الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de RELIZANE
Faculté des Sciences et de la Technologie
Département: Informatique

جامعة غليزان
RELIZANE UNIVERSITY

Polycopié de cours intitulé

Informatique 1
Cours et exercices

Préparé et présenté par :
Dr. BELHADEF Leila
Maître de Conférences classe 'B'

Le cours est destiné aux étudiants de License 1ére année Sciences et Technologie

Année universitaire : 2023/2024

Structure du polycopié

Plan pédagogique du cours

Matière : Informatique 1
Domain : Sciences et Technologie
Filière : 1ère année Licence Sciences et Technologie
Volume Horaire du cours : 1h30 par semaine
Volume Horaire du TP : 1h30 par semaine
Coefficient : 2
Crédits : 4
Evaluation : Contrôle continu : 40%, Examen : 60%.

Introduction

L'informatique est un domaine vaste qui permet de traiter les informations par ordinateurs, ce cours se base principalement sur l'apprentissage de la programmation qui est essentielle dans le parcours des étudiants de toutes les branches en sciences et technologie.

L'objectif du cours est d'avoir une introduction générale sur l'informatique et le fonctionnement de l'ordinateur ensuite de permettre aux étudiants d'apprendre à élaborer des algorithmes et de les programmer avec un langage évolué le Pascal.

Les TP ont pour objectif d'illustrer les notions enseignées durant le cours. Ces derniers doivent permettre à l'étudiant de se familiarisation avec l'utilisation de l'ordinateur et d'acquérir une connaissance de programmation en langage Pascal et de pouvoir résoudre des exercices et de les exécutés sur machine.

Plan

Les principaux points traités dans ce cours sont :

- Introduction à l'informatique
- Notion d'algorithme et de programme

Les principaux points traités dans les TP sont :

- Initiation et familiarisation avec l'ordinateur d'un point de vue matériel et logiciel
- TP d'initiation à l'utilisation de l'environnement de programmation le Pascal
- Résolution d'exercices et exécution des programmes correspondants sur ordinateur par les techniques de programmation vues en cours

Chapitre 1: Introduction à l'informatique

1 Définition de l'informatique

L'informatique (computer science) est une science qui permet de traiter automatiquement l'information (textes, nombre, images, vidéos…) par un ordinateur. Elle est utilisée dans plusieurs domaines tels que:

- Le calcul scientifique
- Les réseaux
- L'intelligence artificielle
- La robotique…

2 Evolution de l'informatique et des ordinateurs

Génération 1 (~1945 - 1955)

a. Machines électroniques composées de circuits à lampes à vide (place importante)
b. performances de d'environ 1000 opérations/s
c. programmation en langage binaire
d. programme et données fournis sous forme de cartes perforées, résultats sur une imprimante (pas de stockage)

Parmi les ordinateurs de cette génération on peut citer ENIAC (**E**lectronic **N**umerical **I**ntegrator and **C**omputer). Vitesse : environ 330 multiplications par seconde. (1946)

Génération 2 (1955 - 1965)

e. Création des transistors qui remplacent les circuits à lampes à vide
f. utilisation des 1ères mémoires
g. performances d'environ 100 000 opérations/s
h. utilisation des **premiers langages évolués** (Fortran en 1957 par John Backus)

Génération 3 (1965 - 1971)

i. Invention du circuit intégré permettant de placer des dizaines de transistors sur une puce de silicium
j. performances de 10^9 à 10^{12} opérations/s
k. utilisation des Systèmes d'Exploitation (1969 UNIX : *UNiplexed Information and Computing Service)*

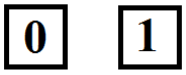
Génération 4 (1971 à nos jours) :

l. Plusieurs dizaines de milliers (millions) de circuits peuvent être intégrés sur une même puce : le micro processeur. En 1971 l'Intel 4004 fut le premier microprocesseur (Intel INTegrated Electronics).
m. développement de l'**ordinateur personnel**. (1981 IBM Personal Computer) International Business Machines

Exemple d'ordinateur:
2011 : Processeur Intel Core i7 2600K
Plus de 1,4 milliards de transistors
128300 MIPS (Million d'Instructions par Seconde)

3 Les systèmes de codage des informations

Les informations traitées par un ordinateur (texte, nombres, etc.) sont représentées et manipulées par l'ordinateur sous forme binaire comme une suite de 0 et de 1. L'unité d'information est le chiffre binaire (0 ou 1) ou **bit** (pour **bi**nary digi**t**). Le bit représente une case mémoire avec une valeur de 0 ou 1.



Bit

3.1 Représentation des nombres

Le Système de numération utilisé en informatique est le système binaire mais pour faciliter la conversion d'autres systèmes peuvent être utilisés comme le système octal et hexadécimal.

Nom de la base	*b*	Chiffres
Binaire	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Décimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadécimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

3.1.1 Les conversions entre les bases

a) Passage d'une base b->10 : $(\alpha)_b = (?)_{10}$

Soit un nombre α écrit en base *b*

$(\alpha)_b = (a_n a_{n-1} \ldots a_1 a_0)_b = (a_n \times b^n + a_{n-1} \times b^{n-1} + \ldots + a_1 \times b^1 + a_0 \times b^0)_{10}$

Exemples:

$(\alpha)_b = (1011)_2 = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$

$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

$= 8 + 0 + 2 + 1 = (11)_{10}$

$(\alpha)_b = (2D1)_{16} = (2 \times 16^2 + 13 \times 16^1 + 1 \times 16^0)_{10}$

$= 2 \times 256 + 13 \times 16 + 1 \times 1$

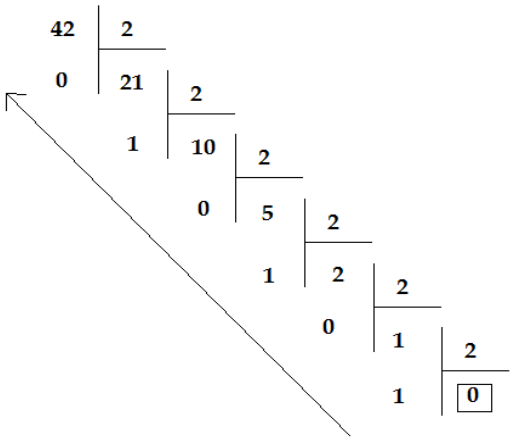
$= 512 + 208 + 1 = (721)_{10}$

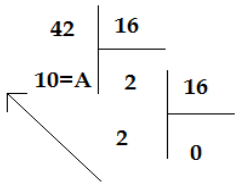
b) Passage d'une base 10->*b* : $(\alpha)_{10} = (?)_b$

Divisions successives de α par *b*.

Les restes obtenus représentent α en base *b* (du chiffre de poids faible au chiffre de poids fort)

Exemple : $(42)_{10}=(101010)_2$ $(42)_{10}=(2A)_{16}$





c) Cas particulier des bases (2,8 et 16)

1-Conversion de 2-> 8 ou 16 : La conversion de chaque 3 chiffres binaire (4 chiffres binaire) Partant du bit du poids le plus faible représentent un chiffre octal (hexadécimal) en binaire ($8=2^3$ et $16=2^4$).

Exemple :

Hexadécimal	**7**	**A**	**B**
Binaire	**0111**	**1010**	**1011**
Octal	**3**	**6**	**5**	**3**

Hexadécimal : $B=1\times2^3+0\times2^2+1\times2^1+1\times2^0$, $A=1\times2^3+0\times2^2+1\times2^1+0\times2^0$

$7=1\times2^2+1\times2^1+1\times2^0$

Octal : $3=1\times2^1+1\times2^0$, $5= 1\times2^2+0\times2^1+1\times2^0$

$6= 1\times2^2+1\times2^1+0\times2^0$

2- Conversion de 8 ou 16-> 2 : La conversion chaque chiffre octal (hexadécimal) représente trois bits (4 bits) en binaire. (la division de chaque chiffre en octal (en hexadécimal) par 2).

Hexadécimal	**7**	**A**	**B**
Binaire	**0111**	**1010**	**1011**
Octal	**3**	**6**	**5**	**3**

3.2 Représentation des caractères:

Le codage des caractères est fait par une table indiquant chaque caractère en binaire. Le code le plus connu est le code ASCII (American Standard Code for Information Interchange). Il représente chaque caractère sur 7 bits.

$A=(65)_{10}= (1000001)_2$

4 Principe de fonctionnement d'un ordinateur

Les deux principaux constituants d'un ordinateur sont :
La *mémoire centrale* (MC) et *le processeur* (UC pour unité centrale).

