UPMC SORBONNE UNIVERSITÉS

A.R.E.

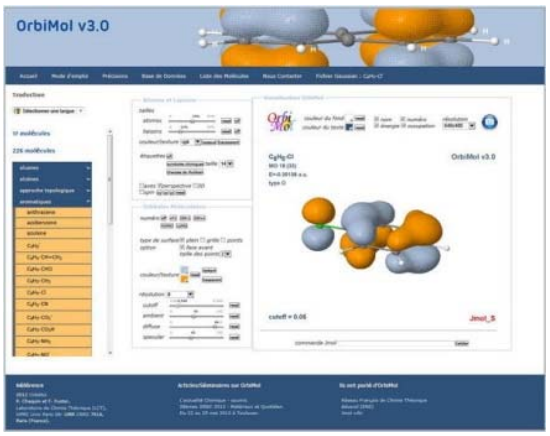
LIVRET DE L'ETUDIANT

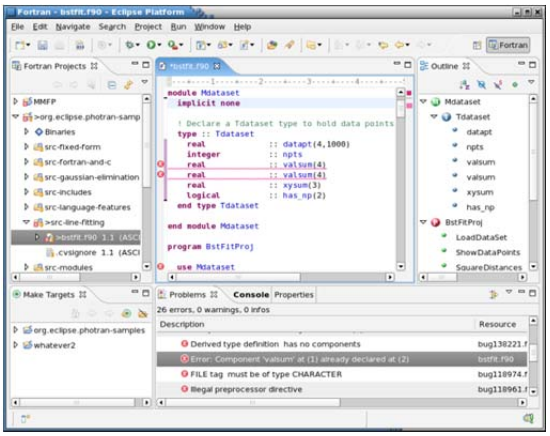
CIT

CHIMIE INFORMATIQUE ET THEORIQUE

FUSTER FRANCK ET CHEVREAU HILAIRE
franck.fuster@upmc.fr chevreau@lct.jussieu.fr

Le but de cet atelier est de comprendre des notions simples de Chimie Théorique et de concevoir un algorithme pour créer un programme informatique. Pour cela, en Chimie, nous reprendrons les notions déjà vues par les étudiants du L1 et nous les approfondirons en TD et en TP informatisé. En informatique, des cours informatisés introduiront, sous forme d'exemples simples, les notions de programmation qui seront nécessaires dans la conception de leur projet de chimie Informatique. Ce projet sera l'élaboration d'un programme appliqué à la Chimie.





Les informations concernant les A.R.E sont affichées sur les panneaux au sous-sol, niveau Saint Bernard près de la cafétéria, niveau SB dans couloir jaune. Le secrétariat est au niveau J+, au bout du couloir.

CALENDRIER DES EVALUATIONS

CE PLANNING PRECISE LES CONTROLES ECRITS ET ORAUX, AINSI QUE LES DEVOIRS OU RAPPORTS DEVANT ETRE RENDUS. AUCUNE EVALUATION NE POURRA ETRE REALISEE PAR LES ENSEIGNANTS EN DEHORS DE CES ELEMENTS.

Semaine	Dates	Rendu attendu durant cette semaine.
1	Du 19 au 25 janvier	
2	Du 26 au 30 janvier	
3	Du 2 au 6 février	
4	Du 9 au 13 février	Notion de Chimie2
5	Du 16 au 20 février	Notion de Chimie2
6	Du 23 au 27 février	Notion de Chimie3
7	Du 2 au 6 mars	Notion de Chimie3
8	Du 9 au 13 mars	Notion de Chimie4
9	Du 16 au 20 mars	Notion de Chimie4
10	Du 23 au 27 mars	TP OrbiMol ; Oral d'informatique
11	Du 30 mars au 3 avril	TP OrbiMol ; Oral d'informatique
12	Du 7 au 13 avril	Rapport écrit

- Toute absence à une séance obligatoire devra être justifiée auprès du secrétariat des U.E. transversales (niveau J+) dans les 2 jours ouvrés suivant la fin de l'absence. 2 absences non justifiées entrainent la non validation de l'UE (note 0/100), sauf décision contraire du jury d'UE et de parcours.
- Toute absence, justifiée ou non justifiée, lors d'un contrôle écrit ou oral entraînera une note de 0 à celui-ci.
- Toute absence ou retard à la restitution d'un devoir ou d'un rapport entraînera une note de 0 à celui-ci.
- En cas de retard (réveil qui ne sonne pas ou train en retard) : L'étudiant doit se présenter à la séance prévue même en retard. Il sera accepté ou non selon les cas. S'il n'est pas accepté, il devra se présenter au secrétariat pour faire constater par celui-ci que son absence en cours est liée à un retard : ceci tiendra lieu de justificatif d'absence.

BAREME DE L'ATELIER

NOTE DE TP /50

CETTE NOTE EST OBTENUE PAR SOMME DES ELEMENTS SUIVANTS :

SUIVI DU TRAVAIL PERSONNEL /10

La ponctualité, l'investissement et le comportement en classe feront l'objet d'une évaluation.

DEMARCHE ET INVESTIGATION SCIENTIFIQUE / 40

Le projet informatique sera évalué par rapport aux critères suivants : problématique, méthode pour sa résolution, typage des variables, analyse et optimisation de l'algorithme. Un rapport écrit sera demandé afin de répondre à tous ces critères, mais aussi pour expliquer le code informatique créé et démontrer sa validité scientifique par un calcul rédigé.

NOTE DE CC /50

CETTE NOTE EST OBTENUE PAR SOMME DES ELEMENTS SUIVANTS :

(AUTO-)APPRENTISSAGE DE CONNAISSANCE /20

En chimie, l'étudiant sera amené à travailler sur trois notions (3*10pts). Ce travail individuel se fera en bibliothèque pendant les heures de tutorat. Un compte-rendu écrit sera demandé en début de chaque séance de TD. Le compte-rendu du TP de Chimie sera noté sur 30 pts.

RESTITUTION ET DEFENSE / 30

Au cours de la dernière séance d'informatique (info5), le projet CIT sera présenté par l'étudiant devant l'ordinateur. Sa validité et sa robustesse seront testées. L'optimisation de l'algorithme sera discutée ainsi que son amélioration possible. Un œil critique sera demandé à l'étudiant.

PROJET INFORMATIQUE
PROGRAMMATION FORTRAN(90)

Le but de ce projet est de mettre en œuvre les notions vues en cours de programmation sur des concepts vus dans la partie chimie, tels que le numéro atomique d'un atome, sa configuration électronique, l'énergie d'un niveau dans le modèle hydrogénoïde, ou dans le modèle corrigé de Slater, les raies de transitions électroniques entre niveaux d'énergie différentes. Ce projet se décompose en plusieurs parties de difficulté croissante, chacune donnant lieu à la réalisation d'un programme qui devra compiler et fonctionner à l'exécution.

