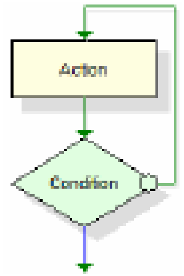


FORMATION AFPA
- DEVELOPPEUR LOGICIEL –

(NIVEAU III)

JJP

ALGORITHME ET PSEUDO-CODE

CORRIGES DES EXERCICES : 7.1 à 7.6

Exercice 7.1

```
ALGORITHME Exo_7_1
        Variables Nb, i en Entier
        Variable Flag en Booleen
        Tableau T() en Entier
Debut
        Ecrire "Entrez le nombre de valeurs :"
        Lire Nb
        Redim T(Nb-1)
        Pour i ← 0 à Nb - 1
                Ecrire "Entrez le nombre n° ", i + 1
                Lire T(i)
        i Suivant
        Flag ← Vrai
        Pour i ← 1 à Nb - 1
                Si T(i) <> T(i – 1) + 1 Alors
                        Flag ← Faux
                FinSi
        i Suivant
        Si Flag Alors
                Ecrire "Les nombres sont consécutifs"
        Sinon
                Ecrire "Les nombres ne sont pas consécutifs"
        FinSi
Fin
```

Cette programmation est sans doute la plus spontanée, mais elle présente le défaut d'examiner la totalité du tableau, même lorsqu'on découvre dès le départ deux éléments non consécutifs. Aussi, dans le cas d'un grand tableau, est-elle dispensieuse en temps de traitement. Une autre manière de procéder serait de sortir de la boucle dès que deux éléments non consécutifs sont détectés.

ALGORITHME ET PSEUDO-CODE

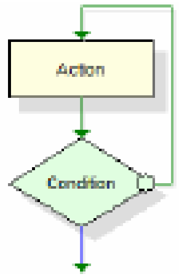
La deuxième partie de l'algorithme deviendrait donc :

```
        i ← 1
        TantQue T(i) = T(i – 1) + 1 et i < Nb - 1
                i ← i + 1
        FinTantQue
        Si T(i) = T(i – 1) + 1 Alors
                Ecrire "Les nombres sont consécutifs"
        Sinon
                Ecrire "Les nombres ne sont pas consécutifs"
        FinSi
```

Exercice 7.2

On suppose que N est le nombre d’éléments du tableau. Tri par insertion :

```
ALGORITHME Exo_7_2
        Variables posmaxi, i, temp, N en Entier
        Tableau T() en Entier
Debut
        Pour i ← 0 à N - 2
                posmaxi = i
                Pour j ← i + 1 à N - 1
                        Si t(j) > t(posmaxi) alors
                                posmaxi ← j
                        Finsi
                j suivant
                temp ← t(posmaxi)
                t(posmaxi) ← t(i)
                t(i) ← temp
        i suivant
Fin
```



ALGORITHME ET PSEUDO-CODE

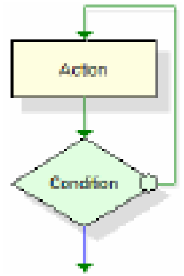
Tri à bulles :

```
ALGORITHME Exo_7_2_Bis
        Variables i, temp, N en Entier
        Variables Yapermut en booleen
        Tableau T() en Entier
Debut
        Yapermut ← Vrai
        TantQue Yapermut
                Yapermut ← Faux
                Pour i ← 0 à N - 2
                        Si t(i) < t(i + 1) Alors
                                temp ← t(i)
                                t(i) ← t(i + 1)
                                t(i + 1) ← temp
                                Yapermut ← Vrai
                        Finsi
                i suivant
        FinTantQue
Fin
```

Exercice 7.3

On suppose que n est le nombre d'éléments du tableau préalablement saisi

```
ALGORITHME Exo_7_3
        Variables i, Temp, N en Entier
        Tableau T() en Entier
Debut
        Pour i ← 0 à (N-1)/2
                Temp ← T(i)
                T(i) ← T(N-1-i)
                T(N-1-i) ← Temp
        i suivant
Fin
```



