Exemples

- Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

- Écrire un algorithme qui permet d'effectuer la saisie d'un nom, d'un prénom et affiche ensuite le nom complet

- Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 7 ou non

- Le prix de disques compacts (CDs) dans espace de vente varie selon le nombre à acheter: 5 DH l'unité si le nombre de CDs à acheter est inférieur à 10, 4 DH l'unité si le nombre de CDS à acheter est compris entre 10 et 20 et 3 DH l'unité si le nombre de CDs à acheter est au-delà de 20.

Écrivez un algorithme qui demande à l'utilisateur le nombre de CDs à acheter, qui calcule et affiche le prix à payer

- Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul

Exemples

Exemple 1: lecture et écriture

Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

```
Algorithme Calcul_du_Carre  Rôle : calcul du carre
Données : un entier
Résultats : le carre du nombre
variables A, B : entier
Début
        écrire("entrer la valeur de A ");
        lire(A);
        B ← A*A;
        écrire("le carré de ", A, " est :", B);
Fin
```

Exemples

Exemple 2: lecture et écriture

Écrire un algorithme qui permet d'effectuer la saisie d'un nom, d'un prénom et affiche ensuite le nom complet

Algorithme AffichageNomComplet
…
variables Nom, Prenom, Nom_Complet : **chaîne de caractères**
Début
  écrire("entrez le nom");
  lire(Nom);
  écrire("entrez le prénom");
  lire(Prenom);
  Nom_Complet ← Nom & " " & Prenom;
  écrire("Votre nom complet est : ", Nom_Complet);
Fin

Exemples

Exemple 3: tests

Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 7 ou non

```
Algorithme Divsible_par7

Variable n : entier
Début
    Ecrire (" Entrez un entier : ");
    Lire (n)
    Si (n%7=0) alors
        Ecrire (n," est divisible par 7");
    Sinon
        Ecrire (n," n'est pas divisible par 7");
    Finsi
 Fin
```

Exemples

Exemple 4: tests imbriqués

- Le prix de disques compacts (CDs) dans espace de vente varie selon le nombre à acheter:

  5 DH l'unité si le nombre de CDs à acheter est inférieur à 10,

  4 DH l'unité si le nombre de CDS à acheter est compris entre 10 et 20 et 3 DH l'unité si le nombre de CDs à acheter est au-delà de 20.

- Écrivez un algorithme qui demande à l'utilisateur le nombre de CDs à acheter, qui calcule et affiche le prix à payer

```
Variables       unites : entier
                prix  : réel
Début

 Ecrire ("Nombre d'unités ");
 Lire (unites);
 Si unites < 10 Alors
    prix ← unites*5;
 Sinon Si unites < 20 alors   prix ← unites*4;
          Sinon prix ← unites*3;
          Finsi
 Finsi

      Ecrire ("Le prix à payer est : ", prix);

 Fin
```

Exemples

Exemple 5: boucle Pour

Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul

```
…
Variables x, puiss : réel
          n, i : entier
Debut
   Ecrire (" Entrez respectivement les valeurs de x et n ");  Lire (x, n);
   puiss ← 1;
   Pour i ←1 à n faire
       puiss← puiss*x;
   FinPour
   Ecrire (x, " à la puissance ", n, " est égal à ", puiss);
Fin
```

TABLEAUX

- Supposons que l'on veut calculer le nombre d'étudiants ayant une note supérieure à 10 pour une classe de 20 étudiants.

- Jusqu'à présent, le seul moyen pour le faire, c'est de déclarer 20 variables désignant les notes **N1, …, N20**:

  - La saisie de ces notes nécessite 20 instructions lire(Ni).
  - Le calcul du nombre des notes>10 se fait par une suite de tests de 20 instructions Si :

```
nbre ← 0;
Si (N1 >10) alors nbre ←nbre+1
FinSi
…
Si (N20>10) alors nbre ←nbre+1
FinSi
```

cette façon n'est pas très pratique

TABLEAUX

- C’est pourquoi, les langages de programmation offrent la possibilité de rassembler toutes ces variables dans **une seule structure de donnée** appelée **tableau qui est facile à manipuler.**

- Un **tableau** est un ensemble d'éléments de même type désignés par un identificateur unique;

- Une variable entière nommée **indice** permet d'indiquer la position d'un élément donné au sein du tableau et de déterminer sa valeur.

