Les structures de données et les structures simples

A. Les structures de données

I. Les constantes

1. Définition

Une constante est une donnée connue. Sa valeur reste inchangée tout le long d'un programme (ou d'un algorithme).

Exemples : E = 2.718281
Pi = 3.14
Trouve = faux
Val = 20

2. Caractéristiques

Une constante est caractérisée par :

- Son nom : un identificateur unique
- Sa valeur

3. Déclaration en algorithme et en pascal

Déclaration en algorithme		
Objet	Type/Nature	Rôle
Nom de la constante	**Constante =** valeur de la constante	Rôle de la constante

Déclaration en pascal
CONST constante1 = valeur 1 ; constante 2 = valeur 2 ; … constante N = valeur N ;

Remarques :

- La valeur de la constante nous renseigne sur son type : réel, entier, booléen, …
- On utilise les constantes pour rendre le programme plus lisible et plus facilement modifiable

Activité : Déclarer une constante **message** de valeur "Bonne chance".

Déclaration en algorithme		
Objet	Type/Nature	Rôle
message	**Constante =** "Bonne chance"	--

En pascal :

Const message = 'Bonne chance' ;

II. Les variables

1. Définition

Une variable est un objet dont la valeur est susceptible d'être modifiée dans le temps. Elle est caractérisée par :

(1) Son nom : un identificateur unique

(2) Son type

(3) Son contenu

Remarques : Les identificateurs

Un identificateur est un mot que le programmeur choisit librement et qui lui permet de nommer son programme, ses données (constante, variables, …)

Les caractères autorisés pour construire un identificateur sont :

(1) Les lettres majuscules et minuscules non accentuées.

(2) Les chiffres

(3) Le caractère de soulignement « _ » **(tiré bas)**

Un identificateur ne peut commencer que par une lettre

Il est conseillé de choisir des noms d'identificateurs évocateurs

Exemples :

(1) Les identificateurs suivants sont corrects : Classe, ELEVE1 ; nombre_d_eleves ;

(2) Par contre ceux-ci sont incorrects :

Elève 1 (lettre accentuée non admise)

Nombre_d'élèves (l'apostrophe est interdite)

1ereclasse (on ne peut pas commencer par un chiffre)

2. Déclaration en algorithme et en pascal

Déclaration en algorithme		
Objet	Type/Nature	Rôle
Nom de la variable	Type de la variable	Rôle de la variable

Déclaration en pascal
VAR variable 1 : type1 ; variable 2 : type 2 ; … variable N : type N ;

Remarques :

(1) Variable 1, variables 2, …, variable N sont les identificateur des variables intervenant dans le programme (ou l'algorithme) et type 1, type 2, …, et type N sont les identificateurs de leurs types respectifs.$

(2) On peut regrouper ensemble les variables déclarées sous le même type.

Exemple :

```
Var   i, j, k: integer;
      Note: real;
      Ch: string;
```

(3) Lors de la déclaration d'une variable, le compilateur réserve dans la mémoire vive (RAM) un certain espace (appelé la taille de la variable) pour stocker la valeur future de cette variable. Suivant le type de celle-ci l'espace réservé ne sera pas de tout le même. Ainsi pour une variable de type entier, on réserve un espace de 2 octets alors qu'un réel occupe 8 octets.

Application :

Soit le programme pascal suivant :

```
Programme premier exemple ;
Uses wincrt ;
Conste cœfficient : 3 ;
Begin
Writeln('Donner trois notes entières') ;
Readln (a, b, c) ;
M := (a+b+c)/coefficient;
Writeln('La moyenne est :', M) ;
End ;
```

Questions :

1) Corriger le programme ci-dessus
2) Déclarer les variables utilisées.

III. Les types de données standard

Le type nous renseigne sur la valeur à affecter à la variable et sur l'ensemble des opérateurs qu'on peut appliquer sur la variable en question.

III.1. Les types numériques

1) Le type entier :

La mémoire vive de l'ordinateur n'étant pas infinie et par conséquent l'ensemble des entiers n'est pas égal à $\mathbb{Z}$. Si une variable est déclarée sous un type entier cela signifie que ses valeurs possibles sont les entiers appartenant à un certain intervalle **[Minint, Maxint].**

L'ensemble des entiers appartient à l'intervalle **[-32768, 32767]**.