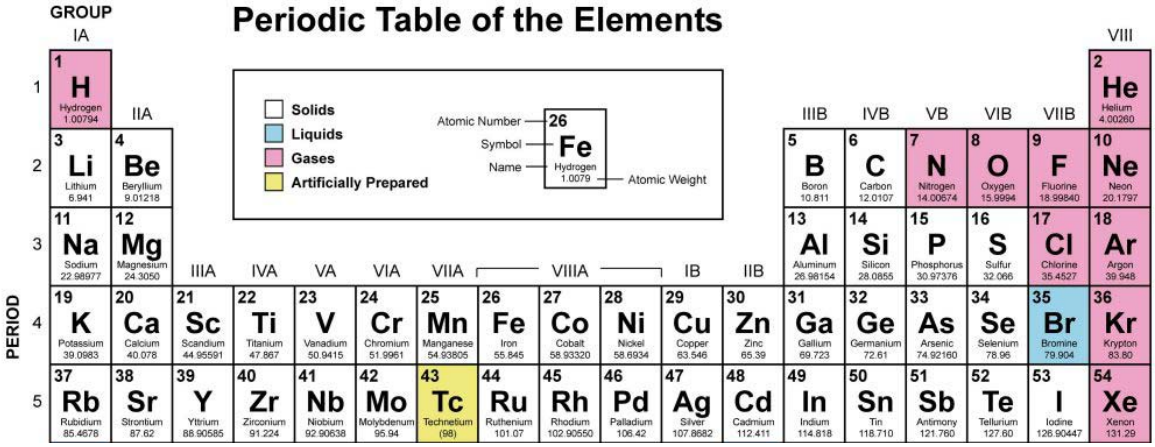
Votre rapport doit faire apparaître le programme fortran, les résultats du programme (exécution) pour compléter les réponses aux questions de Chimie (en italique).

À l'issue de ce projet, une mini-soutenance est prévue, avec pour support un rapport qui devra être rédigé et envoyé en format pdf. Pour le compte-rendu vous expliquerez les choix de programmation effectués (avec présentation et commentaires du code), les difficultés rencontrées, ainsi que les solutions retenues.

Partie I (ex. 2 du cours 3)

Soit un atome de numéro atomique Z (avec Z <= 54) écrire un programme qui indique son numéro de ligne et de colonne dans le tableau de classification périodique.

L'aide de votre programme (exécution) et de vos connaissances, indiquer à quelle période, colonne, bloc et famille, appartiennent les atomes suivants : $_{10}$Ne, $_{13}$Al, $_{26}$Fe et $_{37}$Rh. Utiliser votre programme pour valider vos réponses.

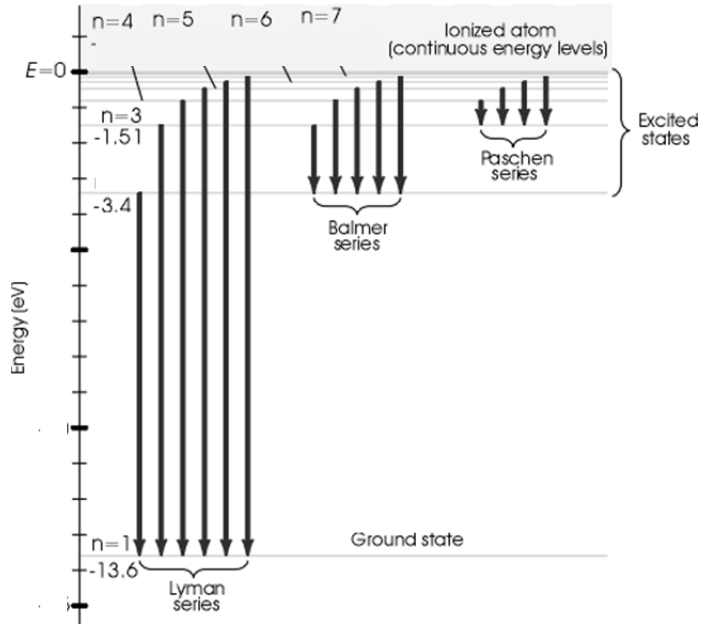


Partie II (ex. 4 du cours 4)

Écrire un programme qui indique la valeur de l'énergie du niveau n (n demandé à l'utilisateur) dans le modèle hydrogénoïde, pour un atome de numéro atomique Z (demandé à l'utilisateur).

Compléter ce programme pour qu'il donne l'énergie d'ionisation, ainsi que les écarts d'énergies pour les 3 premières raies des séries de Lyman (ni → nf = 1), de Balmer (ni → nf = 2), et de Paschen (ni → nf = 3). Améliorer encore le programme en calculant les longueurs d'onde associées à ces raies. Ces raies seront calculées à la main dans un premier temps pour l'atome d'hydrogène, et les valeurs comparées à celles obtenues par le programme (qui lui doit pouvoir fonctionner pour tout Z).

Pour l'atome de fer, calculer, à l'aide de votre programme, les niveaux d'énergie en eV du diagramme énergétique de l'hydrogénoïde associé. Calculer l'énergie d'ionisation en Joule et en eV ainsi que la différence d'énergie pour une des transitions de la série de Balmer. Utiliser votre programme pour valider vos réponses.



Partie III

Soit un atome de numéro atomique Z (avec Z <= 54) écrire un programme qui donne la configuration électronique de cet atome dans son état fondamental. Rappels : l'écriture de la configuration électronique suit le principe d'exclusion de Pauli ainsi que la règle de Klechkowski (on ne rentrera pas dans le détail de l'occupation des cases quantiques avec la règle de Hund). Et pour n : le nombre quantique principal, et ℓ, le nombre quantique secondaire, avec n = 1, 2, 3, 4, … et l tel que 0 <= ℓ <= n-1, (l=0 : orbitale s, ℓ=1, orbitale p, puis 2 : d) l'ordre de remplissage se fait selon n+ ℓ croissant, avec en cas d'égalité pour deux couples (n, ℓ) différents, n croissant. Par exemple, on remplira dans l'ordre 3s (3+0) puis 3p (3+1) = 4, puis 4s (4+0)=4 puis 3d (3+2)=5.

A l'aide de votre programme (exécution), écrire la configuration électronique de l'état fondamental des atomes suivants : $_{10}$Ne, $_{13}$Al, $_{26}$Fe et $_{37}$Rh. Utiliser votre programme pour valider vos réponses.

Partie IV

Afin de corriger le modèle hydrogénoïde, écrire un programme qui calcule le Z* effectif d'un atome de numéro atomique Z, tel que Z* = Z – s, avec s : la somme des constantes d'écran à prendre en compte entre un électron sur une couche donnée et le noyau. Pour simplifier, on prendra 0 pour tout électron situé sur une couche supérieure à l'électron considéré, 0.35 pour tout autre électron situé sur une même couche, 0.85 sur la couche immédiatement inférieure, et 1 pour toute autre couche inférieure.

Par ex : pour l'azote : Z = 7, configuration électronique $1s^2\ 2s^2\ 3p^3$, pour un électron en 3p :

$$Z^* = 7 - 2 * 0.35 - 2 * 0.85 - 2 * 1$$

Pour les différents types d'électrons de l'atome de fer, calculer, à l'aide de votre programme (exécution), la charge effective Z dans le modèle de Slater. Utiliser votre programme pour valider vos réponses.*

Constantes d'écran de Slater

1s	0,3										
2s 2p	0,85	0,35									
3s 3p	1	0,85	0,35								
3d	1	1	1	0,35							
4s 4p	1	1	0,85	0,85	0,35						
4d	1	1	1	1	1	0,35					
4f	1	1	1	1	1	1	0,35				
5s 5p	1	1	1	1	0,85	0,85	0,85	0,35			
5d	1	1	1	1	1	1	1	1	0,35		
5f	1	1	1	1	1	1	1	1	1	0,35	
6s 6p	1	1	1	1	1	1	1	0,85	0,85	0,85	0,35
	1s	2s 2p	3s 3p	3d	4s 4p	4d	4f	5s 5p	5d	5f	6s 6p

Partie V (bonus)

Reprendre la partie II cette fois-ci avec le Z* effectif pour le calcul des transitions, et comparer les résultats obtenus dans les deux modèles pour différents atomes.

