Chapitre 2 : Les algorithmes séquentiels simples

1. Parties d'un algorithme

Un algorithme contient deux parties:

- La partie données (déclaration) : contient les variables et les constantes.
- La partie traitement (code, opérations, corps de l'algorithme) : elle correspond au processus de calcul.

2. Données

Les données sont les objets manipulés dans l'algorithme. Dans un algorithme, toute donnée utilisée doit être déclarée. Les données peuvent être des variables ou des constantes.

Les variables : Une variable correspond à un objet dont la valeur peut varier au cours de déroulement de l'algorithme. Une variable est caractérisée par :

- Le nom (identificateur) qui doit être explicite, i.e. indique le rôle de la variable dans l'algorithme.
- Le type : indique les valeurs qui peuvent être prises par une variable.
- La valeur : indique la grandeur prise par la variable à un moment donné.

Sur le plan technique, une variable correspond à une case mémoire avec : le non de la variable est l'adresse de la case mémoire ; le type indique la taille (le nombre d'octets) de la case ; la valeur représente le contenu de la case.

Les constantes : Une constante est un cas particulier de la variable. Il s'agit d'une variable dont la valeur est inchangeable dans l'algorithme tout entier.

Exemple :

```
Algorithme calcul_surface ;
Constantes
  PI=3.14 ;
Variables
  rayon, surface : réel ;
…
```

En Pascal :

```
program calcul_surface ;
const
  PI=3.14 ;
var
  rayon, surface : real ;
…
```

3. Types

Le type correspond au genre ou la nature de la variable que l'on souhaite utiliser. Il indique donc les valeurs qui peuvent être prises par cette variable.
La déclaration d'une variable consiste à l'associer à un type. Chaque type donné peut être manipulé par un ensemble d'opérations.
Il existe des types simples et des types structurés.
Les types simples : sont des types dont les valeurs sont primitives, élémentaires non décomposables. A leur tour, les types simples peuvent être classés en deux catégories :

1. Types numériques :
   - Entier : par exemple 12, -55. Les opérateurs de manipulation des entiers sont :
     - Les opérateurs arithmétiques classiques : +, -, *.
     - La division entière, notée ÷, avec n ÷ p donne la partie entière du quotient de la division de n par p.
     - Le modulo, noté MOD ou %, avec n MOD p donne le reste de la division de n par p.
     - Les opérateurs de comparaison classiques : <, >, =, etc.
   - Réel : par exemple 12.5, -2.09. Les opérateurs de manipulation des réels sont :
     - Les opérations arithmétiques classiques : +, -, *, /.
     - Les opérateurs de comparaison classiques : <, >, =, etc.
2. Types Symboliques :
   - Booléen : les deux valeurs VRAI et FAUX. Les opérateurs de manipulation sont NON, ET, OU.

- Caractère : comporte les données alphanumériques, symboliques, ponctuation, etc., contenant un seul caractère, par exemple 'a', '? ', '3', ' ; '. Notons qu'un caractère se met entre deux guillemets simples. Les opérations de manipulation des caractères sont les opérateurs de comparaison : >, <, =, etc. Le résultat de la comparaison dépend du code ASCII des caractères comparés.

Les types structurées : c'est tout type dont la définition fait référence à d'autre type ; c.-à-d., basé sur les types simples. Ces types sont aussi dits types complexes ou composés. Parmi ces types on cite : le type tableau, chaîne de caractères, enregistrement et ensemble qui seront vus ultérieurement.

4. Opérations de base

La partie traitement d'un algorithme implique un ensemble d'opérations qui peuvent être :

- Opérations de base ou encore élémentaires qui permettent une exécution séquentielle d'un algorithme.
- Structures de contrôle qui permettent le saut dans un algorithme.

Les opérations de base sont l'affectation et les opérations d'entrée/sortie.