La MC permet de stocker de l'information (programmes et données), tandis que le processeur exécute pas à pas les instructions composant les programmes.

Le processeur et la MC communiquent via des bus (bus d'adresses et bus de données)

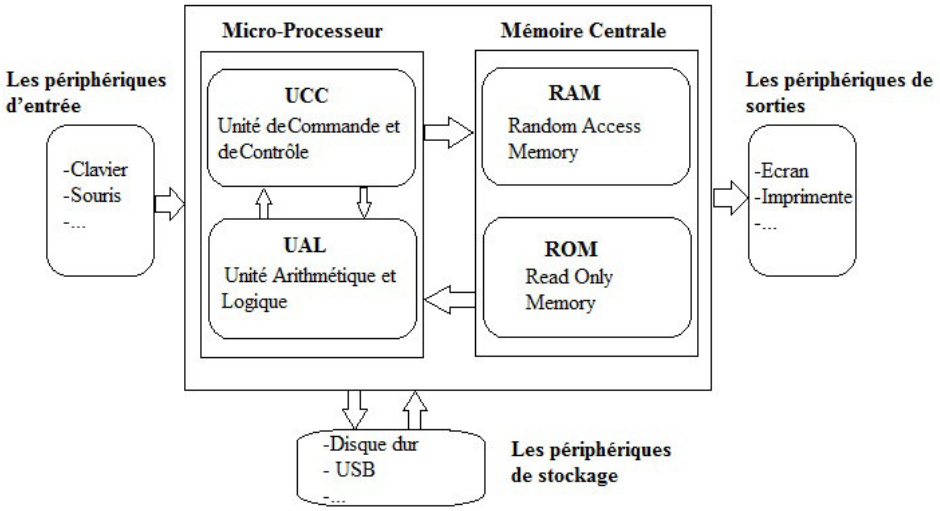


Figure 1.1 Structure d'un ordinateur

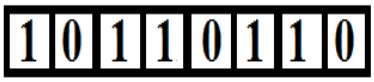
4.1 La mémoire centrale (MC)

Permet de stocker les informations (programmes et données). Elle est représentée par deux types de mémoire :

4.1.1 La mémoire vive RAM (*Random Access Memory*) permet d'enregistrer les informations en cours d'exécution de manière temporaire dés que vous éteignez l'ordinateur, son contenu est effacé.

4.1.2 La mémoire morte ROM (*Read Only Memory)* contient du code et des paramètres système qui permettent à l'ordinateur de s'allumer correctement (BIOS : Basic Input/Output System Programme de base des entrées/sorties). Elle ne s'efface pas lorsque vous éteignez l'ordinateur.

4.1.3 Taille des mémoires La taille des mémoires est mesurée par les octets où un octet est représenté par 8 bits.



1 Octet ou 1 Byte (en anglais)

La taille des mémoires est mesurée par les octets avec :

1 octet = 8bits

1 Ko (Kilo-octet) = 2^{10} = 1024 octet

1 Mo (Méga-octet) = 2^{20} = 1024*1024 octet

1 Go (Giga-octet) = 2^{30} = 1024*1024 *1024 octet

1 To (Téra-octet) = 2^{40} = 1024*1024*1024*1024 octet

4.2 Le processeur ou (micro-processeur)

Exécute les instructions composants les programmes. Il se compose d'une unité de commande et de contrôle UCC, d'une unité arithmétique et logique UAL, d'une horloge, de registres et d'un bus interne qui relie ces unités.

4.2.1 Unité de commande et de contrôle (UCC) Recherche les instructions à partir de la MC, les décode et en supervise leur exécution par l'UAL.

4.2.2 L'unité arithmétique et logique (UAL) contient tous les circuits électroniques qui réalisent les opérations de calcul arithmétiques (+, -, *, /) et les opérations logiques (ET, OU, NON…).

4.2.3 Le registre est une mémoire qui stocke les données, les instructions et les résultats.

4.2.4 L'horloge A chaque top d'horloge le processeur effectue une instruction, Ex: un ordinateur avec une fréquence 1G Hertz (1000MHz=10^6khz=10^9hertz) effectue 1000 millions d'instructions par seconde.

4.2.5 Les bus Le processeur et la MC communiquent via des bus (bus d'adresses et bus de données).

4.3 Etapes d'exécution d'une instruction

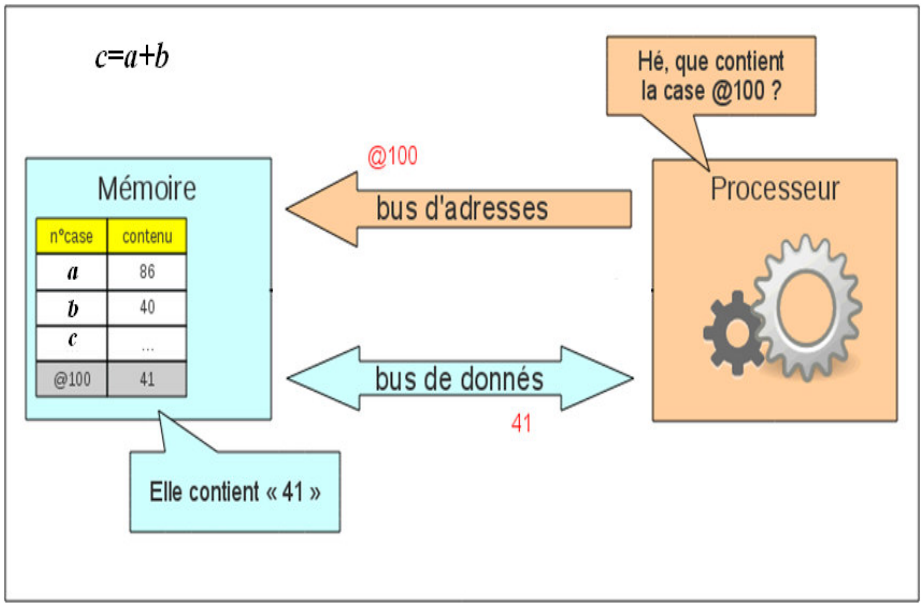


Figure 1.2 : Etapes d'exécution d'une instruction

UCC: 1. Charge une instruction à partir de la MC et la décode
UAL: 2. Exécute l'instruction
UCC: 3. Sauvegarde les résultats à leurs adresses
UCC: 4. Passe à l'instruction suivante

5 Partie matérielle d'un ordinateur (Hardware)

Tous les ordinateurs comportent les mêmes éléments physiques de base :

- Le boîtier (Unité centrale): contient la carte mère, le processeur, la MC
- les périphériques d'entré/sortie et de stockage.

5.1 La carte mère c'est une carte électronique sur laquelle on trouve :

- La MC et le processeur.
- Les Slots : emplacement pour connecter des cartes d'extension, la RAM
- Connecteurs réseau, haut-parleurs, VGA , **Ports :** USB, …ect.
- Socket : emplacement pour connecter le processeur.
- Ensemble de bus : un bus est un ensemble de fils de cuivre utilisé pour le transport des informations entre les composants connectés à la carte mère.

5.2 Les périphériques

C'est tout accessoire qu'on peut connecter à un ordinateur.

5.2.1 Les périphériques d'entrée : Transmettent les informations de l'extérieur vers la mémoire (MC).
Exemple : le clavier, la souris, ….

5.2.2 Les périphériques de sorties Transmettent les informations de la mémoire (MC) vers l'extérieur.
Exemple : l'écran, l'imprimante, …

5.2.3 Les périphériques de stockage Sauvegardent les informations de manière permanente.
Exemple: le disque dur, clé USB, DVD, CD

6 Partie logiciel d'un ordinateur (Software)

Un logiciel est un programme ou un ensemble de programmes destiné au fonctionnement de l'ordinateur. Un programme (software) est un ensemble d'instructions que la partie matérielle (hardware) réalise.

Il existe trois catégories dans cette partie:

- Les systèmes d'exploitation
- Les logiciels d'application
- Les langages de programmation

6.1 Le système d'exploitation (SE) (Operating System: OS)

Est le programme qui gère la communication entre le processeur, les périphériques et l'utilisateur en offrant une interface graphique.
Les SE les plus connus sont :

Mac OS (Macintosh Operating System) (Apple)

Windows (Fenêtres) (Microsoft)

Linux système d'exploitation libre

6.2 Les logiciels d'application : programmes qui permettent de réaliser une fonction précise.
Exemples :
MS Word……..traitement de texte
Photoshop……...logiciel de retouche d'images

6.3 Les langages de programmations

Un langage de programmation est un code de communication permettant aux développeurs de créer des programmes exécutables par ordinateur.

