Séquence 3

Les structures conditionnelles

Partie 1

La structure conditionnelle : Si…Alors…Sinon

Exercice 21

Écrire l'algorithme d'affichage du plus grand de 2 nombres saisis au clavier et répondez à la question suivante : que se passe-t-il si les 2 nombres saisis sont identiques ?

Lexique	Algo Maximum de 2 nombres
n1, n2 (entier, saisis) : nombres à comparer. max (entier) : maximum des 2 nombres.	Début Afficher ("Donner 2 nombres entiers :") Saisir (n1, n2) Si n1 > n2 Alors max ← n1 Sinon max ← n2 FinSi Afficher('Le maximum est : ', max) Fin

Écrire dans un algo **saisir**(n**1,** n**2**) revient à écrire :

saisir(n**1**)

saisir(n**2**),

autrement dit, l'ordinateur attend que l'utilisateur saisisse le premier nombre et presse sur la touche entrée pour le ranger dans n1, puis il attend que l'utilisateur saisisse un deuxième nombre et appuie sur entrée, puis le range dans n2.

On peut aussi écrire l'algo comme suit :

Lexique	Algo Maximum de 2 nombres
n1, n2 (entier, saisis) : nombres à comparer.	Début Afficher ("Donner 2 nombres entiers :") saisir (n1, n2) Si (n1 > n2) Alors Afficher ('Le maximum est : ', n1) Sinon Afficher ('Le maximum est : ', n2) FinSi Fin

Par contre, si on vient à augmenter le nombre de nombres à comparer, la première façon est plus adaptée (à bon entendeur, salut…).

Exercice 22

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.

Lexique	Algo pile ou face
choixUtil(caractère, saisi) : caractère choisi par l'utilisateur (P ou F). hasard(fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres. choixOrdi(entier, calculé) : résultat d'appel de la fonction hasard. carChoixOrdi(caractère, calculé) : caractère résultant du résultat d'appel de la fonction hasard.	Début Afficher ("Taper P pour pile, F pour face") Saisir (choixUtil) choixOrdi ← hasard(0,1) Si choixOrdi = 0 Alors carChoixOrdi ← "P" Sinon carChoixOrdi ← "F" Finsi Si choixUtil = choixOrdi Alors Afficher ("Gagné") Sinon Afficher ("Perdu") FinSi Fin

Je voudrais profiter de cet algo pour faire une remarque concernant la ligne d'instruction **choixOrdi ← casard(0,1)**.

Cette ligne est le cas typique où il ne faut appeler qu'une fois la fonction et stocker son résultat dans une variable en début de traitement.

Je m'explique : imaginez qu'à chaque fois que l'on a besoin du choix de l'ordinateur, on rappelle la fonction **hasard(0,1)**, si on fait ça, on n'est absolument pas assuré qu'à chaque fois que l'on rappelle la fonction **hasard(0,1)** l'ordinateur choisisse le même nombre (il peut très bien choisir une fois 0, une fois 1, ou bien 2 fois 0, ou bien 2 fois 1, on ne sait pas puisque le choix est aléatoire).

On peut écrire plus simplement cet algo. Si on a "Pile" et 0 ou bien "Face" et 1, alors c'est gagné, sinon, c'est perdu :

Lexique	Algo pile ou face
choixUtil(caractère, saisi) : caractère choisi par l'utilisateur. hasard(fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres. choixOrdi(entier, calculé) : résultat d'appel de la fonction hasard.	Début Afficher ("Taper P pour pile, F pour face") Saisir (choixUtil) choixOrdi ← hasard(0,1) Si choixOrdi = 0 et choixUtil = "P" ou choixOrdi = 1 et choixUtil = "F" Alors Afficher ("Gagné") Sinon Afficher ("Perdu") Fin

Il y a une dernière façon d'écrire ce programme de pile ou face, à mon avis la plus futée d'entre toute. Elle consiste à ne pas tenir compte du choix effectif de l'utilisateur pour décider s'il a gagné ou non :

Lexique	Algo pile ou face

choixUtil (caractère, saisi) : caractère choisi par l'utilisateur.
hasard (fonction, résultat : entier) : retourne un nombre choisi aléatoirement entre les 2 entiers passés en paramètres.
choixOrdi (entier, calculé) : résultat d'appel de la fonction hasard.
message (chaîne, calculé) : message indiquant à l'utilisateur s'il a gagné ou perdu sa partie de pile ou face.
	Début
	Afficher ("Taper P pour pile, F pour face")
	Saisir (choixUtil)
	// Voilà, on ne fait que récupérer le choix de
	// l'utilisateur, mais après, on ne s'en sert pas.
	// Le fait que l'utilisateur gagne ou perde
	// dépend en fait uniquement du résultat
	// d'appel de la fonction hasard
	Si hasard (0,1) = 0
	Alors message ← "Gagné"
	Sinon message ← "Perdu"
	Finsi
	Afficher (message)
	Fin

Exercice 23

Écrire l'algorithme qui calcule et affiche le maximum de 3 nombres saisis au clavier par l'utilisateur.