Il existe des sous types du type entier : entier court, entier long, …

Chaque sou type indique un sous ensemble particulier de l'ensemble des nombres entiers.

Type d'entier	Identificateur	Intervalle	Taille
Entier	Integer	-32768..32767	2 octets
Entier court	Shortint	-128..127	1 octet
Octet	Byte	0..255	1 octet
Entier long	Longint	-2147483648..2147483647	4 octets
Mot	Word	0..65535	2 octets

Remarques :

(1) Les opérateurs arithmétiques qu'on peut appliquer sur une variable de type entier sont : l'addition (+), la soustraction (-), la multiplication (*), la division entière (**div**), le reste de la division entière (**mod**) et le changement de signe (-).

(2) Les opérateurs relationnels sont : **<, >, ≤, ≥, ≠, =**

(3) Déclaration en pascal : **Var** nom de la variable : integer ;

2) Le type réel :

Comme pour les entiers, l'ensemble des nombres réels informatiques n'est pas $\mathbb{R}$, c'est un ensemble fini

Les valeurs d'une variable de type réel sont donc délimitées par un intervalle de validité (correspondant cette fois à un espace de 8 octets).

Dans le langage pascal, tout réel est codifié par : un chiffre autre que 0 précédé du signe – si le nombre est négatif puis un point (qui correspond à notre virgule) suivi de dix chiffres et d'un exposant positif ou négatif à deux chiffres précédé de la lettre **E**.

Exemples :

Le nombre 12,5 est mémorisé par pascal sous la forme : 1.2500000000E+01 (qu'il faut le comprendre $1{,}25.10^{1}$). Cette écriture est appelée écriture scientifique des nombres (ou écriture à virgule flottante)

Par contre le nombre 1/3 est mémorisé sous la forme 3.3333333333E-01 (qu'il faut comprendre $3,3333333333.10^{-1}$)

Remarques :

(1) Nous ne sommes pas obligés d'adopter cette écriture dans la conception de nos programmes. A condition d'utiliser le point à la place de la virgule et la lettre **E** pour l'exposant.

(2) Les opérateurs arithmétiques qu'on peut appliquer sur une variable de type réel sont : l'addition (+), la soustraction (-), la multiplication (*), la division (/) et le changement de signe (-).

(3) Les opérateurs relationnels sont : **<, >, ≤, ≥, =, ≠** (En pascal : **<, >, <=, >=, <>, =**)

(4) Déclaration en pascal : **Var** nom de la variable : real ;$

Activité : soient les déclarations suivantes :

```
Const max = 1000 ;
Var     x, y : real ;
        A, B, C : integer ;
```

Compléter le tableau ci-dessous, dans le cas d'invalidité, donner une justification.

Expression	Valide ?	Résultat/Justification
C := A mod B		
C :=(990 - max) div A		
C :=A mod y		
X := A / B		
X := A mod (A / B)		
C := (max – 990) div A		
C := A mod 0		
X := A div B		

III.2. Le type booléen :

1. Présentation :

Une variable déclarée sous le type booléen est dite « booléenne ». Elle ne peut prendre que deux valeurs possibles : vrai (true) ou faux (false).

Les opérateurs possibles applicables sur les variables booléennes sont :

La négation NON (NOT)

La conjonction ET (AND)

La disjonction OU (OR)

Le OU exclusif OUex (XOR)

Activité :

Soient P et Q deux variables booléennes, compléter les tableaux de vérité suivants :

P	NON (P)
Vrai	
Faux	

P	Q	NON (P)	NON (Q)	P ET Q	P OU Q	P OUex Q
Vrai	Vrai					
Vrai	Faux					
Faux	Faux					
Faux	Vrai					

Remarques :

(1) Classement des opérateurs selon l'ordre de priorité décroissante :

NOT

*, /, div, mod, AND

+, -, OR

<, <= , =, <>, >=, >

(2) Les opérateurs se trouvant entre parenthèses sont prioritaires

2. Déclaration en algorithme et en pascal:

Objet	Type/Nature	Rôle
Nom de la variable	**Booléen**	Rôle de la variable

Var nom de la variable : **boolean** ;

Exemple :

Objet	Type/Nature	Rôle
OK	**Booléen**	---

Var OK **: boolean ;**

III.3. Le type caractère :

1. Présentation :

Une variable de type caractère a comme valeur un des 256 caractères connus : lettres minuscules, lettres majuscules, lettres accentuées, chiffres, caractères spéciaux ($, %, …).