4.1. L'affectation

L'opération la plus importante en algorithmique est l'affectation (assignation) qui se note ← (:=en Pascal), et qui consiste à attribuer ou affecter à une variable une valeur appartenant à son domaine de définition (type). La valeur affectée est souvent le résultat de calcul d'une expression arithmétique ou une expression logique.

Exemple :

```
Algorithme calculs ;
…
Début
  X ← 4 ;
  Y ← X * 2 ;
  Z ← Y ;
```

En Pascal :

```
program calculs ;
…
begin
  X := 4 ;
  Y := X * 2 ;
  Z := Y ;
```

```
Z ← Z – 6 ;                    Z := Z – 6 ;
H ← (10>5) ET (2<3) ;          H := (10>5) and (2<3) ;
…                              …
```

L'exemple est lu comme suit : la variable X reçoit la valeur 4. La variable Y reçoit la valeur de X multipliée par 2. La variable Z reçoit la valeur de Y. La variable Z reçoit la valeur courante (actuelle) de Z moins 6. Enfin la variable H reçoit la valeur VRAI.

4.2. Les entrées/sorties

Les échanges d'informations entre l'utilisateur et la machine sont appelés opérations d'entrée-sortie. Les opérations d'entrée-sortie sont :

- Lire () : qui récupère la valeur tapée au clavier et l'affecte à l'espace mémoire désigné par la variable entre parenthèses. En Pascal c'est read();
- Ecrire () : qui récupère la valeur située à l'espace mémoire désigné par la variable entre parenthèses, et affiche cette valeur à l'écran. En Pascal c'est write();

Remarque : Le langage Pascal permet de lire et écrire une variable de type entier, réel, caractère ou chaîne de caractères. Une variable booléenne peut être seulement affichée.

Une chaîne de caractères : est une suite de plusieurs caractères, permettant de représenter des mots ou des phrases. Une chaîne de caractères doit être mise entre deux guillemets simples pour la distinguer d'un identificateur de variable. Par exemple, Ecrire('bonjour'); en Pascal write('bonjour'); permet d'afficher le mot bonjour à l'écran.

5. Construction d'un algorithme simple

La construction d'un algorithme consiste à lui donner un nom, identifier les variables et les constantes et écrire le corps de l'algorithme.

Le corps de l'algorithme est constitué d'une séquence d'opérations mises entre Début et Fin et séparées par des points virgules. Le corps de l'algorithme peut contenir des opérations de lecture, écriture, affection, etc. Pour éclaircir l'algorithme, son corps peut contenir des commentaires mis entre (*…*) ou { }.

Exemple :
L'algorithme de calcul de la surface d'un cercle représenté par énumération des étapes est le suivant :

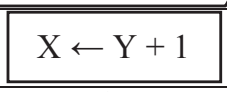
1. Saisir le rayon du cercle (lire le rayon).
2. Calculer la surface par l'expression : Π*(rayon)^2.
3. Afficher le résultat (écrire la surface).

En utilisant un langage algorithmique on obtient :

