Chapitre 1 : Introduction, les structures simples.

1. Algorithme

1.1 Notion d'algorithme:

*"Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données."*l' Encyclopaedia Universalis

A la base de tout programme, il existe donc un algorithme. En fait, un programme n'est rien d'autre qu'une traduction d'un algorithme ne un langage de programmation.

1.2 Exemple :

Si on prend l'exemple suivant:

Données fournies : 2 entiers.

Traitement : Faire la somme de ces deux entiers.

Résultat à afficher : La somme.

Comment écrire un algorithme qui résout cet exemple?

1.3 Language de description : Algorithme

On peut donc dire que tout problème (généralement) se décompose en 3 parties:

- La partie lecture : dans laquelle les données fournies sont "lues"
- La partie traitement : dans laquelle un ou plusieurs opérations sont effectuées pour atteindre les résultats voulus.
- La partie écriture : dans laquelle les résultats sont affichés

En appliquant ceci à notre exemple on trouve qu'on doit écrire cela :

1. La lecture de deux entiers
2. Calculer la somme
3. Afficher cette somme

Si on essaye d'écrire cela convenablement, en langage de description, on doit écrire

0) Début Calcul_Somme
1) Lire (entier1)
2) Lire (entier2)
3) somme <-- entier1 + entier2
4) Ecrire (somme)
5) Fin Calcul_Somme

- entier1, entier2 et somme sont dits *variables* (on expliquera cela après, paragraphe 2)
- Il est conseillé d'utiliser des noms de variables *significatifs*
- On peut distinguer deux verbes dans l'exemple :
    - lire : Introduire une valeur dans une variable
    - écrire : Afficher le contenu de cette variable

- Il est également conseillé d'ajouter quelques messages pour faciliter la compréhension de l'algorithme d'abord, et du programme par l'utilisateur ensuite, notre algorithme deviendra donc

0) Début Calcul_Somme
1) Ecrire ("Donner le premier entier")
2) Lire (entier1)
3) Ecrire ("Donner le premier entier")
4) Lire (entier2)
5) somme <-- entier1 + entier2
6) Ecrire ("La somme est : ", somme)
7) Fin Calcul_Somme

1.4 Forme générale d'un algorithme :

0) Début *Nom_De_L_Algorithme*
1) Etape 1
2) Etape 2
.....
n-1) Etape n-1
n) Fin *Nom_De_L_Algorithme*

1.5 Applications :

- Ecrire un algorithme permettant de calculer la surface d'un cercle ayant son rayon. On rappelle que : Surface cercle = 3.1415*rayon²
- Ecrire un algorithme permettant de lire 2 entiers, et d'afficher leur somme, produit, et différence.

2. Variables, affectation

2.1 Les variables :

Les variables jouent le rôle de **"tiroirs"** dans lesquels on place une valeur durant l'exécution de l'algorithme. En effet, une variable est une *case mémoire* caractérisée par un nom, un type et une valeur. On reviendra dans le paragraphe 3 sur les types de variables.

2.2 Les opérations sur les variables par structures simples :

On peut effectuer, par structures simples, 3 opérations sur les variables :

1. Lecture : on met une valeur entrée dans une variable
2. Ecriture : on affiche le contenu d'une variable (afficher sa valeur)
3. Affectation : On met le résultat d'une expression dans une variable (étape 5 de notre exemple)

2.3 Les constantes :

Il existe aussi une autre forme de représentation de données : les constantes. Une constante est une case mémoire dont la valeur reste inchangée tout le long de l'exécution du programme.

3. Les structures de données (en langage de description et en Pascal)

3.1 Les structures de données simples :

3.1.1 Les entiers : integer

3.1.1.1 Les valeurs prises par un entier :

Un entier est un nombre qui appartient à **Z**, il est compris entre -32768 et 32767 (On ne pourra pas gérer des nombre plus grand avec ce type, cependant il existe d'autre types qui permettent des valeurs beaucoup plus grande. (On verra ça dans le cours de chapitre 6)

3.1.1.2 Les opérations sur les entiers :

5 opérations sont possibles sur les entiers : + pour la somme, - pour la différence, * pour la multiplication, div pour le résultat de la division entière et mod pour le reste de la division entière :

Exemples :

21 div 4 = 5

21 mod 4 = 1

3.1.2 Le type réel : real

3.1.2.1 Les valeurs prises par un réel :

Un réel est un nombre qui appartient à **IR**, de même on ne peut pas gérer des réels très grands ou trés petits, la machine à des limites aussi*!*

3.1.2.2 Les opérations sur les réels :

4 opérations sont permises : + pour la somme, - pour la différence, * pour la multiplication, / pour la division sur un nombre différent de 0.