La valeur d'une variable de type caractère est donnée par le caractère lui même encadré par deux guillemets en algorithme et par deux apostrophes en pascal.

Exemples :

"A", 'a', '2', "", …

Remarques :

(1) Chaque caractère possède un code appelé : Code **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

(2) Le caractère "" est un caractère (caractère vide)

(3) Une variable de type caractère ne peut contenir qu'un seul caractère

(4) Les caractères sont classés selon leurs codes ASCII. ("A"<"B"<"C"<"D"<"E"<…<"Z")

(5) Exemple : Code ("A") = 65, Code ("a") =97

(6) Les opérateurs relationnels sur les caractères sont : **<, >, ≤, ≥, =, ≠**

2. Déclaration en algorithme et en pascal :

Objet	Type/Nature	Rôle
Nom de la variable	**Caractère**	Rôle de la variable

Var nom de la variable : **char** ;

Exemple :

Objet	Type/Nature	Rôle
C	**Caractère**	---

Var C : **char** ;

3. Les fonctions prédéfinies sur les caractères : (activité)

III.4. Le type chaîne de caractères :

1. Présentation :

Une chaîne de caractères est une suite de **n** caractères **(0 ≤ n ≤ 255)**.

Si **n = 0** alors la chaîne est dite vide.

Exemple : "informatique", "programme", … sont des chaînes de caractères.

Remarques :

(1) Le caractère de position **i** dans une chaîne de caractère **ch** est noté, en algorithme et en pascal, par **ch [i]**

(2) Exemple : si ch = "informatique" alors ch [1] = "i" ; ch [3] = "f" ; …

2. Déclaration en algorithme et en pascal :

Objet	Type/Nature	Rôle
Nom de la variable	**Chaîne de caractères**	Rôle de la variable

Var nom de la variable : **String** ;

Exemple :

Objet	Type/Nature	Rôle
CH	**Chaîne de caractères**	---

En pascal :

Var CH : String ;

Remarques :

(1) Si on déclare une variable de type chaîne de caractères de la manière suivante :

Var CH : String ; Le compilateur réserve **256** octets dans la RAM pour stocker les caractères de la chaîne **CH** même si la chaîne contient un nombre de caractères inférieur à **256**.

(2) Au moment de la déclaration d'une chaîne **CH**, on peut fixer le nombre maximal des caractères qui constituent la chaîne **CH** de la manière suivante :

Objet	Type/Nature	Rôle
Nom de la variable	**Chaîne de n caractères**	Rôle de la variable

Var nom de la variable : **String [nombre de caractères]** ;

Exemple :

Objet	Type/Nature	Rôle
CH	**Chaîne de 30 caractères**	---

Var CH : String [30] ;

3. **Les fonctions prédéfinies sur les chaînes : voir page 26**
4. **Les procédures prédéfinies sur les chaînes : voir page 26**

Application :

Donner le résultat d'exécution des instructions suivantes.

Instructions	Résultats
A ← "L'" + "informatique"	A =
B← " un " + CHR (ORD ("a") + 4)	B =
C ← CONCAT ("esti", "mation")	C =
D ← SOUS-CHAINE ("gourmande", 5,5)	D =
EFFACE (B, POS ("e", B), 1)	B =
E ← CONCAT (" ", SOUS-CHAINE (C, 1, 3))	E =
D[2] ← "o"	D =
B ← CONCAT (B, " ", D, " ", SOUS-CHAINE (D, 4, 2), " ")	B =
A ← CONCAT (A, E, " ", B, A [5] +A [3], "c", SOUS-CHAINE (C, 7,4))	A =

III.5. Les types énumérés

1. Présentation :

Un type énuméré permet de représenter des valeurs en les énumérant à l'aide de leurs noms. Une variable de type énuméré ne peut stocker qu'une seule valeur possible parmi une liste finie de valeurs (au maximum 256).

