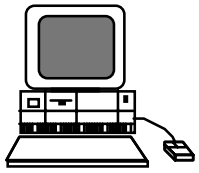
**FACULTE DES SCIENCES DE TUNIS**
**Département de Physique**
**Section : SP4**
**Module : EG22/Informatique**



# Programme

I. Informatique et architecture de l'ordinateur

II. Langages informatiques

III. Systèmes d'exploitation

IV. Introduction à l'algorithmique

1. Les constantes
2. Les structures conditionnelles
3. Les structures répétitives
4. Les chaînes de caractères et les tableaux
5. Les procédures et les fonctions

V. Programmation enTurbo Pascal

VI. Programmation en Fortran

# PREFACE

Dans la réforme des maîtrises instaurée ces dernières années, l'objectif du module Enseignement Général(EG22) de la maîtrise sciences physique (SP4) est de sensibiliser l'étudiant aux applications de cette discipline en sciences et technologies en plus des méthodes pédagogiques de communication.

Cet enseignement doit, en conséquence, assurer une formation permettant à l'étudiant l'intégration des différents concepts et lois déjà acquises par la stimulation de sa capacité d'analyse et de synthèse ainsi que de son sens critique.

Cet enseignement comprend trois parties :

1) La Chimie
2) La Physique
3) L'informatique

En ce qui concerne la partie informatique, nous avons opté pour un enseignement permettant essentiellement :

∗d'apprendre à l'étudiant l'utilisation du micro-ordinateur en tant qu'outil de développement.

∗∗de maîtriser des méthodologies rigoureuses et adéquates pour la résolution des problèmes.

∗∗∗de sensibiliser à une culture informatique et à la technologie de l'information.

Ce cours d'initiation à l'outil informatique pour les physiciens est une introduction à l'informatique et adapté à l'enseignement de la Physique.

On commence par voir l'intérieur d'un ordinateur et l'historique de l'informatique dans un chapitre intitulé architecture de l'ordinateur.

Dans un deuxième chapitre, les langages informatiques sont introduit. Comment l'ordinateur fait passer un nombre de la base 10 (que nous utilisons), il le tranforme en base hexadécimal (base 16) puis en binaire (base 2). Pourquoi, il ne fait pas un passage direct vers le binaire. Comment, à travers le code ASCII, l'ordinateur convertit les lettres et symboles en chiffres pour pouvoir les manipuler (en binaire). Les langages évolués et spécialisés sont invoqués.

Les systèmes d'exploitations sont introduit en troisième chapitre. Quelques instructions MS-DOS et UNIX sont présentées à titre d'exemples.

L'Algorithmique est un chapitre important puisque pour résoudre un problème, il est conseillé de faire d'abord un algorithme. Puis vient la programmation en Turbo Pascal et Fortran. Les exemples sont d'ordre général et élémentaire comme la détermination du périmètre et de l'aire d'un rectangle; mais aussi appliqué à la Physique comme la détermination à partir de la méthode de Newton de la constante de Wien du rayonnement d'un corps noir.

En plus de la dérivation et intégration numérique (méthode de Gauss), nous avons voulu présenté un logiciel de calcul formel : MAPLE. Ceci a permis de trouver analytiquement quelques dérivées et intégrales utilisées courament en Physique, de représenter des fonctions en 2D et 3D, . . .

## Chapitre I Informatique et architecture de l'ordinateur

L'**informatique** est l'ensemble des disciplines scientifiques et des techniques spécifiquement applicable au traitement de l'information, effectué notamment par des moyens automatiques.
Informatique = Traitement automatique de l'information.
Un **ordinateur** est un outil qui peut remplir quatre *fonctions* principales :
- Saisie de l'information
- Mémorisation
- Calculs et traitements
- Restitution

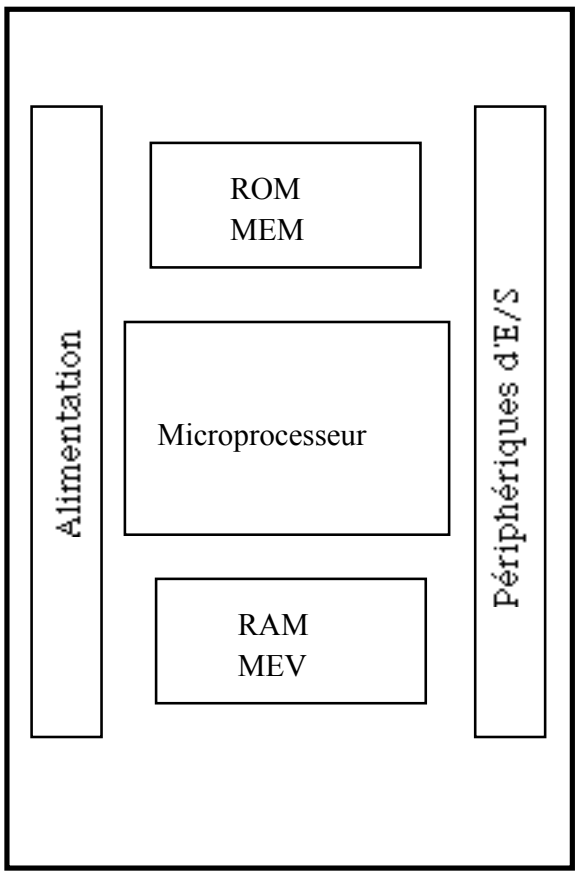
L'**ordinateur** est *constitué* de :
- Unité Centrale de Traitement (Central Processing Unit : C.P.U.)
- Mémoire centrale
- Périphériques d'entrées et sorties

En 1961 : D.E.C. ---> mini-ordinateur
En 1972 : Intel ---> micro-ordinateur contenant un microprocesseur.