Tester votre programme avec l'atome de fer.

Programme d'Informatique

Info1 (2h) : *Linux*

- Introduction et commandes Linux

Info2 (2h) : *programmation*

- Structure d'un programme, déclarations de variables et premières instructions d'entrée sortie

Info3 (2h) : *programmation*

- Aiguillages avec les instructions permettant d'effectuer des tests. Introduction au projet.

Info4 (2h) : *programmation*

- Tableaux et processus itératifs

Info5 (1h30) : *programmation*

- Finalisation du projet et soutenance

Au cours des séances des exercices sont prévus sur machine.

UNIVERSITE PIERRE ET MARIE CURIE

Licence. Module A.R.E
Atelier de Recherche Encadrée

C.I.T
Chimie Informatique et Théorique

INTRODUCTION A LA PROGRAMMATION FORTRAN(90)

Plan :

1 : Introduction et instructions unix

2 : Structure d'un programme Fortran

3 : Aiguillages

4 : Tableaux et Processus itératifs

Contrôle des connaissances :

Projet

Rapport et Soutenance du projet

Objectifs

Cette UE a pour objectif de s'approprier une thématique scientifique, ici les bases de la programmation, en relation avec le domaine de la chimie théorique. L'acquisition des connaissances de base devra permettre à l'étudiant de réaliser un projet en relation avec la chimie, et de mener à bien des raisonnements en informatique. Une soutenance viendra marquer la phase de restitution de ce processus de réflexion et de réalisation personnelle.

Les concepts sont abordés 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 (L2, 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 I –

INTRODUCTION

I.1) HISTORIQUE ET DEFINITIONS

Informatique: traitement des **INFOR**mations par voie auto**MATIQUE:** ensemble des techniques de traitement de l'information utilisant les ordinateurs.
Ordinateur: calculateur doué de mémoire.

Pascal invente la première machine à additionner en 1642, Leibniz, la première machine à multiplier en 1671, Babbage les cartes perforées en 1840. En 1855 International Business Machine est fondée aux Etats-Unis.

Matériel (Hardware, Hard): tout ce qui a une existence physique: l'ordinateur et ses périphériques.

Logiciel (Software, Soft): l'information, elle-même et les langages permettant de la traiter.

L'algorithmique (recherche des algorithmes les mieux adaptés aux ressources de l'ordinateur) et la programmation sont des branches de l'informatique qui s'intéressent à la conception des méthodes de résolution de problèmes sur les machines.

I.2) LES UNITES D'INFORMATION.

Tout symbole de caractère, chaque chiffre, est codé d'une façon particulière en mémoire. Et l'unité élémentaire de mémoire est le BIT ou BInary digiT, valant 0 ou 1, ce qui donne le codage binaire.

Les unités utilisées sont

- l'octet (BYTE) qui vaut 8 bits,
- le kilooctet, K qui vaut 10^3 octets
- le megaoctet, MK qui vaut 10^6 octets. (1K de K)
- le gigaoctet, qui vaut 10^9 octets.

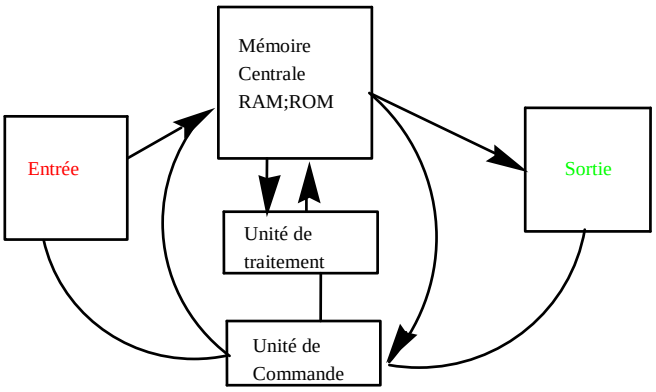
Un entier de quatre octets représenté par 32 bits (dont un pour le signe) devra être au maximum égal a 2^{31}-1= 2 147 483 647 (codé en binaire) soit un nombre de 10 chiffres décimaux;

Le codage d'un texte nécessite, la représentation des chiffres, des lettres de l'alphabet (y compris des caractères spéciaux + - * /). C'est le cas d'un texte fortran.

Un octet permet de définir 2^8=256 caractères. Le code ASCII définit 2^7=128 caractères que l'on peut étendre à 256 en utilisant des octets.

I.3) LE MATERIEL.

Il comprend des organes d'entrée, de traitement et de sortie.



Périphériques :
Entrée : **Clavier**, Unité de disque, lecteur CD-ROM ou DVD, clé USB, réseau de communication ...
Sortie : **Ecran**, Unité de disque ou graveur CD-ROM/DVD, clé USB, Réseau de communication, **Imprimante** ....

Unité Centrale :
Mémoire centrale: là où l'on stocke les informations. Une image simple serait celle d'une armoire avec des tiroirs: chaque tiroir (registre) contient un nombre en binaire.

ROM (Read Only Memory) Mémoire morte dans laquelle le constructeur a préenregistré des logiciels (le système permettant à la machine de fonctionner).

RAM (Random Access Memory) Mémoire vive dans laquelle vous copiez des logiciels (système d'exploitation que l'on peut changer: l'opération de chargement s'appelle Boost; vos programmes)

Organe de traitement :
Microprocesseur; c'est la partie active de l'ordinateur, celle où se font les opérations. Elle est toute petite (en bits). En effet, prenons l'addition comme opération de base. Elle nécessite la présence de trois "tiroirs" si l'on veut conserver les valeurs de départ (une pour les deux nombres à ajouter, une pour le résultat) ou de deux tiroirs seulement (si l'on place le résultat, là où il y avait un des deux nombres à ajouter).

Unité de commande : organe physique permettant de charger les instructions dans un ordre donné (il y a un compteur ordinal) et de les contrôler. Horloge interne.

I.4) login, password : accès aux ordinateurs.

Vous disposez d'un PC relié physiquement au réseau informatique d'enseignement de l'Université Pierre et Marie Curie.

- ***Le matériel informatique à LUTES***

Les machines de Lutes permettent de travailler dans plusieurs configurations en utilisant des machines virtuelles (*MV*).

Les salles d'informatique de Lutes sont équipées de PC qui permettent de travailler soit avec le système d'exploitation Windows, soit avec le système d'exploitation Linux (Mandriva).
Pour les enseignements du module, les étudiants se connecteront sur :

« *Mandriva 2006 avec authentification* »

Pour se Connecter sur le PC :
mettre son nom d'utilisateur, puis taper son mot de passe (ce dernier n'apparaît pas sur l'écran, on voit des *), puis taper sur la touche ENTER pour valider.

Lorsque l'ordinateur sera prêt, vous serez dans une configuration KDE de Linux (écran avec des icônes style Windows).

On va travailler dans une **fenêtre « terminal »** pour saisir des lignes de commandes. Le programme qui fournit cette interface est le shell (ici on aura un shell « bash ») . Dans la fenêtre, on a un prompt :
[login@localhost login]$

Ce prompt suivi par un curseur indique que l'ordinateur attend une commande.

Les répertoires sont organisés comme un arbre, la racine est le premier répertoire existant. Il est toujours indiqué par le signe /
Vous n'avez pas le droit d'écrire dedans (ou d'effacer).
Le nom d'un répertoire est un nom composé par exemple : /home/tartempion
Les différents membres du nom sont séparés par des slash (/). C'est une séquence hiérarchisée ; chaque nom est un sous-répertoire du nom qui le précède. Votre répertoire est situé dans /home/ tartempion, c'est-à-dire un répertoire situé dans / et qui s'appelle home et qui contient votre répertoire tartempion. Dans votre «Home Directory» (tartempion dans notre exemple), vous avez le droit de créer/effacer des fichiers et des répertoires.