Exemple :

Couleur = (Rouge, Bleu, Vert)

Saison = (Automne, Hiver, Printemps)

2. déclaration en algorithme et en pascal :

Objet	Type/Nature	Rôle
Nom de la variable	**Liste des valeurs**	Rôle de la variable

En pascal :

Type Nom_du_type = (valeur1, valeur2, …) ;

Var nom de la variable : **Nom_du_type** ;

Exemple :

Objet	Type/Nature	Rôle
Col	**Rouge, Bleu, Vert**	---
saison	**Automne, Hiver, Printemps, Ete**	

Type Couleur = (Rouge, Bleu, Vert) ;
Saison = (Automne, Hiver, Printemps, Ete) ;
Var Col : **Couleur** ;
S : **Saison** ;

Remarques :

(1) Les valeurs d'un type énuméré sont ordonnées selon leur ordre de déclaration. Chaque valeur énuméré correspond à un numéro (0, 1, 2, …).

(2) Exemple : Rouge porte le numéro 0, Bleu a le numéro 1 et Vert a le numéro 2.

(3) Les opérateurs relationnels sur le type énuméré sont : $<$, $>$, $\leq$, $\geq$, $=$, $\neq$

(4) Une valeur d'un type énuméré ne peut pas paraître dans plus qu'une déclaration de type énumérée. **Exemple** : **type** jours = (Lundi, mardi, mercredi, jeudi) ;
Joursf = (Lundi, dimanche) ;

Cette écriture est invalide, car la valeur Lundi apparaît dans les deux déclarations.

(5) On ne peut pas comparer deux valeurs appartenant à deux types énumérés différents. Exemple : on peut écrire Rouge < Bleu mais non Dimanche > Rouge

(6) Les fonctions prédéfinies :

Exemples :

Soient les déclarations pascal suivantes :

Type jour = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche) ;

Var jour_courant, demain : jour ;

On peut utiliser les fonctions prédéfinies suivantes :

Jour_courant :=mercredi ;

Ord (jour_courant) = 3

Ord (demain) = 4

Succ (jour_courant) = jeudi

Pred (jour_courant) = mardi

Jour_courant := dimanche ;

Succ (jour_courant) est non définit / Déduire alors le rôle de chaque fonction.

(7) On ne peut ni lire, ni écrire une variable de type énuméré. Pour l'afficher, on peut utiliser une instruction case par exemple.

III.6. Les types utilisateurs (intervalles)

1. Déclaration en algorithme et en pascal :

Un type utilisateur est type définit par le programmeur lui-même.

Type
Nom du type = borne_inf ..borne_sup

Objet	Type/Nature	Rôle
Nom de la variable	**Nom du type**	Rôle de la variable

```
Type nom du type = borne_inf .. borne_sup ;
Var nom de la variable : nom du type ;
```

Exemples :

Type
Notes_possibles = 0 .. 20

Objet	Type/Nature	Rôle
Note	**Notes_possibles**	---

```
Type notes_possibles = 0..20 ;
Var note : notes_possibles ;
```

III.7. Les tableaux

1. Les tableaux à une dimension :

a) Définition :

Un tableau unidimensionnel (ou vecteur) est une structure de données permettant de ranger (regrouper) un nombre fini d'éléments de même type.

