calculé) : cumul des diviseurs du nombre. Prend des valeurs successives au cours des itérations. cpteur (entier, calculé) : compteur d'itérations qui sert éventuellement de diviseur. DIV : opération de division entière. MOD : opération qui calcule le reste de la division entière.	Début Afficher ("Saisissez le nombre") Saisir (nb) cumul ← 1 Pour cpteur de 2 à (nb DIV 2) (augmenter de 1) Faire Si nb MOD cpteur = 0 Alors cumul ← cumul + cpteur FinSi FinPour Si cumul = nb Alors afficher (nb, " est un nombre parfait") Sinon afficher (nb, " n'est pas un nombre parfait") FinSi Fin

Dans ce cas, si **nb** vaut **0**, on n'entre pas dans la boucle et **cumul** (qui vaut 1) étant différent de **nb** (qui vaut 0), il n'y a pas de problème.

Si **nb** vaut **1**, on n'entre pas dans la boucle mais **cumul** et **nb** sont égaux. Il faut donc faire exactement la même chose que dans la version précédente, rajouter un test sur la valeur de **nb** :

```
Si      cumul = nb et nb <> 1
Alors   afficher (nb, " est un nombre parfait")
Sinon afficher (nb, " n'est pas un nombre parfait")
FinSi
```

Exercice 36

Écrire l'algorithme du jeu suivant : ce jeu se joue à 2 joueurs.

Le premier joueur saisit un mot à l'abri du regard du deuxième joueur.

Puis, le deuxième joueur doit deviner quel est ce mot et le saisir.

Il a droit à 3 essais.

À chaque essai, le programme lui indique s'il a trouvé le mot ou bien, le cas échéant, quelles sont les bonnes lettres aux bonnes places parmi celles qu'il a saisies.

Exemple : si le mot à trouver est « carpette » et que le joueur a saisi « crateres », le programme lui affiche : c - - - e - - -.

Si, au bout du 3e essai, le joueur n'a pas trouvé le mot, un message lui indique quel était le mot à trouver.

Lexique	Algo : Deviner un mot
motATrouver (chaîne, saisie) : Mot que le joueur 2 doit trouver. essai (chaîne, saisie) : mot saisi par le joueur 2. i (entier, calculé) : compteur d'itération, sur la longueur du mot à trouver. longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.	Début Afficher("Joueur 1, tu dois saisir un mot") Saisir (motATrouver) Afficher ("Joueur 2, tu dois trouver le mot.") Afficher ("Tu as droit à 3 essais.") Afficher ("Premier essai :") Saisir (essai) Si essai = motATrouver Alors Afficher ("Bravo, joueur 2, tu as gagné du premier coup") Sinon Pour i de 1 à longueur (motATrouver) Faire Si essai[i] <> motATrouver[i] Alors essai[i] ← "-" FinSi FinPour Afficher ("Voici les bonnes lettres :", essai) Afficher("Joueur 2, saisis ton deuxième essai") Saisir (essai) Si essai = motATrouver Alors Afficher ("Bravo, joueur 2, tu as gagné du deuxième coup") Sinon Pour i de 1 à longueur (motATrouver) Faire Si essai[i] <> motATrouver[i] Alors essai[i] ← "-" FinSi FinPour Afficher ("Voici les bonnes lettres :", essai) Afficher ("Joueur 2, saisis ton troisième essai") Saisir (essai Si essai = motATrouver Alors Afficher ("Bravo, tu as enfin trouvé le mot") Sinon Afficher ("Tu as définitivement perdu, le mot à trouver était ", motATrouver) FinSi FinSi FinSiSisss Fin

Exercice 37

Écrire un algorithme qui affiche la valeur du plus grand de x nombres saisis, x étant saisi.

Lexique	Algo Maximum de x nombres

qte (entier, saisi) : quantité de nombres.
max (entier, saisi) : maximum des x nombres, initialement saisi.
i (entier, calculé) : compteur d'itérations.
nombre (entiers, saisis) : nombre courant à comparer au maximum.
	Début
	Afficher ("Saisir la quantité de nombres que vous souhaitez saisir")
	Saisir (qte)
	Afficher ("Saisir le premier nombre")
	// Initialisation du maximum avec le premier
	// nombre saisi
	Saisir (max)
	// L'itération commence à 2 car l'utilisateur a
	// déjà saisi un nombre
	Pour i de 2 à qte (augmenter de 1)
	Faire Afficher ("Saisir le ", i, "° nombre)
	Saisir (nombre)
	Si nombre > max
	Alors max ← nombre
	FinSi
	FinPour
	Afficher ("Le maximum est : ", max)
	Fin

Exercice 38

Dans un but statistique, on veut établir des calculs de consommation d'un groupe de 100 abonnés à l'EDF, le dernier et l'avant-dernier relevé de compteur de chacun d'eux étant saisi.

On sait d'autre part, que la tarification se fait par tranches :

- **si la quantité d'électricité est inférieure à 100 kwh, le prix du kwh est de 0.09 euros ;**
- **si la quantité d'électricité est supérieure à 100 kwh, les 100 premiers kwh sont à 0.09 euros et au delà, à 0.05 euros.**

Le coût forfaitaire de location du compteur est de 6 euros hors taxes.

Écrire l'algorithme qui calcule et affiche le pourcentage d'abonnés faisant partie de la première tranche, de la deuxième tranche, ainsi que la consommation moyenne de chaque tranche et du groupe complet d'abonnés.

Lexique	Algo statistique EDF
conso1 (entier, calculé) : cumul des consommations des abonnés de la tranche 1. conso2 (entier, calculé) : cumul des consommations des abonnés de la tranche 2. tranche1 (entier, calculé) : pourcentage d'abonnés faisant partie de la tranche 1. i (entier, calculé) : compteur d'itérations. dernier (entier, saisi) : dernier relevé de compteur de l'abonné courant. avantD (entier, saisi) : avant dernier relevé de compteur de l'abonné courant. conso (entier, calculé) : consommation d'électricité de l'abonné courant.	Début conso1 ← 0 conso2 ← 0 tranche1 ← 0 Pour i de 1 à 100 Faire Saisir (dernier, avantD) conso ← dernier – avantD Si conso <= 100 Alors tranche1 ← tranche1 + 1 conso1 ← conso1 + conso Sinon conso2 ← conso2 + conso Finsi FinPour Afficher (tranche1, " % d'abonnés ont consommé moins de 100 KW entre 2 relevés, avec une consommation moyenne de " , conso1 / tranche1) Afficher (100 – tranche1," % d'abonnés ont consommé plus de 100 KW entre 2 relevés, avec une consommation moyenne de " , conso2 / tranche2) Afficher ("La consommation moyenne, toutes tranches confondues est de : ", (conso1 + conso2)/100) Fin

Exercice 39

Traduction des mots en code secret. On l'a tous fait quand on était petit !

Écrire l'algorithme qui traduit un mot de n'importe quelle longueur saisi au clavier en codes numériques ascii.

Exemple : si l'utilisateur saisit le mot** jour**, l'algorithme affiche le message suivant : "Le code secret du mot jour est 106 111 117 114" (ces nombres étant les codes ascii des caractères j, o, u, et r).