Exemple : Pascal, Fortran, C, C++, Java, Matlab, python….

7. Exercices

Exercice 1

1- Les systèmes de codage : Compléter le tableau de conversion suivant :

Nombre en décimal	Nombre en binaire	Nombre en octal	Nombre en hexadécimal
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

2- Réaliser les conversions suivantes :

$(71)_{10}=(\ )_2$ $(284)_{10}=(\ )_8$ $(469)_{10}=(\ )_{16}$

$(1101001)_2=(\ )_{10}$ $(720)_8=(\ )_{10}$ $(3E8)_{16}=()_{10}$

$(1011000101)_2=(\ )_8=()_{16}$ $(A18BF)_{16}=(\ )_8=(\ )_2$

3- Quel est le plus grand nombre qu'on peut écrire avec 8 bits ?

Exercice 2

Un utilisateur veut enregistrer trois fichiers (F1, F2, F3) qu'il a crée sur un disque amovible (USB). Les tailles de ces fichiers sont : F1=90000 Ko, F2=2 Go, F3=100000 Octet. Et la capacité du disque est 2250 Mo. Est-ce qu'il peut enregistrer ces trois fichiers sur le même disque ? Justifier votre réponse.

Exercice 3 Les composants de l'ordinateur

1- Quels sont les principaux composants (périphériques) externes d'un ordinateur?

2- Citer sept composants importants à l'intérieur de l'unité centrale.

3- Allumer l'ordinateur

4- Explorer le bureau de Windows: menu démarrer, les icones, barre des taches.... ;

5- Obtenir les informations sur le processeur et la mémoire RAM : Démarrer> Panneau de configuration>Système et sécurité>Système.

- Consultez les informations disponibles à propos de l'ordinateur.
- Donner le nom du processeur et sa génération. Quelle est sa vitesse en GHz et MHz?
- Quelle est la capacité de la mémoire RAM en Go et en Mo?

6- Lancer l'application calculatrice : Démarrer>Tous les programmes>Accessoires> Calculatrice, tester une conversion réalisée dans l'exercice 1 (affichage programmeur).

Exercice 4 Saisir et enregistrer un texte : Démarrer>Tous les programmes>Accessoires>Bloc notes

- Tapez le texte suivant :

« Ceci est mon premier TP en Informatique »

- Enregistrer le fichier sous le nom INFO1 dans le dossier **documents**.
- Créer un nouveau dossier **groupeA1** dans **documents** puis déplacer le fichier INFO1 dans le dossier **groupeA1.**
- Renommer le fichier INFO1 par TP1 ensuite afficher **ces propriétés**.

Les fichiers sont déterminés par leur nom et extension. Donner la signification des extensions suivantes :

- .TXT
- .EXE
- .docx

Exercice 5 Créer et supprimer un dossier

-Ouvrir le dossier **documents** et créer un nouveau dossier **Informatique.**
-Créer deux autres dossiers appelés **TP 1** et **TP 2** dans le dossier **Informatique**.
-Supprimer le dossier **TP 2**.

Exercice 6 Utiliser invite de commandes

L'invite de commandes permet d'utiliser certaines commandes Dos comme IPCONFIG:
-Cliquer sur **Démarrer**>**Tous les programmes>Accessoires>Invite de commandes**
Ou bien Cliquer sur **Démarrer>Exécuter** taper ensuite le mot **cmd**
Ensuite tester les commandes suivantes :
CD " Change Directory " Changer de répertoire (dossier)
DIR " Directory " Affiche la liste des fichiers et sous répertoires
MD ou MKDIR " Make directory " Créer un répertoire
HELP Affiche à l'écran toutes les commandes
CLS " clear screen" effacer l'écran

Chapitre 2: Notions d'algorithme et de programme

1 Concept d'un algorithme

L'algorithme est un terme d'origine arabe, par référence au nom du mathématicien **Alkhawarizmi** (qui décrit les méthodes de calcul algébrique).
Un algorithme est une suite d'instructions, qui décrit comment résoudre un problème particulier. **Exemple** : Recette de cuisine, indication d'un chemin.

Un algorithme est représenté par un nombre fini d'opérations selon une suite logique et en traitant tout les cas possibles.
Un problème donné peut être résolu à l'aide de différents algorithmes.
La structure d'un algorithme est composée principalement d'une partie de déclaration suivie des instructions à réalisées (corps de l'algorithme).

2 Représentation en organigramme

L'organigramme est utilisé pour illustrer les étapes d'un algorithme de manière claire et simple grâce à sa représentation graphique. L'organigramme se compose d'un ensemble de symboles les plus importants sont:

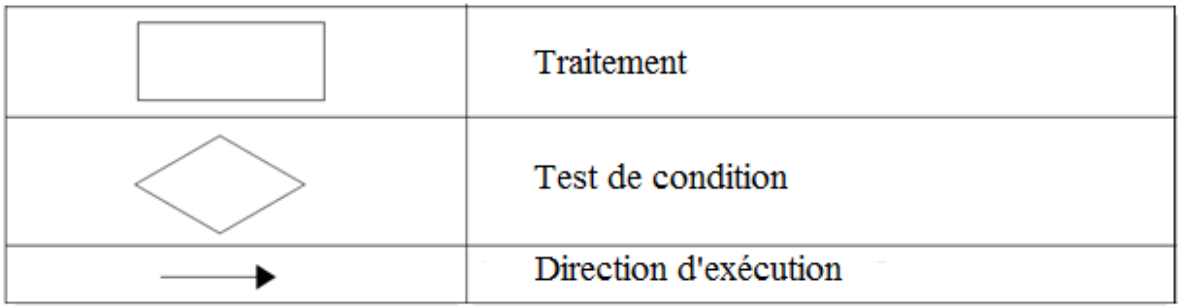


Figure 2.1 : Symboles d'organigramme

3 Structure d'un programme en Pascal

Un programme informatique est la traduction d'un algorithme dans un langage de programmation pour qu'il soit traduit en langage machine afin d'être exécuté par l'ordinateur.

Le langage Pascal a été créé par Wirth au début des années 70, c'est un langage structuré qui utilise des instructions claires pour permettre de traduire facilement un algorithme en programme exécutable par l'ordinateur.
Pour programmer en Pascal il faut installer un environnement de développement intégré (IDE : Integrated development environment) du langage Pascal. IDE comporte plusieurs outils pour créer une application comme l'éditeur et le compilateur. Ainsi après avoir traduit un algorithme en un programme Pascal :

- La première étape consiste à saisir le texte du programme dans l'éditeur
- La seconde étape est la compilation qui consiste à traduire en langage machine le programme saisit dans l'éditeur.
- La dernière étape est l'exécution du programme.

	Structure d'un algorithme	**Structure d'un programme en Pascal**
Entête	**Algorithme** NomProgramme;	**PROGRAM** NomProgramme;
Déclarations	**CONST** *{Déclarations de constantes}* **VAR** *{Déclarations de variables}*	**Uses** wincrt ; **CONST** *{Déclarations de constantes}* **VAR** *{Déclarations de variables}*
Corps	**Début** *{les instructions}* **Fin.**	**BEGIN** *{les instructions}* **END.**

4 La démarche d'analyse d'un problème

Figure 2.2 : Les étapes de programmation

Avant d'écrire l'algorithme d'un problème donné, il faut analyser le problème en déterminant :

1- Les données en entrée
2- Les données de sortie (résultats)
3- Les étapes du traitement

5 Structure des données

Dans un programme on utilise deux types d'objets simples: les constantes et les variables.

5.1 Caractéristiques des objets

5.1.1 L'identificateur le nom de la variable ou la constante, il est représenté par une suite de caractères Alphanumériques [0..9]+[a, z]+[A,Z] et _ du 8 . Il commence par une lettre et peut contenir un chiffre.