Le **matériel** (Hardware) est le côté purement électronique de l'informatique.
Le **logiciel** (Software) est un programme ou un ensemble de programmes (suite d'opérations appelées instructions). mis à la disposition de l'utilisateur et qui appliqué à un matériel donné (Hardware) permet d'utiliser celui-ci.
- **Logiciels de base** ; il comprend :
  - -système d'exploitation (S.E.)
  - -logiciel de communication
  - - logiciels de gestion des périphériques d'E/S.

-**Logiciels d'application**
-**Progiciels** : C'est un programme ou une série de programmes (package) conditionné sous une forme prête à l'utilisation en vue d'applications spécifiques. Exemples : progiciels de traitement de textes, tableurs, gestion de bases de données.
Le **microprocesseur** :
C'est un circuit intégré de quelque milliers de transistors et composants. Il comprend quatre parties principales :
- Unité Arithmétique et Logique (Arithmetic and Logic Unit)
-Registres (servant de mémoire temporaire)
-Décodeur (servant à décoder et préparer les instructions pour l'exécution).
-Circuit de séquencement (pilote le fonctionnement par une horloge)
  (1 ms ---> 100 Hz et 1MHz= $10^6$ Hz )

*Un microprocesseur ne peut être commander qu'avec un langage binaire : c'est la représentation digitale de l'information.*
Le microprocesseur est piloté par une horloge, il détermine le nombre d'instructions élémentaires pouvant être exécuter par seconde; plus ce nombre est élevé plus les programmes s'effectueront rapidement apportant confort et souplesse à l'utilisateur.
L'horloge est caractérisée par sa fréquence en MHz.
MEM (MEmoire Morte) = R.O.M. (Read Only Memory)
MEV (MEmoire Vive) = R.A.M. (Random Acces Memory)


Alimentation
ROM
MEM
Microprocesseur
RAM
MEV
Périphériques d'E/S

## Chapitre II Les langages informatiques

II.1/ Caractéristiques d'un langage

a/ L'alphabet : c'est l'ensemble des signes utilisables pour représenter ou codifier une information.
L'alphabet est numérique s'il n'utilise que des chiffres.
L'alphabet est alphabétique s'il n'utilise que des lettres.
L'alphabet est alphanumérique s'il utilise des chiffres et des lettres.
b/ Unités syntaxiques : Les mots du langage = regroupement de caractères d'un alphabet. On parle d'orthographe ou de syntaxe. Exemples de faute d'orthographe ou de syntaxe : "Alfabé" ou "unprime"
c/ Règles syntaxiques : "grammaire".
Les mots sont regroupés en lignes d'instructions (phrases). En informatique, ces lignes d'instructions sont regroupées en programmes selon des règles de grammaire. Exemples de fautes de règles syntaxiques : "va école Ali à" ou "toto txt print dir".

II.2/ Les différents niveaux de langages

a/ Langage machine ou langage binaire :
C'est un langage numérique, base 2. Un digit binaire = bit (Binary digit) 0 ou 1.
Un octet = 8 bits (Byte).
Conversions de la base décimale à la base binaire :
$(29)_{10} = 2x10^1 + 9x10^0 = 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = (11101)_2$
$(2,25)_{10} = 2x10^0 + 2x10^{-1} + 5x10^{-2} = 1x2^1 + 0x2^{-1} + 1x2^{-2} = (1,01)_2$

b/ Langage hexadécimal ou base 16 :
En base 2, on utilise deux symboles 0 et 1, en base octale ou base 8, on utilise huit symboles (de 0 à 7), en base décimale ou base 10, on utilise dix symboles (de 0 à 9) et en base hexadécimal ou base16, on utilise seize symboles (de 0 à 9 et de A à F).
$(1B)_{16} = 1x16^1 + 11 = 16 + 11 = (226)_{10}$ et $(C)_{16} = (12)_{10} = (1100)_2$

Exercice n°1:

a/ Convertir 13; 7; 21; 25 et 45 de la base décimale vers les bases binaire et héxadécimale.

b/ Convertir 32; A2; 1AB; 2F et BAC de l'hexadécimale vers le binaire.

Exercice n°2:

a/ Calculer en hexadécimal : 125+698; 2AF-1C3 et 948x3

b/ Calculer en binaire : 101101+11011; 11011-1101 et 11001x110

Exercice n°3:

a/ Convertir 0,1011 et 0,1101 en $b_{10}$.

b/ Convertir 0,45 et 1,32 en $b_2$.

| Décimal (base 10) | Binaire (base 2) | Hexadécimal (base 16) |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

$(1\ 1011\ 1101)_2 = (1BD)_{16}$

Langage ASCII (American Standard Code for Information Interchange).
Le code ASCII de "A" est 65 en décimal ou 41 en hexadécimal.

c/ Langage assembleur (langage mémorique)

L'addition sur un microprocesseur est codé par (1100 0110) en binaire, (C6) en hexadécimal ou "ADD" en assembleur.
Le programme d'assemblage ou assembleur est un programme de traduction résident dans la mémoire de l'ordinateur.