Attention, il n'est pas question de « sortir l'artillerie lourde » en cherchant quel est le plus grand nombre, quel est le 2e plus grand et quel est le plus petit.

Tout ce qu'on veut, c'est le plus grand.

Il n'est donc pas nécessaire de comparer ces 3 nombres 2 à 2, ce qui ferait 4 comparaisons.

Deux comparaisons suffisent pour obtenir ce qu'on veut.

Lexique	Algo Maximum de 3 nombres

n1, n2, n3 (entiers, saisis) : nombres à comparer.
max (entier, calculé) : maximum des 3 nombres.
	Début
	Afficher ("Donner 3 nombres :")
	Saisir (n1, n2, n3)
	// On détermine d'abord quel nombre entre **n1**
	// et **n2** est le plus grand et ce nombre devient
	// le **maximum provisoire**.
	Si n1 > n2
	Alors max ← n1
	Sinon max ← n2
	FinSi
	// Maintenant, on cherche à savoir si **max** est
	// réellement le maximum des 3 nombres. Le cas
	// échéant, c'est **n3** qui deviendra le maximum définitif
	// sinon, notre maximum provisoire devient le
	// maximum définitif
	Si n3 > max
	Alors max ← n3
	FinSi
	Afficher ("Le maximum est : ", Max)
	Fin

Partie 2

Imbrication de Si

Exercice 24

Même exercice que l'exercice 21 en prévoyant l'affichage d'un message lorsque les nombres saisis sont égaux.

Algo Maximum	
Lexique	**Algo**
n1, n2 (entier, saisis) : nombres à comparer. max (entier) : maximum des 2 nombres.	Début Afficher ("Donner 2 nombres entiers :") Saisir (n1, n2) Si n1 = n2 Alors Afficher("Les 2 nombres saisis sont égaux") Sinon Si n1 > n2 Alors max ← n1 Sinon max ← n2 Finsi Afficher ("Le maximum est : ", max) FinSi Fin

Ici, la place de ce **finsi** est importante : si on le met sous l'instruction **Afficher('Le maximum est : ', max)**, le message ne s'affichera que dans le cas où n1<= n2.

Avec cet algo, une erreur classique aurait été d'écrire :

Algo Maximum	
Lexique	**Algo**
n1, n2 (entiers, saisis) : nombres à comparer. max (entier) : maximum des 2 nombres.	Début Afficher ("Donner 2 nombres entiers :") Saisir (n1, n2) Si n1 = n2 Alors Afficher("Les 2 nombres saisis sont égaux") Sinon Si n1 > n2 Alors max ← n1 Sinon max ← n2 Finsi FinSi Afficher("Le maximum est : ", max) Fin

Si on écrit cela, alors, même si les 2 nombres saisis sont égaux et qu'on a affiché le message **Les 2 nombres saisis sont égaux**, on affichera aussi le message **Le maximum est : …**, avec éventuellement en prime une erreur à l'exécution car **max** n'a pas été initialisée.

On peut aussi avoir envie de l'écrire comme suit :

Algo maximum	
Lexique	**Algo**
n1, n2 (entier, saisis) : nombres à comparer. max (entier) : maximum des 2 nombres.	Début Afficher ("Donner 2 nombres entiers :") Saisir (n1, n2) Si n1 > n2 Alors max ← n1 Sinon Si n1 > n2 Alors max ← n2 Sinon Afficher ("Les 2 nombres saisis sont égaux") FinSi Finsi Afficher ("Le maximum est : ", max) Fin

Exercice 25

Calcul du salaire d'un employé.

L'utilisateur saisit le nombre d'heures travaillées, le salaire horaire et l'ancienneté de l'employé.

Les retenues de sécurité sociale sont calculées à partir du salaire brut multiplié par le taux de retenue de la sécurité sociale qui est une constante valant 0.19.

L'employé bénéficie d'une prime d'ancienneté qui équivaut à 2% du salaire brut pour + de 10 ans et -20 ans d'ancienneté et 5% du salaire brut pour + 20 ans d'ancienneté.