Pour se Déconnecter du PC :
cliquer en bas à gauche sur K pour obtenir le menu :
Terminer la session
Eteindre l'ordinateur
...

sélectionner : « **Eteindre l'ordinateur** » pour revenir à l'écran de départ de LUTES.

I.5) Les logiciels.

Ensemble de programmes permettant d'utiliser l'équipement précédent.

- Système d'exploitation : **UNIX** il permet d'ordonnancer les taches exécutables par l'ordinateur et d'accéder aux programmes langages.

- Langages de programmation : Basic (**B**eginner's **A**ll **P**urpose **S**ymbolic **I**nstruction **C**ode)- 1965. **Fortran** première version 1954 Fortran IV 1962 Fortran V 1977 : il existe plusieurs versions avec en plus des possibilités d'extensions pas présentes dans les versions standards. Pascal (**P**rogram (**A**) **S**tructured **C**omputer **A**lgorithmic **L**anguage) – 1969. COBOL **Co**mmon **B**usiness **O**riented **L**anguage. ADA. C/C++, Java …

Ces langages doivent être traduits pour que la machine les exécute. En effet l'ordinateur ne comprend que les chiffres binaires; la traduction se fait au moyen de commandes UNIX: c'est la **compilation**. Le résultat obtenu en deux étapes est un module_objet puis un module_exécutable. La première étape "traduit" et crée le module_objet. La deuxième étape relie les modules_objets entre eux pour créer le module exécutable (load module).

Un programme est une suite ordonnée d'instructions. Quel que soit le langage, il comprend trois parties: une entrée (lecture de données), une partie traitement et une sortie (restitution des résultats).

I.5.A) LES COMMANDES UNIX

Une action sur l'ordinateur est déclenchée par une commande. Une commande peut être obtenue en frappant son nom, puis en tapant sur la touche retour ↵ qui envoie l'exécution.

man Des informations peuvent être consultées pour chaque commande par man cmd1 (man renvoie aux pages d'un manuel dans une bibliothèque de l'ordinateur).

Une commande peut être obtenue en frappant une touche préréglée du clavier; ce sont les clefs de commandes. Les clefs sont des touches qui exécutent directement une action sur l'ordinateur. Par exemple CNRTL-Z est défini et son action suspend une commande.

Le mode historique des commandes déjà tapées est souvent utile pour relancer une commande deux fois de suite. Vous activez cette option en lançant l'interpréteur de commande `bash` ↵. L'historique de vos commandes apparaîtra à l'aide des flèches hautes et basses.

I.5.B) LES COMMANDES POUR LES REPERTOIRES.

pwd La commande pwd affiche le nom absolu (à partir de /) du répertoire courant.

mkdir créer un sous-répertoire par mkdir␣rep1

rmdir détruire par rmdir␣rep1
ou bien par rm␣-r␣rep1 (catalogue ayant un contenu)

mv changer son nom par mv␣rep1␣rep2

cp copier par cp␣-r␣rep1␣rep2

I.5.C) LES COMMANDES POUR LES FICHIERS.

LS Taper ls ↵, vous verrez apparaître la liste des fichiers/répertoires qui sont déjà installés dans votre répertoire s'il y en a. Les fichiers contiennent des écritures. Ils peuvent contenir du texte

à consulter dans un langage quelconque (fichier dans lequel on peut lire ou écrire), des listes de commandes, un module objet ou exécutable (dans ces 2 derniers cas le fichier n'est ni lisible ni imprimable. Ne jamais lancer l'impression d'un module exécutable!).

La plupart des commandes sont accompagnées de paramètres (obligatoires ou optionnels) et d'arguments. Par exemple ls␣-ail vous donne plus d'informations: un numéro (ou inode) associé à chaque fichier, des renseignements sur les gens autorisés à lire, écrire ou utiliser vos fichiers... Les paramètres a (all*) i (inode) et l (long) sont abrégés en ail et placés derrière un tiret. Les arguments concernent les objets sur lesquels agissent les commandes, par exemple un nom de fichier ou de répertoire.

more La commande more vous montre le contenu d'un fichier. Par exemple more␣fic1 ↵

Si un fichier est trop long pour être affiché sur un seul écran, il est préférable de l'afficher page par page. Pour cela faire more␣-l␣*«nom du fichier»*. On passe alors de page en page en actionnant la barre d'espace (ou en arrière grâce à CNTRL et B).

grep On peut rechercher une expression avec la commande grep. Par exemple, grep␣mot␣fic1 va rechercher l'expression mot dans le fichier fic1 et affichera à l'écran toutes les lignes contenant l'expression recherchée.

rm détruire par rm␣fic1 (sans confirmation)
rm␣–i␣fic1 (avec confirmation)

mv changer son nom par mv␣fic1␣fic2

cp copier par cp␣fic1␣fic2

lpr imprimer un fichier lpr␣-Pibm7␣fic1

wc compter les lignes d'un fichier par wc␣-l␣fic1
les caractères et mots wc␣-lcw␣fic1

diff comparer 2 fichiers (et afficher la différence) diff␣fic1␣fic2 donne toutes les lignes qui diffèrent

Si une commande est incomplète, il peut y avoir affichage de ? au lieu du prompt. Cela vous indique que l'ordinateur attend un paramètre.
On peut enchaîner deux commandes (en les séparant par des points virgules s'il y a confusion possible entre nouvelle commande et argument de la précédente). La seconde prend en entrée le résultat de la première :

`ls␣-l | more` la commande ls␣-l fait défiler trop d'informations sur l'écran; more enchaîné va permettre de faire défiler ces informations page par page. le signe "pipe" (|) est obtenu en tapant: CNTRL+ALT+F6

Pour créer un fichier nouveau, il suffit d'utiliser un éditeur en nommant un fichier nouveau. Ce fichier sera crée s'il n'existe pas, ouvert s'il existe déjà. Faire par exemple `nedit␣fic1, ou gedit␣fic1`.

cat Pour une première fois, nous allons faire différemment en utilisant la commande cat. Ce sera exceptionnel; par la suite vous utiliserez directement gedit fic1. La commande cat affiche à

l'écran un fichier : cat fic1 La flèche permet de réorienter la sortie: au lieu de l'écran cat >fic1 crée le fichier fic1. Tout ce qui est entré au clavier est placé dans le fichier fic1. Pour sortir de l'édition du fichier, il faut faire CNTRL-D. (ou par CNTRL-Z si l'on a fait une fausse manoeuvre et si l'on veut annuler ce qui vient d'être édité). Le processus cat >fic1 remplace le fichier fic1 antérieur qui porte ce nom si un fichier existe déjà avec ce nom. Si l'on veut conserver le fic1 et écrire à sa suite, faire cat >>fic1.
La commande cat <fic1 >fic2 prend le contenu de fic1 et le place dans fic2 en remplaçant fic2. La commande cat <fic1 >>fic2 prend le contenu de fic1 et le place dans fic2 à la suite de son contenu.

I.5.D) LES COMMANDES POUR LES PROCESSUS.

