SUP'MANAGEMENT
ECOLE SUPERIEURE DE MANAGEMENT DE COMMERCE ET D'INFORMATIQUE
Reconnue par l'Etat

1 A **Informatique** TC

Enseignant: M.HMAMOU

Introduction à l'algorithmique

Déclarations (suite)

- Déclarations

Entête

- Il permet l'identification de l'algorithme
- Mot-clé : Algorithme

Déclarations

- Liste exhaustive des variables, constantes, des structures, des fonctions et des procédures utilisées dans le corps de l'algorithme
- Mots-clés : variable, constante, structure , fonction, …

Corps

- C'est dans cette partie que les tâches (instructions, opérations,…) de l'algorithme sont placées
- Mots-clés : Début, Fin

Déclarations (suite)

- Déclarations

  La partie déclaration dans un algorithme contient une liste exhaustive des déclarations:

  - des variables,
  - des constantes;
  - des structures,
  - des fonctions,
  - des procédures,

  utilisées dans le corps de l'algorithme

  Les variables, constantes, structures, fonctions et procédures utilisées doivent avoir fait l'objet d'une déclaration préalable.

Déclarations (suite)

- Types composites

  On a vu qu'une variable peut être de type :

  - Entier,
  - Réel,
  - Caractère,
  - …

  Ce sont des types simples (élémentaires)

  Il existe d'autres types dits composites ou structurés :

  - Tableaux
  - Structures

Déclarations (suite)

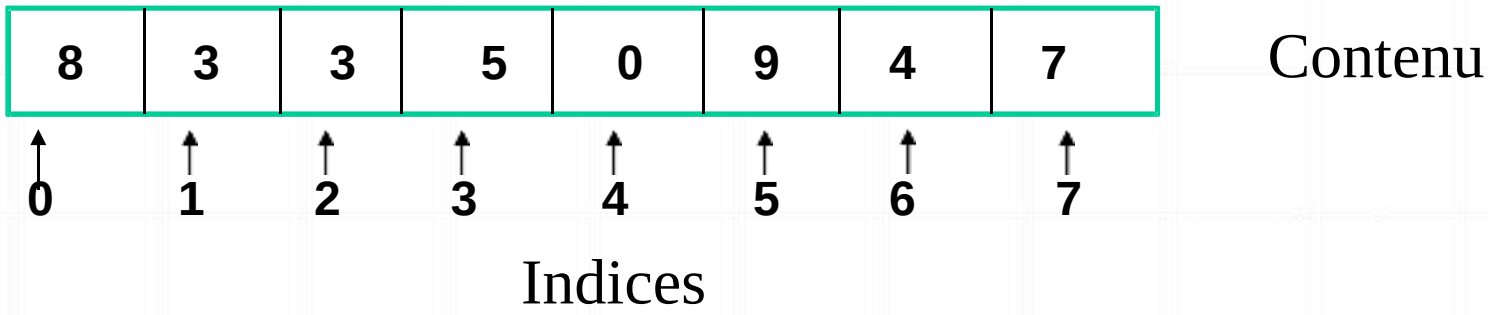
- Tableaux : notions

Heureusement, les langages de programmation offrent la possibilité de rassembler toutes ces variables dans une seule structure de donnée appelée tableau. Donc, un tableau est une collection ordonnée et homogène de valeurs :

- ordonnée car les cases mémoires composant un tableau se suivent
- homogène car toutes les valeurs d'un tableau ont le même type Il est composé d'un certain nombre de cases