L'identificateur ne droit pas être un mot clés du langage : **program, begin, end, var, const,…**

Exemple : *poids, t1, t2*

5.1.2 La valeur C'est le contenu de l'objet

5.1.3 Le type : On distingue 5 types standards

Tableau 2.1 : les types standards

Algorithme	Pascal	Description	Place mémoire	Exemple de déclaration
entier	integer	comprend les valeurs sans virgule	16bits(2octets)	num1, num2 : integer ;
réel	real	comprend les valeurs sans ou avec virgule	32bits(4octet)	poids : real ;
booléen	boolean	peut prendre deux valeurs Vrai ou Faux (True, False)	8bits(1octet)	B1, B2 : boolean ;
caractère	char	comprend les lettres+chiffres+caractères spéciaux. Il contient un seul caractère. lettres={'a','b',..,'z'} et {'A','B',..,'Z'} chiffre={'0','1',...,'9'} caractère spéciaux ={'(',')', '!',…,' '}	8bits(1octet)	Section :char ;
chaîne de caractères	string	comprend une suite de N caractères. $0 \leq N \leq 255$. L'espace est un caractère		ville : string ;

5.2Les objets

5.2.1 Les constantes

Leurs valeurs ne change pas durant l'exécution d'un programme

Déclaration

Const

Nom1=valeur1 ;

Exemple (Algorithme et Pascal) :

Const

epsilon=0.5 ;

5.2.2 Les variables

Leurs valeurs peuvent changer durant l'exécution d'un programme

Déclaration

Var

Nom1, Nom2 : type de données1 ;

Nom3 : type de données2 ;

Algorithme:

Var

x, y : réel;

a : entier;

Pascal :

Var

x, y : real ;

a:integer;

6 Les expressions et les opérateurs

Une expression est un ensemble d'opérandes (constantes ou variables) et d'opérateurs.

6.1 Les opérateurs

Les types d'opérateurs qu'on peut trouver dans une expression sont:

- Arithmétiques (+ ,- ,/ , *, div, mod)
- Logiques : algorithme (et, ou, non),
  Pascal (and, or, not)
- Relationnels : algorithme (≠ ≤ < ≥ > =) ,
  Pascal (= (égale), <> (différent), < (inférieur), > (supérieur),
  <= (inférieur ou égale), >= (supérieur ou égale)

Les opérateurs **mod** et **div** s'appliquent sur des données de type entier.

mod: le reste de division entière, **div** : le résultat de division entière

Exemple :

7 **div** 2=3

7 **mod** 2=1

6.2 Les fonctions standards

Les expressions peuvent contenir des fonctions standards.

Tableau 2.2 : Les fonctions standards

Nom	Rôle	Exemple	Affichage
Sqr (x)	Renvoi le carré d'un nombre	write(sqr(2));	4
Sqrt (x)	Renvoi la racine carrée d'un nombre	write(sqrt(4));	2
Pi	Renvoi la constante Pi	write(Pi);	3.14159265358
Sin(x)	Calcul du sinus (Radians)	write(sin(0), sin(Pi));	0 1
Cos(x)	Calcul du cosinus (Radians)	write(cos(0), cos(Pi));	1 0
ArcTan(x)	Renvoi l'arc tangente	write(ArcTan(4));	0.7853982
Exp(x)	Exponentielle	write(Exp(0));	1
Ln(x)	Logarithme népérien	write(Ln(1));	0
Trunc(x)	Donne la partie entière	write(Trunc(Pi));	3
Int(x)	Renvoi la partie entière	write(Int(5.67));	5.00
Frac(x)	Renvoi la partie fractionnaire (après la virgule)	write(Frac(45.98765));	0.98765
Round(x)	Arrondi à l'entier le plus proche	write(Round(28.7),Round(28.3));	29 28
Pred(x)	le *prédé*cesseur d'une variable de type ordonné (integer, char)	write(Pred(5));	4
Succ(x)	le *succé*sseur d'une variable de type ordonné (integer, char)	write(Succ(5));	6
Odd(x)	Renvoi **true** si impair, **false** si pair	write(Odd(3), Odd(2));	TRUE FALSE
Abs(x)	Renvoi la valeur absolue	write(Abs(-54));	54
Random(x)	Renvoi un nombre aléatoire entre 0 et X	write(Random(10));	6.71593

6.3 L'écriture algorithmique des expressions

Dans un programme les expressions sont représentées par une écriture algorithmique.

Exemple :

$$x^3 - \sqrt{\frac{1}{|x+1|}} \quad : \quad (x * x * x) - sqrt(1/abs(x+1))$$

6.4 Priorités des opérateurs

La priorité des opérateurs utilise les règles suivantes :

1) Les parenthèses et on commence par les plus internes
2) Le **non** logique, et le – unitaire
3) *, /, **mod**, **div**, **et** logique
4) + , - , **ou** logique
5) En cas d'égalité des priorités, on commence par l'opérateur le plus à gauche, exemple :
   3/3*4*5= 20

7 Les instructions de base

7.1 L'affectation

Consiste à affecter une valeur à une variable.

Syntaxe :

Algorithme	**Pascal**
variable←valeur ;	variable :=valeur ;

Exemple :

Algorithme test1;
const a=0.5;
Var X,Y : réel ;
Début
X←4 ;
Y←3*X+a ;
end.

Mémoire

Case	**Contenu**
a	0.5
X	4
Y	12.5

Le type de la valeur doit être du même type de la variable, cependant on peut affecter une valeur de type entier à une variable de type réel.

7.2 La lecture

Cette fonction permet de lire une valeur à partir du clavier, et stocker cette valeur dans une variable.

Syntaxe

Algorithme	**Pascal**
Lire(variable);	read(variable) ;

Exemple:

```
Algorithme test1;
const  a=0.5;
Var X,Y : réel ;
Début
Lire(X) ;
Y←3*X+a ;
end.
```

Mémoire

Case	Contenu
a	0.5
X	1
Y	3.5

7.3 L'écriture

Cette fonction permet d'afficher les résultats sur l'écran.

Syntaxe :

Algorithme	**Pascal**
écrire (objet) ;	write(objet) ;

Exemple:

```
Algorithme test1;
const  a=0.5;
Var X,Y : réel ;
Début
Lire(X) ;
Y←3*X+a ;
écrire(Y) ;
end.
```

Ecran

```
1
3.5
```

Pour afficher un message il faut le mettre dans des guillemets.

Exemple:

Algorithme : écrire ('Sciences et Technologies ST') ;

pascal : write ('Sciences et Technologies ST') ;

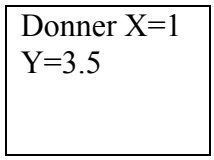
Si on veut afficher plusieurs objets il faut les séparer par des virgules.

Exemple:

Algorithme test1;

const a=0.5;
Var X,Y : réel ;
Début
Écrire ('Donner X=');
Lire(X) ;
Y←3*X+a ;
écrire('Y=',Y) ;
end.

Ecran



8. Exemples d'algorithmes et de programmes

Exemple1 :
Écrire un programme qui affiche le message suivant : Sciences et Technologies ST

Algorithme affichage ;
début
écrire ('Sciences et Technologies ST') ;
fin.

program affichage ;
uses wincrt ;
begin
write (' Sciences et Technologies ST');
end.

Exemple2 :
Écrire un programme qui permet de calculer la surface d'un carré.

Algorithme carré ;

var L, S : **réel ;**
début
écrire('Donner la longueur du carré ') ;
lire(L) ;
S← L*L ;
écrire('La surface=', S) ;
fin.

program carre ;
uses wincrt ;
var L, S:**real;**
begin
write ('Donner la longueur du carré ') ;
read(L) ;
S :=L*L ;
write ('La surface=', S) ;
end.

Exemple 3 :

Ecrire un programme qui permet de saisir trois notes d'un étudiant, et de calculer sa moyenne.

Données en entrée: N1, N2, N3

Données en sortie : Moy

Traitement : Moy=(N1+N2+N3)/3

```
Algorithme moyenne;
var
N1,N2,N3,Moy:réel;
début
écrire('Donner trois notes ');
lire(N1,N2,N3);
Moy ←(N1+N2+N3)/3;
écrire('La moyenne est ',Moy);
fin.
```

```
Program moyenne;
uses wincrt;
var N1,N2,N3,Moy:real;
begin
write('Donner trois notes ');
read(N1,N2,N3);
Moy:=(N1+N2+N3)/3;
writeln('La moyenne est ',Moy);
end.
```

Exemple d'exécution du programme :

Mémoire

N1	12
N2	10
N3	8
Moy	10

Ecran

```
Donner trois notes 12
10
8
La moyenne est 10
```

Remarques :

- L'affichage des données s'effectue par **write** et **writeln**. Avec l'instruction **writeln** un passage à la ligne sera effectué lors du prochain affichage.
- Les instructions **read** et **readln** permettent de lire des données pour les validées. Avec l'instruction **readln** un passage à la ligne sera effectué lors de la prochaine lecture.
- Les valeurs numériques réelles sont affichées par défaut sous forme scientifique. L'écriture 3.74E-3 représente le nombre $3,74 * 10^{-3}$.
- Pour insérer des commentaires : **{**commentaire**}** ou **(***commentaire***)**
- Pour tester les programme télécharger Free Pascal ou MyPascal et installer le sur votre PC.