Lorsqu'une commande est lancée, lorsqu'un travail est soumis, un processus apparaît.

ps donne les informations sur les processus en cours

kill provoque l'arrêt d'un processus spécifié kill -9 23896
Le numéro d'identification (dans l'exemple 23896) est dans PID de ps^u (travaux de l'utilisateur)

Un module exécutable est mis en œuvre par son nom de fichier (voir plus loin). Il est normalement exécuté en interactif (foreground) ce qui implique d'attendre que le processus soit terminé pour retrouver la main. Un processus interactif peut être suspendu par CNTRL-Z

Un processus peut être exécuté en arrière plan (background) ce qui vous laisse la possibilité de continuer à travailler. Cela se fait en accompagnant la commande du symbole &. Les commandes bg et fg font passer de foreground à background et réciproquement.

I.6) COMPILATION ET EXECUTION.

ATTENTION : Les fichiers sources fortran doivent avoir un nom se terminant par .f

La source fortran doit être traduite par un compilateur pour créer un module objet. Le module objet est ensuite relié à d'autres modules objet par l'éditeur de liens pour engendrer le module exécutable. Il y a donc trois versions d'un même programme: les fichiers .f .o et le module exécutable. Seul le fichier .f est lisible.

1ere étape : compilateur fortran, g95, vérifie l'orthographe et la grammaire de votre programme

`g95` ^ `-c` ^ *«nom du fichier fortran»*

La commande `g95 -c perim.f` vérifie l'orthographe et la grammaire de votre programme contenu dans perim.f. Si tout se passe bien, il apparaît dans votre répertoire un fichier .o (faire ls) et le message `Compilation successfull`.

2eme étape : le module objet est relié à d'autres modules objet par le compilateur pour engendrer le module exécutable

`g95` ^ `-o` ^ *«nom du fichier exécutable»* ^ *«liste des modules objets»*

La commande `g95 ^ -o ^ perim ^ perim.o` crée le programme `perim`. En

l'absence de nom le fichier exécutable va s'appeler `a.out` Si l'on doit réunir plusieurs modules objet faire
`g95^-o^exe1^fic1.o^fic2.o^fic3.o` si l'on doit réunir tous les modules objet faire
`g95^-o^exe1^*.o` attention, il ne faut pas avoir alors plusieurs sous-programmes avec le même nom. Il est recommandé de travailler dans un sous-répertoire dans lequel vous réunissez tous les fichiers dont vous avez besoin.

3eme étape : l'exécution du programme est lancée par `perim` ↵ (ou ./`perim` ↵)

Exercices:

I-1). Créer un répertoire TEST dans un répertoire SEANCE1, puis y accéder.

I-2) Dans TEST, Créer et compiler le programme suivant :

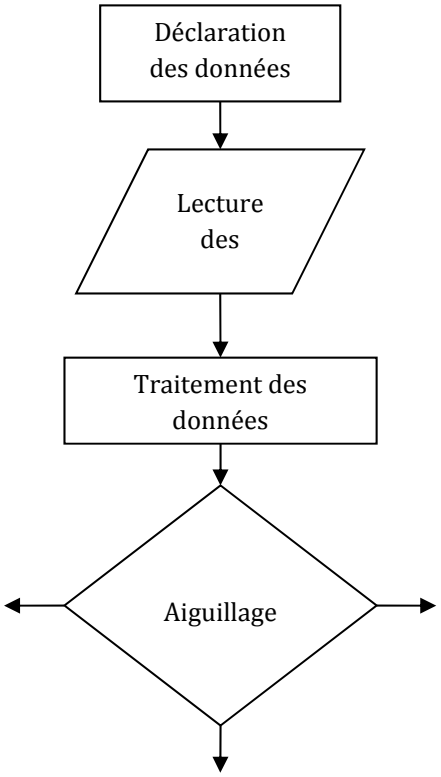
```
^^^^^^PROGRAM^ESSAI
^^^^^^WRITE(*,*) 'bonjour^!'
^^^^^^END
```

- COURS 2-

STRUCTURE D'UN PROGRAMME FORTRAN

II.1) STRUCTURE D'UN PROGRAMME

Quel que 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, ou un point d'exclamation « ! », 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. D'un manière générale, un commentaire débutant par un « ! » peut suivre une instruction, dans ce cas, tout le reste de la ligne est considéré comme un commentaire.

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 etre composee que de deux mots: le mot PROGRAM, suivi (avec un espace) d'un nom de six caracteres (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-bibliothèque...).

II.2.b) Les declarations

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. Autre écriture possible, avec un petit commentaire

```
^^^^^^REAL^*4^:: rayon   ! variable pour stocker le rayon
```

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
^^^^^^CHARACTER (LEN=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 définies à l'aide du mot clef PARAMETER 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.
Caractères : 'RESULTATS'
la chaîne de caractère doit être placée entre cotes
logique : .T. ou .F. *(à revoir plus tard)*

Ex :

REAL, PARAMETER :: pi = 3.14159

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 operateurs arithmetiques (+,-,*,/,) et les puissances entieres (**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 BIBLIOTHEQUE.

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

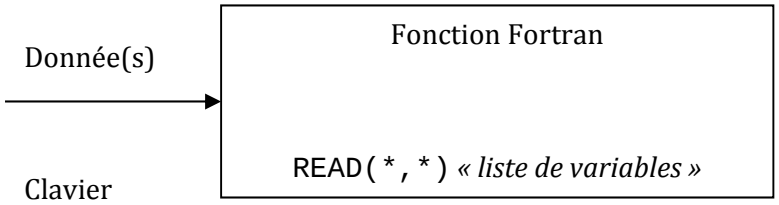
`MOD(A,B)` donne le reste de la division de A par B

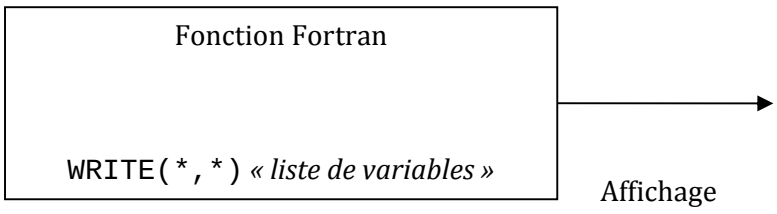
`AINT`(E) renvoie la partie entière de E et `NINT(E)` l'entier le plus près (arrondi)

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

II.3) LECTURE DES DONNEES

II.3.A)LES ENTREES 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'
^^^^^^READ(*,*) N,M
```

Résultat à l'écran :

```
ENTRER DEUX NOMBRES
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 ordres 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)

Exemples de programmes.

Soit le programme somme qui demande à l'utilisateur deux valeurs réelles, les additionne en stockant le résultat dans une variable intermédiaire, et affiche le résultat.

```
PROGRAM somme
IMPLICIT NONE
REAL x, y, s
WRITE(*,*)'' rentrer une valeur reelle ''
READ(*,*) x
WRITE(*,*)'' rentrer une autre valeur reelle''
READ(*,*) y

s = x + y

WRITE(* ,*) ''La somme de x = '',x , '' et y = '', y, ''vaut = '', s
END PROGRAM
```

Exercices:

II-1). Écrire un programme demandant deux valeurs entières à l'utilisateur, calculant le produit de ces valeurs, et affichant le résultat.

II-2) Écrire un programme calculant le périmètre d'un cercle et le volume d'une sphère à partir de leurs rayons.

II-3) Écrire un programme donnant demandant à l'utilisateur un nombre N de secondes, et affichant son équivalence en heure, minute et secondes.

AIGUILLAGES

III.1) Variables booléennes.