Lexique	Algo Calcul de salaire avec prime, s'il vous plaît !

taux = 0.19 (constante)
nbH (entier, saisi) : nombre d'heures travaillées.
salH (réel, saisi) : salaire horaire.
anc (entier, saisi) : ancienneté.
coeffPrime(réel, calculé) : coefficient multiplicateur permettant de calculer la prime à partir du salaire brut.
	Début
	Afficher ("Saisissez le nombre d'heures travaillées, le salaire horaire puis l'ancienneté de l'employé")
	Saisir (nbH, salH, anc)
	// On calcule le pourcentage de prime auquel
	// l'employé a droit, c'est plus judicieux que de
	// calculer le montant de la prime, et ça fait
	// moins de calculs, car n'oublions pas qu'un
	// bon informaticien est un "fainéant
	// intelligent".
	Si anc >= 20
	Alors coeffPrime ← 0.05
	Sinon // On entre dans le sinon si anc < 20
	Si anc >= 10
	
	// on arrive ici si anc < 20
	
	//et anc >= 10
	Alors coeffPrime ← 0.02
	Sinon // Ce sinon signifie **anc < 10 et anc < 20**,
	
	// ce qui finalement signifie **anc < 10**. Non ?
	
	// Si, si.
	
	coeffPrime ← 0
	FinSi
	FinSi
	// Maintenant, on n'a plus qu'à calculer d'un
	// coup le salaire net, carrément dans
	// l'instruction d'affichage puisqu'on ne s'en
	// sert qu'une fois, pour l'afficher.
	Afficher ("Net = ", (nbH*Salh)*(1 + coeffPrime)*(1 – taux)
	Fin

Voici une autre solution, à mon avis un peu moins futée, car elle oblige à calculer le brut pour pouvoir calculer le montant de la prime ce qui est dommage car on n'a pas besoin de calculer le brut ailleurs que dans le calcul du salaire net. En outre cette solution effectue des tests pour rien.

Bien sûr, cette solution fonctionne, mais elle est beaucoup moins optimisée que la précédente : plus on tape de lignes dans un programme, plus on risque d'y faire des erreurs et plus longue est la mise au point.

Lexique	Algo calcul de salaire avec prime, autre version, s'il vous plaît

taux = 0.19 (constante)
nbH (entier, saisi) : nombre d'heures travaillées.
salH (réel, saisi) : salaire horaire.
anc (entier, saisi) : ancienneté.
coeffPrime(réel, calculé) : coefficient multiplicateur permettant de calculer la prime à partir du salaire brut.
brut (réel, calculé) : salaire brut.
	Début
	Afficher ("Saisissez le nombre d'heures travaillées, le salaire horaire puis l'ancienneté de l'employé")
	Saisir (nbH, salH, anc)
	// On calcule le brut pour pouvoir calculer le montant de
	// la prime, c'est l'inconvénient de cette version
	brut ← nbh * salH
	// Calcul de la prime
	Si anc >= 10 et anc < 20
	Alors prime ← brut * 0.02
	Sinon // Ce **sinon** signifie **si** anc < 10 ou anc >= 20
	// Ce qui n'est pas très futé non plus car cela oblige
	// à refaire un test pour savoir si on est entré dans
	// le **sinon** parce que **anc < 10** ou bien parce
	// que anc >= 20
	Si anc >= 20 // Et voilà, je l'avais dit, on est
	// obligé de refaire des tests.
	Alors prime ← brut * 0.05
	Sinon prime ← 0
	FinSi
	FinSi
	Afficher ("Net = ", brut - (brut * taux) + prime)
	Fin

Exercice 26

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

Le premier joueur saisit un mot de 4 lettres 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 cape et que le joueur a saisi rate, le programme lui affiche : - a – e.
Si, au bout de 3 essais, le joueur n'a pas trouvé le mot, un message lui indique quel était le mot à trouver.

Dans les séquences suivantes, nous reprendrons cet exercice pour l'améliorer : pour le moment, la solution est très longue et répétitive et il existe en algo des structures qui permettent de gérer « économiquement » le fait qu'il y ait des répétitions.

Lexique	Algo : deviner un mot
motATrouver (chaîne[4], saisie) : mot que le joueur 2 doit trouver. essai (chaîne[4], saisie) : mot saisi par le joueur 2.	Début Afficher("Joueur 1, tu dois saisir un mot de 4 lettres") Saisir(motATrouver) Afficher("Joueur 2, tu dois trouver le mot de 4 lettres.") Afficher("Tu as droit à 3 essais, que la force soit avec toi") Afficher("Premier essai :") Saisir(essai) Si essai = motATrouver Alors Afficher("Bravo, joueur 2, tu as gagné du premier coup") Sinon Si essai[1] = motATrouver[1] Alors Afficher(essai[1]) Sinon Afficher("-") FinSi Si essai[2] = motATrouver[2] Alors Afficher(essai[2]) Sinon Afficher("-") FinSi Si essai[3] = motATrouver[3] Alors Afficher(essai[3]) Sinon Afficher("-") FinSi Si essai[4] = motATrouver[4] Alors Afficher(essai[4]) Sinon Afficher("-") FinSi Afficher("Joueur 2, saisis ton deuxième essai") Saisir(essai) Si essai = motATrouver Alors Afficher("Bravo, joueur 2, tu as gagné du deuxième coup") Sinon Si essai[1] = motATrouver[1] Alors Afficher(essai[1]) Sinon Afficher("-") FinSi Si essai[2] = motATrouver[2] Alors Afficher(essai[2]) Sinon Afficher("-") FinSi Si essai[3] = motATrouver[3] Alors Afficher(essai[3]) Sinon Afficher("-") FinSi

```
                Si          essai[4] = motATrouver[4]
                Alors       Afficher(essai[4])
                Sinon       Afficher("-")
                FinSi
                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,
                            vraiment, tu n'es pas doué, le mot à
                            trouver était ", motATrouver, " C'était
                            pourtant simple!")


                FinSi
                FinSi
FinSiSisss


Fin
```