L'affichage doit se faire au fur et à mesure ou bien par concaténation dans une chaîne de caractères initialisée à la chaîne vide avant l'entrée dans la boucle,et je vous présente ces deux versions ci-dessous..

Lexique	Algo code secret avec affichage au fur et à mesure
mot (chaîne, saisie) : mot à coder. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.	Début Afficher ("Saisissez le mot de à coder") Saisir (mot) Afficher ("Le code secret du mot ", mot," est ") Pour i de 1 à longueur(mot) Faire Afficher (ascii(mot[i]) + " ") FinPour Fin

Lexique	Algo code secret avec concaténation dans une chaîne initialement vide
mot (chaîne, saisie) : mot à coder. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre. chaineCodee (chaîne, calculée) : chaîne contenant le résultat.	Début Afficher ("Saisissez le mot de à coder") Saisir (mot) chaineCodee ← "" Pour i de 1 à longueur(mot) Faire chaineCodee ← chaineCodee + fchaine(ascii(mot[i])) + " " FinPour Afficher ("Le mot codé est :", chaineCodee) Fin

Exercice 40

Version 1 : cryptage d'un mot. Votre client est agent secret et écrit ses messages en langage crypté. Il vous demande de lui écrire un programme qui crypte le mot qu'il saisit. Ce cryptage consiste à inverser le mot et à intercaler entre chacune de ses lettres la lettre qui a la position symétrique dans l'alphabet.

Je vous explique : l'alphabet comporte 26 lettres : a b c d e f g h i j k l m n o p q r s t u v w x y z. Dans l'alphabet a est la symétrique de z, b est la symétrique de y, c est la symétrique de x,..., et ainsi de suite jusqu'à m qui est la symétrique de n.

Version 2 : cryptage d'une phrase saisie par l'utilisateur. Les mots sont cryptés dans l'ordre inverse où ils sont saisis dans la phrase.

Version 3 : cryptage d'un texte saisi par l'utilisateur. Les phrases sont cryptées dans l'ordre inverse où elles sont saisies dans le texte.

Note : les points et les espaces ne font pas l'objet d'un cryptage particulier.

Pour chacune des 3 versions, je vous propose deux façons de procéder : la première en affichant le résultat du cryptage au fur et à mesure, la seconde en constituant au fur et à mesure une chaîne de caractère que l'on affiche après avoir fini de la construire.

Lexique	Algo cryptage – version 1 avec affichage au fur et à mesure
mot (chaîne, saisi) : mot à crypter. i (entier, calculé) : compteur d'itérations et indice de parcours de la phrase. chr (fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.	Début Afficher ("Saisir le mot à crypter") Saisir (mot) Afficher ("Le mot ", mot, "une fois crypté devient : ") Pour i de longueur(mot) à 1 (diminuer de 1) Faire Afficher (mot[i] + chr(219-ascii(mot[i]))) FinPour Fin

Lexique	Algo cryptage – version 1 avec concaténation
mot (chaîne, saisi) : mot à crypter. i (entier, calculé) : compteur d'itérations et indice de parcours de la phrase. chr (fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre. resultat (chaîne, calculé) : mot crypté.	Début Afficher ("Saisir le mot à crypter") Saisir (mot) resultat ← "" Pour i de longueur(mot) à 1 (diminuer de 1) Faire resultat ← resultat + mot[i] + chr(219 - ascii(mot[i])) FinPour Afficher ("Le mot ",mot , "une fois crypté devient :", resultat) Fin

Lexique	Algo cryptage – version 2 avec affichage au fur et à mesure

phrase (chaîne, saisi) : phrase à crypter.
i (entier, calculé) : compteur d'itérations et indice de parcours de la phrase.
chr (fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre.
ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.
	Début
	Afficher ("Saisir la phrase à crypter")
	Saisir (phrase)
	Afficher ("La phrase ", phrase, "une fois cryptée devient : ")
	// On ne traite pas le point, que l'on rajoutera à la fin, d'où
	// la borne longueur(phrase) - 1
	Pour i de longueur(phrase) - 1 à 1 (diminuer de 1)
	Faire Si phrase[i] = " "
	Alors Afficher (" ")
	Sinon Afficher (phrase[i] + chr(219 - ascii(phrase[i])))
	FinSi
	FinPour
	// On rajoute le point
	Afficher (".")
	Fin

On pourrait aussi écrire le **pour** comme ci-dessous …

```
Pour i de longueur(phrase) - 1 à 1 (diminuer de 1)
Faire    Afficher (phrase[i] + chr(219-ascii(phrase[i])))
         Si       phrase[i] <> " "
         Alors    Afficher (chr(219 - ascii(phrase[i])))
         FinSi
FinPour
```

…car c'est seulement dans le cas où le caractère courant n'est pas un espace que l'on rajoute son symétrique.

Lexique	Algo cryptage – version 2 avec concaténation
phrase (chaîne, saisi) : phrase à crypter. resultat (chaîne, calculé) : phrase cryptée. i (entier, calculé) : compteur d'itérations et indice de parcours de la phrase. chr(fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.	Début Afficher ("Saisir la phrase à crypter") Saisir (phrase) resultat ← "" Pour i de longueur(phrase) - 1 à 1 (diminuer de 1) Faire Si phrase[i] = " " Alors resultat ← resultat + " " Sinon resultat ← resultat + phrase[i] + chr(219 - ascii(phrase[i])) FinSi FinPour // On rajoute le point et on affiche Afficher ("La phrase ", phrase, "une fois cryptée devient :", resultat + ".") Fin

Voici une autre façon d'écrire le **pour** de cette version :

```
Pour i de longueur(phrase) - 1 à 1 (diminuer de 1)
Faire    resultat ← resultat + phrase[i]
         Si       phrase[i] <> " "
         Alors    resultat ← resultat + chr(219 - ascii(phrase[i]))
         FinSi
FinPour
```

Lexique	Algo cryptage – version 3 avec affichage au fur et à mesure
texte (chaîne, saisi) : texte à crypter. i (entier, calculé) : compteur d'itérations et indice de parcours du texte. chr(fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.	Début Afficher ("Saisir le texte à crypter") Saisir (texte) Afficher ("Le texte ", texte, "une fois crypté devient :") Pour i de longueur(texte) - 1 à 1 (diminuer de 1) Faire Si texte[i] = " " ou texte[i] = "." Alors Afficher (texte[i]) Sinon Afficher (texte[i] + chr(219 - ascii(phrase[i]))) FinSi FinPour // On rajoute le dernier point Afficher (".") Fin

Lexique	Algo cryptage – version 3 avec concaténation

// même lexique que
// précédemment
// avec en plus
resultat (chaîne, calculé) : texte crypté.
	Début
	Afficher ("Saisir le texte à crypter")
	Saisir (texte)
	resultat ← " "
	Pour i de longueur(texte) - 1 à 1 (diminuer de 1)
	Faire Si texte[i] = " " ou texte[i] = "."
	Alors resultat ← resultat + texte[i]
	Sinon resultat ← resultat + texte[i] + chr(219 - ascii(texte[i]))
	FinSi
	FinPour
	Afficher ("Le texte ", texte, "une fois crypté devient :", resultat + ".")
	Fin

La version 4 a pour vocation de corriger le fait que dans notre texte crypté de la version 3, on se retrouve avec un espace avant le point au lieu d'un point suivi d'un espace.