Il s'agit d'une variable qui ne peut prendre que deux valeurs, **.TRUE.** (vrai) ou **.FALSE.** (faux). Cette variable doit avoir été déclarée par:

```
^^^^^^LOGICAL^P,N
```

... suite du programme ...

```
^^^^^^P=.TRUE.
^^^^^^READ(*,*) N
```

La donnée N peut être constituée uniquement de T ou F (les points ne sont pas obligatoires).
Une variable booléenne est associée à une proposition qui est vraie ou fausse selon que vous l'avez déclarée telle. Il s'agit soit d'une donnée que vous avez introduite (pour contrôler l'impression de résultats intermédiaires d'un calcul par exemple), soit de variables qui ont pris des valeurs selon des cas de figure prévues dans votre programme. Dans ce dernier cas, elle résulte de la comparaison de variables du programme à un instant donné; cette comparaison se fait au moyen d'opérateurs relationnels.

III.2) Opérateurs Logiques

Ces opérateurs agissent sur les variables booléennes.

.NOT. négation
`.NOT..TRUE.` équivaut à `.FALSE.`
`.NOT.P` équivaut à `.FALSE.` si `P=.TRUE.`

.AND. ajoute deux variables booléennes, la proposition est vraie si les deux propositions sont vraies.

.OR. la proposition résultante est vraie si l'une des deux propositions est vraie.

.EQV. la proposition résultante est vraie si les deux propositions sont équivalentes (toutes deux vraies, ou toutes deux fausses).

.NEQV. la proposition résultante est vraie si les deux propositions ne sont pas équivalentes (l'une vraie et l'autre fausse).

La hiérarchie de ces opérateurs est l'ordre ci-dessus (entre EQV et NEQV, c'est le premier rencontré, le plus à gauche)

III.3) Opérateurs relationnels

Ils permettent de comparer des variables non logiques de même type et de même précision (deux entiers, deux réels, deux chaînes de caractères...) Le résultat de la comparaison est **.TRUE.** (vrai) ou **.FALSE.** (faux).

.GT.	**ou**	**>**	plus grand que	**>**
.GE.	**ou**	**>=**	plus grand que	**≥**
.LT.	**ou**	**<**	plus petit que	**<**
.LE.	**ou**	**<=**	plus petit que	**≤**
.EQ.	**ou**	**==**	égal à	**=**
.NE.	**ou**	**!=**	différent de	**≠**

Ces opérateurs sont prioritaires sur les opérateurs logiques.
Attention aux tests d'égalité sur les réels. Ceux-ci peuvent avoir des valeurs approchées et non rigoureusement égales. Pour que deux réels soient égaux, il faut que tous leurs bits le soient.

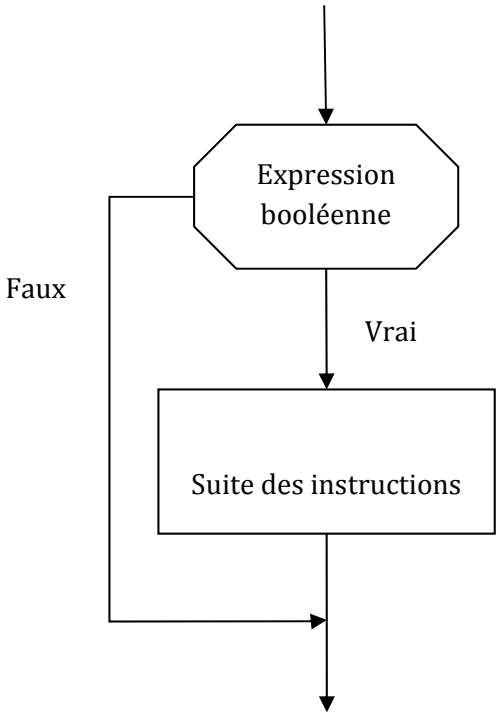
III.4) Ruptures de séquence : les Aiguillages.

Si la condition ne porte que sur une seule instruction :

```
IF ( <<expression booléenne>> )^<<instruction fortran>>
```

L'instruction fortran n'est exécutée seulement si l'expression booléenne (entre parenthèses) est vraie

Si la condition porte sur plusieurs instructions fortran :



l*a lecture du programme est facilitée si vous décalez les écritures des instructions vers la droite (colonne supérieure à 7)*

```
IF ( <<expression booléenne>> ) THEN

    ... suite d'instructions à exécuter
    si l'expression est vraie ...

ENDIF
```

Dans le cas où il y a plusieurs suites d'instructions suivant la justesse de l'expression booléenne :

```
IF ( <<expression booléenne>> ) THEN

    ... suite d'instructions à exécuter
    si l'expression est vraie ...

ELSE
```

... suite d'instructions à exécuter
l'expression est fausse ...

ENDIF

Attention, il est nécessaire de respecter les sauts de lignes !

Il peut ne pas y avoir de ELSE (Si la proposition est fausse, les instructions du bloc suivant le IF..THEN sont omises). Il est possible d'imbriquer plusieurs blocs IF..THEN (Un second bloc IF..THEN-ELSE-ENDIF est alors inclus dans une série d'instructions contrôlée par un IF..THEN; dans ce cas, il y a autant d'ENDIF que de IF..THEN et chaque ENDIF renvoie toujours au dernier IF THEN rencontré.

IF (*<<expression booléenne 1>>*) THEN

... suite d'instructions à exécuter si l'expression 1 est vraie ...

ELSE
IF (*<<expression booléenne 2>>*) THEN

... suite d'instructions à exécuter si l'expression 1 est fausse et l'expression 2 est vraie ...

ELSE

... suite d'instructions à exécuter si les expressions 1 et 2 sont fausses ...

ENDIF [sur l'expression booléenne 2]
ENDIF [sur l'expression booléenne 1]

L'instruction **ELSE IF () THEN** *(sur une seule ligne)* permet d'alléger l'écriture de plusieurs propositions mutuellement exclusives. L'ensemble commandé par le premier IF..THEN se termine alors par un seul ENDIF (pas de ENDIF associés aux ELSE IF sur une seule ligne).

IF (*<<expression booléenne 1>>*) THEN

... suite d'instructions à exécuter si l'expression 1 est vrai ...

ELSE^IF (*<<expression booléenne 2>>*) THEN

... suite d'instructions à exécuter si l'expression 1 est fausse et l'expression 2 est vraie ...

ELSE

... suite d'instructions à exécuter si les expressions 1 et 2 sont fausses ...

ENDIF

Attention:

- *Une seule instruction par ligne; n'ajoutez rien aux lignes où sont déjà écrits if..then ou else.*
- *Il y a autant d'ENDIF que de IF..THEN (Ne pas compter les ELSE IF..THEN)*
- *Le nombre d'ELSE est au plus égal à celui de IF..THEN*
- *Le nombre d'ELSE IF..THEN est quelconque (Les propositions sont alors hiérarchisées - les instructions qui suivent un ELSE IF..THEN sont effectuées dans le cas où toutes les propositions booléennes qui précèdent sont "false").*

Exemple de programme contenant un test :

```
PROGRAM leplusgrand
IMPLICIT NONE
REAL x, y

WRITE(*,*) ''rentrer deux valeurs réelles''
READ(*,*) x, y

IF ( x >= y) THEN
  WRITE(*,*) x, '' plus grand que '', y
ELSE
  WRITE(*,*) y, '' plus grand que x'', x
ENDIF

END PROGRAM
```

Autre exemple de programme :

```
^^^^^^PROGRAM^TEST
^^^^^^IMPLICIT^NONE
^^^^^^READ(*,*) I
^^^^^^IF (I.LT.2) THEN
^^^^^^^^WRITE(*,*) I,' est < 2'
^^^^^^ELSE^IF (I.EQ.3) THEN
^^^^^^^^WRITE(*,*) I,' vaut 3'
^^^^^^ELSE^IF(I.LT.10) THEN
^^^^^^^^WRITE(*,*) I,' est compris entre 4 et 10'
^^^^^^ENDIF
^^^^^^STOP
^^^^^^END
```

Evitez d'écrire au départ beaucoup d'if sans les tester; éviter les else if si vous débutez.