Remarques :

- **Il faut mettre le deuxième essai dans le sinon du premier essai et le troisième essai dans le sinon du deuxième essai, sinon, même si le joueur a trouvé le mot au coup d'avant, le programme lui proposera un deuxième et un troisième essai.**
- **Vous remarquerez dans ce corrigé à quel point l'indentation est importante pour garder une bonne lisibilité des algos et des programmes.**

Il y a une autre solution, que je trouve plus futée et qui, à mon avis, s'adapte mieux au développement graphique et événementiel :

Lexique	Algo : deviner un mot – version que personnellement, je trouve plus futée (c'est un peu long comme titre pour un algo, mais au moins, ça veut bien dire ce que ça veut dire)

motATrouver (chaîne[4], saisie) : mot que le joueur 2 doit trouver.
essai (chaîne[4], saisie) : mot saisi par le joueur 2.
	Début
	Afficher("Joueur 1, tu dois saisir un mot de 4 lettres")
	Saisir(motATrouver)
	Afficher("Joueur 2, tu dois trouver le mot de 4 lettres.")
	Afficher("Tu as droit à 3 essais, que la force soit avec toi")
	Afficher("Premier essai :")
	Saisir(essai)
	Si essai = motATrouver
	Alors Afficher("Bravo, joueur 2, tu as gagné du premier coup")
	Sinon Si essai[1] <> motATrouver[1]
	Alors essai[1] ← "-"
	FinSi
	// Pour chaque caractère du mot trouvé, si ce caractère
	// n'est pas égal au caractère du mot à trouver, on le
	// remplace par un tiret
	Si essai[2] <> motATrouver[2]
	Alors essai[2] ← "-"
	FinSi
	Si essai[3] <> motATrouver[3]
	Alors essai[3] ← "-"
	FinSi
	Si essai[4] <> motATrouver[4]
	Alors essai[4] ← "-"
	FinSi
	Afficher ("Voici les bonnes lettres :", essai)
	Afficher("Joueur 2, saisis ton deuxième essai")
	Saisir(essai)
	// Là, l'utilisateur n'aura qu'à ressaisir les lettres qu'il
	// n'a pas trouvées lors de son premier essai,
	// représentées par des -.
	Si essai = motATrouver
	Alors Afficher("Bravo, joueur 2, tu as gagné du deuxième coup")
	Sinon Si essai[1] <> motATrouver[1]
	Alors essai[1] ← "-"

motATrouver (chaîne[4], saisie) : mot que le joueur 2 doit trouver. essai (chaîne[4], saisie) : mot saisi par le joueur 2.	FinSi Si essai[2] <> motATrouver[2] Alors essai[2] ← "-" FinSi Si essai[3] <> motATrouver[3] Alors essai[3] ← "-" FinSi Si essai[4] <> motATrouver[4] Alors essai[4] ← "-" FinSi 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, vraiment, tu n'es pas doué, le mot à trouver était ", motATrouver, " C'était pourtant simple!") FinSi FinSi FinSiSisss Fin

Exercice 27

Écrire l'algo qui permet d'afficher dans l'ordre alphabétique 4 mots saisis dans un ordre quelconque.