d/ Langages évolués

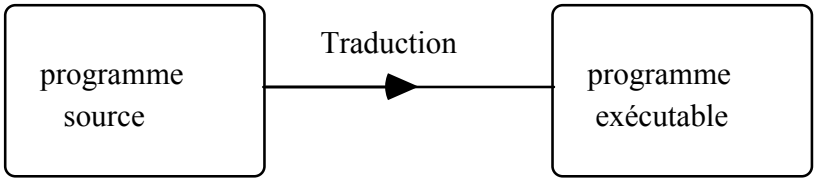
-FORTRAN de FORmula TRANslator (1956)
-COBOL de COmmercial and Business Oriented Langage (1959)
-BASIC de Beginner's All prupose Symbolic Instruction Code (1965)
-PASCAL adapté à la programmation structurée (1970)

e/ Langages spécialisés

Analyses statistiques (Statview)
Traitement de textes (Word)
Tableurs (Excel)
Calculs symboliques (mathematica, matlab, derive, Maple, . . . )

## II.3/ Compilateurs et interpréteurs

Un programme en assembleur ou écrit en langage évolué ou spécialisé ; seul le langage binaire est compris par l'ordinateur.
Il doit donc être traduit en langage machine par un programme spécifique appelé compilateur ou interpréteur dans certains cas. Ce programme spécifique est lié au langage évolué concerné et à l'ordinateur au quel il est destiné; par exemple le Fortran. Pour un langage évolué donné, il existe autant de compilateurs que de type d'ordinateurs.
Le traducteurs a pour rôle d'analyser, de décoder, d'interpréter et de générer des suites de zéro et de un; seuls informations assimilables par l'ordinateur.
Le traducteur travaille à partir d'un programme écrit par un utilisateur en langage évolué appelé programme source




Le compilateur produit à partir d'un programme source, un programme objet (en langage machine) pouvant être effectivement exécuter globalement.
L'interpreteur est un programme qui permet une exécution immédiate d'un programme source sans la fabrication d'un programme objet. Il traite et exécute une ligne d'instruction du programme source à la fois.

# Chapitre III Systèmes d'exploitation

## III.1/Les différents S.E.

Le S.E. est un ensemble de programmes qui réalise l'interface entre le matériel de l'ordinateur et les utilisateurs. Il a deux fonctions essentielles :
D'une part il construit sur la machine physique telle qu'elle est livré par le constructeur, une machine virtuelle plus facile d'emploi et plus conviviale pour l'utilisateur.
D'autre part, il prend en charge la gestion des ressources de l'ordinateur (processeur, mémoire, périphériques d'entrée-sortie, . . . ), en optimise l'utilisation et en permet le partage entre les utilisateurs.
Le S.E. est le premier programme introduit dans la machine, sans lui, la machine ne saurait rien exécuter. Le S.E. est en effet le coordinateur de tous les travaux réaliser par l'ordinateur. Il gère le fonctionnement des différents éléments de l'ordinateur. Il gère les fichiers stockés sur le disque dur en permettant de les classer, d'en afficher la liste sur écran ou imprimante, de les copier sur disquette, de les effacer, . . .. Il permet aux logiciels et autres programmes qu'on utilise de communiquer avec l'unité de traitement et les autres éléments de l'ordinateur.

## II.2/ Exemples de S.E. :

M.S. DOS : MicroSoft Disk Operating System.( Avec l'utilisation de windows 95, le DOS n'est plus une nécessité et les instructions du S.E. sont devenues visuelles et non sous forme d'instructions qu'on écrit)
UNIX: Un S.E. utilisé sur des stations de travail
MacOS : Le S.E. des Macintosh.( Il est visuel depuis sa naissance ! )
VMS/VAX : Un S.E. de grosses machines.

## III.3/ Instructions MS-DOS

Bien que le système MS-DOS n'est plus nécessaire dans les machines récentes, nous présentons l'essentiel des instructions ce qui permet de comprendre les fonctionnalitées d'un S.E.
a/ Commandes internes :
Elles déclenchent l'exécution d'un programme résident en mémoire vive (RAM). L'ensemble des programmes associés aux commandes internes est contenu dans la partie command.com du S.E. qui est chargé en mémoire au lancement.
Les principales commandes internes sont : date, time, copy, mkdir, chdir, rmdir, path, append, . . .
b/ Les commandes externes :
Les programmes correspondant aux commandes externes sont contenus dans des fichiers du disque avec l'extension .exe ou .com L'appel d'une commande externe provoque le chargement du programme correspondant en mémoire puis son exécution.
Les principales commandes externes sont : format, xcopy, diskcopy, chkdsk, tree, backup, restore, . . .

III.4/ Gestion des fichiers et des répertoires :

a/ copie de fichier ( copy a.for b.for )
b/ liste triée des fichiers ( tree )
c/ renommer des fichiers ( rename a.for b.for )
d/ suppressions de fichiers (del a.for )
e/ visualisation de fichiers ( edit a.for )
f/création de répertoires ( md eg22 )
g/ liste des répertoires ( tree )
h/ changement de répertoires ( cd eg22 ou cd.. pour revenir à la racine )
i/ suppression de répertoire vides ( rd eg22 )

III.5/ Gestion des disques et disquettes :

a/ copie de disquette ( diskcopy )
b/ formatage de disquettes ( format a: )
c/ vérification d'une disquette ( chkdsk c: )

III.6/ L'éditeur "edit"

Cet éditeur sert à créer ou modifier un fichier qui contient des lignes de texte.
Exemple D:\> edit add.for

III.7/ Les fichiers de commandes :

Ce sont des fichiers dont l'extension est .bat
On crée un fichier par edit x.bat dont les lignes sont des instructions que l'ordinateur exécute une à une. Les paramètres des fichiers de commandes sont notées %0, %1, . . %9.
Le fichier de commande t.bat par exemple, permet par la simple instruction t add par exemple de compiler le fichier add.for, de l'exécuter et d'éliminer les fichiers crées lors de la compilation.