La manipulation des nombres sera expliquée dans le paragraphe 5.1

3.1.3 Le type caractère : char

Un cacrtère peut être alphanumérique, espace, symbole..., il n y a pas des opérations sur les caractères cependant il y des fonctions prédéfinis qu'on verra plus tard (paragraphe 5.2)

3.1.4 Type booléen : booléen

3.1.4.1 Les valeurs prises par un booléen :

Un booléen peut avoir deux valeurs : Vrai : TRUE ou faux : FALSE.

3.1.4.2 Les opérations sur les booléens :

Pour effectuer ces opérations on utilise les *connecteurs logiques* :

Non : NOT

entrée	sortie
FAUX	VRAI
VRAI	FAUX

Tableau 1.1 Table de vérité : NOT

Et : AND

entrée1	entrée 2	sortie
FAUX	FAUX	FAUX
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX

VRAI	VRAI	VRAI

Tableau 1.2 Table de vérité : AND

Ou : OR

entrée1	entrée 2	sortie
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	VRAI

Tableau 1.3 Table de vérité : OR

Ou Ex (Ou exclusif): XOR

entrée1	entrée 2	sortie
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	FAUX

Tableau 1.4 Table de vérité : XOR

3.1.5 Le type chaîne de caractères : string

une chaîne de caractère consiste au plus en 255 caractères. Il n'y a pas d'opérations sur les chaînes de caractères, en revanche il y a des procédures et fonctions prédéfinies qui permettent de les manipuler, on détaillera ceci dans le paragraphe 5.3.

3.2 Les structures de données composées

On s'intéressera surtout aux tableaux unidimensionnels : les vecteurs. D'autres structures seront vues dans les prochains chapitres.

3.2.1 Exemple :

On veut stocker les notes de 10 élèves pour un certain traitement. On a donc :

Le tableau *Notes* qui contient 10 réels :

13	10.5	15	18.25	17	9	16	15.25	12	16

Ce tableau est un vecteur, qui est remplie comme suit (Algorithme) :

Notes [1] <-- 13

Notes [2] <-- 10.5

….

Notes [10] <-- 16.

Commentaires :

- L'accès à une case du vecteur se fait par accès direct comme pour la chaîne de caractères.

- Pascal ne contient aucune fonction ou procédure prédéfinie qui agit sur les tableaux, sauf l'affectation est donc permise.

Une étude plus détaillée sur les tableaux sera faite dans le chapitre 5.

4. Traduction Pascal

Revenons à notre exemple :

0) Début Calcul_Somme
1) Ecrire ("Donner le premier entier")
2) Lire (entier1)
3) Ecrire ("Donner le premier entier")
4) Lire (entier2)
5) somme <-- entier1 + entier2
6) Ecrire ("La somme est : ", somme)
7) Fin Calcul_Somme

Remarque : avec l'algorithme on doit **impérativement** ajouter un tableau dit **Tableau de déclaration des objets** : **T.D.O.**

Objet	Nature/Type
entier1, entier2, somme	variable/entier

Sa traduction en Pascal est :

```
Program Exemple
uses wincrt
var
        entier1, entier2, somme : integer;
Begin
        writeln ('Donner le premier entier');
        readln (entier1);
        writeln ('Donner le deuxième entier');
        readln (entier2);
        somme:=entier1+entier2;
        writeln ('La somme est ', somme);
end.
```

Les règles de base :

1° Un programme Pascal se compose de trois parties:

1. Un en-tête, caractérisé par le mot PROGRAM
2. Une section déclarative introduite ici par le mot VAR
3. Une section instruction ou corps du programme, délimitée par les mots BEGIN et END.

Attention: Le programme se termine par un point.

2° L'en tête (facultative) sert à donner un nom au programme selon la forme:

PROGRAM identificateur;

3° Un identificateur en Pascal doit débuter par une lettre suivie par un nombre quelconque de **lettres, chiffres** ou de " **_** " (caractère souligné). Les identificateurs ne peuvent contenir d'espacement (caractère "blanc") ou de caractères tels que %? *, ., -,...etc. et ne doivent pas être des noms **réservés** (voir commentaire 8)

4° Les variables doivent faire l'objet d'une déclaration de **type** de la forme:

VAR *liste des variables* : type;

5° Des **points-virgules** sont obligatoires pour séparer les trois parties et pour séparer les instructions.

6° Les instructions de **lecture** et **d'écriture** se traduisent respectivement par **READ** ou **READLN** et **WRITE** ou **WRITELN** (on verra la différence à la fin de ce paragraphe) suivis d'une liste de variables ou d'expressions placées entre parenthèses et séparées par des virgules.

7° L'assignation se représente par " **:=** "

8° Les mots **PROGRAM, VAR, BEGIN, END, DIV, MOD,** ... ont un sens précis dans le langage: ce sont des **mots réservés** qui ne peuvent être choisis comme identificateurs par le programmeur.