Un vecteur est caractérisé par :

Son nom (un identificateur unique. Exemple : **T, V, U**, …)

Sa taille (nombre d'éléments. Exemple : 20, 30, 100, …)

Son type (le type des éléments qu'il contient. Exemple : entier, réel, caractères, …)

b) Déclaration en algorithme et en pascal (première formulation) :

Objet	Type/Nature	Rôle
Nom du tableau	**Tableau de taille de type**	Rôle du tableau

Var nom du tableau : **array** [indmin .. indsup] **of** type ;

Exemple :

Objet	Type/Nature	Rôle
Moyenne	**Tableau de 30 réels**	Pour stocker les moyennes des élèves

Var Moyenne: array [1..30] **of** real ;

c) Déclaration (deuxième formulation) :

Type
Nom du type = tableau de taille et de type

Objet	Type/Nature	Rôle
Nom de la variable	**Nom du type**	Rôle de la variable

Type nom du type = array [indmin .. indsup] of type ;
Var nom de la variable : nom du type ;

Exemple :

Type
Tab = tableau de 20 entiers

Objet	Type/Nature	Rôle
T	**Tab**	---

```
Type Tab = array [1.. 20] of integer ;
Var T: Tab;
```

2. Les tableaux à deux dimensions (les matrices) :

a) Présentation :

Un tableau à deux dimensions peut être simulé à une grille de **n** lignes et de **m** colonnes.

Les éléments stockés dans ce type de tableau sont de même type.

b) Déclaration en algorithme et en pascal (première formulation) :

Objet	Type/Nature	Rôle
Nom du tableau	**Tableau de taille de type**	Rôle du tableau

```
Var nom du tableau : array [indmin1 .. indsup1 , indmin2..insup2] of type ;
```

c) Déclaration (deuxième formulation) :

Type
Nom du type = tableau de taille et de type

Objet	Type/Nature	Rôle
Nom de la variable	**Nom du type**	Rôle de la variable

```
Type nom du type = array [indmin1 .. indsup1 , indmin2..indup2] of type ;
Var nom de la variable : nom du type ;
```

Exemple:

```
Type jeu = array [1..8 , 1..8] of integer ;
     Matrice = array [1..3 , 1..3] of real;
Var  echec: jeu;
     M: matrice;
```

Remarques :

(1) Un élément du tableau est repéré par le numéro de la ligne et le numéro de la colonne. Exemple : M [i, j] avec i et j compris entre 1 et 3.

(2) Sur les éléments d'un tableau, on peut effectuer les mêmes opérations et exécuter les mêmes instructions que sur n'importe quelle variable du même type :

Lire : Ecrire ("entrer coordonnées"), Lire (M [1,1])

Ecrire : Ecrire (M [1,3])

Affecter : M [1,2] ← 1.6

Effectuer un calcul : trace ← M [1,1] +M [2,2] + M [3,3]

Comparer : Si M [1,3] = M [2,3] Alors …

(3) Si T et U sont deux tableaux de même type alors l'instruction T ← U transfère en bloc tout le tableau U dans le tableau T. Cette opération est appelée **affectation de transfert**

(4) Les éléments d'un tableau n'ont pas de valeurs par défaut. Il faut penser à les initialiser avant de les utiliser.

(5) En dehors de l'affectation de transfert, **aucune instruction n'est utilisable pour un tableau en bloc**. Ainsi l'initialisation d'un tableau T à 0 ne peut pas se faire par l'instruction T ← 0. De même, on ne peut pas utiliser les instructions écrire et lire pour afficher ou lire un tableau T en entier.

(6) L'affichage ou la lecture des éléments d'un tableau se fait **un par un**.

B. Les structures simples

I. Introduction

Activité :

Soit l'algorithme suivant :

```
0) Début exercice
1) Ecrire ("entrer un entier : "), Lire (x)
2) Ecrire ("entrer un autre entier :"), Lire (y)
3) M ← (x + y) / 2
4) Ecrire ("la moyenne de ", x, " et ", y, " est : ", M)
5) Fin exercice
```

Questions :

1) Quelles sont les structures de données utilisées ?
2) Quelles sont les actions utilisées pour : faire entrer les deux entiers, pour calculer M et pour afficher le résultat ?
3) Qu'appelle-t-on alors une structure simple ?

Définition :

Une structure simple peut être soit :

Une instruction de lecture (entrée, saisie) de données

Une instruction d'écriture (sortie, affichage) de résultat

Une affectation (modification du contenu d'une variable)

II. L'opération d'entrée (Lecture/Read)

1. Présentation

L'instruction qui permet à l'utilisateur d'entrer (lire ou saisir) des valeurs au clavier (entrée standard) (ou autre source d'entrée) s'appelle opération de lecture (ou d'entrée).