Le principe est simple : au fil du parcours du texte (que l'on commence par la fin), si on rencontre un espace suivi d'un point, alors on échange leurs positions dans le texte crypté.

Lexique	Algo cryptage – version 4
texte (chaîne, saisi) : texte à crypter. resultat (chaîne, calculé) : texte crypté. chr(fonction, résultat : caractère) : retourne le caractère correspondant au code ascii passé en paramètre. ascii (fonction, résultat : entier) : retourne le code ascii du caractère passé en paramètre.	Début Afficher ("Saisir le texte à crypter") Saisir (texte) resultat ← "" Pour i de longueur(texte) - 1 à 1 (diminuer de 1) Faire Si texte[i] = " " et texte[i+1] = "." Alors texte[i+1] ← " " texte[i] ← "." FinSi Si texte[i] = " " ou texte[i] = "." Alors resultat ← resultat + texte[i] Sinon resultat ← resultat + texte[i] + chr(219 - ascii(texte[i])) FinSi FinPour // On rajoute le dernier point et on affiche Afficher ("Le texte ", texte, "une fois crypté devient :", resultat + ".") Fin

Exercice 41

Écrire un algorithme qui simule le jeu de pile ou face.

Déroulement du jeu : l'utilisateur saisit la lettre P pour pile, et F pour face, puis valide sa saisie (ou bien il clique sur le bouton « Pile » ou le bouton « Face » dans le cas d'une interface graphique et événementielle).

Le programme lui, choisit aléatoirement un nombre entre 0 et 1, si le nombre tiré au sort est 0, alors pile est gagnant, face est perdant, si le nombre tiré au sort est 1, alors pile est perdant et face est gagnant. Le programme affiche un message : gagné ou perdu.

L'utilisateur fait 10 essais et c'est au bout des 10 essais que la partie s'arrête, affichant son score à l'utilisateur : nombre de pertes et de gains.

Lexique	Algo pile ou face
partie (entier, saisi) : compteur d'itérations. choixUtil (caractère, saisi) : caractère choisi par l'utilisateur. hasard (fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres. gains(entier, calculé) : nombre de parties gagnées par le joueur.	Début Pour partie de 1 à 10 (augmenter de 1) Faire Afficher ("Taper P pour pile, F pour face") Saisir (choixUtil) Si hasard(0,1) = 0 Alors gains ! gains + 1 Finsi FinPour Afficher ("Sur 10 parties, vous en avez gagné ", gains) Fin

Remarque : on peut rendre notre algorithme plus futé. Je vous propose ci-dessous 2 améliorations.

Amélioration n°1 : on peut avantageusement remplacer …

Si hasard(0,1) = 0 Alors gains ← gains + 1 FinPour

par

gains ← gains + hasard(0,1)

Amélioration n°2 : on peut carrément écrire l'algorithme comme ci-dessous.

Lexique	Algo pile ou face – Version plus futée
partie (entier, saisi) : compteur d'itérations. choixUtil (caractère, saisi) : caractère choisi par l'utilisateur. hasard (fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres. gains(entier, calculé) : nombre de parties gagnées par le joueur.	Début Pour partie de 1 à 10 (augmenter de 1) Faire Afficher ("Taper P pour pile, F pour face") Saisir (choixUtil) FinPour Afficher ("Sur 10 parties, vous en avez gagné ", **hasard(0,10)**) Fin

Dans cette version, on se contente, dans l'itération, de récupérer les choix de l'utilisateur, ce qui lui donne l'impression que l'on prend ses choix successifs en compte dans le calcul du nombre de gains.

Mais en fait, c'est seulement au moment de l'affichage que l'on choisit aléatoirement un nombre de gains compris entre 0 (zéro) et 10.

Exercice 42

Version 1 : écrire l'algo qui compte et affiche le nombre de caractères et de mots dans une phrase saisie au clavier. La phrase saisie est affectée à une chaîne de caractères. On suppose pour le moment que l'utilisateur ne fait pas d'erreur de saisie (du genre 2 espaces au lieu d'1). L'espace entre 2 mots n'est pas considéré comme un caractère, le point n'est pas non plus un caractère.

Version 2 : modifier l'exo 42 pour qu'il sache compter les phrases, les mots et les lettres dans un texte saisi puis nettoyé au préalable (texte auquel on a enlevé tous les espaces inutiles)

Commençons ci-dessous par la version 1 de l'exercice 42.

Une remarque au préalable : on doit initialiser le nombre de mots à 1 et pas à 0 car le dernier mot de la phrase n'étant pas suivi d'un espace, il ne sera pas comptabilisé.

Lexique	Algo exo 42 – Compter les mots et les lettres dans une phrase Version 1

phrase (chaîne, saisie) : phrase dont on veut compter les mots et les caractères.
nbMots (entier, calculé) : nombre de mots de la phrase.
nbCar (entier, calculé) : nombre de caractères de la phrase.
i (entier, calculé) : compteur d'itérations.
longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.
	Début
	Afficher ("Saisissez la phrase")
	Saisir (phrase)
	// Initialisation du nombre de mots et du nombre de
	// caractères
	nbMots ← 1
	nbCar ← 0
	// L'itération s'effectue sur la longueur de la chaîne qui
	// contient la phrase
	Pour i de 1 à longueur(phrase) (augmenter de 1)
	Faire Si phrase[i] = " "
	Alors nbMots ← nbMots + 1
	Sinon nbCar ← nbCar + 1
	FinSi
	FinPour
	Afficher ("Dans cette phrase, il y a ", nbMots, " mots et ", nbCar, "caractères.")
	Fin

Il existe une façon de faire plus rigolote pour cette version 1. Cette autre façon de faire consiste à compter le nombre d'espaces que l'on rencontre dans la phrase et à en déduire le nombre de mots contenus dans la phrase.

Lexique	Algo exo 42 – Compter les mots et les lettres dans une phrase Version 1 – Autre façon de faire

phrase (chaîne, saisie) : phrase dont on veut compter les mots et les caractères.
nbEspaces (entier, calculé) : nombre de mots de la phrase.
i (entier, calculé) : compteur d'itérations.
longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.
	Début
	Afficher ("Saisissez la phrase")
	Saisir (phrase)
	// Initialisation du nombre d'espaces
	nbEspaces ← 0
	// L'itération s'effectue sur la longueur de la chaîne qui
	// contient la phrase
	Pour i de 1 à longueur(phrase) (augmenter de 1)
	Faire Si phrase[i] = " "
	Alors nbEspaces ← nbEspaces + 1
	FinSi
	FinPour
	Afficher ("Dans cette phrase, il y a ", nbEspaces + 1, " mots et ", longueur(phrase) – 1 - nbEspaces, "caractères.")
	Fin

Voilà. Passons maintenant à la version 2 de l'exercice 42.

Nous commençons par la partie de l'algorithme permettant de nettoyer le texte.

