Module

informatique I

- Cours :

  Responsable : BIDI NORIA

  - ***Contact*** :
    - **n_bidi@univ-mascara.dz**
    - **http://www. univ-mascara.dz/**
  - ***Cours*** :
    - **1 séances /sem : dimanche et mardi**

      Amphi K,J ET I

- TP :
  - ***Intervenants*** :
    - BIDI NORIA
    - BENFRIHA HICHAM
    - BENFRIHA HOURIA
    - BOUFERA HADIA
    - RIMANI RACHID
    - CHENAFI SOFIANE

SOMMAIRE

Introduction à l'informatique

Aujourd'hui, on vit dans un monde où l'on ne peut pas se passer de l'**informatique**. L'informatique peut se trouver partout, dans la vie quotidienne de chacun, l'entreprise et les administrations, la société surtout avec l'apparition des différents réseaux sociaux et internet.

1 Introduction à l'informatique

L'informatique est la science du traitement automatique des informations. Cette définition fait appel aux deux mots abstraits "information" et "traitement".

Les informations traitées par l'informatique sont de différentes natures ; des nombres, du texte, des sons, des images, des clips vidéo etc. On définit en général deux grandes catégories d'information : la première est l'information **numérique**, par exemple l'âge d'une personne ou le prix d'une voiture, la seconde est l'information **symbolique** comme par exemple la description d'une personne ou la couleur d'un mur. Quelle que soit l'information à traiter, on se ramène toujours à une information numérique en attribuant une valeur à chaque information. Par exemple, si on a l'ensemble de couleurs {rouge, vert, bleu}, on peut proposer le codage suivant : 1<--> rouge; 2 <--> vert; 3 <--> bleu. En codant toutes les informations par des valeurs numériques, le traitement des informations se ramène toujours à un traitement sur des nombres. Ce traitement numérique peut être une opération mathématique classique (addition, multiplication...), une comparaison entre 2 nombres, une mémorisation, un transfert de données ou une combinaison de ces opérations élémentaires.

1.1 Les grands domaines de l'informatique

L'informatique, comme d'autres disciplines, comporte plusieurs sous-disciplines ou *domaines*. On peut citer les domaines suivants :

• L'architecture des ordinateurs : Il s'agit de toutes les connaissances relatives à la conception des circuits, des cartes, de tous les composants électroniques, en un mot du "hardware" (ce qui est dur, que l'on peut donc sentir au toucher).

• Les systèmes d'exploitation : pour exploiter les ressources de l'ordinateur (L'écran, le clavier, la souris, le lecteur de disquette, le disque dur…) il faut faire appel à un système d'exploitation. Le système d'exploitation est un programme qui gère l'interface entre l'ordinateur et l'utilisateur. On peut citer les systèmes d'exploitation MSDOS, Windows95, OS/2, Unix, Mac OS….

• La Bureautique : La Bureautique est l'ensemble des outils informatiques utilisés pour automatiser des tâches administratives ou de secrétariat. Parmi ces outils, le traitement de texte et le tableur sont les plus connus.

• Les bases de données : Une base de données est un ensemble organisé d'informations. L'exemple le plus connu est celui de l'annuaire téléphonique.

• Les réseaux : Un réseau informatique est un ensemble d'ordinateurs reliés entre eux. Les câbles sont soient dédiés au réseau informatique, soient les mêmes que ceux utilisés pour le téléphone, soient inexistants si on utilise par exemple une liaison laser ou un satellite.

• Les langages informatiques : Toutes les applications informatiques (le traitement de texte, les jeux, …) sont conçues dans un langage de programmation, avec un vocabulaire et une grammaire spécifique par exemple le Pascal, le C, le C++, le Fortran, le LISP, etc.

• L'automatique : L'automatisation des tâches nécessite l'automatisation du traitement de l'information, mais pas obligatoirement un ordinateur. Parmi les différents domaines de l'automatique, la robotique ne fait pas partie de l'informatique, mais elle y est cependant très liée.

• Génie logiciel, les jeux, l'intelligence artificielle, Informatique industrielle, Modélisation et simulation, Télécommunications, etc.

1.2 Le fonctionnement des ordinateurs

Un **ordinateur est une m**achine électronique programmable capable de réaliser des calculs logiques sur des nombres binaires.
C'est une machine (Hardware) :
Le fonctionnement d'un ordinateur est basé sur une architecture matérielle (processeur, support de stockage, interfaces utilisateurs, connexion, . . .) dont le fonctionnement est soumis aux lois de la physique.
C'est une machine programmable (Software) :
Cette machine est capable de remplir des tâches différentes selon les instructions qui lui sont adressées. Ces instructions, rédigées sous forme de programmes par les informaticiens, sont traitées en fin de course par le matériel de l'ordinateur.

A. Partie Matériel (hardware) :

Tout ce qui compose l'ordinateur et ses accessoires, chaque composant possède une fonction particulière, dont les éléments les plus importants sont les suivants :

1-Unité centrale :

Un boîtier contenant l'ensemble des éléments internes du PC qui sert à sauvegarder, traiter et restituer les données en exécutant les instructions du programme en cours

Carte mère :

carte-mère est le système nerveux du pc, c'est sur cette carte que sont connectés tous les éléments de l'unité centrale.

Processeur :

Un processeur (aussi appelé microprocesseur ou CPU pour **Central Processing Unit**) est le cœur de l'ordinateur, ce composant a été inventé par Intel (modèle 4004) en 1971, il est chargé de traiter les informations et d'exécuter les instructions, caractériser par sa vitesse d'horloge mesurer en hertz HZ (vitesse d'exécution des instructions).

Mémoire centrale RAM :

La mémoire se présente sous forme de composants électroniques ayant la capacité de retenir et restituer des informations d'une façon temporaire (volatile), SDRAM, DDRAM, DDRAM2,……

Disque dur :

Le disque dur est l'organe du PC servant à conserver les données de manière permanente, même lorsque le PC est hors tension, contrairement à la mémoire centrale, qui s'efface à chaque redémarrage de l'ordinateur, c'est la raison pour laquelle on parle de **mémoire de masse**, sa capacité exprimée en Go.

Carte graphique :

Elle permet de convertir des données numériques brutes en données pouvant être affichées sur un périphérique destiné à cet usage (écran, vidéo projecteur, etc...).

2-Les Périphériques

Les Périphériques désignent tous les appareils susceptibles d'échanger des informations.