TABLEAUX

- La **déclaration** d'un tableau s'effectue en précisant le **type** de ses éléments et sa **dimension** (le nombre de ses éléments)

- **Syntaxe** :

Variable tableau identificateur [dimension] : <TypeTab>

- **Exemple** :

variable tableau notes[20] : réel

- On peut définir des tableaux de tous types : tableaux d'entiers, de réels, de caractères, de booléens, de chaînes de caractères, …

TABLEAUX

Les tableaux à une dimension (vecteurs)

variable **tableau tab[10] : entier**

0	1	2	3	4	5	6	7	8	9
45	54	1	-56	22	134	49	12	90	-26

- Ce tableau est de longueur 10. Chacun des dix nombres du tableau est repéré par son rang, appelé **indice**
- Pour **accéder** à un élément du tableau, il suffit de préciser entre crochets l'indice de la case contenant cet élément.
- Pour accéder au 5ème élément (22), on écrit : **Tab[4]**
- Les instructions de **lecture**, **écriture** et **affectation** s'appliquent aux tableaux comme aux variables.

x ← Tab[0] (x=45)

Tab[6] ← 43

TABLEAUX

- Pour accéder au **i^{ème}** élément, on écrit tab[i-1] (avec 0<=i<10)

- Selon les langages, le premier indice du tableau est soit 0, soit 1. Le plus Souvent c'est 0 (c'est ce qu'on va utiliser en pseudo-code).

- Dans ce cas, tab[i] désigne l'élément i+1 du tableau tab.

- Un grand avantage des tableaux est qu'on peut traiter les données qui y sont stockées de façon simple en utilisant des boucles.

- Les éléments d'un tableau s'utilisent comme des variables.

Tableaux : accès et modification

- Les instructions de lecture, écriture et affectation s'appliquent aux tableaux comme aux variables.

- Exemples :

x ← tab[0];

La variable x prend la valeur du premier élément du tableau (45 selon le tableau précédent).

tab[6] ← 43;

Cette instruction a modifié le contenu du 7ème élément du tableau (43 au lieu de 49).

Relation entre tableaux et boucles

- Les boucles sont extrêmement utiles pour les algorithmes associés aux tableaux.
- Pour parcourir les éléments du tableau selon l'ordre croissant (ou décroissant) des indices, on utilise des boucles.
- Le traitement de chacun des éléments étant souvent le même, seule la valeur de l'indice est amenée à changer.
- Une boucle est donc parfaitement adaptée à ce genre de traitements.

TABLEAUX

Pour le **calcul** du nombre d'étudiants ayant **une note supérieure à 12** avec les tableaux, on peut écrire :

```
Constante     N=20 : entier
Variables     i, nbre : entier
               tableau notes[N] : réel
 Début
        nbre ← 0;
        Pour i ← 0 à N-1 faire
                Si (notes[i] >12) alors
                    nbre ←nbre+1;
                FinSi
        FinPour

        écrire ("le nombre de notes supérieures à 12 est : ", nbre);
Fin
```

Tableaux : saisie et affichage

```
Constante   Max=200 : entier
Variables          i, n        : entier

                    tableau Notes[max] : réel
        ecrire("entrer la taille du tableau :") ;
        lire(n);
   Debut
/* saisie */
        Pour i ← 0 à n-1 [par 1] faire
            écrire ("Saisie de l'élément ", i + 1);
            lire (T[i] );
        FinPour
/* affichage */
        Pour i ← 0 à n-1 [par 1] faire
             écrire ("T[",i, "] =", T[i]);
        FinPour
   Fin
```

Tableaux : Exercice

- Ecrire un programme qui déclare un tableau de 100 entiers et qui l'initialise par les nombres entiers de 1 à 100.

- Ecrire un programme qui permet de saisir 20 nombres réels en simple précision et les stocker dans un tableau

Tableaux : Exercice

```
Algorithme : tableau test
Variable   tableau a [100], i: entiers
 début
        pour i <- 0 à 99 [par 1] faire
              a[i]<- i+1
         Fin Pour
Fin
```

Tableaux : Exercice corrigé

```
Algorithme : saisie_tab
Variable   tableau a [20]: réels en simple précision
                i : entier
 début
        pour i <- 0 à 19 [par 1] faire
            écrire ("Saisie de l'élément ", i + 1);
            lire(a[i])
Fin Pour
Fin
```

TABLEAUX

Tableaux à deux dimensions

- Les langages de programmation permettent de déclarer des tableaux dans lesquels les valeurs sont repérées par **deux indices**.
- Ceci est utile par exemple pour représenter des matrices.
- En pseudo-code, un tableau à deux dimensions se **déclare** ainsi :

  variable **tableau** identificateur**[dim1] [dim2] : type**
- **Exemple** :
- une matrice A de 3 lignes et 4 colonnes dont les éléments sont réels :
  variable **tableau** A**[3][4] : réel**
- **A[i][j]** permet d'accéder à l'élément de la matrice qui se trouve à l'intersection de la ligne i+1 et de la colonne j+1
- Les tableaux peuvent avoir n dimensions.