Lexique	Algo compter les phrases, les mots et les lettres dans un texte nettoyé // Partie de l'algo concernant uniquement le nettoyage du texte

texte (chaîne, saisi) : texte dont on veut compter les mots et les caractères.
textePropre(chaîne, calculé) : texte nettoyé dont on veut compter les mots et les caractères.
indPropre(entier, calculé) : indice du caractère courant dans textePropre.
i (entier, calculé) : compteur d'itérations et indice du caractère courant dans texte.
longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.
	Début
	Afficher("Saisissez le texte")
	Saisir (texte)
	// Nettoyage du texte
	textePropre ← ""
	indPropre ← 1
	Pour i de 1 à longueur(texte)
	Faire Si (i <> 1 et texte[i – 1] <> " ") ou (texte[i] <> " ")
	Alors textePropre[indPropre] ← texte[i]
	indPropre ← indPropre + 1
	FinSi
	FinPour
	// Le texte est nettoyé
	….
	Fin

Voici maintenant l'algorithme complet :

Lexique	Algo compter les phrases, les mots et les lettres dans un texte nettoyé

texte (chaîne, saisi) : texte dont on veut compter les mots et les caractères.
textePropre(chaîne, calculé) : texte nettoyé dont on veut compter les mots et les caractères.
indPropre(entier, calculé) : indice du caractère courant dans textePropre.
i (entier, calculé) : compteur d'itérations et indice du caractère courant dans texte.
longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.
nbPhrases (entier, calculé) : nombre de phrases du texte.
nbMots (entier, calculé) : nombre de mots du texte.
nbCar (entier, calculé) : nombre de caractères du texte.
	Début
	Afficher("Saisissez le texte")
	Saisir (texte)
	// Nettoyage du texte
	textePropre ← ""
	indPropre ← 1
	Pour i de 1 à longueur(texte)
	Faire Si (i <> 1 et texte[i – 1] <> " ") ou (texte[i] <> " ")
	Alors textePropre[indPropre] ← texte[i]
	indPropre ← indPropre + 1
	FinSi
	FinPour
	// Le texte est nettoyé
	// Initialisation du nombre de mots, de phrases et de
	// caractères
	nbPhrases ← 0
	nbMots ← 1
	nbCar ← 0
	// L'itération s'effectue sur la longueur de la chaîne qui
	// contient le texte nettoyé
	Pour i de 1 à longueur(textePropre) (augmenter de 1)
	Faire Si textePropre[i] = "."
	Alors nbPhrases ← nbPhrases + 1
	Sinon Si textePropre[i] = " " ettextePropre[i+1] <> "." et textePropre[i-1] <> "."
	Alors nbMots ← nbMots + 1
	Sinon Si textePropre[i] <> "." et textePropre[i] <> " "
	Alors nbCar ← nbCar + 1
	FinSi
	FinSi
	FinPour
	Afficher ("Dans ce texte, il y a ", nbPhrases, " phrases, ", nbMots, " mots et ", nbCar, "caractères.")
	Fin

On peut, comme pour la version 1 de cet exercice, compter le nombre de points et d'espaces et en déduire le nombre de caractères, le nombre de mots et le nombre de phrases.

Le nombre de mots est égal au nombre d'espaces + 1, le nombre de phrases est égal au nombre de points, et le nombre de caractères, c'est la longueur du texte moins le nombre de points et d'espaces.

Passons maintenant au corrigé des exercices avec boucles imbriquées.

Exercice 43

Écrire l'algo d'affichage des nombres parfaits compris entre 0 et 20000.

Lexique	Algo nombres parfaits compris entre 0 et 20000
i (entier, calculé) : compteur des 20000 itérations. cumul (entier, calculé) : cumul des diviseurs du nombre courant. Prend des valeurs successives au cours des itérations. cpteur (entier, calculé) : compteur d'itération qui sert éventuellement de diviseur du nombre courant. DIV : opération de division entière. MOD : opération qui calcule le reste de la division entière.	Début Afficher ("Liste des nombres parfaits compris entre 1 et 20000 :) Pour i de 1 à 20000 Faire // test sur le nombre i, pour voir s'il est parfait ou non cumul ← 1 Pour cpteur de 2 à (i DIV 2) (augmenter de 1) Faire Si i MOD cpteur = 0 Alors cumul ← cumul + cpteur FinSi FinPour Si cumul = i et i <> 1 Alors afficher (i) FinSi FinPour Fin

Exercice 44

Écrire l'algorithme qui affiche les tables de multiplication de 1 à 10.

Lexique	Algorithme tables du fois
i, j (entiers, calculés) : compteurs d'itérations.	Début Pour i de 1 à 10 Faire Afficher ("Table du ", i) Pour j de 1 à 10 Faire Afficher (i, " * ", j, " = ", i * j) FinPour FinPour Fin

Exercice 45

Écrire l'algorithme qui affiche la moyenne de x élèves d'une classe à des devoirs ainsi que la moyenne générale de la classe. x et le nombre de devoirs pour chaque élève sont saisis. Le nombre de devoirs n'est pas forcément le même pour chaque élève.

Lexique	Algo moyenne
totMoy (entier) : cumul des moyennes . nbEleves (entier, saisi) : nombre d'élèves. compteur (entier) : compteur des boucles effectuées sur le nombre d'élèves. nbDev (entier, saisi) : nombre de devoirs de l'élève courant. totEleve (entier, calculé) : cumul des notes de l'élève courant. cptDev (entier, calculé) : compteur d'itérations sur le nombre de devoirs de l'élève courant. note (entier, saisi) : note courante. moyEleve (entier, calculé) : moyenne de l'élève courant. moyenne (réel, calculé) : moyenne générale.	Début totMoy ← 0 Saisir (nbEleves) Pour compteur de 1 à nbEleves Faire Afficher("Nombre de devoirs du", compteur, "° élève) Saisir (nbDev) totEleve ← 0 Pour cptDev de 1 à nbDev Faire Afficher("Entrez la ", cptDev, "° note ") Saisir (note) totEleve ← totEleve + note FinPour moyEleve ← totEleve/nbDev Afficher (moyEleve) totMoy ← totMoy + moyEleve FinPour moyenne ← totMoy / nbEleves Afficher ("La moyenne générale est de :", moyenne) Fin

Partie 2

La structure TantQue…FinTantQue

Exercice 46 *(reprise de l'exo 31)*

Écrire l'algo de calcul et d'affichage de la moyenne d'un groupe d'élèves à un devoir, le nombre d'élèves étant saisi.

Une remarque : j'ai prévu ici le cas où l'utilisateur saisisse 0 comme nombre d'élèves. Si nbEleves vaut 0 alors on n'entrera pas dans le TantQue car à l'entrée dans le TantQue, compteur vaut 0. Par contre, il faut ensuite tester la valeur de nbEleves (ou de compteur) car s'il valent 0, il ne faut pas calculer la moyenne (une division par 0 entraînant automatiquement une erreur lors de l'exécution, c'est-à-dire un plantage carabiné).

Lexique	Algo moyenne
somme (entier, calculé) : somme des notes successives saisies . compteur (entier, calculé) : compteur de boucles. nbEleves (entier, saisi) : nombre d'élèves. note (entier, saisi) : note courante. moyenne (réel, calculé) : moyenne à calculer.	Début somme ← 0 compteur ← 0 Saisir (nbEleves) TantQue compteur <> nbEleves Faire compteur ← compteur + 1 Afficher("Entrez la ", compteur, "° note ") Saisir (note) somme ← somme + note FinTantQue Si nbEleves <> 0 Alors moyenne ← somme / nbEleves Afficher ("La moyenne est de :", moyenne) FinSi Fin

Le comportement de cet algo est exactement le même que celui de l'algo correspondant utilisant un **pour**.