Chaque case peut prendre une valeur

- Une case est repérée par son indice

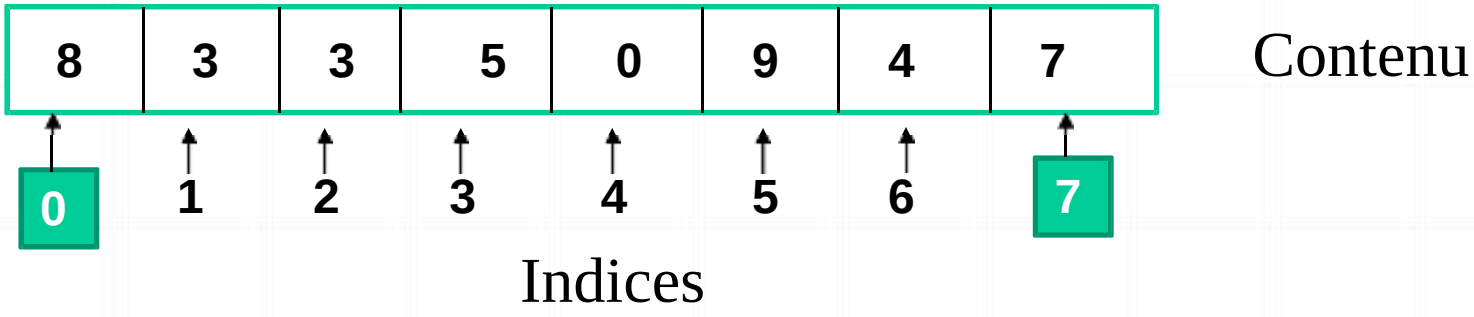


Déclarations (suite)

- Tableaux : notions

Un tableau a une taille fixe : cardinal, correspondant au nombre des valeurs pouvant être stockées dans le tableau.

Si N est la taille du tableau alors les indices des cases vont de 0 à N-1



Déclarations (suite)

- Tableaux : syntaxe

Syntaxe

variable *nom_var* : Tableau[*Taille*] de Type

Exemple:

variable notes : Tableau[8] d'entiers

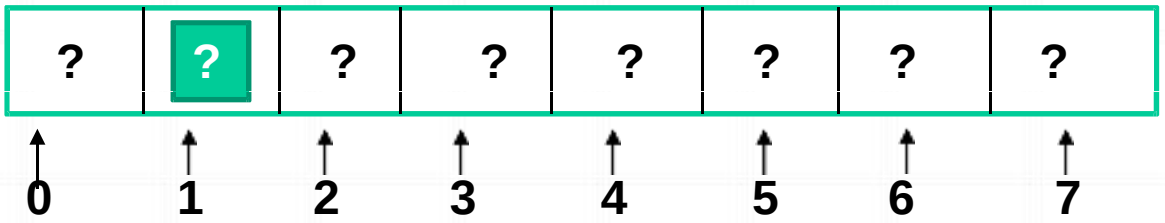
Ceci va créer une variable T qui est un tableau contenant 8 cases.
Chaque case peut contenir une valeur entière

Déclarations (suite)

- Tableaux : syntaxe

Attention : Lorsqu'on déclare un tableau, il n'est pas initialiser, c'est-à-dire, les valeurs des cases ne sont pas définies

variable notes : Tableau[8] d'entiers



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

Déclarations (suite)

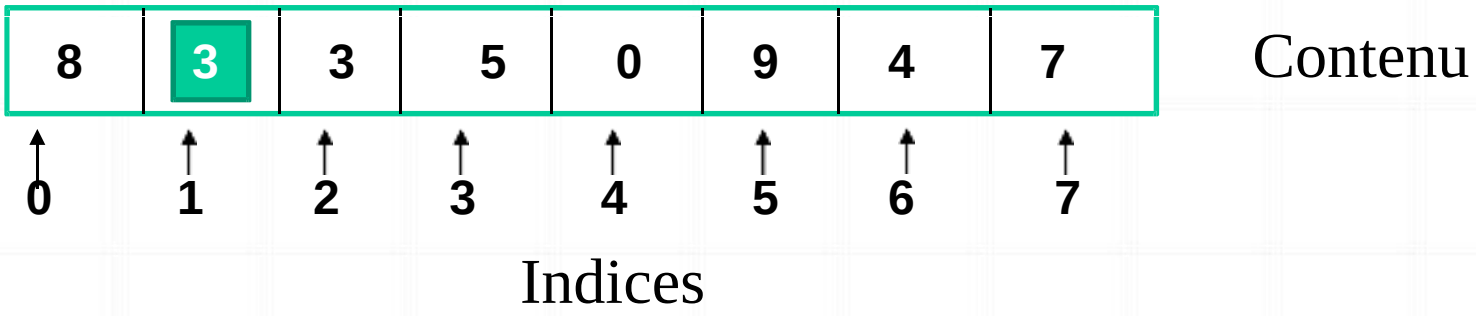
- Tableaux : accès aux tableaux

Accès aux cases des tableaux : se fait à travers l'indice de la case dans la tableau