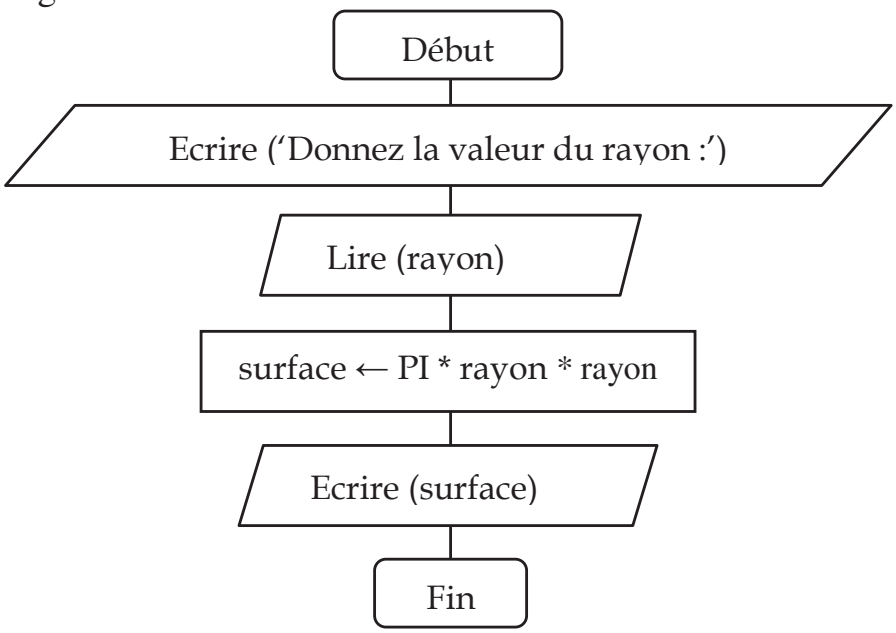
```
Algorithme Calcul_Surface ;
(* algorithme de calcul de la surface d'un cercle *)
Constantes
  PI = 3.14159 ;
Variables
  rayon, surface : réel ;
Début
  Ecrire ('Donnez la valeur du rayon :') ;
  Lire (rayon) ;
  surface ← PI * rayon * rayon;
  Ecrire (surface) ;
Fin.
```

6. Représentation d'un algorithme par un organigramme

Un algorithme peut être représenté aussi par un organigramme facilitant sa compréhension. Un organigramme est une représentation graphique de la solution d'un problème. Pour cela, on utilise les symboles géométriques suivants:

	Début ou fin de l'algorithme.
	Les opérations d'entrée/sortie.
	Un traitement (une affectation par exemple).
	Un test d'une condition pour une décision ou une sélection.
	La liaison entre les différents points et indiquent aussi l'ordonnancement des opérations.

L'algorithme de calcul de la surface peut être représenté par l'organigramme suivant :



7. Traduction en langage Pascal

7.1. Exemple

La traduction de l'algorithme de calcul de la surface en un programme Pascal sera comme suit :

```
program Calcul_Surface ;  (* Calcul de la surface d'un cercle *)
const
  pi = 3.14159 ;
var
  rayon, surface : real ;
begin
  writeln('Donnez la valeur du rayon') ;
  readln(rayon) ;
  surface := pi * sqr(rayon) ; (* sqr c'est le carré *)
  writeln(surface) ;
end.
```

7.2. Règles de base

Dans ce qui suit, nous citons quelques règles qui doivent être respectées lors de l'écriture d'un programme Pascal :

1. Un programme Pascal se compose de trois parties : (1) Un en-tête commençant par le mot program ; (2) une section déclarative introduite par le mot var ; (3) une section instructions ou corps du programme, délimitée par les mots begin et end. Le programme se termine par un point.
2. Un identificateur en Pascal (nom du programme, identificateurs de variables ou de constantes, …) ne doit pas dépasser 127 caractères. Il ne doit pas contenir de caractères accentués, ni d'espace, ni des caractères tels que %, ?, *, ., - , etc. Il doit exclusivement être composé des 26 lettres de l'alphabet, des 10 chiffres et du caractère de soulignement. Un chiffre ne peut pas être placé au début d'un identificateur. On note aussi qu'il ne faut pas dupliquer les identificateurs dans un programme.
3. Les constantes sont définies par le mot clé const.
4. Les points virgules sont obligatoires pour séparer les instructions.
5. Les opérations de lecture et d'écriture se traduisent respectivement par read et write (ou readln et writeln) suivies d'une liste de variables ou d'expressions placées entre parenthèses, et séparées par des virgules. L'ajout de ln après write et read force le passage à la ligne lors de l'affichage suivant à l'écran.
6. L'assignation se représente par :=.
7. Les opérateurs arithmétiques sont : +, -, *, /. Pascal utilise deux opérateurs supplémentaires : DIV fournissant la partie entière du quotient de deux nombres entiers, et MOD fournissant le reste de la division de deux nombres entiers.
8. Dans une affectation, le type de l'expression doit correspondre au type de la variable de destination. Cette règle admet les exceptions suivantes : une variable réelle peut recevoir une valeur entière ; une variable chaîne de caractères peut recevoir la valeur d'une variable caractère.
9. Les mots program, var, begin, end, div, mod, integer, read, write, etc. sont des mots réservés ou mots clés qui ne peuvent être utilisés comme identificateurs par le programmeur.

