UNIVERSITE PIERRE ET MARIE CURIE

Licence de Chimie UE-209

INTRODUCTION A LA PROGRAMMATION FORTRAN

Plan :

1 : Introduction

2 : Structure d'un programme Fortran

3 : Aiguillages

4 : Processus itératifs

5 : Variables indicées

6 : Sous-programmes

Contrôle des connaissances :

CC N°1 (*10 min - 5pts*) pendant le deuxième cours

CC N°2 (*10 min - 5pts*) pendant le troisième cours

Examen (*1h - 30pts*) pendant le quatrième cours

Objectifs

L'objectif de cette UE est d'approfondir les connaissances des étudiants dans les objets et concepts mathématiques de base utilisés dans les différents domaines de la chimie. L'enseignement est donc axé sur les besoins des différents enseignements dispensés en L2 et L3 de la mention chimie, dont seront tirées les principales applications. Les concepts sont illustrés et complétés par des enseignements d'informatique qui vont familiariser les étudiants avec des notions de Linux et de Fortran. Ces notions d'informatique permettront ensuite aux étudiants lors de leurs études (L3 et Master) d'aborder l'informatique appliquée aux sciences chimiques.

Conventions utilisées

Le texte écrit comme ceci est un message affiché sur l'écran ou un texte à taper au clavier

Le texte écrit *«comme ceci»* est un texte que vous devez adapter selon le contexte.

la touche ENTER (entrée) est indiquée par le signe ↵

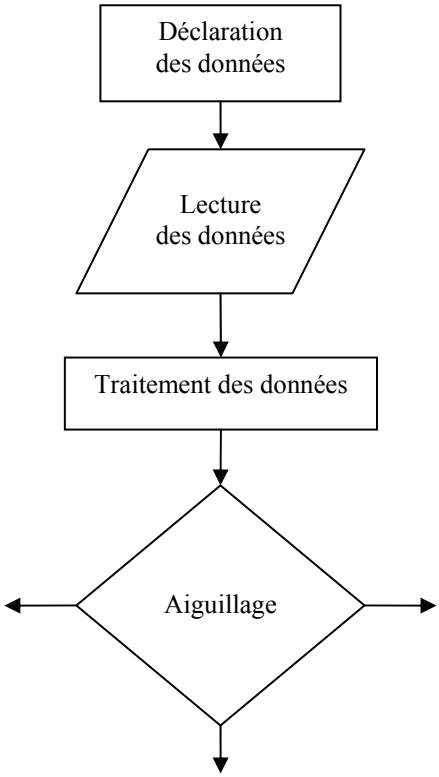
le caractère espace est indiqué par le signe ^

- COURS 2-

STRUCTURE D'UN PROGRAMME FORTRAN

II.1) Structure d'un programme

Quelque soit le langage utilisé, un programme se présente comme une succession d'instructions. L'algorithmique étudie la structure des programmes. Cette structure peut se présenter sous forme d'un **organigramme** où apparaissent de façon schématique les opérations à faire et leur séquence. Les entrées-sorties sont représentées dans des parallélogrammes, les traitements par des rectangles, les ruptures de séquence (ou aiguillages qui donnent des alternatives dans la séquence des instructions) par des polygones. Un organigramme est une représentation graphique de l'enchaînement des suites d'instructions. Il débute ainsi:



II.2) Structure d'un fichier Fortran:

Le fichier fortran doit posséder **un suffixe `.f`** pour être reconnu par le compilateur (par exemple : `perim.f`). On y trouve **une seule instruction par ligne**.

La ligne comprend 80 caractères. (C'était le format des cartes perforées). Elle comprend plusieurs zones (1-5 type, 6 suite, 7-72 ordre fortran, 73-80 identification)
Les colonnes 1 à 5 contiennent le type:

Si la **colonne 1** contient un C, cette ligne est ignorée par le programme, cela vous permet de mettre des commentaires pour faciliter la lecture de votre programme (ou des lignes blanches pour l'aérer). Il est important que votre programme soit bien lisible; les commentaires aident beaucoup.

Les **colonnes 1 à 5** contiennent éventuellement des étiquettes, c'est-à-dire un nombre repérant une ligne et qui sert de référence ailleurs.

La **colonne 6** indique, si elle contient un caractère, qu'il faut trouver dans cette ligne la suite de l'instruction de la ligne précédente. (Celle-ci est trop longue pour tenir sur une seule ligne). Le caractère peut être quelconque (excepté 0). En règle générale, il vaut mieux éviter d'écrire des instructions longues.

Les **colonnes 7 à 72** contiennent l'instruction proprement dite.

Les **colonnes 73 à 80** sont ignorées par le compilateur. Vous pouvez y mettre un commentaire; elles servaient autrefois pour numéroter les cartes.