Périphériques entrées :

Des organes et des appareils servants à recueillir les informations qui sont ensuite transformées (numérisées et codées en binaires) pour être utilisables par l'unité centrale...).

Clavier : AZERTY(français)/QWERTY(anlglais)
Souris : optique….
Lecteur CD/DVD : X56 vitesse de lecture
Modem : Adsl, routeur, Wifi….

Périphériques de sorties :

Des dispositifs qui transmirent l'information binaire de l'unité centrale vers l'extérieur sous une forme compréhensible par l'utilisateur humain.

Ecran : tube cathodique, Tft (écran plat)
imprimante : jet d'encre, laser
Graveur CD/DVD :
Haut parleur :

B -Logiciels (software) :

Ce nom désigne l'ensemble des programmes qui permettent aux utilisateurs de travailler avec un ordinateur.

Un programme est un enchainement d'instructions, écrit dans un langage de programmation, exécuté par une ordinateur, permettant de traiter un problème et de renvoyer des résultats.

Système d'exploitation :

Le système d'exploitation est un logiciel « système » qui contient l'ensemble des instructions et des informations intermédiaire entre le matériel informatique et les logiciels applicatifs.

La famille Windows de Microsoft :
La famille Mac Os d'Apple :
Unix : communauté universitaire (open source).
Et d'autre comme pour les smart phones.

Logiciel standard :

Des programmes commerciaux, destiné a un large usage, Microsoft office : Word, Excel,…..

Logiciel utilisateur
Logiciel spécifique :

Un logiciel applicatif ou application informatique contient les instructions et les informations relatives à une *activité* automatisée, développé spécialement pour une entreprise, il peut s'agir d'une activité de *production ou de gestion (*logiciel de la poste).

Drivers

Un **driver (ou pilote)** sert de lien entre le matériel et le système d'exploitation, si les drivers n'existaient pas, le système d'exploitation devrait reconnaître tous les périphériques du marché, d'où une prise d'espace disque importante, carte graphique, carte son,…….

1.3 Le codage des informations

Pour effectuer des calculs, il faut un moyen de représenter les nombres. De par la nature de ses composants électroniques, le robot ne perçoit que deux états: composant allumé et composant éteint. De cette perception découle le langage binaire, qui utilise par convention les deux symboles 0 (éteint) et 1 (allumé). Ne connaissant que le 0 et le 1, l'ordinateur utilise un code pour représenter une information aussi simple qu'un nombre entier ou un caractère. Ce code est un programme, qui différencie chaque type d'information et transforme une information (donnée numérique ou alphabétique) en valeurs binaires. À l'inverse, ce programme sait aussi transformer un nombre binaire en valeur numérique ou alphabétique. Il existe autant de codes que de types d'informations.

Introduction à l'algorithme

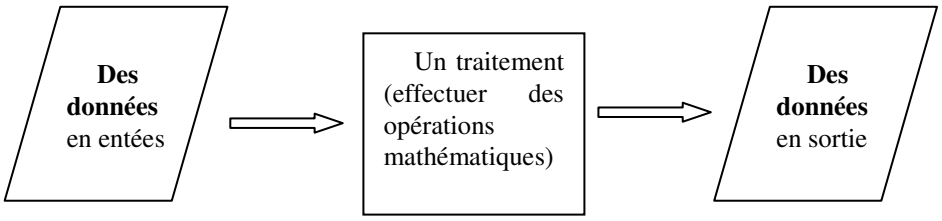
1. Qu'est ce qu'un algorithme ?

D'une façon générale, un algorithme est une suite d'instructions (actions) logiquement ordonnées, qui permet de résoudre un problème données.

Exemple : recette de cuisine, mode d'emploi, notice de montage,…

Un algorithme est une suite d'instructions logiquement ordonnées, qui transforment **des données** en entées (inputs) vers des données en sortie (outputs). Ces derniers outputs représentent la solution d'un problème donné.

2. Principe général



Exemple : Considérons le traitement qui consiste à calculer la moyenne d'un groupe d'étudiants pour un module donné. Dans ce cas :

- Pour effectuer le calcul de la moyenne, on a besoin de **la note** de chaque étudiant dans le module en question
- Ensuite, on effectue le calcul comme suit :
  - On calcule la **somme** des notes,
  - On **divise** par le nombre d'étudiants,
- Enfin, on affiche le résultat (la moyenne).

Généralement, lorsqu'on cherche un algorithme permettant d'automatiser un traitement donné, on doit se poser trois questions :

- Qu'est ce qu'on doit obtenir comme résultat ?
- Quelles sont les données dont on a besoin ?
- Comment faire (traitement proprement dit) ?

3. Comment écrire un algorithme ?

L'algorithme suit les étapes suivantes :

1- Saisir les données déterminées d'une manière très précise nécessaires pour la résolution du problème.
2- Le traitement de données saisis pour résoudre le problème.
3- L'affichage des résultats obtenus après le traitement.

4. Structure d'un algorithme :

Un algorithme manipule des données, les données avant de les utiliser il faut les identifier et les
déclarer en utilisant les identificateurs. Un algorithme est constitué de trois parties :

– ***Entête :*** dans cette partie on déclare le nom de l'algorithme à travers un identificateur.
– ***Déclarations :*** dans cette partie on déclare toutes les données utilisées par l'algorithme.
– ***Corps :*** dans cette partie de l'algorithme, sont placées les tâches (instructions, opérations) à exécuter.

Pour écrire un algorithme, il faut suivre la structure suivante :

Un algorithme se compose d'un **entête**(le nom), **déclarations**(variable et constants), et d'un **corps** (Début, suite instruction, fin).

Algo (le nom de l'algorithme) <identificateur_algo>

Variable (déclaration des variables) <Déclarations>

Début
Instruction 1 ;
Instruction2 ;
. . ..
Instruction N ;
Fin.

Le nom de l'algorithme, les variables et les constants sont caractérisées par leur nom qui doit être significatif et respecter les règles d'un identificateur.

Un identificateur

Un identificateur Sert à donner un nom à un objet. Ce nom contient les lettres (un caractère de 'a'..'z' ou 'A'..'Z' ou '_') , les nombres (un caractère de '0'..'9').
Un identificateur est une suite de lettres ou de digit accolées, commençant par une lettre.