10. Le langage Pascal ne différencie pas entre majuscule et minuscule.
11. Un programme Pascal contient éventuellement des commentaires mises entre (* *) ou { }.

7.3. Différents formats d'affichage

Pour personnaliser le format d'édition, l'instruction write peut être utilisée de différentes manières :

- write(*valeur_entière*) affiche la valeur entière.
- write(*valeur_entière*:n) affiche la valeur entière sur n positions. Par exemple, write(X:5), avec (X=123), affiche ^^123 (^ symbole d'espacement).
- write(*valeur_réelle*) affiche le nombre en notation scientifique à virgule flottante (^x.xxxxxxxxxE±xx). Par exemple, si la variable réelle X contient la valeur 123.4567 alors write(X) affiche ^1.2345670000E+02. on lit 1.234567 * 10^2.
- write(*valeur_réelle*: n1:n2) affiche le nombre sur n1 positions avec n2 décimales (avec ajustement). Par exemple, si la variable réelle X contient la valeur 123.4567 alors write(X:8:2) affiche ^^123.46.
- write(*chaîne*:n) affiche la chaîne de caractères sur n positions. Par exemple, si la variable X de type chaîne de caractères contient la valeur 'AZERTY' alors write(X) affiche AZERTY, write(X:8) affiche ^^AZERTY, et write(X:3) affiche AZERTY.

7.4. Manipulation des nombres

Pascal définit 5 types d'entier :

Type	Intervalle	Occupation en mémoire
SHORTINT	de -128 à +127	1 octet
BYTE	de 0 à 255	1 octet
INTEGER	de -32768 à +32767	2 octets
WORD	de 0 à 65535	2 octets
LONGINT	de -2147483648 à 2147483647	4 octets

Pascal définit 4 types de réels :

Type	Valeurs autorisées	Occupation en mémoire
SINGLE	$1.5 * 10^{-45}$ .. $3.4 * 10^{38}$	4 octets
REAL	$2.9 * 10^{-39}$ .. $1.7 * 10^{38}$	6 octets
DOUBLE	$5.0 * 10^{-324}$ .. $1.7*10^{308}$	8 octets
EXTENDED	$3.4 * 10^{-4932}$ .. $1.1 * 10^{4932}$	10 octets

Remarque : Un nombre réel appartenant aux valeurs autorisées peut prendre le signe positif (+) ou négatif (-).
Les fonctions mathématiques du langage Pascal sont détaillées dans le tableau suivant :

Notation math.	Fonction Pascal	Type de x	Type du résultat	Signification
|x|	ABS(x)	Entier ou réel	Type de x	Valeur absolue de x
x^2	SQR(x)	Entier ou réel	Type de x	Carré de x
$\sqrt{x}$	SQRT(x)	Entier ou réel	Réel	Racine carrée de x
sin(x)	SIN(x)	Entier ou réel	Réel	sin de x (x en radian)
cos(x)	COS(x)	Entier ou réel	Réel	cos de x (x en radian)
arctg(x)	**ARCTAN(x)**	Entier ou réel	Réel	Angle (en radian) dont la tangente vaut x
e^x	EXP(x)	Réel	Réel	Exponentielle de x
ln(x)	LN(x)	Réel	Réel	Logarithme népérien de x
[x]	TRUNC(x)	Réel	Entier	Partie entière de x
[x]	INT(x)	Réel	Réel	Partie entière de x
arrondi de x	**ROUND(x)**	Réel	Entier	Entier le plus proche de x
décimal de x	FRAC(x)	Réel	Réel	Partie décimale de x