Le programme est structuré de la manière suivante :

– Le mot PROGRAM suivi d'un titre
– déclarations des variables
– lecture des données
– traitement des données
– sortie des résultats
– le mot END pour signaler la fin du programme

II.2.a) Le titre

La ligne-titre ne peut être composée que de deux mots: le mot PROGRAM, suivi (avec un espace) d'un nom de six caractères (lettre ou chiffre) et commençant par une lettre:

```
^^^^^^PROGRAM^PERI
```

Ce nom ne doit pas entrer en conflit avec un autre nom ayant aussi une signification (nom d'une variable de votre programme, nom d'une fonction-bilbliothèque...).

II.2.b) Les déclarations

Pour forcer à déclarer TOUTES les variables du programme on utilisera l'instruction Fortran suivante :

```
^^^^^^IMPLICIT^NONE
```

Dans les **déclarations**, vous allez définir les noms et les types de toutes les variables utilisées dans le programme. Soit

`INTEGER`	pour des nombres entiers
`REAL`	réels
`DOUBLE PRECISION`	réels longs
`COMPLEX`	complexes
`CHARACTER`	texte
`LOGICAL`	logiques

Les variables sont des cases-mémoires repérées par un nom. Ce nom contient six caractères; il doit débuter par une lettre et ne pas avoir déjà reçu une autre signification (il n'est pas possible de réutiliser le nom du programme).

Exemples de déclarations:

```
^^^^^^INTEGER^*2^I, J, K
```

I, J et K sont des entiers stockés sur deux octets.

```
^^^^^^REAL^*4^RAYON, PERIM
```

RAYON et PERIM sont des réels stockés sur quatre octets.

Si l'on ne précise pas le nombre d'octet, la simple précision correspond sur IBM à 4 octets et la double à 8.

```
^^^^^^DOUBLE PRECISION^A, B
```

```
^^^^^^CHARACTER^*12^C
```

Le choix réel ou double précision est important pour les manipulations de ces nombres. Il n'est pas indifférent de déclarer REAL*8 A ou DOUBLE PRECISION A car cela va déterminer le choix de fonctions bibliothèque qui vont être appelées (par exemple la fonction sinus est différente suivant les déclarations SIN() ou DSIN()).

Les constantes contiennent des valeurs qui ne peuvent pas être modifiées lors de l'exécution du programme.

exemples de constantes:
entières : 15, -4 *le signe + peut être omis*
réelles : 3.14159, -12.3, 1.E3, 1.6E-19
double precision : 0.D0, 6.023D-23

C'est le point qui correspond à la virgule en français.

Un réel utilisant 4 octets (soit 32bits) est codé de la façon suivante: Le nombre réel est converti en binaire et la virgule est décalée vers la gauche. 4.25 s'exprime par 1.0001 $*2^2$. Un bit définit le signe, 8 bits sont réservés pour l'exposant (ici 2) et le reste (23 bits) sert pour la mantisse (10001). Les valeurs limites pour l'exposant sont -126 et +127 ce qui conduit en binaire à $1*2^{-126}$ et à $2*2^{127}$ c'est à dire en base 10 à $1.2*10^{-38}$ et à $3.4*10^{38}$.

Caractères : 'RESULTATS'

la chaîne de caractère doit être placée entre cotes

logique : .T. ou .F. *(à revoir plus tard)*

On peut également affecter des valeurs à une série de variables au moyen d'un DATA placé après les déclarations.

```
^^^^^^DATA^E,AVOG,PLANC^/1.6E-19,6.023E23,6.62E-34/
```

```
^^^^^^DATA^ZERO,NUL,O^/3*0.0/
```

Affectation de valeurs: Cela se fait par le signe =

```
^^^^^^N=2
```

```
^^^^^^N=M
```

```
^^^^^^PI=3.14159
```

```
^^^^^^IM=(3.14,0.0)
```

```
^^^^^^NOM='RESULTAT'
```

signifie que l'on met 2 dans la case mémoire appelée N (cette instruction se lit de droite à gauche), M dans la case N, 3.14159 dans la case PI...

La nouvelle valeur remplace toute autre valeur précédemment définie.
Au terme de la séquence

```
^^^^^^N=2
^^^^^^N=3
```

N vaudra 3

`^^^^^^N=N+1` signifie que l'on calcule d'abord N+1 et que l'on attribue cette valeur calculée à N.

ATTENTION, il doit y avoir compatibilité (Homogénéité) entre les déclarations de N et M. Sinon, vous aurez un message d'erreur. Si l'on définit un entier à partir d'un réel, celui-ci prendra la valeur entière la plus proche. Toute nouvelle définition remplace la précédente; une variable N a la valeur attribuée par la dernière instruction N= exécutée.

L'affectation se fait à la compilation.

II.2.c) Les Quatre opérateurs arithmétiques (+,-,,/,) et les puissances entières (**N)*

```
^^^^^^N=A+B
^^^^^^N=M**2
^^^^^^N=A+((3./DELTA)*-X-1.)
```

L'ordre de priorité se lit de gauche à droite en l'absence de hiérarchie ou de parenthèses.
Hiérarchie: les puissances ont priorité sur les produits et les divisions qui ont elles-mêmes priorité sur les sommes et les différences.