Un identificateur doit être significatif.
Un identificateur doit être écrit sur 8 positions (Taille <= 8 caractères).
Un identificateur doit commencer obligatoirement par une lettre.
Un identificateur ne doit pas comporter le caractère « espace ».
Un identificateur ne doit pas comporter de caractères spéciaux autres que le tiret de soulignement (_).
Un identificateur doit être unique dans un algorithme

Un constant (CONST) est un objet qui ne change pas de valeurs lors de l'exécution d'un programme. Par exemple : La constante PI = 3.14 La constante e=2.7.

Une variable (VAR) est un objet ou une « boite » dans laquelle on met une valeur (un nombre, un mot…) qui peux changer (si l'utilisateur modifie la valeur par exemple). Elle est caractérisée par

- un identificateur (son nom)
- une valeur (qui peut varier au cours du programme)
- un type (qui détermine sa taille (de la place occupée dans la mémoire) et les opérations possibles)

5. Notion de Type :

Une variable utilisé dans un algorithme ne peut prendre qu'un ensemble de valeurs connues à l'avance ; toute variable possède un domaine de définition. En terme informatique, ce domaine est appelé le Type de la variable. Un type est alors caractérisé par : ses valeurs et les opérations qui peuvent s'effectuer sur des variables ayant ce type.

Les Types de base sont au nombre de cinq :

- Le Type Entier.
- Le Type Réel.
- Le Type Caractère.
- Le Type chaîne de Caractère.
- Le Type Booléen ou Logique.

a) Le Type Entier :

Le type Entier comprend un sous ensemble fini de nombres entiers, dont la taille varie en fonction des performances techniques de la machine et celles du langage de programmation utilisé. Pour la représentation Algorithmique de type entier, il suffit d'indiquer devant le nom de la variable son type.

Var Nomvar : Entier (Nomvar étant la variable qui sera utilisée dans l'algorithme)

Si on plusieurs variables en même temps, ce n'est pas la peine de les déclarer séparément ; on peut les regrouper en les séparant par une virgule. Var Nomvar1, Nomvar2, ……., NomvarN : Entier.

Les Opérations de base sur le type Entier :

Sur le type Entier, on distingue deux catégories d'opérations : les opérations arithmétiques (addition (+), soustraction (-) , Multiplication(*) , Division entière(DIV), Modulo (MOD), Comparaisons (≥, =, ≠)) les opérations algorithmiques telles que la lecture, l'écriture, etc .

b) Le Type Réel ;

Le type Réel comprend un sous ensemble fini de nombres réels, dont la taille varie en fonction des performances techniques de la machine et celles du langage de programmation utilisé

Var Nomvar : Réel.

Var Nomvar1, Nomvar2, ……., NomvarN : Réel

Les Opérations de base sur le type Réel : Sur le type Réel, on distingue deux catégories d'opérations : les opérations arithmétiques (addition (+), soustraction (-) , Multiplication(*) , Division (/), Comparaisons (<, >, ≥, =, ≠)) les opérations algorithmiques telles que la lecture, l'écriture, etc

c) Le Type Caractère :

Le type Caractère est ensemble de caractères comportant : les 26 lettres alphabétiques en majuscules ('A' jusqu'à 'Z') les 26 lettres alphabétiques en minuscules ('a' jusqu'à 'z') les 10 chiffres arabes ('0' jusqu'à '9'). quelques caractères spéciaux. Remarque : Chaque valeur d'un caractère est délimitée par deux apostrophes ' '.

Var Nomvar : Caractère

Var Nomvar1, Nomvar2, ……., NomvarN : Caractère Remarque :

Une variable de type caractère contient à un instant donnée un seul caractère.

Les Opérations de base sur le type Caractère : Sur le type Caractère, on distingue deux catégories d'opérations : les opérations propres aux caractères (comparaisons , ≥, =, ≠ (exemple 'a' < 'z')) et les opérations algorithmiques telles que la lecture, l'écriture, etc.

d) Le Type Chaîne de caractère : Ce sont des séquences obtenues en concaténant plusieurs caractères

Exemple: 'bonsoir' 'il fait froid' '254.25' 'j"imagine'

Remarque: les constantes caractères ainsi que les constantes chaînes apparaîtront entres apostrophes.

La constante apostrophe sera représentée par une double apostrophe.

Les Opérations de base sur le type Chaîne caractère

Concaténation (&) Longueur (chaîne) Extraction (sous-ch, ch).

e) Le Type Booléen ou Logique

Un type booléen dit également logique est un ensemble qui est constitué de deux éléments dont la valeur de l'un contredit celle de l'autre. Les valeurs attribuées à ces éléments varient d'un contexte à un autre. Pour représenter des nombres binaires, on déclare un type booléen formé de 0 et 1. Pour représenter une variable réponse, on déclare un type booléen formé de Oui et Non. Pour représenter une variable logique, on déclare un type booléen formé de Vrai et Faux. Pour représenter l'état d'une lampe, on déclare un type booléen formé de Allumée et Éteinte. Pour représenter l'état d'un interrupteur, on déclare un type booléen formé de Ouvert et Fermé.

Var Nomvar : Booléen.

Var Nomvar1, Nomvar2, ……., NomvarN : Booléen

Les Opérateurs Logiques Sur le type booléen, on applique des opérateurs logiques pour constituer une expression logique ; ces opérateurs sont : Comparaison (=, ≠), négation(NON), conjonction (ET) disjonction(OU).

Les instructions

Une instruction est une action élémentaire commandant à la machine un calcul, ou une communication avec un de ses périphériques d'entrés ou de sorties.
Les types d'instructions sont

- L'instruction d'entrée.
- L'instruction de sortie.
- Les instructions de traitements.

L'instruction d'entrée : L'instruction d'entrée ou de lecture permet à l'utilisateur de saisir des données au clavier pour qu'elles soient utilisées par le programme. (Permet d'assigner (affecte) à une variable, une valeur entrée au clavier)
Syntaxe : Lire (identificateur) Exemple : Lire (A) Cette instruction permet à l'utilisateur de saisir une valeur au clavier qui sera affectée à la variable A.
Remarque : Lorsque le programme rencontre cette instruction, l'exécution s'interrompt et attend que l'utilisateur tape une valeur. Cette valeur est rangée en mémoire dans la variable désignée.

L'instruction de sortie: L'instruction de sortie (d'écriture) permet d'afficher des informations à l'utilisateur à travers l'écran. (Permet d'écrire le contenu d'une variable ou d'une expression à l'écran).
Syntaxe : Ecrire (expression) Expression peut être une valeur, un résultat, un message, le contenu d'une variable... Exemple : Ecrire (A) Cette instruction permet d'afficher à l'écran la valeur de la variable A.