Lexique	Algo Tri de 4 mots saisis – Solution pas très futée – Tri par insertion

mot1, mot2, mot3, mot4 (chaînes, saisies) : mots à trier et à afficher dans l'ordre lexicographique
m1, m2, m3, m4 (chaînes, calculées) : liste des 4 mots triés.
	Début
	Saisir (mot1, mot2, mot3, mot4)
	// Tri des deux premier mots
	Si mot1 < mot2
	Alors m1 ← mot1
	m2 ← mot2
	Sinon m1 ← mot2
	m2 ← mot1
	FinSi
	// Insertion du 3e mot dans la liste triée

```
Si          mot3 < m1
Alors       // mot3 doit prendre la place de m1,
            // m1 doit aller en m2 et m2 doit aller en m3
            // Pour ne pas écraser les valeurs figurant déjà dans m1 et m2
            // Il faut décaler les mots dans l'ordre qui suit
            m3 ← m2
            m2 ← m1
            m1 ← mot3
Sinon       // mot3 >= m1, il faut trouver s'il est entre m1 et m2, ou s'il // est
            entre m2 et m3
            Si          mot3 < m2
            Alors       // Il faut décaler m2 en m3 et mettre mot3 dans m2
                        m3 ← m2
                        m2 ← mot3
            Sinon       // mot3 >= m2
                        // mot3 doit être inséré en fin de liste, dans m3
                        m3 ← mot3
            FinSi
FinSi
// les trois premiers mots saisis sont triés
// Insertion du 4e mot saisi au bon endroit dans la liste
Si          mot4 < m1
Alors       m4 ← m3
            m3 ← m2
            m2 ← m1
            m1 ← mot4
Sinon       Si          mot4 < m2
            Alors       m4 ← m3
                        m3 ← m2
                        m2 ← mot4
            Sinon       Si          mot4 < m3
                        Alors       m4 ← m3
                                    m3 ← mot4
                        Sinon       m4 ← mot4
                        FinSi
            FinSi
FinSi
Afficher (m1, m2, m3, m4)
Fin
```

Suivant le même principe de tri par insertion, il existe une solution plus légère qui consiste à ne pas utiliser 4 variables supplémentaires, mais les mots eux-mêmes dont on permute les places. Il suffit pour cela d'utiliser une variable temporaire dans laquelle on range un des mots à permuter. Cette façon de faire permet de diminuer légèrement le nombre de lignes de code qui pour l'instant reste assez volumineux compte tenu du fait qu'on ne trie que 4 mots :

Lexique	Algo Tri de 4 mots saisis – Solution un peu plus futée – Tri par insertion

mot1, mot2, mot3, mot4 (chaînes, saisies) : mots à trier et à afficher dans l'ordre lexicographique.
tempo (chaîne, calculée) : variable temporaire utilisée lors des permutations de mots.
	Début
	Saisir (mot1, mot2, mot3, mot4)
	// Tri des deux premiers mots
	Si mot1 >= mot2
	Alors // On permute mot1 et mot2
	tempo ← mot1
	mot1 ← mot2
	mot2 ← tempo
	// Vous remarquerez qu'il n'y a pas de Sinon car si mot1 < mot2, on
	// ne fait rien
	FinSi
	// Insertion du 3e mot dans la liste triée
	Si mot3 < mot1
	Alors // mot3 doit prendre la place de mot1,
	// mot1 doit aller en mot2 et mot2 doit aller
	// en mot3
	tempo ← mot1
	mot1 ← mot3
	mot3 ← mot2
	mot2 ← tempo
	Sinon // mot3 >= mot11, il faut trouver s'il est entre mot1 et mot2,
	// ou s'il est entre mot2 et mot3
	Si mot3 < mot2
	Alors // Il faut permuter mot2 et mot3
	tempo ← mot2
	mot2 ← mot3
	mot3 ← tempo
	// Là non plus, pas de Sinon car si mot3 >= mot2, on
	// ne permute rien
	FinSi
	FinSi
	// les trois premiers mots saisis sont triés
	// Insertion du 4e mot saisi au bon endroit dans la liste
	Si mot4 < mot1

```
Alors   // mettre mot4 dans mot1 et décaler les 3 autres mots
        // en mot2, mot3 et mot4
        tempo ← mot1
        mot1 ← mot4
        mot4 ← mot3
        mot3 ← mot2
        mot2 ← tempo
Sinon   Si      mot4 < mot2
        Alors   // décaler mot2 en mot3 et mot3 en mot4 et mettre
                // mot4 dans mot2
                tempo ← mot2
                mot2 ← mot4
                mot4 ← mot3
                mot3 ← tempo
        Sinon   Si      mot4 < mot3
                Alors   // permuter mot4 et mot3
                        tempo ← mot3
                        mot3 ← mot4
                        mot4 ← tempo
                // Sinon rien, donc, pas de Sinon
                FinSi
        FinSi
FinSi
Afficher (mot1, mot2, mot3, mot4)
Fin
```

La version suivante (utilisant les principes du tri à bulles) n'utilise pas tout à fait le même principe : on commence par chercher parmi les 4 mots celui qui sera en haut de la liste. Ceci une fois fait, on cherche le premier mot parmi les 3 mots restant à trier. Ensuite, on cherche le premier mot parmi les 2 mots restant à trier. Et c'est fini.