Exemple

variable notes : Tableau[8] d'entiers

Pour accéder à deuxième case du tableau en écrit notes[1]



Attention : si vous essayez d'accéder à une case dont l'indice est supérieure à la taille du tableau, il se produit une erreur d'exécution

Corps (suite)

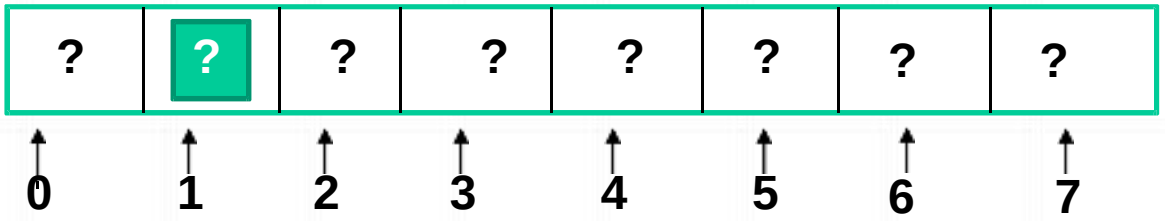
- Tableaux : accès aux tableaux

  Opérations sur les tableaux : l'affectations

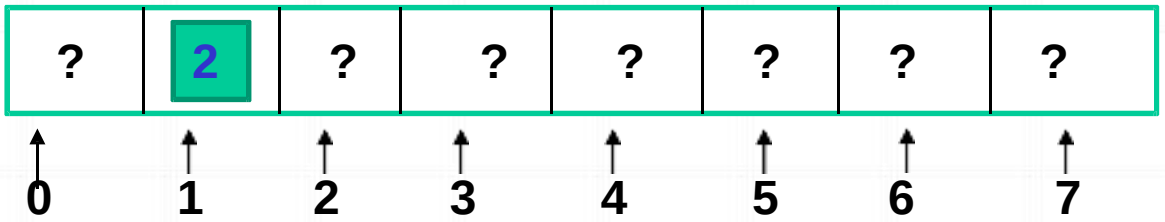
  Pour changer la valeur d'une case, on utilise ←

  Exemple

  variable notes : Tableau[8] d'entiers



  Maintenant si : notes[1] ← 2, on obtient



Corps (suite)

- Tableaux

Exemple : initialisation d'un tableau

```
Algorithme  initialisation

Constante MAX ← 20 : Entier
Variable tab : Tableau [ MAX ] d'entiers
Variable i : entier
Début
    Pour i ← 0 à MAX -1 Faire
        tab[ i ] ← 0
    Fin pour
Fin
```

Déclaration (suite)

- 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

Un tableau à deux dimension peut être vu comme une matrice

Syntaxe

variable *nom_var* : Tableau[*Taille1*][Taille2] de Type

Exemple:

variable notes : Tableau[10][5] d'entiers

Ceci va créer une variable T qui est un tableau contenant 10*5 = 50 cases.

Déclaration (suite)

- Tableaux à deux dimensions

Exemple : saisie des valeurs d'une matrice 10*20

```
Algorithme saisieMatrice
Variable matrice : Tableau[10][20] d'entiers
Variable i, j : Entier
Début
    Pour i ← 0 à 9 Faire
        Pour j ← 0 à 19 Faire
            Lire(matrice[ i ][ j ])
        Fin Pour
    Fin Pour
Fin
```

Déclaration (suite)

- Structures

On a vu que les tableaux permettent de regrouper des informations du même type (Entier, ….).

Mais, comment regrouper des informations de types différents.

Par exemple, une personne à

- Nom (Chaîne)
- Prénom (Chaîne)
- Date de Naissance (elle-même composée du jour + mois + année)

Déclaration (suite)

- Structures : syntaxe

Rôle : Représenter un ”objet” composé d’un ensemble fini d’éléments (champs ou attributs) de types différents.