# Chapitre IV Introduction à l'algorithmique

## IV.1/ Définitions

Un **problème** est un énoncé pour lequel il existe une ou plusieurs solutions.

Un **processeur** est une entité capable de comprendre un énoncé et d'exécuter le travail indiqué.

L'**environnement** est l'ensemble des objets nécessaires à l'exécution d'un travail.

Une **action** est un événement qui modifie l'environnement. Pour un processeur donné, une action est primitive si l'énoncé de cette action est à lui seul suffisant pour qu'il puisse l'exécuter sans informations supplémentaires. Dans le cas contraire, l'action est dite composée.

Etant donné un processeur bien défini et un traitement à exécuter par ce processeur, un **algorithme** du traitement est l'énoncé d'une séquence d'actions primitives réalisant ce traitement. L'algorithme est donc une succession d'opérations ou d'actions qui permet de faire passer l'environnement de l'état initial à l'état final désiré.

Un algorithme est aussi la description du processus permettant d'atteindre un but précis. Le mot algorithme provient de la latinisation du nom du mathématicien arabe Al Khawarizmi (VIIIe-IXe siècle), a qui l'on doit aussi la fonction mathématique logarithme.

Un **organigramme** est la représentation avec des tableaux de l'algorithme trouvé.

Un **programme** est une traduction dans un langage donné de l'algorithme permettant de répondre à l'étude recherchée.

La difficulté de la recherche vient du fait qu'il n'y a pas de méthode permettant d'obtenir pour tout problème un algorithme efficace.
Avant d'utiliser un micro-ordinateur pour la résolution d'un problème donné il faut d'abord dégager ce que le programme doit faire puis séparer le problème en trois parties :
* une phase de lecture
** une phase de calcul
*** une phase d'écriture ou impression
Il est donc conseillé de définir d'abord l'environnement du problème (variables d'entrées, variables de sorties, . . . ).
Un **objet** est considéré comme une boîte, portant une étiquette son nom, d'une certaine forme : son type et qui contient une information sa valeur.

Un objet est considéré comme un objet-donné d'un algorithme lorsque sa valeur initiale est

utilisée par l'algorithme et reste inchangée après exécution de l'algorithme.

Un objet est considéré comme un objet-résultat si sa valeur initiale est sans signification

pour l'algorithme, mais l'exécution de l'algorithme affecte à un tel paramètre une nouvelle

valeur.

Un objet-Donné-Résultat est un objet dont la valeur initiale est utilisée par l'algorithme

mais remplacée en fin d'exécution par une valeur finale.

IV.2/ Types élémentaires d'une variable :

Les **types élémentaires** sont :

a- Type numérique : C'est l'ensemble des valeurs numériques destinées à tous les calculs.

Elles s'écrivent sous forme entière, décimale ou en notation scientifique.

b- Type alphanumérique : C'est une chaîne de caractères formée de lettres, de chiffres et de

signes spéciaux mais considérés comme texte.

c- Type logique : C'est un type qui ne peut prendre que l'une des deux valeurs suivantes :

vraie ou faux.

IV.3/ Structures élémentaires en algorithmique :

1. Les constantes

a/ Entrée d'une donnée : C'est l'instruction : Lire

b/ Sortie d'une donnée : C'est l'instruction Ecrire

c/ Action d'affectation : C'est l'instruction reçoit ( ←)

Une variable est une position de la mémoire qui peut conserver une valeur. Si x est un nom de variable alors x représente une position dans la mémoire. Si x est une variable de type Integer, par exemple, la position dans la mémoire x ne peut contenir que des valeurs entières ( 4, 30, 89, . . . ).

Pour attribuer une valeur à un objet, on utilise donc le symbole reçoit :

nomobjet ← valeur

nomobjet est le nom de l'objet auquel le processeur doit attribuer une valeur, valeur est la valeur à affecter qui peut être une constante, une expression ou le nom d'un autre objet précédemment définie.

Exemple d'algorithme simple :

*Algorithme du double :*

```
var n, m : entier
début
    Lire n
    m ← 2*n
    Ecrire m
```

Fin

2. Les structures conditionnelles

a/ Structure conditionnelle simple :

Si <condition>

alors

début

<traitement>

fin

ou

Si <condition>

alors

début

<traitement1>

sinon

début

<traitement2>

fin

b/ Structure conditionnelle à choix multiples :

selon <selection> faire

<valeur1> : <traitement1>

<valeur2> : <traitement2>

" : "

<valeurn> : <traitementn>

sinon

<traitement>

fin

Exemple d'algorithme à choix multiple :

*Algorithme du jour de la semaine :*

var numerojour : entier

début

Ecrire('Donner le numéro du jour')

Lire (numerojour)

selon numerojour faire

1 : Ecrire('C"est un Lundi')

2 : Ecrire('C"est un Mardi')

" : " " "

7 : Ecrire('C"est un Dimanche')

sinon

Ecrire('Le numéro du jour donné est incorrect !')

fin

fin

3. Les structures répétitives

a/ Schéma répéter : La forme générale de la boucle "répéter-jusqu'à" peut être donnée de la façon suivante :

Répéter

<traitement>

Jusqu'à <condition>

b/ Schéma Pour :

Sa forme générale est :

Pour I de <vi> à <vf> Faire

début

<traitement>

fin

I est la variable de contrôle (compteur du type scalaire, ou indice), <vi> sa valeur initiale et <vf> sa valeur finale.