L'instruction de traitement

L'affectation : L'affectation permet d'affecter une valeur à une variable. Symbolisée en algorithmique par "←", elle précise le sens de l'affectation (permet de changer la valeur d'une variable).

Syntaxe :

Variable ← Expression
Une Expression : peut être soit : identificateur, constante, expression arithmétique et expression logique
Une affectation peut être définie en deux étapes :

- Evaluation de l'expression qui se trouve dans la partie droite de l'affectation
- Placement de cette valeur dans la variable.

Le déroulement

L'historique d'exécution(le déroulement d'un algorithme) : Pour bien comprendre le fonctionnement d'un algorithme, on peut construire un tableau représentant l'évolution de l'état de la mémoire au cours d'une exécution de l'algorithme.
Construire un historique d'exécution
1. Numéroter les instructions (n)
2. Sélectionner les variables à suivre (m)
3. Construire un tableau n+2 lignes * m+1 colonnes :

N° de ligne Variable	Variable 1	Variable 2
Avant exécution de l'instruction 1		
Après exécution de l'instruction		
Après exécution de l'instruction2		
Après exécution de l'instruction n		

Valeur de la variable v1 après l'exécution de toutes les instructions

Exemple 1 :
Algorithme Calcul ;
Variables A, B, C, D : entier ;
Debut
1.A ← 10 ;
2.B ← 30 ;
3.C ← A+B ;
4.D ← C*A ;
Fin.
Nous pouvons expliquer ce qui ce passe par le tableau suivant :

N° de ligne Variable	**A**	**B**	**C**	**D**
Avant 1	/	/	/	/
Après 1	10	/	/	/
Après 2	10	30	/	/
Après 3	10	30	40	/
Après 4	10	30	40	400

Exemple2 :
Algorithme deroulement ;
Variables A, B,C : entier ;
Debut
A← 5 ;
B←3 ;
C←A + B ;
A← 7 - A ;
B← B + 4 ;
C← B -2 ;
Ecrire (A , B, C) ;
Fin.

N° de ligne Variable	**A**	**B**	**C**
Avant 1	/	/	/
Après 1	5	/	/
Après 2	5	3	/
Après 3	5	3	8
Après 4	2	3	8
Après 5	2	7	8
Après 6	2	7	5

Exemple3 :
Quelles seront les valeurs des variables A ,B ,C ,D et E après exécution des instructions suivantes ?

Algorithme calcule ;
Variables A, B,C ,D: entier ;
E: réel ;
Debut
A← 14 ;
B←3 ;
C←A div B ;
D←A mod B ;
E←A / B ;
Ecrire (A , B, C,D ,E) ;

Fin.

N° de ligne Variable	**A**	**B**	**C**	**D**	**E**
Avant 1	/	/	/	/	/
Après 1	14	/	/	/	/
Après 2	14	3	/	/	/
Après 3	14	3	4	/	/
Après 4	14	3	4	2	/
Après 5	14	3	4	2	4.666666

Exemple 4 :

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

Algorithme 1

Algo echange1 ;
Var A, B : entier ;
Debut
A ← 5 ;
B ← 2 ;
A ← B ;
B ← A ;
Ecrire (A , B) ;
Fin.

N° de ligne Variable	A	B
Avant 1	/	/
Après 1	5	/
Après 2	5	2
Après 3	2	2
Après 4	2	2

Algorithme 2

Algo echange2 ;
Var A, B : entier ;
Debut
A ← 5 ;
B ← 2 ;
B ← A ;
A ← B ;
Ecrire (A , B) ;
Fin.

N° de ligne Variable	A	B
Avant 1	/	/
Après 1	5	/
Après 2	5	2
Après 3	5	5
Après 4	5	5

les deux dernières instructions permettent elles d'échanger les deux valeurs de B et A ?
Réponse : non

Exercice 5: Ecrire un algorithme qui permet d'échanger les valeurs de deux variables a et b.

Solution :

Algorithme permutation ;
Var a,b,c : réel ;
Début
Ecrire ″Donnez valeur de a :″ ;
Lire (a) ;
Ecrire ″Donnez valeur de b :″ ;
Lire (b) ;
c ← a ;
a ← b :
b ← c ;
Fin.

La variable c est une variable temporaire qu'on est obligé de l'utiliser (C'est est elle qui permet l'échange des valeurs des variables), elle doit être déclarée de même type que a et b. Les trois variables peuvent être déclarées d'un autre type (entier, booléen,…)

Je donne A := 8, B:=77.

N° de ligne Variable	**A**	**B**	**C**
Avant 1	/	/	/
Après 1	8	/	/
Après 2	8	77	/
Après 3	8	77	8
Après 4	77	77	8
Après 5	77	8	8

Exercice 6

Que produisent les algorithmes suivants ?

Algo N° 1	Algo N° 2	Algo N° 3	Algo N°4
Algo a6 ; Var A ,B,C : entier; Début A ← 6 ; B ← 3 ; C ← A A ← B ; B ← C ; Fin.	Algo a7 ; Var A ,B, C: chaîne de caractères; Début A ← "423" B ← "12" C ← A + B Fin	Algo a8 ; Var A ,B C: chaîne de Caractères; Début A ← "423" B ← "12" C ← A & B Fin	Algo a9 ; Var A ,B C: booléen; I, J: entire; Début A ← 5 ; B ← vrai ; C ← vrai ; I←5 ; J←10 ; B ← I + J ; Fin
A =3 B =6 C =6	Erreur (+ est n'est pas un opérateur de type chaîne de caractères)	A ="423" B ="12" C ="12423"	Erreur (type A et B bouléen)

Exercice 7 : Algorithme qui demande à un utilisateur d'entrer trois nombres et qui affiche la moyenne de ces trois nombres.

- Lire un nombre a
- Lire un nombre b
- Lire un nombre c

Caluler la moyenne de a ,b et de c.

- Afficher cette moyenne.