Lexique	**Algo Tri de 4 mots saisis – Solution un peu plus futée**
mot1, mot2, mot3, mot4 (chaînes, saisies) : mots à trier et à afficher dans l'ordre lexicographique. tempo (chaîne, calculée) : variable temporaire utilisée lors des permutations de mots.	Début Saisir (mot1, mot2, mot3, mot4) // Recherche du plus petit mot parmi les 4 Si mot1 >= mot2 Alors // On permute mot1 et mot2 tempo ← mot1 mot1 ← mot2 mot2 ← tempo FinSi

```
// mot1 est le minimum provisoire entre mot1 et mot2
Si          mot3 < mot1
Alors       // mot3 doit prendre la place de mot1,
            // et devient le minimum provisoire entre mot1, mot2 et
            // mot3
            tempo ← mot1
            mot1 ← mot3
            mot3 ← tempo
FinSi
// Recherche du minimum provisoire entre mot1, mot2, mot3
// et mot4
Si          mot4 < mot1
Alors       // Il faut permuter mot1 et mot4
            tempo ← mot1
            mot1 ← mot4
            mot4 ← tempo
FinSi
// A ce stade, on a trouvé le minimum parmi les 4 mots.
// Ce minimum est rangé dans mot1.
// Il reste à trier mot2, mot3 et mot4 entre eux
// On recherche ici le plus petit des 3 mots et
// on veut le ranger dans mot2
Si          mot3 < mot2
Alors       tempo ← mot2
            mot2 ← mot3
            mot2 ← tempo
            // mot2 est le minimum provisoire mot2 et mot3
FinSi
Si          mot4 < mot2
Alors       tempo ← mot2
            mot2 ← mot4
            mot4 ← tempo
FinSi
// Voilà, on tient notre 2e mot de la liste.
// Il est rangé dans mot2
// Il ne reste plus qu'à trier mot3 et mot4 entre eux
Si          mot4 < mot3
Alors       // permuter mot4 et mot3
            tempo ← mot3
            mot3 ← mot4
            mot4 ← tempo
FinSi
// Et voilà, fini !
Afficher (mot1, mot2, mot3, mot4)
Fin
```

Cette version a l'avantage de ne pas contenir trop de **Sinon** dans tous les sens. Vous avez pu remarquer qu'elle était très répétitive : on retrouve tout le temps la série suivante d'instructions :

tempo ← unMot
unMot ← lautreMot
lautreMot ← tempo.

C'est un cas typique où la création d'une **fonction** ou d'une **procédure** s'impose. Une procédure, c'est comme une fonction sauf qu'on ne range pas son résultat dans une variable.

Il nous suffit de décider qu'il existe une procédure, choisissons de l'appeler **permuter** dont le rôle est de permuter les valeurs des 2 variables qu'on lui passe en paramètre.

À ce moment là, notre algo devient :

Lexique	Algo Tri de 4 mots saisis – Solution un peu plus futée – Tri à bulles

mot1, mot2, mot3, mot4 (chaînes, saisies) : mots à trier et à afficher dans l'ordre lexicographique.
permuter (procédure) : permute les valeurs des 2 variables passées en paramètre.
	Début
	Saisir (mot1, mot2, mot3, mot4)
	// Recherche du plus petit mot parmi les 4
	Si mot1 >= mot2
	Alors permuter (mot1, mot2)
	FinSi
	// mot1 est le minimum provisoire entre mot1 et mot2
	Si mot3 < mot1
	Alors permuter (mot1, mot3)
	FinSi
	// Recherche du minimum provisoire entre mot1, mot2, mot3
	// et mot4
	Si mot4 < mot1
	Alors permuter (mot1, mot4)
	FinSi
	// A ce stade, on a trouvé le minimum parmi les 4 mots
	// On recherche ici le plus petit des 3 mots et on veut le ranger dans
	// mot2
	Si mot3 < mot2
	Alors permuter (mot2, mot3)
	FinSi
	Si mot4 < mot2
	Alors permuter (mot2, mot4)
	FinSi
	// Voilà, on tient notre 2e mot de la liste
	// Il ne reste plus qu'à trier mot3 et mot4 entre eux
	Si mot4 < mot3
	Alors permuter (mot3, mot4)
	FinSi
	// Et voilà, fini !
	Afficher (mot1, mot2, mot3, mot4)
	Fin

Ca raccourcit bigrement la taille de l'algo n'est-ce pas ? Bien entendu, Si cette procédure **permuter** n'existe pas, il faut l'écrire. Entre temps, nous verrons également comment écrire plus simplement cet algo à l'aide de structures itératives dans la séquence suivante.

Partie 3

Les choix multiples

Exercice 28

Écrire un algorithme qui propose un menu affiché à l'écran, et qui, en fonction du choix fait par l'utilisateur, effectue soit la somme, soit le produit, soit la moyenne de 2 nombres saisis. Prévoir le cas où l'utilisateur a fait une erreur de saisie.