Syntaxe

```
variable nom_variable : structure {
    champ1 : type1
    champ2 : type 2
    ……
}
```

Déclaration (suite)

- Structures : exemple

variable personne : structure (

Nom : Chaîne

Prénom : Chaîne

Date : Tableau[3] d'entiers)

L'accès aux éléments se fait par la notation pointée

personne.Nom ← "Ben …"

personne.Prénom ← "Ahmed"

personne.Date[0] ← 29

personne.Date[1] ← 06

personne.Date[0] ← 1978

Déclaration (suite)

- Type

Maintenant, si je veux remplir les informations relatives à deux personnes, je dois faire :

variable personne1 : structure {

Nom : Chaîne

Prénom : Chaîne

Date : Tableau[3] d'entiers }

variable personne2 : structure {

Nom : Chaîne

Prénom : Chaîne

Date : Tableau[3] d'entiers }

Puis remplir personne1 et personne2!!! Ce n'est pas pratique du tout

Déclaration (suite)

- Type : syntaxe

Pour éviter ca, je peux créer mon propre type.

Type Personne : structure {

Nom : Chaîne

Prénom : Chaîne

Date : Tableau[3] d'entiers }

Le nouveau type Personne est manipulé de la même façon que les types prédéfinis :

Variable moi : Personne

Variable toi : Personne

Déclaration (suite)

- Type : exemple

```
Algorithme  formulaire
Type Personne : structure {
        Nom : Chaîne
        Prénom : Chaîne
        Date : Tableau[3] d'entiers }
Variable toi : Personne
Début
    toi.Nom ←"Ton nom"
    toi.Prénom ←"Ton prénom"
    toi.Date[0] ← 29
    toi.Date[1] ← 06
    toi.Date[0] ← 1978
Fin
```

Déclaration (suite)

- Type : exemple

```
Algorithme  init
Constante MAX ←10 : Entier
Type  Vecteur : Tableau[MAX] d'entiers
Variable V1 : Vecteur
Variable  V2  :  Vecteur
Variable index : Entier
Début
      Pour index de 0 à MAX -1 Faire
            V1[index] ← 0
             V2[index] ← 0
      Fin Pour
Fin
```

Déclaration (suite)

- Exercice

  Supposez que vous avez un point P dans un plan, il est définit par le couple x et y. Un point peut être déplacer dans le plan.

  Ecrire un algorithme qui demande à l'utilisateur de saisir x et y et une valeur d et déplace le point P.

Déclaration (suite)

- Corrigé

```
Algorithme  deplacerPoint
Variable d : réel
Type Point :   structure {
      x : réel
      y : réel }
Variable P : Point
Début
      Ecrire("Donner x, y et d")
      Lire(P.x, P.y, d)
      P.x  ← P.x + d
       P.y ← P.y + d
Fin
```

```
Algorithme  deplacerPoint
Variable d : réel
Variable P :   structure {
      x : réel
      y : réel }

Début
      Ecrire("Donner x, y et d")
      Lire(P.x, P.y, d)
      P.x  ← P.x + d
       P.y ← P.y + d
Fin
```

Déclarations (suite)

- Déclarations

Entête

- Il permet l'identification de l'algorithme
- Mot-clé : Algorithme

Déclarations

- Liste exhaustive des variables, constantes, des structures, des fonctions et des procédures utilisées dans le corps de l'algorithme
- Mots-clés : variable, constante, structure , fonction, …

Corps

- C'est dans cette partie que les tâches (instructions, opérations,…) de l'algorithme sont placées
- Mots-clés : Début, Fin

Déclarations (suite)

- Déclarations

  La partie déclaration dans un algorithme contient une liste exhaustive des déclarations:

  - des variables,
  - des constantes;
  - des structures,
  - des fonctions,
  - des procédures,

  utilisées dans le corps de l'algorithme

  Les variables, constantes, structures, fonctions et procédures utilisées doivent avoir fait l'objet d'une déclaration préalable.

Fonctions et Procédures

- Analyse descendante

Problème : Comment traiter les problèmes trop complexes pour être appréhendés en un seul bloc ?