Exercices:

III-1).Écrire un programme qui calcule la valeur absolue d'une variable x réelle dont la valeur est demandée à l'utilisateur

III-2). Écrire un programme qui lit trois valeurs, et affiche la plus grande des trois

III-3). ***Partie I du projet****.*

III-4) Résolution de l'équation du second degré AX^2+BX+C *(faire lire A, B et C; Prévoir tous les cas possibles. Ne pas déclarer de variable complexe).*

TABLEAUX ET PROCESSUS ITERATIFS

IV.1) Déclaration des tableaux

Il est fréquent d'avoir à désigner sous un même nom de variable des quantités de même nature qui diffèrent par un système d'indice (séries, calcul matriciel,...). On peut définir un tableau de variables. La définition se fait lors des déclarations par :

```
^^^^^^INTEGER^V(40)
^^^^^^REAL^W(20,10)
```

qui signifient que V est un tableau unidimensionnel (vecteur) de 40 valeurs et W un tableau bidimensionnel (matrice) de 200 valeurs (20 lignes, 10 colonnes). Les mémoires correspondantes sont alors réservées. Dans le tableau, les valeurs sont rangées sur une seule ligne; le premier indice varie le premier, le second (qualifié pour un tableau bidimensionnel d'indice de colonne) ensuite et ainsi de suite dans le cas de dimension supérieure.

La déclaration a pour fonction

- de réserver la place nécessaire aux tableaux
- de déclarer la structure: variable d'un ou de plusieurs indices

Si l'indiçage débute par une valeur différente de 1, il est possible de le spécifier. Les valeurs extrêmes des indices sont séparées par deux points.

```
^^^^^^REAL^V(20:60), W(1O:30,10:20)
```

Autre manière de faire en f90 :

```
^^^^^^REAL, DIMENSION (20 :30) :: V
```

Pour des tableaux contenant N cases par exemple, il est préférable d'utiliser l'instruction PARAMETER pour définir N comme une constante

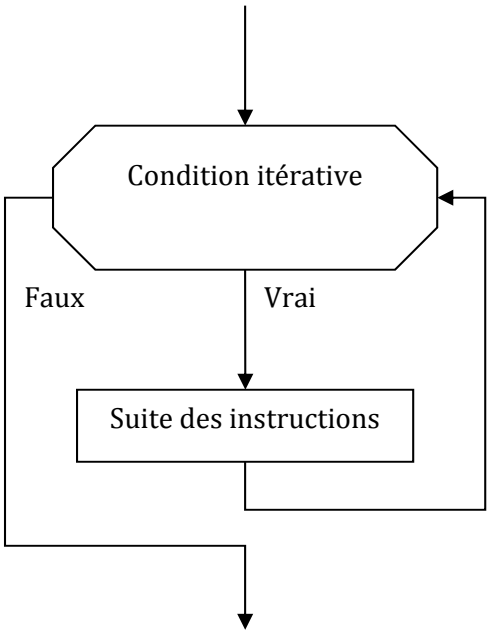
```
^^^^^^INTEGER, PARAMETER :: N = 10
^^^^^^REAL, DIMENSION (1:N) :: T
```

Ainsi, une éventuelle modification de la valeur de N est localisée en un seul endroit dans le programme.

L'accès à une case du tableau se fait par la valeur de (ou des) indice(s) de cette case, par exemple la deuxième case de T est T(2). Le plus souvent, les tableaux sont utilisés dans des boucles qui permettent le traitement de toutes les cases, ou d'une partie des cases du tableau, et on utilise alors les indices de boucles (voir infra) pour accéder aux cases. Autrement dit, il est possible d'accéder à la deuxième case du tableau T avec l'instruction T(i) pour i variable entière valant 2.

IV.2) Les séquences itératives.

Un programme itératif permet de répéter des opérations plusieurs fois. Une séquence itérative est une suite d'instructions que l'on exécute plusieurs fois à l'intérieur d'un même programme.



IV.3) Les boucles arithmétiques.

La lecture du programme est facilitée si vous décalez les écritures des instructions vers la droite (colonne supérieure à 7)

```
DO^«condition itérative»

   … suite des instructions …

ENDDO
```

La condition itérative porte sur une variable (entière, réelle ou double précision) dont on donne la valeur de départ, la valeur de fin et, le cas échéant, la valeur du pas de l'incrémentation :

```
«nom de la variable» = «départ», «fin», «pas»
```

PERMIS
Les valeurs de l'instruction DO (valeur initiale, valeur finale, incrément) peuvent être des expressions arithmétiques comportant des variables. On a le droit de réutiliser ces variables à l'intérieur de la boucle (les valeurs initiales, finales et l'incrément sont calculés dès l'entrée de la boucle et sont conservés tels quels).

INTERDIT
On n'a pas le droit de rentrer dans une boucle sans passer par l'instruction DO.
Il est interdit de modifier la valeur du compteur de boucle lors de l'exécution de cette boucle.

INITIALISATION
Pour cumuler des valeurs, il ne faut pas oublier d'initialiser les variables. Il n'est jamais garanti que les valeurs non déclarées soient zéro.

L'initialisation doit bien sûr se faire en dehors de la boucle.

Par exemple :

```
^^^^^^Prod=1
^^^^^^DO^X=1,5,1
^^^^^^^^^Prod=Prod*X
^^^^^^ENDDO
^^^^^^WRITE(*,*) 'LA VALEUR DU PRODUIT VAUT ',Prod
```

Autre exemple : soit un programme qui gère un tableau T de N=10 cases, initialisé de telle sorte que chaque case i prenne la valeur i, et affiche les valeurs de ces cases dans l'ordre décroissant des i :

```
PROGRAM tab

INTEGER :: i
INTEGER, PARAMETER :: N = 10
INTEGER, DIMENSION (1 : N) :: T

DO i = 1, N
  T(i) = i
ENDDO

DO i = N, 1, -1
 WRITE(*,*) i, T(i)
ENDDO

END PROGRAM
```

IV.3) Boucles arithmétiques multiples.

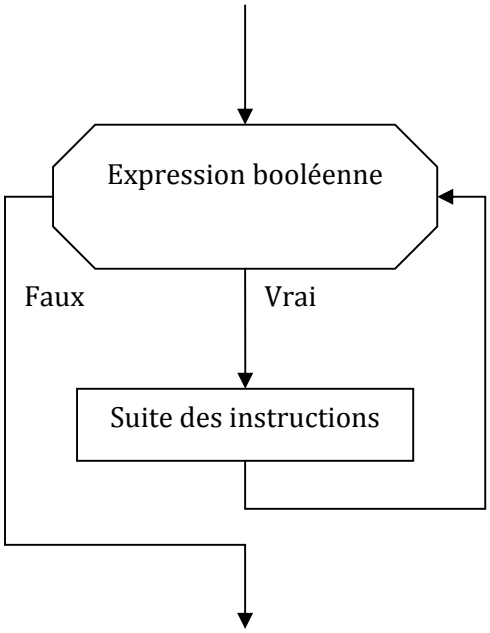
Il est permis d'imbriquer des boucles pourvu qu'elles aient des compteurs différents. Ces boucles ne doivent cependant pas se chevaucher; cela correspondrait à rentrer dans une boucle sans passer par l'instruction DO. Ce sont les boucles internes qui sont achevées en premier.

