***LES TABLEAUX***

**EXERCICE 1**

Algorithme creation\_de\_tableau ;

Var

Tab (1 ; 10) : tableau de type entiers ;

i : entier ;

Début

Pour i —> 1 à 10 à pas 1 faire

Afficher ‘’ Entrez un nombre de la case ‘’, i ;

Saisir tab [i] ;

Fin pour

Fin.

**EXERCICE 2**

Algorithme editer\_tableau ;

Var

tab (1 ; 10) : tableau de type entiers ;

i : entier ;

Début

Pour i —> 1 à 10 à pas 1 faire

Afficher ‘’ Le contenu de la case ‘’, i, ‘’ est ‘’, tab[i] ;

Fin pour

Fin.

**EXERCICE 3**

Algorithme calcul\_somme ;

Var

tab (1 ; 10) : tableau de type entiers ;

i, som : réels ;

Début

som <— 0

Pour i —> 1 à 10 à pas 1 faire

som <— som + tab[i] ;

Fin pour

Afficher ‘’ La somme des réels dans le tableau est : ‘’, som ;

Fin.

**EXERCICE 4**

Algorithme afficher\_minimum ;

Var

tab (1 ; 10) : tableau de type entiers ;

i, min : réels ;

Début

i <— 1

min <— tab[i]

Pour i —> 1 à 10 à pas 10 faire

Si tab[i] < min Alors

min <— tab [¡] ;

Fin si

Fin pour

Fin.

**EXERCICE 5**

Algorithme rechercher\_nom ;

Var

tab (1 ; 10) : tableau de type entiers ;

i : réels ;

trouver : booléen ;

nom : chaîne de caractère ;

Début

Afficher ‘’ Veillez entrer un nom ‘’ ;

Saisir nom ;

trouver <— 0

Pour i —> 1 à 10 à pas 1 faire

Si nom = tab[i] Alors

trover <— 1 ;

Fin si

Fin pour

Si trover = 1 Alors

Afficher ‘’ Le nom a été trouvé ‘’ ;

Sinon

Aficher ‘’ Le n’a pas été trouvé ‘’ ;

Fin si

Fin.

**EXERCICE 6**

Algorithme occurrence ;

Var

tab (1 ; 50) : tableau de type entiers ;

i, occurrence : réels ;

nom : chaîne de caractère ;

Début

Afficher ‘’ Veillez entrer un nom ‘’ ;

Saisir nom ;

occurrence <— 0

Pour i —> 1 à 50 à pas 1 faire

Si nom = tab[i] Alors

occurrence <— occurrence + 1 ;

Fin si

Fin pour

Afficher ‘’ Le nom a été trouvé ‘’, occurrence, ‘’ fois ‘’ ;

Fin.

**EXERCICE 7**

Algorithme suppression ;

Var

tab(1,10) : tableau de type entier ;

i, nombre, position\_trouver : entiers ;

trouver : booléen ;

Début

Afficher ‘’ Veillez entrez un nombre ‘’ ;

Saisir nombre ;

trouver <— 0 ; i <— 1 ;

Pour i —> 1 à 10 à pas 1 faire

Si tab[i] = nombre Alors

trouver <— 1 ;

position\_trouver <— i ;

Fin si

Fin pour

Si trouver = 0 Alors

Afficher ‘’ l’élément est introuvable ‘’ ;

Sinon

Pour i —> position\_trouver à 9 à pas 1 faire

tab[i] <— tab[i + 1]

Fin pour

Fin si

tab[10] <— 0

Fin.

**EXERCICE 8**

Algorithme somme\_tableau ;

Var

tab1(1,5), tab2(1,5), tab3(1,5) : tableau de type entiers ;

i : entier ;

Début

Pour i —> 1 à 5 à pas 1 faire

tab3[i] <— tab1[i] + tab2[i]

Fin pour

Fin.

**EXERCICE 9**

Algorithme tri\_croissant ;

Var

tab(1, 5) : tableau de type entiers ;

min, i, permute, j, position : entiers ;

Début

Pour i —> 1 à 5 à pas 1 faire

min <— tab[i] ;

position <— i ;

Pour j —> i à 5 à pas 1 faire

si tab[j] < min Alors

min <— tab[j] ;

position <— j ;

Fin si

Fin pour

permute <— tab[i] ;

tab[i] <— min ;

tab[position] <— permute ;

Fin pour

Fin.

**EXERCICE 10**

Algorithme tri\_croissant ;

Var

tab(1, 5) : tableau de type entiers ;

max, i, permute, j, position : entiers ;

Début

Pour i —> 1 à 5 à pas 1 faire

max <— tab[i] ;

position <— i ;

Pour j —> i à 5 à pas 1 faire

si tab[j] > max Alors

max <— tab[j] ;

position <— j ;

Fin si

Fin pour

permute <— tab[i] ;

tab[i] <— max ;

tab[position] <— permute ;

Fin pour

Fin.