Solution : On peut appliquer la méthode "Diviser pour régner", et décomposer le problème en sous-problèmes plus simples : c'est l'analyse descendante

Fonctions et Procédures

- Analyse descendante

Rôle : Méthode pour écrire un algorithme de qualité : lecture plus facile d'un algorithme, modularité du code

Principe :

- Abstraire

Repousser la plus loin possible l'écriture de l'algorithme (codage)

- Décomposer

Décomposer la résolution du problème initial en une suite de "sous problèmes" que l'on considère comme résolus

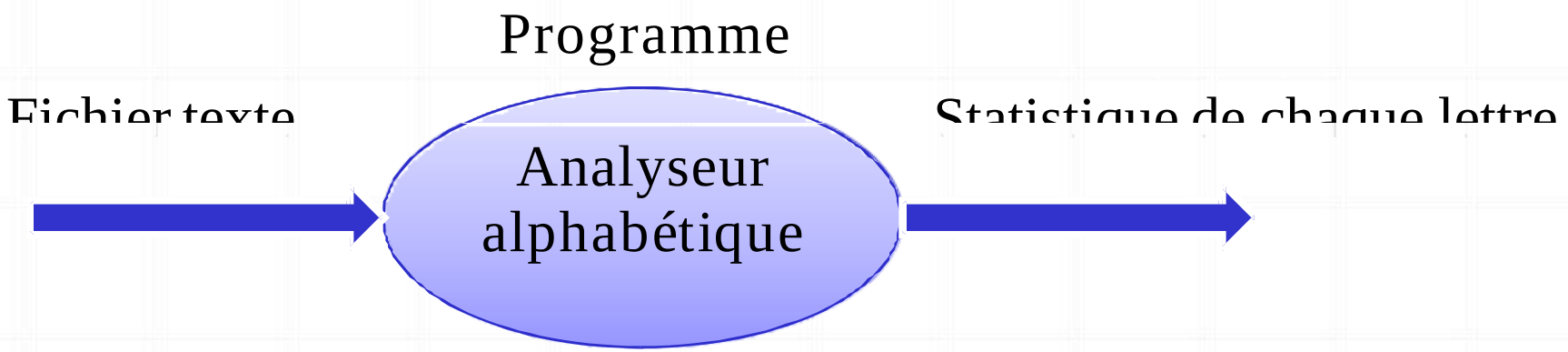
- Combiner

Résoudre le problème initial par combinaison des abstractions des "sous-problèmes"

Fonctions et Procédures

- Analyse descendante : exemple

Problème d'analyseur alphabétique : Écrire un programme analysant un fichier texte et indiquant le nombre d'occurrence de chaque lettre de l'alphabet dans le fichier



Fonctions et Procédures

- Analyse descendante : exemple
- Résoudre le problème revient à :