Cette boucle exprime la répétition d'un traitement. la condition de maintien de la boucle est exprimée par le fait que I doit être rester compris entre <vi> et <vf>. Le compteur I est incrémenté de un à chaque exécution du traitement. L'exécution s'arrête quand le contenu du compteur atteint la valeur finale. Dans un schéma Pour, le processeur prend lui-même en charge la variation de la variable de contrôle et l'arrêt de l'itération.

c/ Schéma Tant que :

Sa forme générale est :

Tant que <condition> faire

début

<traitement>

fin

4. Les chaînes de caractères et les tableaux

a/ Déclaration d'une chaîne de caractère.

Une variable du type chaîne de caractères sera noté du type chaîne :

**var nom : chaîne**

b/ Longueur d'une chaîne de caractère.

La fonction long détermine la longueur d'une chaîne de caractères c'est à dire son nombre de caractères. (l est le nombre de caractère de la chaîne nom) :

**l ← long(nom)**

c/ Concaténation de deux chaînes de caractère.

La concaténation est la réunion de deux ou plusieurs chaînes :

**ch ← concatenation(ch1,ch2)**

d/ Partie d'une chaîne de caractère.

La fonction copie extrait une partie d'une chaîne de n caractères à partir de la position p :

**ch ← copie(chaine,p,n)**

e/ Position d'une chaîne dans une autre.

L'instruction position retourne un entier qui est la position, la première fois, de la chaîne ch1 dans la chaîne ch. Si ch1 n'est pas trouvée, la valeur affectée à position est zéro.

**p ← position(ch1,ch)**

Les Tableaux

1°) Définition

Un tableau est une structure de données constituée d'un nombre fini d'éléments de même type.

2°) Déclaration

var T : tableau [$v_i$ . . $v_f$] de type-élément

$v_i$ est l'étiquette de début

$v_f$ est l'étiquette de fin

Exemples :

En Algorithmique : T : tableau [1 . . 15] de réels

En Turbo Pascal : T : array [1 . . 15] of reals;

En Fortran : dimension T(15)

5. Les procédures et les fonctions

Ce sont des parties de programmes indépendantes, qu'on appelle chaque fois qu'on a besoin.(voir exemples)

## Chapitre V Programmation en Turbo Pascal

Le langage Pascal a été développé et mis au point en 1970 par le professeur Niklaus Wirth et ses collaborateurs à l'Ecole Polytechnique de Zurich.

### V.1/ Les touches rapides de Turbo Pascal

[F1] Fait apparaître une fenêtre d'aide avec des renseignement sur votre position actuelle.

[F2] Sauvegarde le fichier contenu dans l'éditeur.

[F3] Vous permet de charger un fichier.

[F5] Réduit/Agrandit la taille de la fenêtre active.

[F6] Change de fenêtre active.

[F9] Lance une compilation sélective.

[F10] Appelle le menu principal.

[Alt][F1] Fait apparaître le dernier écran d'aide demandé.

[Alt][F3] Permet de choisir un fichier à charger.

[Alt][F5] Vous ramène vers l'écran depuis lequel T. Pascal a été lancé.

[Alt][F9] Compile votre programme.

[Alt][F6] Passe à la fenêtre suivante.

[Alt][R] Exécute votre programme.

[Alt][X] Quitte le T. Pascal.

### V.2/ Les commandes de l'éditeur

a/ Commandes de déplacement

[Ctrl] S caractère à gauche.

[Ctrl] D caractère à droite.

b/ Commandes d'insertion et d'effacement :

[Ctrl] N insertion d'une ligne.

c/ Commandes de bloc :

d/ Commandes de recherche/remplacement :

e/ Commandes diverses :

V.3/ Structure simplifiée d'un programme Turbo Pascal :

```
program calcul;
{c'est un programme qui calcule . . . }
uses crt;
var {déclaration des variables};
        procedure proc(parametres);
        var {déclaration des variables locales};
        begin
        {Bloc principale de la procédure};
        end;

        function func(parametres);
        begin
        {bloc de la fonction};
        end;
```

```
begin
{bloc principal du programme avec appel des procédures et fonctions};
end.
```

V.4/ Les types scalaires prédéfinies en T. Pascal :

($2^{16}$-1= 32767 et $2^{38}$-1=2147483647)

- Types entiers

| | |
|---|---|
| shortint | -128 . . 127 |
| integer | -32768 . . 32767 |
| longint | -2147483648 . . 2147483647 |
| byte | 0 . . 255 |
| word | 0 . . 65535 |

- Types réels

| | |
|---|---|
| real | $2.9\ 10^{-39}$ . . $1.7\ 10^{38}$ |
| single | $1.5\ 10^{-45}$ . . $3.4\ 10^{44}$ |
| double | $5.0\ 10^{-324}$ . . $1.7\ 10^{308}$ |
| extended | $1.9\ 10^{-4951}$ . . $1.1\ 10^{4932}$ |

- Type boolean

Il prend deux valeurs True (pour vrai) et False (pour faux).