7.5. Manipulation des caractères

Un caractère est déclaré par le mot clé CHAR. Une variable caractère (char) occupe 1 octet en mémoire. Chaque caractère

possède un code ASCII (American Standard Code for Information Interchange). A titre d'exemple, les lettres majuscules de 'A' à 'Z' sont codées dans l'ordre par les codes 65 à 90.
La table ASCII est un tableau de 256 caractères, numérotés de 0 à 255, où les 23 premiers sont des caractères de contrôle, associés à des fonctions de base (Suppr, End, Inser, Enter, Esc, Tab, Shift...), et tous les autres sont directement affichables (lettres, ponctuations, symboles, caractères graphiques, chiffres).
Les fonctions avec lesquelles on peut manipuler des caractères sont détaillées dans le tableau suivant :

Fonction Pascal	Type du résultat	Signification
CHR(x)	caractère	Caractère correspondant au code ASCII spécifié entre parenthèses. On peut mettre #x
ORD(c)	entier	Le code ASCII du caractère spécifié entre parenthèses
SUCC(c)	caractère	Le successeur du caractère spécifié entre parenthèses
PRED(c)	caractère	Le prédécesseur du caractère spécifié entre parenthèses

Remarques :
- Chaque caractère doit être mis entre deux guillemets simples pour le distinguer d'un identificateur.
- Il est possible de comparer les caractères suivant l'ordre du code ASCII. Ainsi, ('A'>'B') retourne False.

7.6. Manipulation des booléens

Les variables booléennes, déclarées en Pascal par le mot clé BOOLEAN, peuvent prendre soit la valeur TRUE (VRAI), soit la valeur FALSE (FAUX). Une variable booléenne (boolean) occupe 1 octet en mémoire.
Sur les booléens, on peut effectuer les opérations suivantes : AND, OR et NOT, ainsi que les opérateurs de comparaison : <, >, <>, >=, <=, =.
Il est à noter qu'en Pascal, les priorités données aux opérateurs, de la plus élevée à la plus basse sont :

NOT
* / DIV MOD AND

+ - OR
= <> < <= > >=
Lorsqu'une expression contient plusieurs opérateurs de même priorité, les opérations sont effectuées de gauche à droite. Pour modifier cet ordre, il suffit d'introduire des parenthèses.

8. Exercices corrigés

8.1. Exercices

Exercice 1 :
Ecrire un algorithme permettant de saisir trois nombres, d'en effectuer la somme, le produit et la moyenne, puis les afficher. Traduire l'algorithme en Pascal.

Exercice 2 :
Ecrire un algorithme permettant de saisir deux nombres, de les permuter puis les afficher. Traduire l'algorithme en Pascal.

Exercice 3 :
Ecrire un algorithme qui calcule le périmètre et la surface d'un rectangle. Traduire l'algorithme en Pascal.

8.2. Corrigés

Solution 1 :

```
Algorithme calculs ;
Variables
  somme, produit, moyenne, nb1, nb2, nb3 : réel ;
Début
  Ecrire('Entrez vos trois nombres') ;
  Lire(nb1, nb2, nb3) ;
  somme ← nb1+ nb2 + nb3 ;
  produit ← nb1 * nb2 * nb3 ;
  moyenne ← somme / 3 ;
  Ecrire('La somme de ces trois nombres est : ', somme) ;
  Ecrire('Le produit de ces trois nombres est : ', produit) ;
  Ecrire('La moyenne de ces trois nombres est : ', moyenne) ;
Fin.
```

On peut ne pas utiliser les variables intermédiaires *somme*, *produit* et *moyenne*, et on met directement :

```
  Ecrire('La somme de ces trois nombres est : ', nb1 + nb2 + nb3) ;
  Ecrire('Le produit de ces trois nombres est : ', nb1 * nb2 * nb3) ;
  Ecrire('La moyenne de ces trois nombres est : ', (nb1 + nb2 + nb3) / 3) ;
```

C'est une question de style : dans le premier cas, on favorise la lisibilité de l'algorithme ; dans le deuxième, on favorise l'économie de l'espace mémoire.