Un certain nombre de mots tels que **INTEGER, READLN, WRITE,** ... ont une signification prédéfinie. Pour éviter toute erreur, on s'abstiendra de les choisir comme identificateur.

9° Les mots du langage et les identificateurs doivent être séparés les uns des autres par un ou plusieurs blancs.

Lecture.
Pascal admet la procédure **READ** (qui a le même effet que **READLN** sans passage à la ligne pour le prochain affichage) mais il s'est révélé à l'usage, que celle-ci était parfois source de problème et il est préférable de l'éviter.

Ecriture.
Les écritures se font de façon semblable aux lectures, à l'aide de la procédure **WRITE.**

Pour améliorer la lisibilité, on peut:

Utiliser la procédure **WRITELN** qui force le passage à la ligne suivante pour le prochain affichage

Faire usage des **formats d'édition** qui précisent le nombre de caractères à utiliser pour afficher chacun des résultats:

- **WRITE** (valeur_*entière* : n) affiche la valeur entière sur n positions (insertion d'espacement à gauche du nombre si il y a trop peu de chiffres et ajustement automatique, si n est insuffisant)
- **WRITE** (valeur_*réelle)* affiche le nombre en notation scientifique (x.xxxxxE+x précédé d'un espacement)
- **WRITE** (valeur_*réelle* : n) affiche le nombre en notation scientifique sur n positions
- **WRITE** (valeur_*réelle* : n1 : n2) affiche le nombre sur n1 positions avec n2 décimales (avec ajustement).
- **WRITE** (chaîne : n) affiche la chaîne de caractère sur n positions (insertion d'espacement à gauche de la chaîne si il y a trop peu de caractères et ajustement automatique, si n est insuffisant)

Exemples:

- Si la variable entière x contient 12345, (^ **symbolise l'espacement)**
    - **WRITE** (x) affiche 12345
    - **WRITE** (x:8) affiche ^^^12345
    - **WRITE** (x:2) affiche 12345
- Si la variable réelle x contient 123.4567, (^ **symbolise l'espacement)**

- **WRITE** (x) affiche ^1.23456E+2
- **WRITE** (x:7) affiche ^1.2E+2
- **WRITE** (x:8:2) affiche ^^123.46
- **WRITE** (x:2) affiche 1.2E+2

- Si la variable du type chaîne x contient 'AZERTY', (^ **symbolise l'espacement)**
  - **WRITE** (x) affiche AZERTY
  - **WRITE** (x:8) affiche ^^AZERTY
  - **WRITE** (x:3) affiche AZERTY

5. Manipulation de variables :

5.1 Manipulation des nombres : Fonctions mathématiques

Algorithme	Fonction Pascal	Type de x	Type du résultat	Signification
Abs(x)	ABS(x)	Entier ou réel	Type de x	Valeur absolue de x
Carré(x)	SQR(x)	Entier ou réel	Type de x	Carré de x
Racine (x)	SQRT(x)	Entier ou réel	Réel	Racine carré de x
Sin(x)	SIN(x)	Entier ou réel	Réel	sin de x (x en radians)
Cos(x)	COS(x)	Entier ou réel	Réel	cos de x (x en radians)
Arctg(x)	ARCTAN(x)	Entier ou réel	Réel	Angle (en radians) dont la tangente vaut x
Exp(x)	EXP(x)	Réel	Réel	Exponentielle de x
Ln(x)	LN(x)	Réel	Réel	Logarithme népérien de x
Tronc(x)	TRUNC(x)	Réel	Entier	Partie entière de x
Int(x)	INT(x)	Réel	Réel	Partie entière de x
Arrondi(x)	ROUND(x)	Réel	Entier	Entier le plus proche de x
Frac(x)	FRAC(x)	Réel	Réel	Partie décimale de x

Tableau 1.5 : Fonctions mathématiques en Pascal

5.2 Manipulation des caractères :

Nom	Rôle	Exemples
Ord(c)	Renvoie le code ASCII du caractère c	Ord ("A")=65
Chr(x)	Renvoi le caractère dont le code ASCII est x	Chr (97)="a"
Succ(c)	Renvoie le caractère successeur de c	Succ ("D")="E"
Pred(c)	Renvoie le caractère prédécesseur de c	Pred ("B")="A"

Tableau 1.6 Fonction prédéfinies sur les caractères

5.3 Manipulations des chaînes de caractères :

Notation en LDA	Fonction Pascal	Type du résultat	Signification
Long (Ch)	LENGTH (Ch)	Entier	Nombre de caractères dans Ch
Concat (Ch1,Ch2)	CONCAT(Ch1,Ch2) Ch1 + Ch2	Chaîne	Concaténation (juxtaposition) de Ch1 et Ch2
Copier (Ch, i, j)	COPY (Ch, i, j)	Chaîne	Extraction, dans Ch, des caractères du postions i j caractères

Tab 1.7 Fonctions prédéfinies sur les chaînes de caractères