Vous avez remarqué qu'on a dû rajouter l'instruction **compteur ← compteur + 1** car dans cette structure, le changement de valeur de la condition d'entrée dans la boucle n'est pas automatique.

Exercice 47 *(reprise de l'exo 31 avec modification de la condition d'arrêt)*

Écrire l'algo de calcul et d'affichage de la moyenne d'un groupe d'élèves à un devoir, l'algo demandant à l'utilisateur : "Saisir encore une note (o pour oui, n pour non) ?"

La correction de cet exo est dans le cours.

Exercice 48 *(reprise exo 34)*

Ecrivez en utilisant une structure TantQue l'algorithme qui calcule et affiche le salaire de chacun des employés d'une entreprise, et qui calcule et affiche également le salaire net moyen des employés. Pour chaque employé, on affiche le nom et le salaire net. Le salaire horaire, le nombre d'heures travaillées, le taux de retenue, le montant de la prime et le nom de l'employé sont saisis au clavier.

Dans cette entreprise, les employés sont payés à l'heure (ils ont tous le même salaire horaire) et bénéficient d'une prime d'ancienneté. La règle de calcul d'un salaire net est la suivante : ((nombre d'heures travaillées * salaire horaire) + prime) * (1 – taux de retenue). La fin de la saisie est signalée par une réponse de l'utilisateur à un message. On n'entre dans le traitement de calcul du salaire que si l'utilisateur est d'accord pour saisir le premier employé et on n'affiche les résultats du traitement que si l'utilisateur a saisi au moins 1 employé.

Lexique	Algo calcul du salaire net

salHor (réel, saisi) : salaire horaire.
tauxRet (réel, saisi) : taux de retenue.
unMoinsTauxRet (réel, calculé) : 1 moins le taux de retenue.
rep(caractère, saisi) : réponse de l'utilisateur.
totSal (réel, calculé) : valeurs successives du cumul des salaires nets.
nbEmp (entier, calculé) : nombre d'employés.
nom (chaîne, saisie) : nom de l'employé courant (c'est-à- dire en train d'être traité).
nbH (entier, saisi) : nombre d'heures travaillées par l'employé courant.
prime (réel, saisi) : prime de l'employé courant.
	Début
	Afficher ("Saisissez le salaire horaire")
	Saisir (salHor)
	Afficher ("Saisissez le taux de retenue")
	Saisir (tauxRet)
	unMoinsTauxRet ← 1 – tauxRet
	Afficher ("Voulez-vous saisir le premier salaire ? (o pour oui, n pour non))
	Saisir (rep)
	// Initialisation du cumul des salaires et du nombre d'employés
	totSal ← 0, nbEmp ← 0
	TantQue rep = "o"
	Faire nbEmp ← nbEmp + 1
	Afficher ("Saisissez le nom de l'employé")
	Saisir (nom)
	Afficher ("Saisissez le nombre d'heures travaillées")
	Saisir (nbH)
	Afficher ("Saisissez le montant de la prime")
	Saisir (Prime)
	salNet ← (nbH * salHor + prime) * unMoinTauxRet
	// Calcul du cumul
	totSal ← totSal + salNet
	Afficher ("L'employé (e) ", nom, " doit percevoir un salaire net de ", salNet, "euros.")
	Afficher ("Un autre employé ? (o pour oui, n pour non)")
	Saisir (rep)
	FinTantque
	// Calcul et affichage du salaire moyen si au moins 1 employé a
	// été saisi
	Si nbEmp <> 0
	Alors Afficher ("Le salaire moyen est de ", totSal/nbEmp, "euros.")
	Sinon Afficher ("Aucune saisie, fin de ce programme")
	FinSi
	Fin

Exercice 49 *(reprise de l'exo 35)*

Un nombre est dit « parfait » s'il est égal à la somme de tous ses diviseurs sauf lui-même. Écrire l'algorithme qui indique si un nombre saisi au clavier est parfait ou non.

Lexique	Algo Nombre parfait, pour montrer que parfois, utiliser un TantQue au lieu d'un pour est inutile
nb (entier, saisi) : nombre à tester. cumul (entier, calculé) : cumul des diviseurs du nombre. Prend des valeurs successives au cours des itérations. cpteur (entier, calculé) : compteur d'itérations qui sert également de diviseur éventuel. DIV : opération de division entière. MOD : opération qui calcule le reste de la division entière.	Début Afficher("Saisissez le nombre") Saisir (nb) cumul ← 1 cpteur ← 2 TantQue cpteur <= (nb DIV 2) Faire Si nb MOD cpteur = 0 Alors cumul ← cumul + cpteur FinSi cpteur ← cpteur + 1 FinPour Si cumul = nb et nb <> 1 Alors afficher (nb, " est un nombre parfait") Sinon afficher (nb, " n'est pas un nombre parfait") FinSi Fin

Exercice 50

Découverte du nombre magique.

Le but du jeu est de découvrir un nombre compris entre 1 et 100 choisi par l'ordinateur.

Déroulement du jeu

L'utilisateur saisit un nombre. L'ordinateur lui indique si ce nombre est égal, plus petit ou plus grand que le nombre à découvrir.

En cas d'égalité, l'ordinateur donnera le nombre de coups qui ont été nécessaires pour découvrir ce nombre, l'utilisateur a droit à 5 coups au maximum.

Lexique	Algo Nombre magique version 1 (avec une valeur initiale bidon pour le nombre utilisateur)

hasard (fonction) : choisit aléatoirement un nombre entier entre 1 et la valeur spécifiée en paramètre.
nbOrdi (entier, calculé) : valeur du nombre choisi aléatoirement.
nbCoups (entier, calculé) : compteur de tentatives.
nb (entier, saisi) : nombre courant proposé par l'utilisateur.
	Début
	nbOrdi ← hasard(100), nbCoups ← 0
	nb ← -1
	//On initialise nb à une valeur à propos de laquelle on est
	// sûr que l'ordinateur ne pourra pas la choisir aléatoirement
	TantQue nb <> nbOrdi et nbCoups < 5
	Faire Afficher ("Entrer votre nombre ")

```
        Saisir (nb)
        Si (nb > nbOrdi)
        Alors   Afficher ("Votre nombre est trop grand ")
        Sinon   Si (nb < nbOrdi)
                Alors Afficher ("Votre nombre est trop petit ")
                FinSi
        FinSi
        nbCoups ← nbCoups +1
FinTantQue
// On sort si nbCoups = 5 ou nb = nbOrdi
Si (nb = nbOrdi)
// On teste pour savoir pourquoi on est sorti de la
// boucle : si c'est parce que l'utilisateur a trouvé le nombre magique,
// ou si c'est parce qu'il est arrivé au bout de ses 5 essais
Alors   Afficher ("Bravo vous avez gagné en ", nbCoups, " coups")
Sinon   Afficher("Dommage, le nombre choisi était : ",nbOrdi)
FinSi
Fin
```