Que valent les réels suivants?

```
^^^^^^R1=12./3./2.
^^^^^^R2=12./3.*2.
^^^^^^R3=2/3
^^^^^^R4=R1+24.*3./(-2.)**3+2.*(5/2+1)
^^^^^^R5=12/(3/2)
```

Réponses (2, 8, 0, -1, 12)

L'élévation à une puissance non entière est possible, mais le compilateur passe alors par les logarithmes d'où une perte de précision et l'impossibilité de calculer -2.**3. alors que -2.**3 est faisable.

II.2.d) *Les fonctions bibliothèque.*

Il existe en fortran des fonctions mathématiques courantes (trigonométriques, logarithmes et exponentielles, racines carrées, valeur absolue, partie entière...). Ces fonctions comportent des arguments entre parenthèse. Ces arguments peuvent être des constantes, des

variables ou des expressions. L'argument des fonctions trigonométriques doit être exprimé en radians.

`PI=ACOS(-1.)` ou `PI=DACOS(-1.D0)`

La fonction donne un résultat en double précision si la première lettre est un D : `DSQRT( )` au lieu de `SQRT( )`, `DSIN( )` au lieu de `SIN( )`.

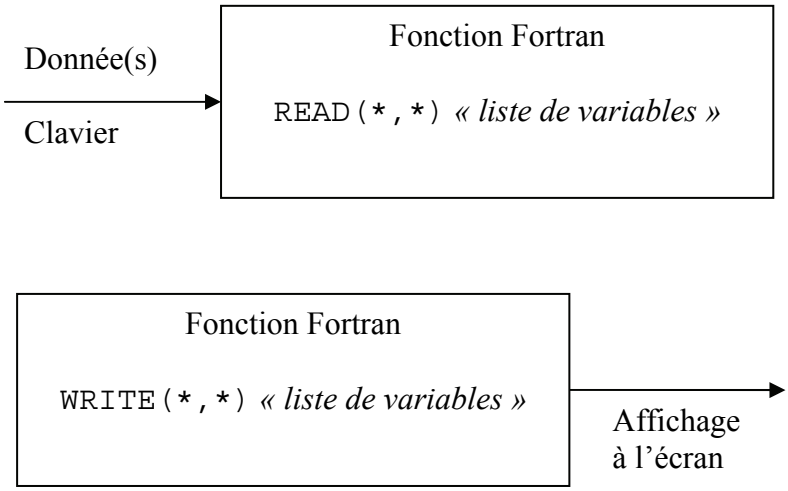
`MOD(A,B)` donne le reste de la division de A par B

`AINT`(E) renvoie la partie entière de E et `NINT(E)` l'entier le plus près (arrondi)

Vous avez à votre disposition un tableau contenant toutes les fonctions bibliothèque Fortran.

II.3) Lecture des données

II.3.a)Les entrées et sorties; les READ et WRITE



La lecture externe d'une donnée.

```
^^^^^^READ(*,*) N,M
```

Cet ordre vous permet d'introduire une donnée au moment de l'exécution. Le programme va attendre que vous donniez des valeurs pour N et M en utilisant le clavier. Pour que vous soyez averti de ce qu'attend le programme, il faut faire apparaître auparavant un message sur l'écran:

```
^^^^^^WRITE(*,*) 'ENTRER DEUX NOMBRES SEPARES PAR UNE VIRGULE'
^^^^^^READ(*,*) N,M
```

Résultat à l'écran :

```
ENTRER DEUX NOMBRES SEPARES PAR UNE VIRGULE
12 ↵
28 ↵
```

Si la chaîne contient une cote, il faut la doubler. 'L"APOSTROPHE'
De la même façon on peut faire apparaître un résultat (la variable S) sur l'écran:

```
^^^^^^WRITE(*,*) 'LE RESULTAT VAUT : ',S
```

Les deux étoiles correspondent à des choix prédéfinis: On a fixé pour vous l'unité où se font les ordre READ et WRITE (le clavier et l'écran) et on utilise un format dit libre. Il y a des façons de choisir vous même le lieu et la forme d'une lecture ou d'une impression. Nous verrons cela plus tard.

Lors de l'exécution les ordres "read "et "write" ne précisant pas de fichier spécifique renvoient au clavier et à l'écran. Si l'on veut que l'exécution utilise des fichiers entrée et sortie à la place du clavier et de l'écran, la commande unix devient :

```
exe1^<data>result^& ↵
```

Dans ce cas, les flèches correspondent au sens de saisie des informations :
à partir de : < ou vers : >.

La double flèche permet d'écrire à la suite d'un fichier existant :

```
exe1^<data>>result^& ↵
```

Le signe & vous permet de lancer un processus (l'exécution) en vous laissant la main. Sans ce signe, il faut attendre que l'exécution soit achevée pour que vous puissiez continuer à travailler. (Voir commandes de processus)

Exercices:

II-1) Ecrire un programme calculant le périmètre d'un cercle et le volume d'une sphère à partir de leurs rayons.

II-2) Ecrire un programme donnant l'angle compris entre 0 et 180° (en degrés, minutes et secondes) dont le cosinus a une valeur donnée.