Lexique	Algo choix d'opérations
n1, n2 (réels, saisis) choix (entier, saisi) : choix de l'utilisateur	Début Afficher ("Donner 2 nombres : "); Saisir (n1, n2) Afficher ('1 - Somme ') Afficher ('2 - Produit') Afficher (3 - Moyenne) Afficher ("Saisissez votre choix :") Saisir (choix) SelonCas choix faire 1 : Afficher ("La somme est ", n1 + n2) 2 : Afficher ("Le produit est ", n1 * n2) 3 : Afficher ("La moyenne est ", (n1 + n2)/2) sinon Afficher ("Choix incorrect") FinSelon Fin

Exercice 29

L'utilisateur saisit la date au format jj/mm/aaaa, et le programme affiche la date avec le mois en lettres.

Exemple : si l'utilisateur saisit 15/10/2007, alors le programme affiche : Aujourd'hui, on est le 15 Octobre 2001.

Si le mois saisi est erroné, un message s'affiche, indiquant à l'utilisateur qu'il a fait une erreur de saisie.

Lexique	Algo date avec mois
date (chaîne, saisie) : date à traduire. SousChaîne (fonction) rés chaîne : fonction qui retourne un morceau de la chaîne de caractère passée en paramètre, à partir du point spécifié en 2e paramètre, de la longueur spécifiée en 3e paramètre. mois (chaîne, calculée) : nom du mois saisi.	Début Afficher ("Saisissez la date au format jj/mm/aaaa") Saisir (date) selonCas SousChaîne (date,4,2) faire "01" : mois ← "Janvier" "02" : mois ← "Février" "03" : mois ← "Mars" "04" : mois ← "Avril " "05" : mois ← "Mai" .... "11" : mois ← "Novembre " "12" : mois ← "Décembre " sinon mois ← "" FinSelon Si mois <> "" Alors Afficher ("Aujourd'hui, on est le ", SousChaîne (date, 1,2) + " " + mois + " " + SousChaîne (date, 7,4)) Sinon Afficher ("Le mois saisi n'existe pas") FinSi Fin

Exercice 30

Écrire l'algorithme qui affiche la conjugaison d'un verbe du premier groupe saisi par l'utilisateur à la personne choisie.

Au lancement, un message invite l'utilisateur à saisir le verbe qu'il souhaite conjuguer.

Deux messages s'affichent ensuite, lui permettant de choisir d'une part le pronom de conjugaison, d'autre part le nombre (singulier ou pluriel) souhaités.

Si l'utilisateur saisit *arriver* puis choisit *2e personne* et *singulier*, le message *Tu arrives* s'affiche.

Lexique	Algo conjugaison

verbe (chaîne, saisi) : verbe à conjuguer.
pers (entier, saisi) : numéro de la personne à utiliser pour la conjugaison.
nombre (caractère, saisi) : s pour singulier, p pour pluriel.
racine (chaîne, calculé) : racine du verbe à conjuguer.
finVerbe (chaîne, calculée) : valeur de la fin du verbe conjugué.
Souschaîne (fonction) rés chaîne : renvoie le découpage de la chaîne passée en paramètre à partir de la position spécifiée en 2e paramètre sur la longueur spécifiée en 3e paramètre.
Longueur(fonction) rés entier : renvoie le nombre de caractères de la chaîne passée en paramètre.
	Début
	Afficher ("Saisissez le verbe à conjuguer")
	Saisir (verbe)
	Afficher ("Choisissez 1 pour la 1ère personne, 2 pour la 2e, 3 pour la 3ème")
	Saisir (pers)
	Afficher ("Choisissez s pour singulier, p pour pluriel")
	Saisir (nombre)
	// Récupérer la racine du verbe
	// On découpe le verbe en partant de la fin, lui enlevant
	// les 2 derniers caractères, en utilisant la fonction
	// longueur
	racine ← SousChaîne (verbe, longueur(verbe) –2, 2)
	SelonCas pers faire
	1 : Si nombre = "s"
	Alors personne ← "je"
	finverbe ← "e"
	Sinon personne ← "nous"
	finverbe ← "ons"
	FinSi
	2 : Si nombre = "s"
	Alors personne ← "tu"
	finverbe ← "es"
	Sinon personne ← "vous"
	finverbe ← "ez"
	FinSi
	3 : Si nombre = "s"
	Alors personne ← "il, elle, on"
	finverbe ← "e"
	Sinon personne ← "ils, elles"
	finverbe ← "ent"
	FinSi
	FinSelon
	Afficher (personne + "" + racine + finVerbe)
	Fin

On peut très bien dans cet algo, utiliser des structures **SelonCas** à la place des **Si**…**Alors**…**Sinon**…**FinSi** :