Lexique	Algo Nombre magique version 2 (avec une saisie avant d'entrer dans la boucle)
hasard (fonction) : choisit aléatoirement un nombre entier entre 1 et la valeur spécifiée en paramètre. nbOrdi (entier, calculé) : valeur du nombre choisi aléatoirement. nbCoups (entier, calculé) : compteur de tentatives. nb (entier, saisi) : nombre courant proposé par l'utilisateur.	Début nbOrdi ← hasard(100), nbCoups ← 1 Afficher ("Saisissez votre premier essai") Saisir (nb) TantQue nb <> nbOrdi et nbCoups < 5 Faire Si (nb > nbOrdi) Alors Afficher ("Votre nombre est trop grand ") Sinon Si (nb < nbOrdi) Alors Afficher ("Votre nombre est trop petit ") FinSi FinSi Afficher ("Entrer votre nombre ") Saisir (nb) nbCoups ← nbCoups +1 FinTantQue Si (nb = nbOrdi) Alors Afficher ("Bravo vous avez gagné en ", nbCoups, " coups") Sinon Afficher("Dommage, le nombre choisi était : ", nbOrdi) FinSi Fin

Remarque : écrire cet algo avec un TantQue n'est pas « le top du top », en effet, on est obligé de « bidouiller » pour entrer la première fois dans la boucle : on donne une valeur bidon au nombre magique.

Le traitement itératif le plus adapté à cet algo est celui que l'on va voir plus loin : le traitement **répéter**... pour lequel on est sûr d'entrer au moins une fois dans la boucle.

Exercice 51

Algo PV : si un conducteur a entre 0.5 et 0.8 gramme d'alcool par litre de sang, il a un retrait de permis de 6 mois et une amende de 200 euros par décigramme au dessus de 0.5 gramme, 0.5 gramme inclus.

Si le conducteur a entre 0.8 et 1 gramme d'alcool dans le sang, il a un retrait de permis de 24 mois et une amende de 300 euros par décigramme au dessus de 0.5 gramme, 0.5 gramme inclus.

Au-delà de 1 gramme, le conducteur repasse son permis et paye une amende de 5 000 euros.

Un litre d'alcool pur pèse 800 grammes. Un homme adulte possède environ 8 litres de sang. Une bouteille d'alcool à 45° possède 45% d'alcool.

À titre indicatif, un verre de 10 cl de boisson alcoolisée à 10° donne une alcoolémie de 0.1 g par litre de sang.

Écrire un algorithme qui calcule l'amende à payer et le temps de retrait de permis, la quantité de boisson et la teneur en alcool de chaque boisson étant saisies. L'utilisateur indique en saisissant le nombre 0 qu'il ne désire plus saisir de nouvelle boisson.

Lexique	Algo PV avec plusieurs boissons

alcoolemie (réel, calculé) : taux d'alcool par litre de sang.
qteBue (entier, saisi) : quantité bue en cl.
teneur (entier, saisi) : teneur en alcool de la boisson bue.
retrait (chaîne, calculé) : durée du retrait de permis.
amende (entier, calculé) : montant de l'amende à payer.
	Début
	alcoolemie ← 0
	// Pour voir si l'utilisateur veut saisir au moins une boisson
	Afficher ("Saisissez la quantité de boisson bue ("0" pour arrêter))
	Saisir (qteBue)
	// On récupère la première quantité qu'il saisit
	// et on n'entre dans la boucle que si elle est <> de 0
	TantQue qteBue <> 0
	Faire Afficher ("Saisissez la teneur en alcool de cette boisson")
	Saisir (teneur)
	alcoolemie ← alcoolemie + teneur * qteBue / 1000
	Afficher ("Saisissez la teneur et la quantité de boisson ("0" pour arrêter))
	Saisir (qteBue)
	// On récupère à chaque tour de boucle
	// la quantité de la boisson suivante, si cette quantité
	// est nulle, on ne re-rentre pas dans la boucle.
	FinTantQue
	Si alcoolemie >= 0.5

```
Alors   Si      alcoolemie < 0.8
        Alors   retrait ← "6 mois"
                amende ← (alcoolemie – 0.4) * 2 000
        Sinon   Si      alcoolemie < 1
                Alors   retrait ← "2 ans"
                        amende ← (alcoolemie – 0.4) * 3 000
                Sinon   retrait ← "retrait définitif"
                        amende ← 5 000
                FinSi
        FinSi
        Afficher ("Vous avez un retrait de ", retrait, et une amende de ", amende)
Sinon   Afficher ("Votre taux d'alcoolémie de ", alcoolemie," ne dépasse pas le seuil autorisé")
Finsi
Fin
```

Vous avez remarqué qu'on est obligé de répéter 2 fois

Afficher ("Saisissez la teneur et la quantité de boisson ("0" pour arrêter))

Saisir (qteBue)

Une fois avant d'entrer dans la boucle, et une fois dans la boucle. La première des 2 fois s'appelle une **lecture avancée**, on fait ça pour être sûr qu'on doit entrer au moins une fois dans la boucle.

Exercice 52 *(reprise de l'exo 37)*

Écrire l'algorithme d'affichage du plus grand de x nombres saisis au clavier, la fin de la saisie étant signalée par une réponse de l'utilisateur à un message du style "encore un nombre? (o pour oui, n pour non)".

Lexique	Algo Maximum de x nombres
nombre (entier, saisi) : nombre courant à comparer au maximum. rep (caractère, saisi) : réponse de l'utilisateur. max (entier, calculé) : maximum des nombres.	Début Afficher ("Saisir le premier nombre") Saisir (max) Afficher ("encore un nombre?(o pour oui, n pour non)") Saisir (rep) TantQue rep <> "n" Faire Afficher ("Saisir un nombre") Saisir (nombre) Si nombre > max Alors max ← nombre FinSi FinTantQue Afficher ("Le maximum est : ", max) Fin

Exercice 53 *(reprise de l'exo 36)*

Écrire l'algorithme du jeu suivant : ce jeu se joue à 2 joueurs.

Le premier joueur saisit un mot à l'abri du regard du deuxième joueur. Puis, le deuxième joueur doit deviner quel est ce mot et le saisir. Il a droit à un nombre d'essais fixé par le premier joueur. A chaque essai, le programme indique au joueur s'il a trouvé le mot ou bien, le cas échéant, quelles sont les bonnes lettres aux bonnes places parmi celles qu'il a saisies.

Exemple : si le mot à trouver est « carpette » et que le joueur a saisi « cratères », le programme lui affiche : c - - - e - - -.

Si, au bout du nombre d'essais autorisés, le joueur n'a pas trouvé le mot, un message lui indique quel était le mot à trouver.