- Type char

Type caractère. Exemple :

```
var nom : char
```

```
begin
nom:='ali';
end.
```

V.5/ Les opérateurs standards du T. Pascal :

| | | | | | |
|---|---|---|---|---|---|
| **:=** | affectation | > | supérieur | **and** | et logique |
| = | égalité | < | inférieur | **or** | ou logique |
| + | addition | >= | supérieur ou égal | **not** | non logique |
| **-** | soustraction | <= | inférieur ou égal | **xor** | ou exclusif |
| * | multiplication | <> | différent | **in** | est un élément de |
| / | division | **mod** | modulo | **div** | division entière |

V.6/ Les fonctions standards du Turbo Pascal :

| | |
|---|---|
| abs(x) valeur absolue | int(x) partie entière |
| cos(x) cosinus de x | sin(x) sinus de x |
| pi valeur de π | arctan(x) arc tangente de x |
| exp(x) exponentielle de x | ln(x) logarithme népérien de x |
| sqrt(x) racine carré de x | sqr(x) le carré de x |
| int(x) partie entière de x | frac(x) x-int(x) |
| round(x) arrondi à la valeur la plus proche | trunc(x) le même résultat que int(x) mais le résultat est longint |

randomise initialise le générateur des nombres aléatoires

random(n) retourne un réel aléatoire compris entre 1 et n (entier)

V.7/ Structures élémentaires en T. Pascal:

1. Les constantes

a/ Entrée d'une donnée : Readln(n);

b/ Sortie d'une donnée : Writeln(n);

c/ Action d'affectation : x:=a+1;

2. Les structures conditionnelles

```
if condition then
        begin
                traitement
        end;
```

ou

```
if condition then
        begin
                traitement1
        end
else
        begin
                traitement2
        end;
```

ou encore pour un choix multiple :

```
case selection of
        valeur1 : traitement1;
        valeur2 : traitement2;
                "     :    "
        valeurn : traitementn;
else
        traitement;
end;
```

3. Les structures répétitives

a/ Schéma répéter (repeat) :

```
repeat
        traitement;
until condition;
```

b/ Schéma Pour (For) :

```
for c:=vi to vf do
        traitement;
```

c/ Schéma Tant que (While) :

```
while condition do
begin
```

```
        traitement;
    end;
```

4. Les chaînes de caractères et les tableaux

a/ Déclaration d'une chaîne de caractère.

**var nom : string;**

b/ Longueur d'une chaîne de caractère.

**l:=length(nom);**

c/ Concaténation de deux chaînes de caractère.

**ch := concat(ch1,ch2);**

d/ Partie d'une chaîne de caractère.

**ch := copy(chaine,p,n);**

e/ Position d'une chaîne dans une autre.

**p := pos(ch1,ch);**

## Chapitre VI Programmation en Fortran

Le langage Fortran est surtout utilisé pour faire des calculs scientifiques.

VI.1/ Compilation et exécution d'un programme fortran :

La programmation en Fortran s'est bien développée. On utilise généralement Le compilateur Microsoft Fortran 90, sous windows. Il a un environnement intégré, ou on peut éditer, corrigé et compiler un programme. Ce qui est mis à la disposition des étudiants, dans ces salles d'informatique est une vieille version sous MS-DOS. On utilise, l'éditeur edit pour écrire le programme (edit x.for), et on compile et exécute par un bat t.bat (t x pour le programme x.for).

VI.2/ Structure simplifiée d'un programme Fortran :

```
      program calcul
c     c'est un programme qui calcule . . .
c     bloc principal
      stop
      end
      subroutine sub(paramètres)
c     bloc principal du sous-programme
      return
      end
      function func(paramètres)
c     bloc principal de la fonction
```

```
      return
      end
```

VI.3/ Les types scalaires prédéfinies en Fortran :

-Types entiers

integer * 2 -32768 à 32767

integer * 4 -2147483648 à 2147483647

-Types réels

real *4 $2.9\ 10^{-39}$ . . $1.7\ 10^{38}$

real * 8 15 chiffres significatifs

Exemples :

implicit integer * 2 (J,K)

implicit real * 8 (A-C,X-Z)

logical * 4, logical * 2 ou character * n

Le système du Fortran a des déclarations de types et de longueur par défaut. Cette longueur n'a donc pas besoin d'être indiquée. Le type d'une variable numérique est défini par la première lettre de son nom. Ce type est entier (integer) si cette lettre est I, J, K, L, M ou N. Il est réel (real) si la première lettre n'est pas l'une de ces lettres.

VI.4/ Les opérateurs standards du Fortran :

| | | | | | |
|---|---|---|---|---|---|
| = | affectation | **.GT.** | supérieur à | **and** | et logique |
| **.EQ.** | égalité | **.LT.** | inférieur à | **or** | ou logique |

| | | | | | |
|---|---|---|---|---|---|
| + | addition | **.GE.** | supérieur ou égal | **not** | non logique |
| **-** | soustraction | **.LE.** | inférieur ou égal | **xor** | ou exclusif |
| * | multiplication | **.NE.** | différent | **in** | est un élément de |
| / | division | **mod** | modulo | **div** | division entière |

VI.5/ Les fonctions standards du Fortran :

| | |
|---|---|
| abs(x) valeur absolue | int(x) partie entière |
| cos(x) cosinus de x | sin(x) sinus de x |
| pi valeur de π | arctg(x) arc tangente de x |
| exp(x) exponentielle de x | alog(x) logarithme népérien de x |
| sqrt(x) racine carré de x | x**2 le carré de x |
| int(x) partie entière de x | frac(x) x-int(x) |
| round(x) arrondi à la valeur la plus proche | trunc(x) le même résultat que int(x) mais le résultat est longint |
| randomise initialise le générateur des nombres aléatoires | random(n) retourne un réel aléatoire compris entre 1 et n (entier) |