Algorithme moyenne ;

```
Var a, b, c : réel ;     /*a, b & c sont des variables qui désignent des nombres réels*/
      Moy : réel ;    /*moy est la variable qui contiendra la  valeur moyenne des variables a, b
& c. C'est un nombre réel */
Début
Ecrire(''Veuillez saisir au clavier trois nombres'');
/*Entrée ou saisie des valeurs des nombres a, b et c*/
Lire(a);
Lire(b);
Lire(c);
/*Calcul de la moyenne des nombres a, b et c */
Moy ← (a + b + c) / 3 ;
/*Affichage de la moyenne des nombres a, b et c */
Ecrire (''La moyenne de ces trois nombres est de :'', moy) ;
Fin.
```

Les structures de contrôle

Structures de contrôle (Les tests) : Un algorithme (ou programme) comporte deux types d'instructions :

- Les instructions de base : qui permettent la manipulation de variables telles que l'affectation, la lecture et l'écriture.

- Les instructions de structuration d'un programme : qui précisent l'enchainement chronologique des instructions de base.

Dans les notions précédentes, on a vu des algorithmes dans lesquels les instructions s'exécutent dans l'ordre de leur écriture (exécution séquentielle). Mais, la puissance d'un programme (algorithme) provient du fait qu'il peut effectuer des choix dans la façon d'exécuter les instructions.

Par exemple, dans un programme de calcul d'impôts, le montant à payer diffère selon le revenu déclaré. Le programme doit donc être capable d'appliquer à chaque tranche de revenu, un taux différent. Il effectuera par conséquent des choix sur les opérations à exécuter en fonction du revenu. Donc, il s'agit du deuxième type d'instructions appelées structures conditionnelles, ou structures alternatives ou tout simplement tests.

La plupart des autres structures de contrôle utilise la notion de condition (expression booléenne) :

Une condition a une valeur qui est, soit vraie, soit fausse.

Pour déterminer la réalité de cette valeur on utilise :

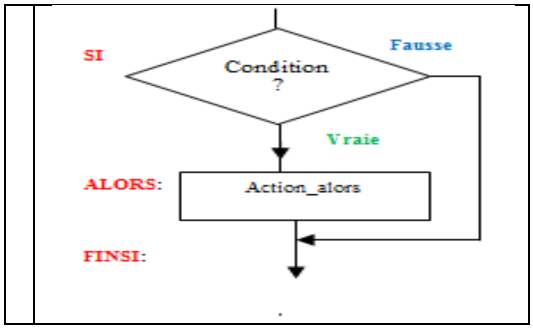
- les opérateurs de comparaisons = , ≤ , ≥ , ≠
- les opérateurs booléens (logique) : ET, OU, NON

L'Alternative SI_ALORS :

Elle permet d'effectuer tel ou tel traitement en fonction de la **valeur d'une condition.**

Syntaxe :	**Principe de fonctionnement :**
SI <condition> **ALORS** < action _alors> **FINSI**	**1 :** la condition est évaluée **2 : Si** la condition a la valeur **vraie** on exécute <**action_alors**> **Si** la condition a la valeur **fausse** on sortir

Organigramme : **Alternative SI_ALORS**

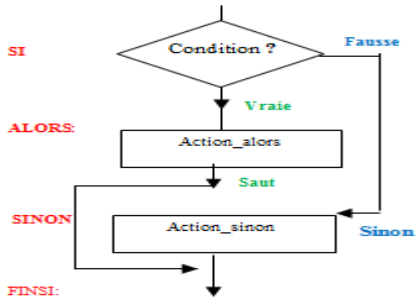


ALGORITHME	PROGRAMME
ALGO resultat ; VAR note :REEL ; DEBUT LIRE (note) ; SI note ≥ 10 ALORS ECRIRE(‘'Admis") FINSI ; *FIN*	PROGRAM resultat ; VAR note :REAL ; BEGIN READE (note) ; IF note ≥ 10 THEN WRITE (‘'Admis") ; END.

L'alternative SI-ALORS-SINON

Syntaxe :	**Principe de fonctionnement :**
SI <condition> **ALORS** < action _alors> **SINON** < action _SINON> **FINSI**	**1 :** la condition est évaluée **2 : Si** la condition a la valeur **vraie** on exécute <**action_alors**> **Si** la condition a la valeur **fausse** on exécute <**action_sinon**>

Organigramme : **Altérnative SI_ALORS_SINON**



ALGORITHME	PROGRAMME
ALGO resultat ; VAR note :REEL ; DEBUT LIRE (note) ; SI note ≥ 10 ALORS ECRIRE(‘'Admis") SINON ECRIRE(‘'Ajourné") ; FINSI ; *FIN.*	PROGRAM resultat ; VAR note :REAL ; BEGIN READE (note) ; IF note ≥ 10 THEN WRITE (‘'Admis") ; ELSE WRITE (‘'Ajourné") ; END.

Remarque :

Les <action_alors> ou <action_sinon> peuvent être soit :

- des **actions élémentaires**
- des **actions composées** (bloc)

Tests imbriqués:

Dans les cas précédents on a seulement deux situation possibles (jour et nuit) , (admin, non admis) mais si il y'a plusieurs situations par exemple dans la mention (non admin, passable, assez bien, bien, très bien) une structure alternative simple ne suffit pas, donc pour traiter ce cas on doit passer par plusieurs structures alternative on parle (test imbriqué)

Syntaxe :

Si <condition 1> alors

<Instruction1>

Sinon Si <condition 2> alors
<Instruction1>
...
Sinon
<Instruction1>
<Instruction N>
Fin Si

Exercice1: Ecrire un programme qui calcule et affiche la racine carrée d'un nombre.
Solution:

```
program racine ;
uses crt;
var x,r:real;
begin
clrscr;
write('x=');
read(x);
if x>0 then
begin
r:=sqrt(x);
write('r=',r);
end
else if x=0 then
write('racine nulle')
else
write('error');
//commentaire
readln;readln;end.
```

Exercice 2 : Ecrire un algorithme qui permet de tester le signe d'un nombre .

Solution : Dans le cas de signe d'un nombre on a trois situation : (négatif, nul, positif).

ALGORITHME	PROGRAMME

Algo signe_nombre :
Var nb: entier ;
Début
Ecrire ('Saisir un nombre : ') ;
Lire(nb) ;
SI nb<0 ALORS
Ecrire('le nombre est négatif ')
SINON SI nb > 0 ALORS
Ecrire('le nombre est positif ')
SINON
Ecrire('le nombre est nul').
FIN SI
Fin.
	PROGRAM signe_nombre :
	VAR nb: INTEGER ;
	BEGIN
	WRITE ('Saisir un nombre : ') ;
	READE (nb) ;
	IF (nb<0)
	THEN WRITELN ('le nombre est négatif ')-
	ELSE IF (nb > 0) THEN WRITELN ('le nombre est positif ')
	ELSE WRITELN ('le nombre est nul') ;
	READLN;
	END.