Lexique	Algo conjugaison

verbe (chaîne, saisi) : verbe à conjuguer.
pers (entier, saisi) : numéro de la personne à utiliser pour la conjugaison.
nombre (caractère, saisi) : s pour singulier, p pour pluriel.
racine (chaîne, calculé) : racine du verbe à conjuguer.
finVerbe (chaîne, calculée) : valeur de la fin du verbe conjugué.
Souschaîne (fonction) rés chaîne : renvoie le découpage de la chaîne passée en paramètre à partir de la position spécifiée en 2e paramètre sur la longueur spécifiée en 3e paramètre.
Longueur(fonction) rés entier : renvoie le nombre de caractères de la chaîne passée en paramètre.
	Début
	Afficher ("Saisissez le verbe à conjuguer")
	Saisir (verbe)
	Afficher ("Choisissez 1 pour la 1re personne, 2 pour la 2e, 3 pour la 3eme")
	Saisir (pers)
	Afficher ("Choisissez s pour singulier, p pour pluriel")
	Saisir (nombre)
	// Récupérer la racine du verbe
	// On découpe le verbe en partant de la fin, lui enlevant
	// les 2 derniers caractères, en utilisant la fonction
	// longueur
	racine ← SousChaîne (verbe, longueur(verbe) –2, 2)
	SelonCas pers faire
	1 : SelonCas nombre faire
	"s" : personne ← "je"
	Finverbe ← "e"
	"p" : // ou Sinon
	personne ← "nous"
	Finverbe ← "ons"
	FinSelon
	2 : SelonCas nombre faire
	"s" : personne ← "tu"
	Finverbe ← "es"
	"p" : personne ← "vous"
	Finverbe ← "ez"
	FinSelon
	3 : SelonCas nombre faire
	"s" : personne ← "il, elle, on"
	Finverbe ← "e"
	"p" : personne ← "ils, elles"
	Finverbe ← "ent"
	FinSelon
	FinSelon
	Afficher (personne + "" + racine + FinVerbe)
	Fin

Certains d'entre vous ont peut-être écrit l'algo de la façon suivante :

Lexique	Algo conjugaison

Verbe (chaîne, saisi) : verbe à conjuguer
pers (entier, saisi) : numéro de la personne à utiliser pour la conjugaison
racine (chaîne, saisi) : racine du verbe à conjuguer
Souschaîne (fonction) rés chaîne : renvoie le découpage la chaîne passée en paramètre à partir de la position spécifiée en 2e paramètre sur la longueur spécifiée en 3e paramètre
Longueur(fonction) rés entier : renvoie le nombre de caractères de la chaîne passée en paramètre.
	Début
	Afficher ("Saisissez le verbe à conjuguer")
	Saisir (verbe)
	Afficher ("Choisissez 1 pour la 1ère personne du singulier, 2 pour la 2e personne du singulier, 3 pour la 3e personne du singulier , 4 pour la 1ère personne du pluriel, 5 pour la 2e personne du pluriel, 6 pour la 3e personne du pluriel ")
	Saisir (pers)
	// Récupérer la racine du verbe
	// On découpe le verbe en partant de la fin, lui enlevant
	// les 2 derniers caractères, en utilisant la fonction
	// longueur
	racine ← SousChaîne (verbe, longueur(verbe) –2, 2)
	SelonCas pers faire
	1 : personne ← "je"
	finverbe ← "e"
	2 : personne ← "tu"
	finverbe ← "es"
	3 : personne ← "il, elle, on"
	finverbe ← "e"
	4 : personne ← "nous"
	finverbe ← "ons"
	5 : personne ← "vous"
	finverbe ← "ez"
	6 : personne ← "ils, elles"
	finverbe ← "ent"
	FinSelon
	Afficher (personne + "" + racine + finVerbe)
	Fin

Bien entendu, cet algo conjugue correctement le verbe saisi à la personne saisie. Mais il ne fait pas exactement ce qui a été spécifié dans l'énoncé :

« Deux messages s'affichent ensuite, lui permettant de choisir d'une part le pronom de conjugaison, d'autre part le nombre (singulier ou pluriel) souhaités. »

Mon objectif ici est de vous montrer à quel point une analyse précise des demandes des clients est importante.

Bon, eh bien nous sommes arrivés à la fin de cette séquence 3.

Alors :

Si	vous avez envie de programmer
Alors	Commencez le TP de la séquence3
Sinon	Détendez vous un peu car comme vous pouvez le constater, les Si…Alors…Sinon, ça finit par faire travailler du chapeau.
FinSi	

Quel que soit votre choix, ne prenez pas de retard.

La séquence 4 est consacré à une autre famille de structures : les structures répétitives (ou itératives). Vous allez voir, ça n'est pas mal non plus.
À bientôt!