  Extraire un mot du fichier

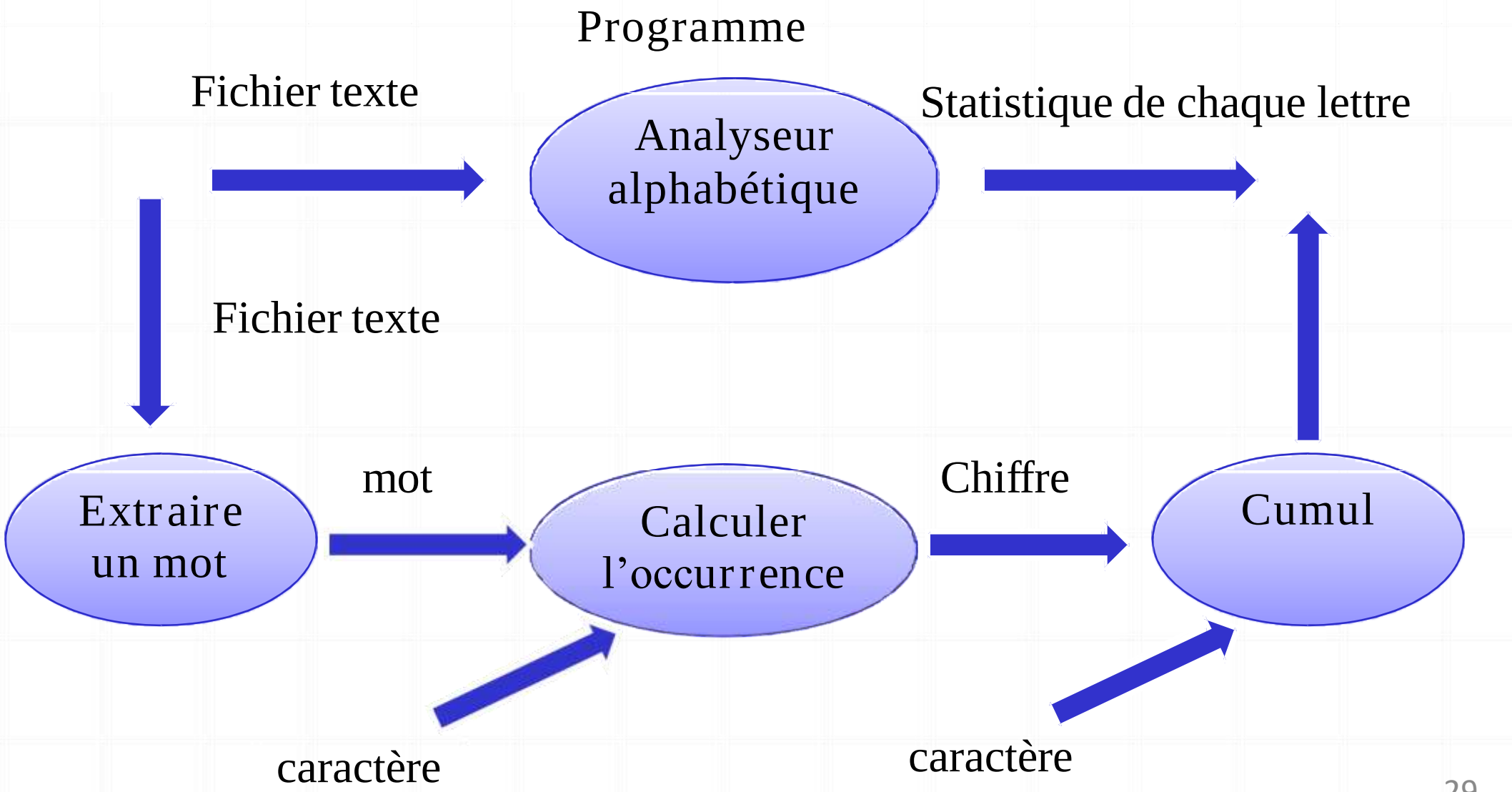
    Pour chaque, lettre de l'alphabet, compter le nombre d'occurrence

    Calculer le cumul d'occurrence de chaque lettre

    Répéter la traitement jusqu'au dernier mots du fichier.
- Chacun de ces sous-problèmes devient un nouveau problème à résoudre
- Si on considère que l'on sait résoudre ces sous-problèmes, alors on sait "quasiment" résoudre le problème initial

Fonctions et Procédures

- Analyse descendante : exemple



Fonctions et Procédures

- Analyse descendante :exemple

  Donc écrire un programme qui résout un problème revient toujours à écrire des sous-programmes qui résolvent des sous parties du problème initial. Il faut comprendre les mots "programme" et de "sous-programme" comme "programme algorithmique" indépendant de toute implantation

- En algorithmique il existe deux types de sous-programmes :
  - Les fonctions
  - Les procédures
- Un sous-programme est obligatoirement caractérisé par un nom (un identifiant) unique
- Lorsqu'un sous programme a été explicité (on a donné l'algorithme), son nom devient une nouvelle instruction, qui peut être utilisé dans d'autres (sous-)programmes

Fonctions et Procédures

- Analyse descendante : exemple

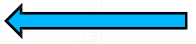
Considérons de nouveau le problème d'analyseur alphabétique

Fonctions et Procédures

- Analyse descendante :exemple

```
Algorithme analyseurAlphabétique
Variable fichier, mot : Chaîne
Variable tabOccurrences : Tableau[26] d'entiers
Début
    Ecrire ("Donner le nom du fichier à analyser« )
    Lire (fichier)
    mot ← extraireMotSuivant(fichier)
    Tant que mot ≠ " "  Faire
        CompterOccurrence(mot, tabOccurrences)
    Fin Tant que
Fin
```