Exercice 3: Ecrire un algorithme qui lit deux valeurs entières puis affiche la plus petite.
Solution :
Rq : min et minimum de deux nombres.
Algo min;
Var note1; note2, mini: entier;
DEBUT
ECRIRE ("entrez les 2 notes : ")
LIRE (note1) ;
LIRE (note2) ;
SI (note1 <= note2)
ALORS mini ← note1
SINON mini ←note2
FINSI
ECRIRE("minimum : ", mini)
FIN.
2) Minimum de 3 notes.
Algo min;
Var note1; note2, note3, mini: entier;
DEBUT
ECRIRE("entrez les 3 notes : ")
LIRE (note1, note2, note3)
SI (note1 <= note2) ET (note1 <= note3)
ALORS mini = note1
SINON SI (note2 <= note1) ET (note2<=note3)
ALORS mini ← note2
SINON mini ←note3 ;
FINSI ;
ECRIRE("minimum : ", mini) ;
FIN.

Exercice4 : Ecrire un algorithme qui permet de résoudre l'équation $ax^2 + bx + c = 0$ dans l'ensemble des réels (a,b et c NON NUL).

Solution :

L'algorithme de résolution de l'équation $ax^2 + bx + c = 0$ dans l'ensemble des réels est le suivant :

1. Calcul du discriminant, soit delta
2. Si delta > 0 alors il ya deux solutions données par les formules :
X1 = (-b + racine carrée(delta)) / 4ac
X1 = (-b - racine carrée(delta)) / 4ac
3. Si Delta = 0 alors il y a une solution double donnée par la formule :
X = -b / 2a
4. Si Delta < 0 alors il n'y a pas de solution dans l'ensemble des réels.

ALGORITHME	PROGRAMME

ALGO Equation
VAR A, B, C, DELTA : REEL
DEBUT
LIRE (A, B, C)
Delta := (B**2) – (4*A*C)
SI (Delta > 0) alors
Ecrire ('Première racine : ', (-B + Racine(Delta)) /(4* A*C)) ;
Ecrire ('Deuxième racine : ', (-B - Racine(Delta)) / (4 *A*C))
SINON
SI (Delta = 0) alors
Ecrire('Une racine double :', -B / (2*A))
SINON
Ecrire('Pas de racine réelle') ;
FSI
FSI
FIN
	PROGRAM Equation
	VAR A, B, C, DELTA : REAL
	BIGIN
	READ (A, B, C)
	Delta :=(B**2)- (4*A*C)
	IF (Delta > 0) THEN BEGIN
	WRITE ('Première racine : ', (-B + Racine(Delta)) / (4* A*C));
	WRITELN('Deuxième racine : ', (-B - Racine(Delta)) / (4 *A*C)) ;
	END
	ELSE
	IF (Delta = 0) THEN
	WRITELN('Une racine double :', -B / (2*A)
	ELSE
	Ecrire('Pas de racine réelle') ;
	END.

Exercice 5 : Écrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minime » de 10 à 11 ans
- « Cadet » après 12 ans

Solution 1	***Solution 2***
Algorithme categorie_enfant : Variable age : entier ; Début Ecrire ("Saisir l'age: ") ; Lire(age) ; SI age < 6 ALORS Ecrire("Saisir un age supérieur ou égal 6") SINON SI age <=7 ALORS Ecrire("Poussin") SINON SI age <=9 ALORS Ecrire("Pupille") SINON SI age <=11 ALORS Ecrire("Minime") SINON Ecrire("Cadet") FIN SI Fin	Algorithme categorie_enfant : Variable age : entier ; Début Ecrire ("Saisir l'age: ") ; Lire(age) ; SI(age < 6) ALORS Ecrire("Saisir un age supérieur ou égal 6") SINON SI(age >=6 ET age <=7) ALORS Ecrire("Poussin") SINON SI(age >=8 ET age <=9) ALORS Ecrire("Pupille") SINON SI (age >=10 ET age <=11) ALORS Ecrire("Minime") SINON Ecrire("Cadet") FIN SI Fin

Exercice 6: Écrire un algorithme qui à partir d'un nombre compris entre 1 et 7 affiche le jour correspondant .

Solution :

```
Algorithme   jour_semaine ;
  Variable jour: entier ;
Début
  Ecrire ("Saisir un jour : ") ;
  Lire(nb) ;


  SI jour=1 ALORS
     Ecrire("Lundi")
  SINON SI jour=2 ALORS
     Ecrire("Mardi")
  SINON SI jour=3 ALORS
     Ecrire("Mercredi")
  SINON SI jour=4 ALORS
     Ecrire("Jeudi")
  SINON SI jour=5 ALORS
     Ecrire("Vendredi")
  SINON SI jour=6 ALORS
     Ecrire("Samedi")
  SINON SI jour=7 ALORS
     Ecrire("Dimanche")
  SINON
     Ecrire("jour invalide")
  FIN SI
Fin.
```

Exercice 7: écrire l'algorithme qui affiche la valeur absolue de X

ALGORITHME	**PROGRAMME**

ALGO essai
VAR X,Y : REEL ;
DEBUT
LIRE(X) ;
SI X ≥ 0 ALORS
Y ← X
SINON
Y ← -X ;
ECRIRE(Y) ;
FINSI ;
FIN
	program essai
	VAR X,Y : REAL ;
	DEBUT
	RIDE(X) ;
	IF X ≥ 0 THEN
	Y := ← X
	ELSE
	Y := -X ;
	WRITE(Y) ;
	END.

Exercice8 : écrire un programme qui donne l'état de l'eau selon sa température doit pouvoir choisir entre trois réponses possibles (solide, liquide ou gazeux).

Une première solution serait la suivante :	Une autre solution consiste à imbriquer les tests de la manière suivante :
Algo eau Var Temp : entier Début Ecrire "Entrez la température de l'eau :" Lire Temp Si Temp =< 0 Alors Ecrire "Etat solide" FinSi Si Temp > 0 Et Temp < 100 Alors Ecrire "Etat liquide" Finsi Si Temp >= 100 Alors Ecrire "Etat gazeux" Finsi Fin	Algo eau Var Temp : entier Début Ecrire "Entrez la température de l'eau :" Lire Temp Si Temp =< 0 Alors Ecrire "Etat solide" Sinon Si Temp < 100 Alors Ecrire " Etat liquide" Sinon Ecrire "Etat gazeux" Finsi Finsi Fin

Structure à choix multiples SELON-QUE

La structure SELONQUE permet d'effectuer tel ou tel traitement en fonction de la valeur des conditions 1ou 2 ou ..n .