9 Exercices

Exercice1

Classer les objets suivants en constantes ou variables en indiquant le type des variables :
Nom d'étudiant, numéro de groupe, section, Poids, Pi, Potentiel hydrogène (ph), gravité de la terre, l'adresse d'un employé, coefficient d'un module.

Exercice 2

1- Donner la structure générale d'un algorithme et d'un programme.

2- Donner l'écriture algorithmique de l'expression suivante :

$$\frac{1}{\sqrt{x^2+x+1}} \qquad \frac{x^3}{x^2-1} \qquad \frac{1}{1-|x|}$$

Exercie3

Donner la valeur des expressions arithmétique suivante :

```
R1 : =(16 mod 4)*5+ 2/(sqrt(4)*sqr(2)−(10 div 3));

R2 : =2/int(2.5)*frac(1.3)+ (round(5.5)*trunc(0.3));
```

Exercice 4

1- Donnez la trace d'exécution des programmes suivant :

Programme TEST1:

```
Program TEST1;
Uses wincrt;
const   D= 5 ;
var A, B, C : integer ;
begin
A :=5 ;
B :=3 ;
C:=A + B ;
write(C);
A:= 2 ;
C:= B – A;
write(C);
end.
```

Programme TEST2 :

```
Program TEST2;
Uses wincrt;
Var A, B, C, D : String ;
begin
A := '51' ;
B := '17' ;
D:= 'a';
C :=B+ A+D ;
write (C);
end.
```

2-

- Exécuter le programme TEST1.
- Remplacer les deux instructions **write** par **writeln** et refaire l'exécution

Exercice 5

Écrire un programme qui affiche 'Algorithmique et programmation' à l'écran.

Série d'exercices avec solution

Exercice 6

Donnez la trace d'exécution du programme suivant :

```
Program Tp2 ;
Uses wincrt;
var    A,B,C : Boolean ;
Begin
A:=true ;
writeln(A) ;
B:=A ;
writeln(B) ;
A:=false ;
B:=not A ;
writeln(B) ;
C:=A and B ;
writeln(B) ;
C:=not (A or B) ;
writeln(C) ;
end.
```

Rappel des tables de vérités :

a	**b**	**a** and **b**
false	false	**false**
false	true	**false**
true	false	**false**
true	true	**true**

a	**b**	**a** or **b**
false	false	**false**
false	true	**true**
true	false	**true**
true	true	**true**

Solution :

Instruction	**A**	**B**	**C**	**écran**
A:=true ;	true	/	/	true
B:=A ;	true	true	/	true
A:=false ;	false	true	/	/
B:=**not** A ;	false	true	/	true
C:=A **and** B ;	false	true	false	true
C:=**not** (A **or** B) ;	false	true	false	false

Exercice 7

Ecrire un programme qui affiche le Produit, la somme, la différence et la moyenne de trois nombres réels.

Pour améliorer l'affichage des nombre réels utiliser les instructions :

1- writeln (element : largeur : chiffres après la virgule) ou writeln(element : chiffres après la virgule)

2- Pour afficher un résultat par exemple le produit représenté par la variable P deux possibilités, **soit**:

```
Writeln('le produit =') ;
Writeln(P) ;
```

Ou

```
Writeln('le produit = ' , P ) ;
```

Solution:

Ecrire l'algorithme et le traduire en Pascal.

L'affichage du résultat doit être fait avec 2 chiffres après virgule en ajoutant dans le programme à l'instruction **write :4 :2**.

```
Algorithme nbres ;
Var X,Y, Z, Prod , Diff , Som, Moy :
Réel ;
Début
Ecrire ('Donner trois nombres : ') ;
Lire(X,Y,Z) ;
Som ←X+Y+Z ;
Prod ←X*Y*Z ;
Diff ←X-Y-Z ;
Moy ← (X+Y+Z)/3 ;
Ecrire ('le produit =', Prod) ;
Ecrire ('la somme =', Som) ;
Ecrire ('la différence =', Diff) ;
Ecrire  ('la Moyenne=', Moy) ;
Fin.
```

```
program nbres;
uses wincrt;
var X,Y,Z,Prod,Diff,Som, Moy:real;
begin
write('Donner trois nombres : ');
read(X,Y,Z);
Som :=X+Y+Z ;
Prod:=X*Y*Z ;
Diff :=X-Y-Z ;
Moy:=(X+Y+Z) /3 ;
writeln ('le produit =', Prod :4:2) ;
writeln ('la somme =', Som :4:2) ;
writeln ('la différence =', Diff :4:2) ;
writeln ('la Moyenne =', Moy :4:2) ;
end.
```

Exercice 8

Ecrire un programme qui permet de calculer le reste et le quotient de la division entière entre deux nombres entiers.

Solution:

```
Algorithme division ;
Var
A,B,R,Q : entier ;
Debut
Ecrire ('Donner deux entiers: ') ;
Lire(A,B) ;
R←A mod B ;
Q←A div B ;
Ecrire ('le reste =', R) ;
Ecrire ('le quotient =', Q) ;
Fin.
```

```
program division ;
uses wincrt;
var A,B,R,Q : integer;
begin
write('Donner deux entiers: ');
read(A,B);
R :=A mod B ;
Q :=A div B ;
writeln ('le reste =', R) ;
writeln ('le quotient =', Q) ;
end.
```

Exercice 9

Ecrire un programme qui permet de faire une permutation entre deux nombres réels.
Exemple : X=9 et Y=41 **permutation :** X=41 et Y=9

Solution:

```
Algorithme permutation;
Var
X,Y,Z :réel ;
Début
Ecrire ('Donner la valeur de X : ') ;
Lire (X) ;
Ecrire ('Donner la valeur de Y : ') ;
Lire (Y) ;
Z ←X ;
X←Y ;
Y←Z ;
Ecrire ('X=', X) ;
Ecrire ('Y =', Y) ;
Fin.
```

```
program permutation;
uses wincrt;
var X,Y,Z: real;
begin
write ('Donner la valeur de X : ') ;
read(X) ;
write ('Donner la valeur de Y : ') ;
read (Y) ;
Z :=X;
X :=Y;
Y:= Z ;
writeln ('X=', X) ;
writeln ('Y =', Y) ;
end.
```

Chapitre 3 : Les Structures de contrôle

1 Introduction

Les instructions utilisées dans un algorithme sont organisées en fonction de leur enchainement suivant trois groupes :

- Structure séquentielle
- Structure conditionnelle
- Structures répétitives

2 Les Structures de contrôle conditionnelles

Une instruction conditionnelle permet à un programme de modifier son traitement en fonction d'une condition.

2.1 si… alors (if …then)

a) Structure:

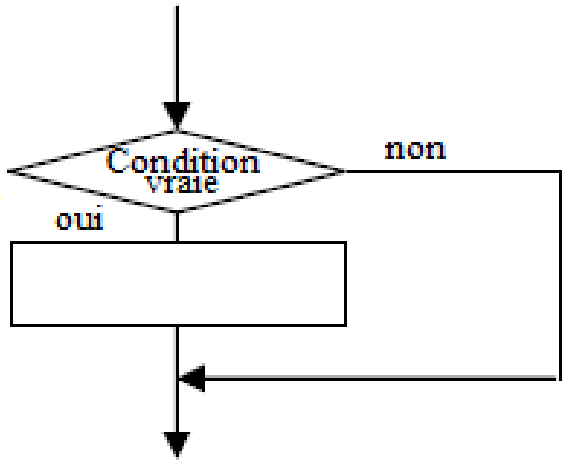
Algorithme

si (Condition) **alors**
  Instruction 1 ;
  Instruction 2 ;
  ……
  Instruction N ;
finsi ;

Pascal

if(Condition) **then**
begin
  Instruction 1 ;
  Instruction 2 ;
  ……
  Instruction N ;
end ;

b)Organigramme



Exemple 1
Ecrire un algorithme qui permet d'afficher la valeur absolue d'un nombre saisi au clavier.

```
Algorithme Valeur_Absolue;
Var x : réel ;
Début
Ecrire ('Entrer un nombre');
Lire(x) ;
Si (x<0) alors
   x←-x ;
finsi ;
ecrire('La valeur absolue est ',x);
fin.
```

```
Program Valeur_Absolue;
Uses wincrt ;
Var x: real;
begin
write('Entrer un nombre');
read(x) ;
if (x<0) then
begin
x :=-x ;
end;
write('La valeur absolue est ',x);
end.
```