Lexique	Algo : Deviner un mot

motATrouver (chaîne, saisie) : Mot que le joueur 2 doit trouver.
essaisAutor (entier, saisi) : nombre d'essais autorisés.
essai (chaîne, saisie) : mot saisi par le joueur 2.
nbEssais (entier, calculé) : nombre d'essais courant.
i (entier, calculé) : compteur d'itérations, sur la longueur du mot à trouver.
longueur (fonction) résultat : entier. Fonction qui retourne le nombre de caractères contenus dans la chaîne passée en paramètre.
	Début
	Afficher ("Joueur 1, tu dois saisir un mot ")
	Saisir (motATrouver)
	Afficher ("Nombre d'essais autorisés")
	Saisir (essaisAutor)
	Afficher ("Joueur 2, tu dois trouver le mot ")
	Afficher ("Tu as droit à ", essaisAutor," essais")
	Afficher ("Premier essai :")
	Saisir (essai)
	nbEssais ← 1
	TantQue essai <> motATrouver et nbEssai < essaisAutor
	Faire Pour i de 1 à longueur(motATrouver)
	Faire Si essai[i] <> motATrouver[i]
	Alors essai[i] ← "-"
	FinSi
	FinPour
	Afficher ("Voici les bonnes lettres :", essai)
	nbEssais ← nbEssais + 1
	Si nbEssais = essaisAutor
	Alors Afficher ("Saisis ton dernier essai")
	Sinon Afficher ("Joueur 2, saisis ton", nbEssais, "° essai")
	FinSi
	Saisir (essai)
	FinTantQue
	// Si on sort, c'est que le joueur a trouvé le mot ou bien qu'il
	// est arrivé au bout du nombre d'essais autorisés
	Si essai = motATrouver
	Alors Afficher ("Bravo, tu as enfin trouvé le mot")
	Sinon Afficher ("Tu as définitivement perdu, le mot à trouver était ", motATrouver)
	FinSi
	Fin

Exercice 54 *(Reprise de l'exo 40 sauf que …)*

Votre client est agent secret et écrit ses messages en langage crypté. Il vous demande de lui écrire un programme qui crypte le texte qu'il saisit. Ce cryptage consiste à inverser le mot et à intercaler entre chacune de ses lettres la lettre qui a la position symétrique dans l'alphabet. Je vous explique : l'alphabet comporte 26 lettres : a b c d e f g h i j k l m n o p q r s t u v w x y z. Dans l'alphabet a est la symétrique de z, b est la symétrique de y, c est la symétrique de x,…, et ainsi de suite jusqu'à m qui est la symétrique de n.

Repartez de la dernière version du corrigé de l'exo 40 mais modifiez-le de manière à ce que les phrases et les mots soient cryptés dans l'ordre où ils sont saisis dans la phrase.

Note : les points et les espaces ne font pas l'objet d'un cryptage particulier.

Lexique	Algo cryptage de texte

texte (chaîne, saisi) : texte à crypter.
texteCrypte (chaîne, calculé) : texte crypté.
i (entier, calculé) : indice de parcours du texte.
mot (chaîne, calculé) : mot courant à crypter.
motCrypte (chaîne, calculé) : mot courant crypté.
	Début
	Afficher ("Saisir le texte à crypter")
	Saisir (texte)
	texteCrypte ← "", i ← 1
	TantQue i < longueur(texte) – 1
	// On itère sur longueur(texte) – 1 car
	// on sait que le dernier caractère est un point et aussi car on utilise
	// plus bas texte[i+1] et si on itère sur longueur(texte), i+1 peut
	// dépasser la longueur du texte de 1 caractère et générer une erreur
	Faire // On recherche un mot
	// On parcourt le texte jusqu'à ce qu'on rencontre un espace ou
	// bien un point et un espace
	// initialisation du mot courant et du mot crypté courant
	mot ← "", motCrypte ← ""
	TantQue non (texte[i] = " " ou texte[i] = "." et texte[i+1] = " ")
	// J'avais la flemme de calculer la négation de cette expression
	// booléenne, alors je la laisse comme ça,
	// Ben quoi, vous avez jamais la flemme vous ?
	Faire mot ← mot + texte[i]
	i ← i + 1
	FinTantQue
	// Si on sort, c'est qu'on a un mot. On le crypte.
	Pour j de longueur(mot) à 1 (diminuer de 1)
	Faire motCrypte ← motCrypte + texte[i] + chr(219 - ascii(texte[i]))
	FinPour
	// On rajoute le mot crypté au texte crypté
	texteCrypte ← texteCrypte + motCrypte
	// On rajoute aussi l'espace ou le point et l'espace et on avance
	Si texte[i] = "."
	Alors texteCrypte ← texteCrypte + "."
	i ← i + 1
	FinSi

```
        Si          texte[i] = " "
        Alors       texteCrypte ← texteCrypte + " "
                    t ← i + 1
        FinSi
FinTantQue
// On rajoute le dernier point et on affiche
Afficher ("Le texte ", texte, "une fois crypté devient :", texteCrypte + ".")
Fin
```

Bon c'est sûr, il suffit qu'on ne mette pas le point à la fin du texte pour que le dernier caractère du dernier mot soit « mangé » (c'est-à-dire ne soit pas crypté). Il aurait fallu encore et encore rajouter des tests, et comme je vous l'ai dit, j'avais la flemme. Mais rien ne vous empêche de le faire… à moins que… vous aussi… un petit coup de flemme ?

Exercice 55

(alors là, accrochez-vous, personnellement je l'adore celui-là)

Écrire l'algo qui affiche les déplacements d'une balle de ping-pong qui rebondit sur les bords d'un carré, dont la taille est choisie par l'utilisateur.

La balle part d'une position aléatoire au bas du carré. La direction que prend la balle après avoir rebondi est le résultat d'un choix aléatoire entre gauche ou droite.

L'algo s'arrête quand la balle a rebondi 40 fois.

La balle est représentée par un petit bouton dont on change la position par programmation. On suppose que la taille du bouton est de 1 sur 1.

On suppose également qu'il existe pour les objets graphiques les propriétés « colonne » et « ligne » qui contiennent les coordonnées de l'objet par rapport au bord de la fenêtre sur laquelle il est posé, ces coordonnées pouvant se modifier par programmation.

Lexique	Algo balle de ping-pong

tailleCarre (entier, constante) : taille du carré.
droite, haut (booléens, calculés) : vrais si on doit aller vers la gauche, la droite.
colonne, ligne (entiers, calculés) : numéro de la colonne, ligne courante.
hasard (fonction, résultat : entier) : retourne un entier choisi aléatoirement, compris entre les 2 bornes passées en paramètre.
i (entier, calculé) : compteur de trajets.
	Début
	// Taille du carré saisie
	saisir (tailleCarre)
	// Choix aléatoire entre gauche ou droite
	si hasard (0,1) = 0
	alors droite ← vrai
	FinSi
	// Les lignes d'instructions précédentes peuvent être
	// avantageusement remplacées par : droite ← (hasard(0,1) = 0)
	// Initialisation de la direction verticale
	haut ← vrai
	// Choix aléatoire pour la colonne de départ
	balle.colonne ← hasard (Carre.colonne, Carre.colonne + taillecarre)
	// Positionnement de départ
	balle.ligne ← Carre.ligne + tailleCarre
	Pour i de 1 à 40 (augmenter de 1)
	Faire // On décrit un trajet
	Tantque (balle.colonne >= carre.colonne)
	et (balle.colonne <= carre.colonne + carre.largeur)
	et (balle.ligne >= carre.ligne)
	et (balle..ligne <= carre.ligne + carre.hauteur)
	Faire si droite
	Alors balle.colonne ← balle.colonne +1
	Sinon balle.colonne ← balle.colonne -1
	finsi
	si haut
	alors balle.ligne ← balle.ligne - 1
	sinon balle.ligne ← balle.ligne + 1
	finsi
	FinTantQue
	// Test pour changement de direction verticale ou horizontale
	si i mod 2 = 0
	alors haut ← non haut
	sinon droite ← non droite
	finsi
	FinPour
	Fin

Partie 3

La structure Répéter…jusqu'à

Exercice 56 *(reprise de l'exo 31 et de l'exo 46)*

Écrire l'algo de calcul et d'affichage de la moyenne d'un groupe d'élèves à un devoir, le nombre d'élèves étant saisi.