La valeur lue sera affectée (recopiée) (d'une manière interne) à la case mémoire de la variable : c'est une affectation implicite (indirecte).

2. Syntaxe en analyse, en algorithme et en pascal :

Syntaxe en analyse	Syntaxe en algorithme	Syntaxe en pascal
Variable = donnée	Lire (Variable)	Read (variable) ;

Remarques :

(1) Dés que le compilateur rencontre une instruction de lecture, l'exécution s'arrête en attendant que l'utilisateur saisisse une valeur pour la variable puis il valide la saisie par la touche Entrée. Après la validation, le compilateur passe à l'instruction suivante.

(2) On peut regrouper plusieurs variables dans la même instruction, dans ce cas la lecture se fait variable par variable. Les valeurs entrées seront affectées aux variables successives. Exemple : Lire (var1, var2, …, var n)

(3) La procédure **ReadLn** effectue, après la lecture des données, un passage à la ligne. **Exemple** : Readln (x, y, z) ;

(4) Par souci de clarté nous ferons toujours précéder les lectures de données par un message qui demande à l'utilisateur qu'il doit entrer.

Exemple : note = donnée **<==>** Ecrire ("entrer une note :"), Lire (note)

(5) En pascal, les instructions successives sont séparées par des points virgules.

(6) Le point virgule : pour ne pas se tromper, il suffit de se souvenir que le point virgule ne désigne pas la fin d'une ligne ou d'une instruction **mais** qu'il sépare deux instructions successives.

III. L'opération de sortie (Ecriture/Write)

1. Présentation

L'instruction qui permet d'afficher (d'écrire ou de sortir) des résultats (texte ou des valeurs) à l'écran (sortie standard) (ou autre périphérique de sortie) s'appelle opération de sortie.

2. Syntaxe en algorithme et en pascal :

Afficher ?	Syntaxe en algorithme	Syntaxe en pascal
Texte	Ecrire ("Texte")	Write ('Texte') ;
Contenu d'une variable	Ecrire (variable)	Write (variable) ;
Texte + Contenue de variable	Ecrire ("Texte", variable)	Write ('Texte', variable);

Remarques :

(1) On peut afficher le résultat d'une expression par : Ecrire (expression).

Exemples : Ecrire (x/2-1), Ecrire (X>Y), …

(2) Afficher un mélange : Ecrire (a, "+", b, "+", c, "=", m)

(3) La procédure **WriteLn** permet d'effectuer un retour à la ligne après l'affichage d'un résultat.

(4) Les procédures write et writeln permettent des effets de mise en page : il suffit de faire suivre le nombre ou la variable à afficher d'indications de mise en page. Ces indications s'écrivent :

Write (variable : champ : nombre des décimaux) ;

Où champ = espace réservé pour l'affichage des chiffres, virgule comprise

Exemple :

Si x est une variable réelle ayant pour valeur 10^{-6}, l'instruction write (x : 8 : 6) affichera 0.000001 (essayer avec d'autres valeurs sur machine pour voir la différence)

IV. L'opération d'affectation

1. Définition

L'affectation est l'instruction qui permet d'attribuer (affecter) une valeur à une variable ou de modifier la valeur qu'elle a déjà. Sa syntaxe est :

Syntaxe en algorithme	Syntaxe en pascal
Variable ← expression	Variable := expression

Remarques :

(1) Expression : est une expression arithmétique dont la valeur sera affectée à la variable.

(2) Il doit y avoir une comptabilité entre le type de la variable et celui de l'expression.

(3) Exemples :

Exemples	Commentaires
Note ← 12	Affecte à Note la valeur 12
X ← Y	Affecte à X la valeur de la variable Y
X ← 3*Y+1	Calcule la valeur de 3*Y+1 et l'attribue à X
Z ← exp (y-1)	Calcule la valeur y-1 puis son exponentielle et attribue sa valeur à Z
K ← K+1	Ajoute 1 à la valeur de K (incrémentation)
K ← K-1	Diminuer la valeur de K de 1 (décrémentation)