Syntaxe :	Méthode de Fonctionnement :
SELONQUE <condition 1> : <action 1> <condition 2> : <action 2> ... <condition n> : <action n> SINON : <action_sinon> FINSELONQUE	1 : la condition 1 est évaluée : • Si la condition 1 est vraie, alors on exécute l'action correspondante et on quitte la structure selon-que • Si la condition 1 est fausse, on évalue la condition 2...et ainsi de suite. 2 : Si aucune n'est vraie on effectue l'action sinon (au cas où l'action sinon n'existe pas alors aucune action n'est exécutée !).

Exemple:

SELONQUE
Note ≥ 16 : ECRIRE (‘'TB")
Note ≥ 14 : ECRIRE (‘'B")
Note ≥ 12 : ECRIRE (‘'AB")
Note ≥ 10 : ECRIRE (‘'Passable")
SINON : ECRIRE (‘'ajourné")
FINSELONQUE

Remarque:

En programmation, cette structure peut exister mais avec une forme ou un fonctionnement éventuellement différent. Si elle n'existe pas, il faut se souvenir que, en fait, SELONQUE est un raccourci d'écriture pour des **SI imbriqués**.

STRUCTURES RÉPÉTITIVES

On appelle boucle ou structure répétitives tout ensemble d'instructions qui peuvent être exécuté plusieurs fois. Par exemple, un programme de calcul de la paye répètera pour chaque employé les mêmes instructions pour établir une fiche de paye.

Idée : répéter un ensemble d'opérations, arrêter la répétition en fonction d'une condition.

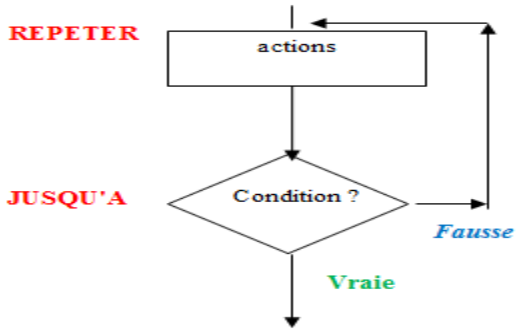
Généralement, on peut considérer deux cas pour les boucles. Dans le premier cas, le nombre de répétitions n'est pas connu ou est variable, il existe deux structures pour ce cas : la structure « Répéter… jusqu'à » et la structure « Tant que … faire ». Dans le deuxième cas, le nombre de répétitions est connu, la structure utilisée est « Pour ».

1. Structure REPETER

Syntaxe :	*Fonctionnement :*
REPETER *<actions simples>* ***JUSQU'A*** *<conditio>*	1. *on exécute le corps* 2. *on évalue la condition, puis* • *si la condition est vraie : on quitte le REPETER* • *si la condition est fausse on recommence*

Remarques :

Il y a toujours **au moins une exécution du corps**. La structure REPETER permet de répéter un traitement **1** ou **plusieurs fois**.



2. Structure TANTQUE

Syntaxe :

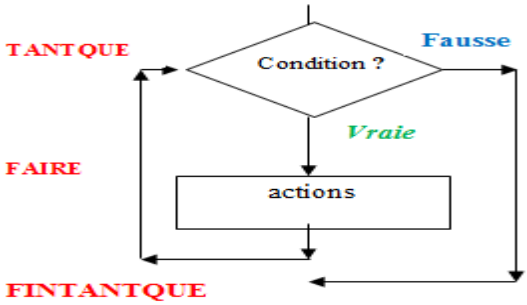
TANTQUE <condition>

FAIRE

<actions>

FINTANTQUE

Ces actions peuvent être simples ou composées ! !



Remarque :

- Le contenu de la structure TANTQUE **peut ne jamais être exécuté.** Donc cette structure permet en réalité de répéter un traitement 0, 1 ou plusieurs fois.
- La condition étant évaluée au début, les variables utilisées dans la condition doivent avoir été initialisées.
- On doit s'assurer de la terminaison (sinon le programme ne se termine jamais). **Pour cela, il faut nécessairement que dans le corps de la structure, la condition soit modifiée quelque part.**
  - Pour choisir entre REPETER et TANT QUE il faut se poser la question : faut-il éventuellement ne jamais faire le traitement ? Si oui : il faut utiliser tant que, sinon utiliser la structure REPETER qui exécute au moins une fois l'action.

  EXEMPLE : BOUCLE TANTQUE_REPETER

BOUCLE TANTQUE	BOUCLE REPETER
val ← -1 **TANTQUE** (val <=0) **FAIRE** ECRIRE (" ENTREZ un nombre positif !!! :') LIRE(val) **FINTANTQUE**	**REPETER** ECRIRE (" ENTREZ un nombre positif :!!!') LIRE(val) **JUSQU'À** (val >0)

EXEMPLE :Montrez l'évolution du contenu des variables x et y, puis déduire le résultat à l'écran du programme suivant :

```
program monwhile;
 var x,y : integer;
begin
x := 0; y :=1;
while (x <=2) do
begin
x := x + 1;
y := y + x;
end;
writeln ('La valeur finale de y est : ', y);
end;
```

SOLUTION

Evolution des variables x et y. x 0 1 2 3

y 1 2 4 7 Affichage à l'écran : La valeur finale de y est : 7

3. Structure POUR

Il est fréquent que le **nombre de répétitions** soit **connu** à l'avance, et que l'on ait besoin d'utiliser le numéro de l'itération afin d'effectuer des calculs ou des tests. Le mécanisme permettant cela est la **boucle POUR.** Cette boucle permet de **parcourir un intervalle** en répétant un traitement pour chacune des valeurs de cet intervalle

Syntaxe :

POUR <id_variable> **DE** <val_inférieure> **A** <val_supérieure>**FAIRE**

<actions>

FINPOUR

Exemple : (* Ecrit 4 fois "ok" à l'écran *)

Algorithme simple:	**L'équivalent avec la boucle POUR:**

program quatreok;
var i : integer;
begin
write('ok');
write('ok');
write('ok');
write('ok');
end.
	program quatreok;
	var i : integer;
	begin
	for i:=1 to 4 do
	write('ok');
	end.