La correction de cet exo est dans le cours.

Exercice 57 *(reprise de l'exo 50)*

Découverte du nombre magique.

Le but du jeu est de découvrir un nombre compris entre 1 et 100 choisi par l'ordinateur.

L'utilisateur saisira un nombre. L'ordinateur lui dira alors si ce nombre est plus petit ou plus grand que celui qu'il a choisi. En cas d'égalité, l'ordinateur donnera le nombre de coups qui ont été nécessaires pour découvrir ce nombre, l'utilisateur a droit à 5 coups au maximum.

Lexique	Algo Nombre magique

nbOrdi (entier) : valeur du nombre choisi aléatoirement.
hasard (fonction) : choisit aléatoirement un nombre entier entre 1 et la valeur spécifiée en paramètre.
nbCoups (entier) : compteur de tentatives.
nb (entier, saisi) : nombre courant proposé par l'utilisateur.
	Début
	nbOrdi ← hasard(100)
	nbCoups ← 0
	Répéter
	Afficher ("Entrer votre nombre ")
	Saisir (nb)
	Si (nb > nbOrdi)
	Alors Afficher ("Votre nombre est trop grand ")
	Sinon Si (nb < nbOrdi)
	Alors Afficher ("Votre nombre est trop petit ")
	FinSi
	FinSi
	nbCoups ← nbCoups +1
	Jusqu'à ((nbCoups = 5) ou (nb = nbOrdi))
	// Si on sort du Répéter, c'est soit parce que l'utilisateur a trouvé le
	// nombre, soit parcequ'il a utilisé les 5 coups.
	Si (nb = nbOrdi)
	Alors Afficher ("Bravo vous avez gagné en ", nbCoups, " coups")
	Sinon Afficher ("Dommage, le nombre choisi était : ", nbOrdi)
	FinSi
	Fin

Exercice 58 *(reprise de l'exo 51)*

Algo PV : si un conducteur a entre 0.5 et 0.8 gramme d'alcool par litre de sang, il a un retrait de permis de 6 mois et une amende de 200 € par décigramme au dessus de 0.5 gramme, 0.5 gramme inclus.

Si le conducteur a entre 0.8 et 1 gramme d'alcool dans le sang, il a un retrait de permis de 24 mois et une amende de 300 € par décigramme au dessus de 0.5 gramme, 0.5 gramme inclus.

Au-delà de 1 gramme, le conducteur repasse son permis et paye une amende 5 000 €.

Un litre d'alcool pur pèse 800 grammes. Un homme adulte possède environ 8 litres de sang. Une bouteille d'alcool à 45° possède 45% d'alcool.

A titre indicatif, un verre de 10 cl de boisson alcoolisée à 10° donne une alcoolémie de 0.1 g par litre de sang.

Écrire un algorithme qui calcule l'amende à payer et le temps de retrait de permis, la quantité de boisson et la teneur en alcool de chaque boisson étant saisies. L'utilisateur indique en saisissant le nombre 0 qu'il ne désire plus saisir de nouvelle boisson.

Lexique	Algo PV avec plusieurs boissons

alcoolemie (réel, calculé) : taux d'alcool par litre de sang.
rep (chaîne, saisie) : réponse de l'utilisateur.
qteBue (entier, saisi) : quantité bue en cl.
teneur (entier, saisi) : teneur en alcool de la boisson bue.
retrait (chaîne, calculé) : durée du retrait de permis.
amende (entier, calculé) : montant de l'amende à payer.
	Début
	alcoolemie ← 0
	// Pour voir si l'utilisateur veut saisir au moins une boisson
	Afficher ("Voulez-vous calculer votre alccolémie ?(oui/non)")
	Saisir (rep)
	// et on n'entre dans la boucle que si la réponse est "oui"
	Si rep = "oui"
	Alors répéter Afficher ("Saisissez la teneur en alcool de la boisson")
	Saisir (teneur)
	Afficher ("Saisissez la quantité de boisson")
	Saisir (qteBue)
	alcoolemie ← alcoolemie + teneur * qteBue / 1000
	Afficher ("Encore une boisson (oui/non)?")
	Saisir (rep)
	Jusqu'à rep = "non"
	Si alcoolemie >= 0.5
	Alors Si alcoolemie < 0.8
	Alors retrait ← "6 mois"
	amende ← (alcoolemie – 0.4) * 2 000
	Sinon Si alcoolemie < 1
	Alors retrait ← "2 ans"
	amende ← (alcoolemie – 0.4) * 3 000
	Sinon retrait ← "retrait définitif"
	amende ← 5 000
	FinSi
	FinSi
	Afficher ("Vous avez un retrait de ", retrait, et une amende de ", amende)
	Sinon afficher ("Votre taux d'alcoolémie de ", alcoolémie," ne dépasse pas le seuil autorisé")
	Finsi
	Fin

Exercice 59 *(Reprise de l'exo 41)*

Écrire un algorithme qui simule le jeu de pile ou face.

Déroulement du jeu : l'utilisateur saisit la lettre P pour pile, et F pour face puis valide sa saisie (ou bien il clique sur le bouton « Pile » ou le bouton « Face » dans le cas d'une interface graphique et événementielle).

Le programme lui, choisit aléatoirement un nombre entre 0 et 1, si le nombre tiré au sort est 0, alors pile est gagnant, face est perdant, si le nombre tiré au sort est 1, alors pile est perdant et face est gagnant.

L'utilisateur fait le nombre d'essais qu'il souhaite et indique qu'il veut arrêter de jouer en saisissant un autre caractère que « P » ou « F ». Alors, la partie s'arrête, affichant son score à l'utilisateur : nombre de gains sur le nombre total de parties.

Lexique	Algo pile ou face

nbParties (entier, calculé) : nombre de parties jouées.
choixUtil (caractère, saisi) : caractère choisi par l'utilisateur.
hasard (fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres.
gains (entier, calculé) : nombre de parties gagnées par le joueur.
	Début
	// Initialisation du compteur de parties
	nbParties ← 0, gains ← 0
	Répéter
	Afficher ("Taper P pour pile, F pour face, une autre lettre pour arrêter")
	Saisir (choixUtil)
	// Ici, on est obligé de mettre un test, car
	// si l'utilisateur a saisi autre chose que
	// "P" ou "F", il ne faut pas lui rajouter de // gains
	si choixUtil = "P" ou choixutil = "F"
	alors gains ← gains + hasard (0,1)
	nbEssais ← nbEssais + 1
	finsi
	Jusqu'à
	choixutil <> "P" et choixUtil <> "F"
	Afficher ("Sur ", nbParties, " parties, vous en avez gagné ", gains)
	Fin

On peut, bien sur, simuler le comptage, comme dans la version que je vous ai proposée dans la partie **TantQue** de ce corrigé.