Appel de Fonction

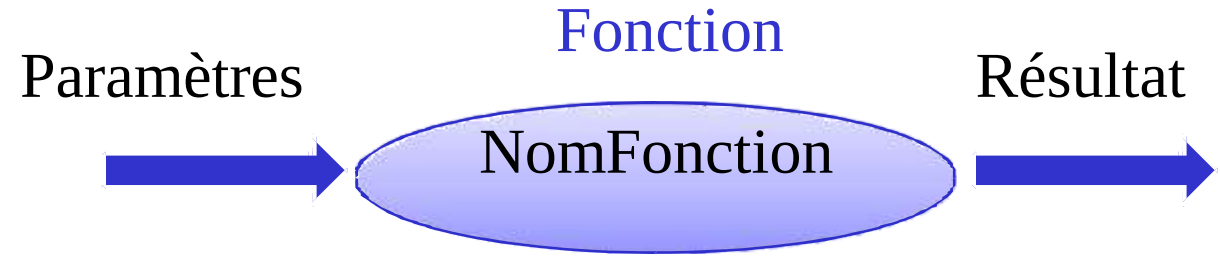


Appel de Procédure

Fonctions et Procédures

- Fonctions : syntaxe

Les fonctions sont des sous-programmes admettant des paramètres et retournant un seul résultat (comme les fonctions mathématiques y=f(x,y,. . .))



Syntaxe :

Fonction nomFonction (Paramètre1 : type1, ...) : typeRetour

variable variableLocale : type

….

Début

instructions

retourne ...

FinFonction

Fonctions et Procédures

- Fonctions

Exemple 1:

```
Fonction ValeurAbsolue (X : Entier) : Entier
Début
    Si X ≥ 0 alors
        retourner X
    Fin Si
    Sinon
        retourner -X
    Fin Sinon
Fin
```

Fonctions et Procédures

- Fonctions : appel de fonction

```
Algorithme abs
Variable a : Entier
Fonction ValeurAbsolue (X : Entier) : Entier
Variable valeurAbsolue : Entier
Début
        Si X ≥ 0 alors
              valeurAbsolue ← X
        Sinon
              valeurAbsolue ← -X
         Fin Sinon
        retourner  valeurAbsolue
Fin
Début
        a ← valeurAbsolue(-3)
        Ecrire ("la valeur absolue de 3 est ",a)
Fin
```

Fonctions et Procédures

- Fonctions : appel de fonction

```
Algorithme abs
Variable a, b : Entier
Fonction ValeurAbsolue (X : Entier) : Entier
Variable valeurAbsolue : Entier
Début
        Si X ≥ 0 alors
            valeurAbsolue ← X
        Sinon
            valeurAbsolue ← -X
         Fin Sinon
        retourner –X
Fin
Début
        Ecrire ("Entrez un entier :")
        Lire(a)
        b ← valeurAbsolue(a)
        Ecrire ("la valeur absolue de ",a," est ",b)
Fin
```

Fonctions et Procédures

- Fonctions

Exemple 2:

```
Algorithme min
variable a, b, c, mini : Entier
Fonction min2 (a , b : Entier) : Entier
Début
    Si a ≥ b alors
        retourner b
    Sinon
        retourner a
    Fin Sinon
Fin
```

```
Fonction min3 (a , b, c : Entier) : Entier
Début
    retourner min2(c, min2(a,b))
Fin
Début
    mini ← min3(3,5,8)
    Ecrire ("Le minimum est",mini)
Fin
```

Fonctions et Procédures

- Fonctions

Exemple 2:

```
Algorithme min
variable a, b, c, mini : Entier
Fonction min2 (a , b : Entier) : Entier
Début
    Si a ≥ b alors
        retourner b
    Sinon
        retourner a
    Fin Sinon
Fin
```

```
Fonction min3 (a , b, c : Entier) : Entier
Début
        retourner min2(c, min2(a,b))
Fin
Début
    Ecrire ("Saisir trois entier")
    Lire(a,b,c)
    mini ← min3(a,b,c)
    Ecrire ("Le minimum est",mini)
Fin
```