2.2 La structure si… alors …sinon (if …then…else)

a) Structure :

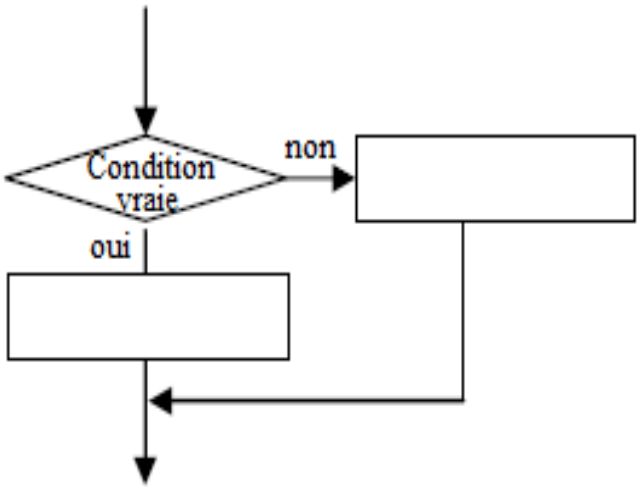
Algorithme

```
si (Condition) alors
   Instruction 1.1 ;
   Instruction 1.2 ;
    ……
   Instruction 1.N ;
Sinon
   Instruction 2.1 ;
   Instruction 2.2 ;
   ……
   Instruction 2.N ;
Finsi ;
```

Pascal

```
if (Condition) then
  begin
    Instruction 1.1 ;
    Instruction 1.2 ;
     ……
    Instruction 1.N ;
  end (*pas de ; *)
else
  begin
    Instruction 2.1 ;
    Instruction 2.2 ;
    ……
    Instruction 2.N ;
  end ;
```

c)organigramme :



Exemple 2 :
Ecrire un algorithme qui permet de trouver le maximum entre deux nombres entiers saisis au clavier.

```
Algorithme Max ;

Var a, b : Entier ;
Début
Ecrire ('Entrez deux nombres : ') ;
Lire (a, b );
Si (a >b) Alors
  Ecrire ('La valeur maximale est=',a)
 Sinon
  Ecrire ('La valeur maximale est=',b) ;
Finsi ;
Fin.
```

```
program Max ;
Uses wincrt;
Var a, b :integer ;
begin
write('Entrez deux nombres : ') ;
read (a, b );
if (a >b) then
begin
 write ('La valeur maximale est=',a) ;
end
else
begin
write ('La valeur maximale est=',b) ;
end;
end.
```

2.3 La structure si… alors …sinon si… alors …sinon (if …then…else if …then…else)

a)Strucrure :

Algorithme

```
si (Condition1) alors
   Instruction 1.1 ;
   Instruction 1.2 ;
    ……
   Instruction 1.N ;

sinon si (Condition2) alors
        Instruction 2.1 ;
        Instruction 2.2 ;
        ……
        Instruction 2.N ;

sinon
        Instruction 3.1 ;
        Instruction 3.2 ;
        ……
        Instruction 3.N ;
       finsi ;
finsi ;
```

Pascal

```
if (Condition1) then
 begin
   Instruction 1.1 ;
   Instruction 1.2 ;
    ……
   Instruction 1.N ;
 end
else if (Condition 2) then
 begin
        Instruction 2.1 ;
        Instruction 2.2 ;
        ……
        Instruction 2.N ;
 end
 else
 begin
        Instruction 3.1 ;
        Instruction 3.2 ;
        ……
        Instruction 3.N ;
 end ;
```

Remarque :

Par convention la partie **else** correspond à l'instruction **if** la plus proche.

Exemple 3

Ecrire un algorithme qui permet de trouver le maximum entre trois nombres entiers saisis au clavier.

Solution :

```
Algorithme Max3 ;
Var a, b : Entier ;
Début
Ecrire ('Entrez trois nombres : ') ;
Lire (a, b, c );
Si (a >b) et (a>c)Alors
  Ecrire ('La valeur maximale est=',a)
 Sinon
    si (b>c)
    Ecrire ('La valeur maximale est=',b) ;
    Sinon
    Ecrire ('La valeur maximale est=',c) ;
    Finsi ;
Finsi ;
Fin.
```

```
program Max3 ;
uses wincrt ;
Var a, b, c:integer ;
begin
write('Entrez trois nombres : ') ;
read (a, b, c );
if (a >b) and (a>c) then
begin
write ('La valeur maximale est=',a) ;
end
else if (b >c) then
      begin
      write ('La valeur maximale est=',b) ;
      end
      else
      begin
      write ('La valeur maximale est=',c) ;
      end ;
end.
```

Remarques:

1- Dans le langage Pascal on enlève le point virgule **;** de l'instruction qui se situe directement avant le **else**.

2- Dans un programme si la structure de **if** ou **else** concerne l'exécution d'une seule instruction, on peut ne pas mettre cette instruction entre **begin** et **end ;** par exemple le programme Max3 pourra être écrit comme suit :

```
program Max3 ;
uses wincrt ;
Var a, b, c:integer ;
begin
write('Entrez trois nombres : ') ;
read (a, b, c );
if (a >b) and (a>c) then
    write ('La valeur maximale est=',a)
else  if (b >c) then
       write ('La valeur maximale est=',b)
       else
       write ('La valeur maximale est=',c) ;
end.
```

2.4 La sélection sur choix multiples selon (case of)

S'il y a plus de deux choix possibles, l'instruction selon permet une facilité l'écriture du programme.

a) Structure :

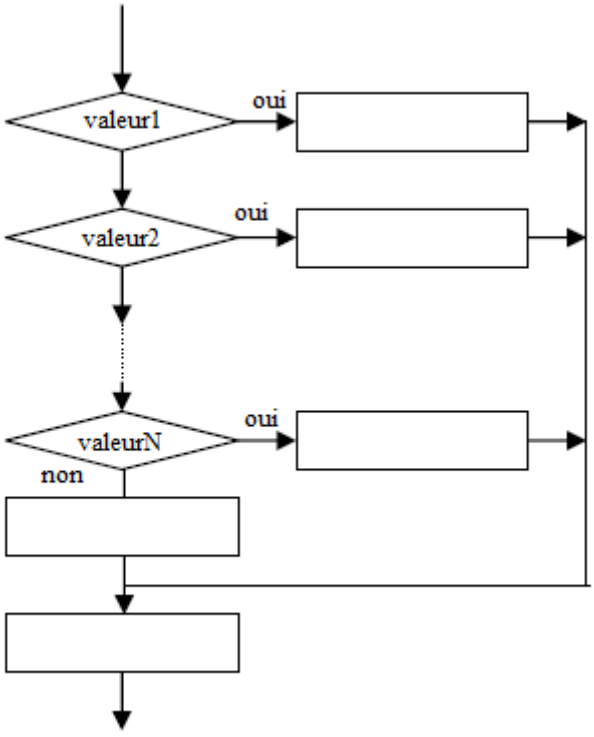
Algorithme

selon variable
valeur1 : instructions 1 ;
valeur2 : instructions 2 ;
…
valeurN : instructions N ;
Sinon instructions ;
Fin selon;

Pascal

```
case variable of
 valeur1  : begin
            instructions 1 ;
            end;
 valeur2 :  begin
            instructions 2;
            end ;
 …
 valeurN: begin
            instructions N;
            end;
 Else   begin
        instructions;
        end ;
end ;
```

b) Organigramme :



Remarque : Comme pour la structure de **if** on peut omettre les **begin** et **end ;** s'il y a une seule instruction par cas, sauf le dernier **end ;** il fait parti de la structure de **case of**.
La structure de **case of** peut être remplacée par la structure **if**..**else** selon le nombre des cas.

Exemple 4

Écrire un programme qui permet de :1-Lire deux réels X et Y (avec Y non nul)
2-En utilisant l'instruction **case of** (selon) calculer selon le nombre C saisi par l'utilisateur:
Si C=1 : X+Y.
Si C= 2 : X*Y.
Si C= 3 : X/Y. Sinon il affiche le message « pas de calcul ».