VI.6/ Structures élémentaires en Fortran :

1. Les constantes

a/ Entrée d'une donnée : Read(a,b)n

où a est l'unité de lecture, * par défaut (le clavier) et b le format de lecture, * par défaut.

b/ Sortie d'une donnée : Write(a,b)n

où a est l'unité d'écriture, * par défaut (l'écran)et b le format de d'écruture, * par défaut.

c/ Action d'affectation : x=a+1

2. Les structures conditionnelles

Le fortran a un if logique :

```
if expression logique then traitement1
```

ou

```
if expression logique then
    traitement1
else
    traitement2
end if
```

et un if arithmétique :

```
if (expression arithmétique) n1, n2, n3
```

Il y a transfert à l'étiquette n1, si le résultat est négatif, à n2 s'il est nul et à n3 s'il est positif.

3. Les structures répétitives

a/ Schéma Faire (Do standard) :

```
    Do n var = vi, vf
    traitement
n   continue
```

b/ Schéma Faire (Do-end do) :

```
    Do var = vi, vf
    traitement
```

end do

c/ Schéma Faire Tant que (Do While) :

Do while condition

traitement

End do

4. Les chaînes de caractères et les tableaux

a/ Déclaration d'une chaîne de caractère.

**character * nom**

b/ Longueur d'une chaîne de caractère.

**l=len(nom)**

c/ Concaténation de deux chaînes de caractère.

**ch = ch1//ch2**

d/ Partie d'une chaîne de caractère.

**ch = chaine(p:p+n)**

e/ Position d'une chaîne dans une autre. **p = index(ch1,ch)**

# Exemples de programmes en Turbo Pascal :

```
program age;
var a : integer;
begin
writeln('Donnez votre age'); readln(a);
case a of;
   0 . . 15 : writeln('Vous êtes très jeune');
  16 . . 30 : writeln('Vous êtes encore jeune');
  31 . . 40 : writeln('Vous adulte');
```

```
  41 . . 50 : writeln('ça va encore ! ! ');
  51 . . 61 : writeln('Vous êtes un peu vieux');
  61 . . 70 : writeln('Etes-vous Hadj ? Il est temps !);
  71 . . 90 : writeln('Bonne santé !  !');
  91 . . maxint : writeln('Menteur ! ! ');
  end;
end.

program fact;
uses crt;
var x : integer;
function fact(n : integer ) : real ;
begin
if n=0 then fact := 1
else fact := n*fact(n-1);
end;
begin
clrscr;
goto(20,1);writeln('CALCUL DE FACTORIEL');writeln;
write('Donner un entier');readln(x);
writeln('Le factoriel de ',x,' est ',fact(x));
end.




program sumn
{Programme qui calcule s la somme des n premiers entiers}
        uses crt;
        var n, s, k : integer;
        begin
        clrscr;
        writeln('Donner n');
        readln(n);
        k:=0;
        s:=0;
```

```
        while k<=n do
        begin
        s:=s + k;
        k:=k + 1;
        end;
        writeln('n=',n:5,'s=',s:10);
        readln;
        end.

        program sumn
{Même programme que le précédent mais qui utilise la boucle for}
        uses crt;
        var n, s, k : integer;
        begin
        clrscr;
        writeln('Donner n');
        readln(n);
        s:=0;
        for k:=1 to n do
        begin
        s:=s + k;
        end;
        writeln('n=',n:5,'s=',s:10);
        readln;
        end.

program initiales;
uses crt;
var nom,prenom,inip,inin : string;
begin
clrscr;
writeln('Donner votre nom');
readln(nom);
writeln('Donner votre prénom');
readln(prenom);
inip:=copy(prenom,1,1);
inin:=copy(nom,1,1);
writeln('Vos ',upcase('i'),'nitiales sont ',inip,'.',inin);
readln;
end.
```

```
program conjugaison;
uses crt;
var
verbe, ver : string;
begin
clrscr;
write('Donner un verbe du premier groupe');
readln(verbe);
clrscr;
writeln('CONJUGAISON DU VERBE ',verbe,' au présent');
ver:=copy(v,1,length(verbe)-2);
writeln('je ',ver,'e');
writeln('tu ',ver,'es');
writeln('il ',ver,'e');
      if copy(ver,length(ver),1) = 'g' then writeln('nous ',ver,'eons') else
        writeln('nous ',ver,'ons');
writeln('vous ',ver,'ez');
writeln('ils ',ver,'ent');
readln;
end.

program integ_Gauss_Laguerre;
uses crt;
var
y : real;
function fn(x:real) : real;
begin
fn:=x*x*x/(exp(x)-1);
end;


begin
clrscr;
y:=0;
y:=y+1.07769*fn(0.41577); y:=y+2.76214*fn(2.29428); y:=y+5.60109*fn(6.2899);
writeln('le res est',y);
readln;
end.

program astro;
uses crt;
var
   jj, mm, aa, m, a, ca, cb, jd, tt, tto : real;
   nj, njd : longint;
begin
         clrscr;
         write('donnez le jour ');
```

```
        readln(jj);
        write('donnez le mois ');
        readln(mm);
        write('donnez l''année ');
        readln(aa);
            if (mm=1) or (mm=2) then
                begin
                   a:=aa-1;
                   m:=mm+12;
                end
            else
                begin
                   m:=mm;
                   a:=aa;
                end; jd:=int(365.25*a)+int(30.6001*(m+1))+jj+1720994.5;
if ((aa+mm/100+jj/10000) > 1582.1015) then
     begin
            ca:=int(a/100);
            cb:=2-ca+int(ca/4);
            jd:=jd+cb;
     end;
tt:=(jd-2415020.0)/36525;
tto:=0.276919398+100.0021359*tt+0.000001075*tt*tt; tto:=24*frac(tto);
njd:=trunc(jd);
nj:=njd mod 7;

case nj of
     0 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un mardi ');
     1 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un mercredi ');
     2 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un jeudi ');
     3 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un vendredi ');
     4 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un samedi ');
     5 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un dimanche ');
     6 : writeln('le',jj:2:0,'/',mm:2:0,'/',aa:4:0,' est un lundi ');
   else writeln('c''est bizare',nj);
     end;
writeln('Le Jour Julien est ',jd:8:2);
writeln('Le temps sidéral à Greenwich à 0h TU est ',tto);
readln;
end.


program motdepass;
   uses crt;
      const  vmp='ASTRONOMIE';
      var ne:         integer;
            mp:       string;
      begin
```

```
clrscr;
   ne:=0;
   repeat
      ne:=ne+1; write('Mot de passe= ');readln(mp);
   until (ne=3) or (mp=vmp);
if (mp=vmp) then writeln('oui,vous pouvez acceder a l''application ') else
writeln('Tu m''enerves ');
   readln;
   end.
```