Le programme Pascal :

```
program calculs ;
var
  somme, produit, moyenne, nb1, nb2, nb3 : real ;
begin
  writeln ('Entrez vos trois nombres : ') ;
  readln (nb1, nb2, nb3) ;
  somme := nb1 + nb2 + nb3 ;
  produit := nb1 * nb2 * nb3 ;
  moyenne := somme / 3 ;
  writeln ('La somme de ces trois nombres est : ', somme) ;
  writeln ('Le produit de ces trois nombres est : ', produit) ;
  writeln ('La moyenne de ces trois nombres est : ', moyenne) ;
end.
```

Solution 2 :

Permutation en utilisant une variable intermédiaire :

```
Algorithme Permuter1 ;
Variables  nb1, nb2, nb3 : réel ;
Début
  Ecrire('Entrez deux nombres') ;
  Lire(nb1, nb2) ;
  nb3 ← nb1 ;
  nb1 ← nb2 ;
  nb2 ← nb3 ;
  Ecrire('Voici les deux nombres permutés') ;
  Ecrire('nb1 = ', nb1) ;
  Ecrire('nb2 = ', nb2) ;
Fin.
```

Le programme Pascal :

```
program permuter1 ;
var  nb1, nb2, nb3 : real ;
```

```
begin
  writeln ('Entrez deux nombres : ');
  readln (nb1, nb2) ;
  nb3 := nb1 ;
  nb1 := nb2 ;
  nb2 := nb3 ;
  writeln ('Voici les deux nombres permutés : ') ;
  writeln ('nb1 = ', nb1) ;
  writeln ('nb2 = ', nb2) ;
end.
```

Permutation sans variable intermédiaire :

```
Algorithme Permuter2 ;
Variables
  nb1, nb2 : réel ;
Début
  Ecrire('Entrez deux nombres') ;
  Lire(nb1, nb2) ;
  nb1 ← nb1 + nb2 ;
  nb2 ← nb1 – nb2 ;
  nb1 ← nb1 – nb2 ;
  Ecrire('Voici les deux nombres permutés') ;
  Ecrire('nb1 = ', nb1) ;
  Ecrire('nb2 = ', nb2) ;
Fin.
```

Le programme Pascal :

```
program permuter1 ;
var
  nb1, nb2 : real ;
begin
  writeln ('Entrez deux nombres : ');
  readln (nb1, nb2) ;
  nb1 := nb1 + nb2 ;
  nb2 := nb1 - nb2 ;
  nb1 := nb1 - nb2 ;
  writeln ('Voici les deux nombres permutés : ') ;
  writeln ('nb1 = ', nb1) ;
  writeln ('nb2 = ', nb2) ;
end.
```

Solution 3 :

```
Algorithme rectangle ;
Variables
  longueur, largeur, périmètre, surface : réel ;
Début
  Ecrire('Entrez la longueur et la largeur du rectangle') ;
  Lire (longueur, largeur) ;
  Périmètre ← 2 * (longueur + largeur) ;
  surface ←  longueur * largeur ;
  Ecrire('Le périmètre du rectangle est : ', périmètre) ;
  Ecrire('La surface du rectangle est : ', surface) ;
Fin.
```

Le programme Pascal :

```
program rectangle ;
var
  longueur, largeur, perimetre, surface : real ;
begin
  writeln ('Entrez la longueur et la largeur du rectangle : ') ;
  readln (longueur, largeur) ;
  perimetre := 2 * (longueur + largeur) ;
  surface := longueur * largeur ;
  writeln('Le périmètre du rectangle est : ', perimetre) ;
  writeln('La surface du rectangle est : ', surface) ;
end.
```