Solution :

```
Algorithme  calcul ;
Var
X,Y, P, S, D: réel ;
C:entier ;
début
écrire ('donner deux nombres') ;
lire(X,Y);
écrire('Entrer la valeur de votre choix ') ;
écrire ('1 : somme') ;
écrire ('2 : produit') ;
écrire ('3 :division') ;
lire(C) ;
Selon  C
1 :  S← X+Y ;
     write ('La somme=',S) ;
2 :  P←X*Y ;
   écrire ('Le produit =',P) ;
3 :  D←X/Y ;
   écrire ('La division =',D) ;
Sinon écrire (' pas de calcul ') ;
fin selon;
fin.
```

```
program  calcul ;
uses wincrt;
var
X,Y, P, S, D: real ;
C:integer ;
begin
writeln ('donner deux nombres') ;
Read(X,Y);
writeln('Entrer la valeur de votre choix ') ;
writeln('1 : somme') ;
writeln('2 : produit') ;
writeln('3 :division') ;
read(C) ;
case  C of
1 :  begin
     S:= X+Y ;
     write ('La somme=',S) ;
     end ;
2 : begin
    P:= X*Y ;
    write ('Le produit=',P) ;
    end ;
3 : begin
    D:= X/Y ;
    write ('La division=',D) ;
    end ;
else write (' pas de calcul ') ;
end ;
end .
```

3 Exercices sur les structures conditionnelles

Exercice 1

Ecrire un programme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on inclut le traitement du cas où le produit peut être nul).

Exercice 2

1- En utilisant l'instruction **case of** (selon) écrire un programme qui permet de lire un numéro comprit entre 1 et 12 et d'afficher le nom du mois correspondant. Si le numéro entré est en dehors de cet intervalle, un message d'erreur doit être affiché.

2- Récrire le programme avec l'instruction **if** à la place de **case of**.

Série d'exercices avec solution

Exercice 3

Ecrire un programme qui lit un entier et affiche ensuite s'il est pair ou impair.

Solution :

```
Algorithme pair_impair ;
Var
a : Entier ;
Début
Ecrire('Donner la valeur d'un entier: ') ;
 Lire(a) ;
Si (a mod 2 = 0) Alors
Ecrire(a, ' est pair')
Sinon
Ecrire(a, ' est impair') ;
Finsi ;
Fin.
```

```
Program pair_impair ;
uses wincrt ;
var
a : integer ;
Begin
Write('Donner la valeur d'un entier: ') ;
 Read(a) ;
if(a mod 2 = 0) then
begin
Write(a, ' est pair') ;
end
else
begin
Write(a, ' est impair') ;
end;
end.
```

L'exercice peut être résolu en utilisant la fonction **odd(a)** (voir chapitre 2) à la place de l'opérateur **mod.**

Exercice 4

Ecrire un programme qui permet de résoudre dans *R* une équation du premier degré de la forme :

ax+b = 0

Solution :

```
Algorithme equa1d ;
Var
a, b, x : Réel ;
Début
écrire ('Donner la valeur de a et b') ;
Lire(a, b) ;
Si (a<>0) Alors
x←-b/a ;
écrire(x)
Sinon
  Si  (a=0) et (b=0) Alors
  Ecrire('l''ensemble R')
  Sinon
  ecrire('l''ensemble vide') ;
  Finsi ;
Finsi ;
Fin.
```

```
Program equa1d ;
uses wincrt;
var
a, b, x : Real ;
begin
write ('Donner la valeur de a et b'') ;
Read(a, b) ;
if (a<>0) then
begin
x :=-b/a ;
write(x) ;
end;
else  if (a=0) and (b=0) then
begin
  Write('l''ensemble R') ;
end
  else
begin
  write('l''ensemble vide') ;
  end ;
end.
```

Exercice 5

Ecrire un programme qui permet de résoudre dans *R* une équation du second degré de la forme :

$ax^2+bx+c = 0$ (dans le cas ou a est différent de 0).

Solution

```
Algorithme equa2d ;
Var
a, b, c, delta,x1,x2 : Réel ;
Début
Ecrire('Donner la valeur de a : ') ;
Lire(a) ;
Ecrire(' Donner la valeur de b : ') ;
 Lire(b) ;
Ecrire(' Donner la valeur de c : ') ;
Lire(c) ;
delta ← b*b – 4*a*c ;
Si (delta < 0) Alors
Ecrire('pas de solution dans R')
Sinon
  Si (delta = 0) Alors
    x1←-b/(2*a);
    Ecrire('x1 = x2 = ',x1)
  Sinon
    x1←(-b-sqrt(delta))/(2*a);
    x2←(-b+sqrt(delta))/(2*a);
   Ecrire('x1 =',x1) ;
   Ecrire('x2 =',x2) ;
   Finsi ;
Finsi ;
Fin.
```

```
program equa2d ;
uses wincrt;
Var
a, b, c, delta,x1,x2 : Real ;
begin
write ('Donner la valeur de a : ') ;
read(a) ;
write ('Donner la valeur de b : ') ;
read(b) ;
write ('Donner la valeur de c : ') ;
read(c) ;
delta := b*b - 4*a*c ;
if (delta < 0) then
begin
write('pas de solution dans R') ;
end
else
  if (delta = 0) then
  begin
    x1:=-b/(2*a);
    write('x1 = x2 = ',x1:2) ;
    end
  else
  begin
    x1:=(-b-sqrt(delta))/(2*a);
    x2:=(-b+sqrt(delta))/(2*a);
   writeln('x1 =',x1:2) ;
   writeln('x2 =',x2:2) ;
  end;
end.
```

Exercice 6

Soit l'algorithme suivant :

Algorithme Test
Const d=5
Var a,b,c,som :entier
Début
lire(a,b,c)
Si (a<b) **alors**
a←b*2
Sinon si (a<c) **alors**
a←b+c
Sinon
c←c+d
b←b+d
Finsi
Finsi
som←a+b+c
écrire(som)
Fin

1) Traduire l'algorithme en Pascal.

2) Compléter les tableaux de la trace d'exécution de l'algorithme pour :

Cas1 : a=5, b=1 et c=8.

Instruction	a	b	c	d	som	Écran
Lire(a,b,c)						

Cas2 : a=6, b=2 et c=4.

Instruction	a	b	c	d	som	Écran
Lire(a,b,c)						

Solution : 1) Traduire l'algorithme en Pascal.

```
Program Test ;
Uses wincrt ;
Const d=5 ;
Var       a,b,c,som :integer ;
begin
read(a,b,c) ;
if(a<b) then
begin
a:=b*2 ;
end
else if (a<c) then
begin
a :=b+c ;
end
else
begin
c :=c+d ;
b :=b+d ;
end;
som :=a+b+c ;
write(som);
end.
```

2) Compléter les tableaux de la trace d'exécution de l'algorithme pour :

Cas1 : a=5, b=1 et c=8.

Instruction	a	b	c	d	som	Écran
Lire(a,b,c)	5	1	8	5	/	/
(a<c) oui: a←b+c	9	1	8	5	/	/
som←a+b+c	9	1	8	5	18	18

Cas2 : a=6, b=2 et c=4.

Instruction	a	B	c	d	som	Écran
Lire(a,b,c)	6	2	4	5	/	/
(a<b)non (a<c)non c←c+d b←b+d	6	7	9	5	/	/
som←a+b+c	6	7	9	5	22	22

4 Les Structures de contrôle répétitives

La structure itérative (boucle) répète l'exécution d'une ou plusieurs instructions.

4.1 La boucle pour …faire (for…do)

a) Structures :

Algorithme :

Pour variable←valeur initiale **à** valeur finale **faire**
Instruction 1 ;
Instruction 2 ;
……
Instruction N ;
Fin pour ;

- La boucle **pour** peut indiquée la valeur du pas utilisée :

Pour variable←valeur initiale **à** valeur finale **pas** valeur du pas **faire**
Instruction 1 ;
Instruction 2 ;
……
Instruction N ;
Fin pour ;

Pascal :

for variable **:=** valeur initiale **to** valeur finale **do**
begin
Instruction 1 ;
Instruction 2 ;
……
Instruction N ;
end ;

On peut inverser la boucle (à l'envers) par l'utilisation du mot clé **downto** à la place du **to**:
for variable **:=** valeur finale **downto** valeur initiale **do**
begin
Instruction 1 ;
Instruction 2 ;
……
Instruction N ;
end ;

b) Principe :

- La variable représente le compteur qui détermine le nombre de répétition, sa valeur est incrémentée de 1 à chaque exécution des instructions à répéter
- Le nombre de répétition = *(valeur finale - valeur initiale+1)* fois.
- Le pas est égal à 1 (**to**) ou -1 (**downto**).