# Exemples de programmes en Fortran :

```
      program add
      write(*,*)'Donner un premier réel'
      read(*,*)a
      write(*,*)'Donner un deuxième réel'
      read(*,*)b
      c=a+b
      write(*,*)c
      stop
      end

      program prodn
      write(*,*)'Donner un entier'
      read(*,*)n
      lp=1
      do 38 k=1,n
      lp = lp*k
38    continue
      write(*,43)n,lp
43    format(2x,'Le produit des ',I5,' premiers nombres est ',I9)
      stop
      end
```

nx dans format c'est pour laisser n blanc en début de la ligne (ici n=2, par exemple).

```
      program sca
      dimension A(100),B(100)
      write(*,*)'Donner la dimension de l''espace vectoriel'
      read(*,*)n
      write(*,*)'Donner les composantes du premier vecteur'
      do 10 k=1,n
      read(*,*)A(k)
10    continue
      write(*,*)'Donner les composantes du deuxième vecteur'
      do 20 k=1,n
      read(*,*)B(k)
20    continue
      sca=0.0
      do 30 k=1,n
      sca=sca + A(k)*B(k)
30    continue
      write(*,40)sca
40    format(1x,'Le produit scalaire est ',F8.2)
      stop
      end
```

En utilisant des fichiers de lecture et d'écriture, on peut faire le programme suivant :

```
      dimension v1(10),v2(10)
      open(1,file='scal.dat')
      open(2,file='scal.res')
      read(1,*)(v1(i),i=1,10)
      read(1,*)(v2(i),i=1,10)
      ps=0
      do 5 i=1,10
      ps=ps + v1(k)*v2(k)
5     continue
      write(2,*)ps
      stop
      end
```

```
      program wien
      z=5
      do 98 k=1,10
      f=z*exp(z) + 5*exp(-5)
      df=(1+z)*exp(z)
      z=z-f/df
      write(*,*)z
98    continue
      stop
      end

      program gauss
      gl=1.07769*f(0.41577)
      gl=gl + 2.76214*f(2.29428)
      gl=gl + 5.60109*f(6.2899)
      write(*,*)gl
      stop
      end
      function f(x)
      f=x*x*x/(exp(x)-1)
      return
      end
```

Examen de TP 1996-97

Exercice n°1:

```
VAR
X, MY : Reel
I: Entier
DEBUT
MY¨ 0
pour I allant de 1 à 10
      DEBUT
      lire (X)
      MY¨ MY+X/10
      FIN
Ecrire MY
FIN
```

1°/ Que fait cet algorithme ?.

2°/ Ecrire le programme turbo pascal correspondant

3°/

a) Tapez ce programme

b) Faire l'exécution pour les valeurs de X suivantes:

1.238001
6.529134
10.100389
5.99321
21.00389
16.98751
4.1054
1.1061
23.0002
20.0497

Exercice n°2:

```
X,Y,S : Entier
DEBUT
Lire X,Y
Si X>0
        Alors
                S ¨ 1
        Si non
                Si Y<0
                        Alors
                        S ¨ 1
                        Si non
                        S ¨ 0
                Fin Si
Fin Si
Ecrire S
```

1°/ Que fait cet algorithme ?

2°/ Ecrire le programme turbo pascal correspondant

3°/

a) Exécuter ce programme

b) Remplir le tableau suivant

| X | Y | S |
|---|---|---|

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Examen de rattrappage 1996-97

EXERCICE 1:
Qn donne l'algorithme suivant:

DEBUT

Lire X, Y, Z

SAUVE ← X

X ← Z

Z ← Y

Y ← SAUVE

Ecrire X, Y, Z

FIN

1°/ Que fait cet algorithme.

2°/ Ecrire le programme turbo pascal correspondant

3°/ Faire l'exécution pour X=10, Y=9, Z=8

EXERCICE 2:

1) Ecrire un algorithme qui permet de calculer la moyenne d'une série de 6 entiers
2) Ecrire le programme turbo pascal correspondant
3) Faire l'exécution pour la série suivante :

19 , 524, 1291 , 2301 ,99 201

Première édition Mars 1998