TABLEAUX

- La matrice A dans la déclaration suivante :

  variable **tableau** A**[3][5] : réel**

peut être explicitée comme suit : les éléments sont rangées dans un tableau à deux entrées.

	0	*1*	*2*	*3*	*4*
0	12	28	44	2	76
1	23	36	51	11	38
2	43	21	55	67	83

- Ce tableau a 3 lignes et 5 colonnes. Les éléments du tableau sont repérés par leur numéro de ligne et leur numéro de colonne.

- . Par exemple A[1][4] vaut 38

Exemples : lecture d'une matrice

- La saisie des éléments d'une matrice :

```
Constante N=100 :entier
Variable      i, j, m, n : entier
              tableau A[N][N] : réel
Début
   ecrire("entrer le nombre de lignes et le nombre de colonnes :") ;
   lire(m, n);
   Pour i ← 0 à m-1 [par 1] faire
          écrire ("saisie de la ligne ", i + 1);
          Pour j ← 0 à n-1 [par 1] faire
              écrire ("Entrez l'élément : ligne ", i+1, " et colonne ", j+1);
              lire (A[i][j]);
           FinPour
    FinPour
Fin
```

TABLEAUX

- **Affichages des éléments d'une matrice :**

```
Constante N=100 : entier
Variable        i, j, m, n : entier
                tableau A[N][N]: réel

Début
  ecrire("entrer le nombre de lignes et le nombre de colonnes :") ;
  lire(m, n);
     Pour i ← 0 à m-1 [par 1] faire
            Pour j ← 0 à n-1 [par 1] faire
                    écrire ("A[",i, "] [",j,"]=", A[i][j]);
            FinPour
      FinPour
Fin
```

Initialisation de matrice

- Pour initialiser une matrice on peut utiliser par exemple les instructions suivantes :

T_1[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
T_2[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
T_3[4][4] = {{1,2,3}, {4,5,6}, {7,8,9}};
T_4[4][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

TABLEAUX

Traitements opérant sur des tableaux

- **Créer** des tableaux
- **Ranger** des valeurs dans un tableau
- **Récupérer**, **consulter** des valeurs rangées dans un tableau
- **Rechercher** si une valeur est dans un tableau
- **Mettre à jour** des valeurs dans un tableau
- **Modifier** la façon dont les valeurs sont rangées dans un tableau (par exemple : les trier de différentes manières)
- Effectuer des **opérations entre tableaux** : comparaison de tableaux, multiplication,...

TABLEAUX

Exercices:

1. Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs entiers en les mettant toutes à zéro.

2. Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet

3. Ecrire un algorithme permettant, à l'utilisateur de saisir les notes d'une classe. Le programme, une fois la saisie terminée, renvoie le nombre de ces notes supérieures à la moyenne de la classe.

Corrigés:

1.

Variable i , Tableau tab[6] **:** entiers
Debut
    Pour i ← 0 à 6 **faire**
        tab[i] ← 0
    i **Suivant**
    Finpour
Fin

2.

Variable Tableau tab[6] **:** caractère
Debut
  Tab[0] ← "a"
  Tab[1] ← "e"
  Tab[2] ← "i"
  Tab[3] ← "o"
  Tab[4] ← "u"
  Tab[5] ← "y"
Fin

TABLEAUX


```
Variables Nb, Som, Moy, Nbsup :réels
          Tableau T[30],i: entiers
Debut
Ecrire ("Entrez le nombre de notes à saisir : ")
Lire (Nb)
      Pour i ← 0 à Nb – 1 faire
          Ecrire( "Entrez le nombre n° ", i + 1)
          Lire (T[i])
          i Suivant
      finpour
Som ← 0
      Pour i ← 0 à Nb – 1 faire
          Som ← Som + T[i]
          i Suivant
      finpour
Moy ← Som / Nb
NbSup ← 0
Pour i ← 0 à Nb – 1 faire
  Si T[i] > Moy Alors
    NbSup ← NbSup + 1
  FinSi
i Suivant
finpour
Ecrire (NbSup, " élèves dépassent la moyenne de la classe")
Fin
```