Exemple

Ecrire le programme qui calcule la somme de n premiers nombres entiers
som=1+2+…+n

```
Algorithme Somme ;
var
    i, n, som : entier ;
début
Lire(n) ;
som←0 ;
Pour i ← 1 à n faire
som←som+i ;
Fin pour ;
Ecrire ('La somme est :  ',som) ;
Fin.
```

```
program somme ;
uses wincrt;
var
     i,n,som : integer ;
begin
read(n) ;
som :=0 ;
for i :=1 to n do
begin
 som :=som+i;
end;
write ('La somme est :  ',som) ;
end.
```

4.2 La boucle Tant que…faire (while …do)

a) Structures :

Algorithme :

```
Tant que (condition) faire
Instruction 1 ;
Instruction 2 ;
  ……
Instruction N ;
Fintq ;
```

Pascal :

```
While  (condition) do
begin
Instruction 1 ;
Instruction 2 ;
  ……
Instruction N ;
end ;
```

b) Principe :

- Le nombre de répétitions dépend de la condition.
- Tant que la condition est vraie les instructions sont exécutées sinon on sort de la boucle.

Exemple

Ecrire le programme qui calcule la somme de n premiers nombres entiers
Som=1+2+…+n

```
Algorithme Somme ;
Var    i,n,som : entier ;
Début
Lire(N) ;
 i←1 ;
som←0 ;
Tant que (i <= N) faire
 som ←som+I;
 i←i+1 ;
 Fintq ;
Ecrire ('La somme est :  ',som) ;
Fin.
```

```
program Somme ;
uses wincrt;
var    i,n,som : integer ;
begin
read(N) ;
 i:=1 ;
som :=0 ;
while (i <= N) do
begin
 som :=som+I;
 i :=i+1 ;
end;
write ('La somme est:  ',som) ;
end.
```

4.3 La boucle Répéter…Jusqu'à (repeat…until)

a) Structure :

Algorithme :

```
répéter
Instruction 1 ;
Instruction 2 ;
  ……
Instruction N ;
jusqu'à (Condition) ;
```

Pascal:

```
repeat
Instruction 1 ;
Instruction 2 ;
  ……
Instruction N ;
until (Condition) ;
```

b) Principe :

- Les instructions sont réalisées au moins une fois puis elles sont répétées jusqu'à ce que la condition soit vraie.

Exemple :

Ecrire le programme qui calcule la somme de n premiers nombres entiers
Som=1+2+…+n

```
Algorithme Somme ;
Var    i,n,som : entier ;
 Début
Lire(N) ;
 i←1 ;
som←0 ;
répéter
 som ←som+i;
 i←i+1 ;
jusqu'à (i>n) ;
Ecrire ('La somme est :  ',som) ;
Fin.
```

```
program somme ;
uses wincrt;
var    i,N,som : integer ;
begin
read(N) ;
 i :=1 ;
som :=0 ;
repeat
 som :=som+i;
 i :=i+1 ;
until(i>n);
write ('La somme est :  ',som) ;
end.
```

Remarque :

- Une structure de contrôle répétitive peut contenir une autre structure de contrôle répétitive ou conditionnelle et le contraire et aussi vraie.
- Dans le programme si la structure de contrôle répétitive **for** ou **while** répète une seule instruction le **begin** et **end ;** de la structure ne sont pas obligatoires.

5 Exercices sur les structures répétitives

Exercice 1
Tracer l'exécution des parties des programmes suivants:

```
i :=0 ;
S:=0 ;
while (i <=5) do
begin
i:=i+1;
S:= S+ i+1;
end;
writeln ('S=',S);
writeln ('i=', i);
```

```
i:=1 ;
while (i<=5) Do
begin
writeln ('i=', i);
end;
```

```
i:=0 ;
S:=1 ;
Repeat
i:=i+2 ;
S:= S* i;
Until (i >7);
writeln ('i=', i) ;
writeln ('S=',S);
```

```
S:=0 ;
For  i :=1 To 8 do
begin
S:= S+ i;
writeln ('i=', i);
writeln ('S=',S);
end ;
```

```
S1:=0 ;
S2:=0;
for i :=1 To 10 do
begin
  if ( i mod 2 = 0)then
  S1:= S1+ i
  else
  S2:=S2+i;
end ;
writeln ('S1=',S1);
writeln ('S2=',S2);
```

```
for i :=6 Downto 1 do
begin
writeln ('i=', i);
end;
```

```
S:=0 ;
for i :=1 To 6 do
begin
  if ( i mod 5 = 0)then
  begin
  S:= S+ i;
  writeln ('i=', i);
  end;
writeln ('S=',S);
end ;
```

Série d'exercices avec solution

Exercice 2
Ecrire un programme qui lit un entier positif n puis calcule et affiche sa factorielle selon la formule **n! = 1 * 2 *…* n**. (avec la boucle **for**, **while** et **repeat**).

Solution factorielle avec la boucle for:

```
Algorithme factorielle ;
Var
i,n,f : entier ;
Début
écrire(' Donner la valeur de n ') ;
lire(n) ;
f←1;
pour i←1 à n faire
f←f*i;
fin pour;
écrire ('factorielle=',f);
fin.
```

```
program factorielle ;
uses wincrt;
Var
i,n,f : integer ;
begin
write(' Donner la valeur de n ') ;
read(n) ;
f:=1;
for i:=1 to n do
begin
f:=f*i;
end;
write('factorielle=',f);
end.
```

Solution factorielle avec la boucle while:

```
Algorithme factorielle ;
Var
i,n,f : entier ;
début
écrire(' Donner la valeur de n ') ;
lire(n) ;
f←1;
i←1;
tant que (i≤n) faire
f←f*i;
 i←i+1;
fintq;
écrire('factorielle= ',f);
fin.
```

```
program factorielle ;
uses wincrt;
Var
i,n,f : integer ;
begin
write(' Donner la valeur de n ') ;
read(n) ;
f:=1;
i:=1;
while (i<=n) do
begin
f:=f*i;
 i:=i+1;
end;
write('factorielle= ',f);
end.
```

Solution factorielle avec la boucle repeat:

```
Algorithme factorielle ;
Var
i,n,f: entier ;
début
écrire('Donner la valeur de n ') ;
lire(n) ;
f←1;
i←1;
répéter
f←f*i;
i←i+1;
jusqu'à(i>n);
écrire('factorielle=',f);
fin.
```

```
program factorielle ;
uses wincrt;
Var
i,n,f : integer ;
begin
write('Donner la valeur de n ') ;
read(n) ;
f:=1;
i:=1;
repeat
f:=f*i;
 i:=i+1;
until(i>n);
write('factorielle=',f);
end.
```

Exercice 3

1. Ecrire un programme permettant de calculer l'expression suivante par la boucle **while** :

$$s = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots \pm \frac{1}{n}$$

2. Dérouler le programme pour n=3.

Solution :1-

```
program exo3 ;
uses wincrt;
Var
n, p,i: integer;
s : real;
begin
writeln('Donner la valeur de n : ') ;
read(n) ;
s  :=0 ;
p:=1;
i:=1;
while(i<= n) do
begin
s := s + p/i;
p :=  -p;
i:=i+1;
end;
writeln('s = ',s:8:4);
end.
```

2.

Etapes	**n**	**s**	**p**	**i**	**écran**
Avant la boucle	3	0	1	1	/
Début de la boucle	3	1	-1	2	/
	3	1-1/2	1	3	/
Fin de la boucle	3	1-1/2+1/3	-1	4	0.8333

Exercice 4

Ecrire le programme qui permet de calculer la somme suivante via la boucle **repeat** et **for** avec x réel et n entier:

$$s = 1 + \frac{x^2}{2} + \frac{x^4}{x^2} + \frac{x^6}{2^3} + \cdots + \frac{x^{2n}}{2^n}$$

```
program exo4 ;
uses wincrt;
Var i,n :integer;
T, Som,x : real;
begin
write ('Donner la valeur de n et x') ;
read(n,x) ;
som :=0 ;
T:=1 ;
for i:=0 to n do
begin
Som:=Som+T ;
T:=T*x*x/2 ;
end;
writeln ('Le resultat est : ', Som);
end.
```

Bibliographie

- Toufik Djouder, Exercices corrigés en Algorithmique, Dar El Hadith Lil-Kitab, 2002.

- Cours informatique1, 1éré année Sciences et Technologie, 2014-2015, Université frères Mentouri Constantine 1, consulté en 2015.

- Edouard Thiel, Algorithmes et programmation en Pascal, 2004, Université Aix-Marseille.

- Adnen A., Rappels sur l'Architecture de base d'un ordinateur, 2009-2010, Institut Supérieur d'Informatique et des Technologies de Communication Sousse.