ALGORITHME ET PSEUDO-CODE

Exercice 7.4

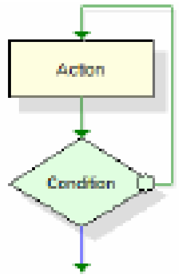
```
ALGORITHME Exo_7_4
        Variables i, Temp, N en Entier
        Tableau T() en Entier
Debut
        Ecrire "Rang de la valeur à supprimer ?"
        Lire S
        Pour i ← S à N-2
                T(i) ← T(i+1)
        i suivant
        Redim T(N–1)
Fin
```

ALGORITHME ET PSEUDO-CODE

Exercice 7.5

N est le nombre d'éléments du tableau Dico(), contenant les mots du dictionnaire, tableau préalablement rempli.

```
ALGORITHME Exo_7_5
        Variables Sup, Inf, Comp en Entier
        Variables Fini en Booléen
Début
        Ecrire "Entrez le mot à vérifier"
        Lire Mot
        // On définit les bornes de la partie du tableau à considérer
        Sup ← N - 1
        Inf ← 0
        Fini ← Faux
        TantQue Non Fini
                // Comp désigne l'indice de l'élément à comparer. En bonne rigueur, il faudra veiller à ce que Comp soit bien un
                // nombre entier, ce qui pourra s'effectuer de différentes manières selon les langages.
                Comp ← (Sup + Inf)/2
                // Si le mot se situe avant le point de comparaison, alors la borne supérieure change, la borne inférieure ne
                // bouge pas.
                Si Mot < Dico(Comp) Alors
                        Sup ← Comp - 1
                // Sinon, c'est l'inverse
                Sinon
                        Inf ← Comp + 1
                FinSi
                Fini ← Mot = Dico(Comp) ou Sup < Inf
        FinTantQue
        Si Mot = Dico(Comp) Alors
                Ecrire "le mot existe"
        Sinon
                Ecrire "Il n'existe pas"
        Finsi
Fin
```



ALGORITHME ET PSEUDO-CODE

Exercice 7.6

```
ALGORITHME 7_6_a_b
// Copyright Tristan
Tableau tab[n] en Numérique // on suppose n la taille du tableau
Variable  iMax, iMin, i, iEcart  en Numérique
DEBUT
        iMax <-- tab[0]
        iMin <-- tab[0]
        POUR i <-- 1 à n-1
                SI (tab[i] > iMax ) ALORS
                        iMax <-- tab[i]
                SINONSI (tab[i] < iMin ) ALORS
                        iMin <-- tab[i]
                FINSI
        i SUIVANT
        iEcart <-- iMax - iMin
FIN
```

ALGORITHME ET PSEUDO-CODE

Exercice 7.7

Les deux tableaux de départ, A(m) et B(n), sont déjà triés : pas question donc de les empiler simplement pour se relancer dans un (long) tri. On prend simplement les deux tableaux, et on avance dans l'un puis dans l'autre selon celui des deux éléments auquel on est parvenu est le plus petit (il suffit de s'imaginer devant deux tas de papiers triés par date, et de vouloir constituer un tas unique, pour comprendre ce qu'on va faire). Le truc est qu'on ne sait pas par avance où on va en être à un moment donné dans un tableau et dans l'autre : il nous faut donc deux compteurs différents pour noter notre position dans chacun des deux tableaux. On appelle C le tableau de destination, et ic la variable qui indique où on en est dans celui-ci.

```
Début
    (...)
    Afini ← faux
    Bfini ← faux
    ia ← 0
    ib ← 0
    ic ← -1
    TantQue Non(Afini) ou Non(Bfini)
        ic ← ic + 1
        Redim C(ic)
        Si Afini ou A(ia)>B(ib) Alors
            C(ic) ← B(ib)
            ib ← ib + 1
            Bfini ← ib > n
        Sinon
            C(ic) ← A(ia)
            ia ← ia + 1
            Afini ← ia > m
        FinSi
    FinTantQue
Fin
```