Exercice N°1 :

Ecrire l'algorithme permettant d'afficher la table de multiplication par 9.

Algorithme simple:	**L'équivalent avec la boucle POUR:**
ALGORITHME TABLE_MULTI **DEBUT** ECRIRE(1*9) ECRIRE(2*9) ECRIRE(3*9) ECRIRE(4*9) ECRIRE(5*9) ECRIRE(6*9) ECRIRE(7*9) ECRIRE(8*9) ECRIRE(9*9) ECRIRE(10*9) **FIN**	**ALGORITHME** TABLE_MULTI **VAR** i : **ENTIER** **DEBUT** **POUR** i **DE** 1 **A** 10 **FAIRE** ECRIRE(i*9) **FINPOUR** **FIN**

Exercice N°2 :

Ecrire un programme qui demande à l'utilisateur une valeur de n et qui affiche le résultat de n!.

Solution avec la boucle POUR:	**L'équivalent avec la boucle while:**

program factorielle1;
var i, n: integer; fact : real;
begin writeln('Entrez un nombre entier');
readln (n);
fact := 1;
for i:=2 to n do
fact := fact * i;
writeln (n,'! =', fact :8:0);
end.
	program factorielle2;
	var i, n : integer; fact : real;
	begin writeln('Entrez un nombre entier');
	readln (n);
	fact := 1;
	i:=2;
	while (i <= n) do
	begin
	fact := fact * i;
	i := i + 1;
	end;
	writeln (n,'! =', fact:8:0);
	end.

Exercice N°3:

Demande d'un nombre à l'utilisateur et vérification qu'il respecte une condition imposée N entre 1 et 100:

Solution :

```
program nombre_1_100;
var  n: integer;
begin
writeln ('Tapez au clavier un nombre compris entre 1 et 100');
readln ( N );
while ( (N<1) ) OR (N>100) ) do
begin
 writeln ('Donnez-moi un autre nombre ! ');
readln ( N ); end ; end.
```

Exercice N°4:

Ecrire un algorithme permettant de calculer la valeur de l'expression E, telle que E=(1+2)x(1+2+3)x(1+2+3+4)x…x(1+2+3+…+(N-2)+(N-1)+N), et (N≥2).

Solution :

```
Algorithme SommeE ;
Var I,N,E,S :entier ;
Début
Lire(N) ;
E ←1 ;
S ←1;
Pour I ←2 à N Faire
S ←S+I;
E ←E*S;
Fin pour;
Ecire('E=',E) ;
Fin.
```

Exercice N°5 :

Ecrire un algorithme qui demande un nombre , calcule et affiche la somme $\sum_{i=1}^{n} i^3$.

Solution :

```
algorithme SommeCubique;
Var i, n, som : entier;
début
Ecrire (‘ Donnez n’);
Lire (n);
som ← 0;
 pour i=1 à n
      faire som ← som+i*i*i
finpour
  Ecrire (‘ La somme cubiques des n entiers est :’,som);
  fin
```

Exercice N°6 :

Ecrire un algorithme qui demande des nombres (nbVal) et fait la somme des nbVal données, arrêt à la lecture de -1.

Solution :

```
Algorithme FaitLeTotal ;
constante (STOP : entier) ←-1
variables val, totalValeurs: entiers
début
totalValeurs←0
afficher("Donnez une valeur,", STOP, " pour finir.") {amorçage}
saisir(val)
  tant que val ≠STOP faire
      totalValeurs←totalValeurs+ val {traitement}
      afficher("Donnez une autre valeur,", STOP, " pour finir.")
      saisir(val) {relance}
 ftq
afficher("La somme des valeurs saisies est", totalValeurs)
fin.
```

Exercice N°7 :

Ecrire un algorithme qui saisit des données et s'arrêter dès que leur somme dépasse 500. (solution avec les deux boucles répéter et tant que).

boucles répéter	boucles tant que
Algorithme saisir ; variables somme, val: entiers début somme ←0 répéter saisir(val) somme ←somme + val tant que somme ≤500 fin.	Algorithme saisir ; variables somme, val: entiers début saisir(val) somme ←val tant que somme ≤500 faire saisir(val) somme ←somme + val ftq.

Université de Mascara
Faculté des Sciences et de la Technologie
Département de Tronc Commun

TP Informatique 1

Exercice N°1 :

A. Ecrire un programme qui affiche à l'écran : "Bonjour".

(Vue en cours)

B. Ecrire un programme qui demande son prénom à l'utilisateur, et qui lui réponde par un charmant « Bonjour » suivi du prénom. On aura ainsi le dialogue suivant :

machine : Quel est votre prénom ?
utilisateur : Yacine
machine : Bonjour, Yacine !

Exercice N°2 :

Créer un algorithme pour calculer la somme, la différence, le produit, le quotient, le quotient entier et le reste d'un deux nombres entiers

Exercice N°3 :

Ecrire un algorithme qui permet d'échanger les valeurs de deux variables A et B.

Exercice N°4 : (devoir à rendre sur double feuille)

Ecrire un programme qui demande à l'utilisateur la valeur d'une durée exprimée en secondes et qui affiche sa correspondance en heures minutes secondes.
Ex : 3800 s --> 1 heure3 minutes 20 secondes.

Université de Mascara
Faculté des Sciences et de la Technologie
Département de Tronc Commun

TP Informatique 2

Exercice N°1 :

Ecrire un programme qui calcul la racine d'un nombre x.

Exercice N°2 :

Ecrire un programme qui affiche le signe d'un nbr x (x>0 :positive, x<0 négative ; x=0 : nulle).

Exercice N°3 :

Ecrire un programme qui lit trois valeurs entières puis affiche la plus grande des trois nombre.

Exercice N°4 : (devoir à rendre sur double feuille)

Ecrire un programme qui saisit trois entiers a, b, c et déterminer dans R les racines de l'équation ax2 + bx + c = 0

Exercice N°5 :

Ecrire un programme qui demande un nombre :

- calcule et affiche le factoriel de ce nombre N !
- calcule et affiche la somme $\sum_{i=1}^{n} i^3$.

Exercice N°6 :

Ecrire un programme qui demande des nombres (nbVal) et fait la somme des nbVal données , arrêt à la lecture de -1.

Exercice N°7 :

Ecrire un programme qui saisit des données et s'arrêter dès que leur somme dépasse 500 (solution avec les deux boucles répéter et tant que)

.