On peut écrire:

```
^^^^^^DO^I=1,5
^^^^^^^^DO^J=1,3
^^^^^^^^^^^WRITE(*,*) I,J
^^^^^^^^ENDDO
^^^^^^ENDDO
```

IV.4) Les séquences booléennes itératives.

Tant que l'expression booléenne est vérifiée, le programme est dans un processus itératif.



La lecture du programme est facilitée si vous décalez les écritures des instructions vers la droite (colonne supérieure à 7)

```
DO WHILE ( «expression booléenne» )

   … suite des instructions …

ENDDO
```

Le bloc d'instruction compris entre DO WHILE () et ENDDO n'est effectué que si la condition entre parenthèses est remplie. Si les variables entrant dans la condition sont modifiées, les instructions sont recommencées jusqu'à ce que la condition devienne fausse.

Par exemple :

```
      I=1
      N=10
      DO WHILE (I.LE.N)
         WRITE(*,*) I
         I=I+1
      ENDDO
```

Cette écriture remplace une boucle I=1,10

```
^^^^^^I=1
^^^^^^N=10
^^^^^^DO^WHILE (I.LE.N)
^^^^^^^^^^WRITE(*,*) I
^^^^^^^^^^N=N-1
^^^^^^ENDDO
```

Cette écriture remplace une boucle N=10,1

```
^^^^^^READ(*,*)A
^^^^^^DO^WHILE((A.GE.-1.).AND.(A.LE.1.))
^^^^^^^^^^^^X=ACOS(A)
^^^^^^^^^^^^WRITE(*,*) ' L''ANGLE VAUT  ',X
^^^^^^ENDDO
```

Cette écriture remplace un IF () THEN…ENDIF

Exercices:

IV-1) On jette trois dés à jouer. On cherche toutes les combinaisons possibles d'obtenir un total donné (compris entre 3 et 18). Écrire ce programme en évitant les solutions redondantes (1+2+3 +2+1+3).

IV-2) Écrire un programme qui contient la déclaration d'un tableau T à deux dimensions N=3, M=3, et qui initialise ce tableau de telle façon que T(i,j) = i+j. Afficher les valeurs des cases de ce tableau.

IV-3) Écrire le programme PREMIER décomposant un nombre en ses diviseurs, nombres premiers.

*IV-4). **Partie II du projet**.*

IV-5) Calculer le rapport du volume d'une sphère de rayon a par rapport au volume du cube d'arête 2a, en tirant aléatoirement des points dans le cube, en cherchant s'ils sont situés à l'intérieur d'une sphère et en sommant le nombre de réponses positives. En déduire la valeur du nombre π en fonction du nombre de points tirés.

Pour réaliser ce programme, il est nécessaire d'utiliser un générateur de points aléatoires.

Le générateur RAND() donne une liste de valeurs dites aléatoires comprises entre 0. et 1. Cette liste est toujours engendrée de la même façon et donc est toujours la même. Elle est constituée de façon à obtenir une distribution homogène dans un intervalle de nombres.

La variable RAND() va alors contenir un réel compris entre 0 et 1.
Pour obtenir une variable réelle E comprise entre 0 et 1 écrire

E=RAND()

IV-5) Recherche de la racine d'une fonction F(x) par dichotomie:

Le principe est de réduire l'écart (x_a,x_b) entre deux valeurs de x conduisant à des valeurs F(x) opposées. Si l'on a trouvé deux valeurs initiales, F(x_a) et F(x_b) de signe opposé, on calcule F($\frac{x_a+x_b}{2}$) qui est de signe contraire soit de x_a, soit de x_b. La racine est donc comprise dans l'intervalle correspondant aux deux valeurs contraires. On a ainsi divisé par deux l'intervalle (x_a,x_b). En procédant par itération, on obtient un intervalle aussi réduit que l'on veut.

***Écrire l'organigramme**. Réaliser le programme en prévoyant que les premières valeurs (x_a,x_b) ne soient pas forcément satisfaisantes. Tester le programme pour F(x)=sinx+2x-2*

Pour le calcul de F(x) dans le programme, insérer après l'instruction end de la fin de votre programme, les instructions suivantes :

```
Real function F(R)
implicit none
Real R

   F = Sin(R) + 2*R -2

end
```

Un appel de la fonction dans votre programme se fera alors simplement par l'instruction F(x) pour une variable réelle x donnée :

x = 0.5 ! juste pour l'exemple…
y = F(x)

Programme de Chimie

Chimie1 (2h) : *atomistique*

- nucléons et électrons (masse, charge), numéro atomique Z et nombre de masse A, cations et anions, Configuration électronique d'un atome dans son état fondamental et représentation, Couche de valence,
- *notion de chimie2 à faire en autonomie.*

Chimie2 (2h) + 1h de tutorat : *quantification de l'énergie d'un atome*

- hydrogénoïde, spectres des atomes à un électron (hydrogénoïde) – Diagrammes énergétiques,
- *notion de chimie3 à faire en autonomie.*

Chimie3 (1h30) + 1h30 de tutorat : *modèle de Slater*

- méthode de Slater, constante d'écran,
- *notion de chimie4 à faire en autonomie.*

Chimie4 (2h) + 1h de tutorat : *orbitales moléculaires*

- notion de recouvrement de deux OA, principe de construction des OM par combinaison linéaire d'OA de même symétrie, indice de liaison, diagramme d'énergie des OM. Configuration électronique d'une diatomique.

Chimie 4 (2h) + 1h de tutorat : *TP sur la Base de données OrbiMol* - CR noté sur 30 pts

Chimie 1 : *atomistique*

Exercice N°1

1) Indiquer la composition des noyaux et le nombre d'électrons de $^{9}_{4}Be^{2+}$, $^{19}_{9}F^{-}$, $^{208}_{82}Pb$ et $^{16}_{8}O^{2-}$
2) Donner la configuration électronique et une représentation sous forme de cases quantiques de l'état fondamental des atomes suivants (différencier les électrons de valence des électrons de cœur).

$_{7}N$ $_{17}Cl$ $_{19}K$ $_{23}V$ $_{24}Cr$

3) Écrire la configuration électronique des ions suivants dans leur état fondamental

$_{8}O^{2-}$ $_{17}Cl^{-}$ $_{19}K^{+}$ $_{30}Zn^{2+}$

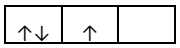
4) Écrire la configuration électronique du lithium $_{3}Li$ dans son état fondamental. Que peut-on dire d'un lithium de configuration électronique $1s^2 2p^1$?

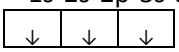
Exercice N°2

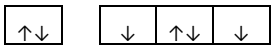
Parmi les configurations électroniques suivantes, quelles sont celles qui ne sont pas acceptables (indiquer la règle qui n'est pas respectée) ?

a) $1s^2 2s^2 2p^8$
b) $1s^2 2s^2 2p^6 3s^2$
c) $1s^2 2s^2 2p^6 3s^2 3p^6 3d^1$
d) $1s^2 2s^2 2p^6 3s^2 3p^6 4s^2 3d^{12} 4p^5$
e) $1s^2 2s^2 2p^6 3s^2 3p^6 4s^2 3d^{10} 4p^6 5s^1$
f) $1s^2 2s^2 2p^6 3s^2 3p^6 4s^2 3d^{10} 4p^6 5s^2 4d^{10} 5p^6 6s^2 5d^6$

g)

h) 

i) 

j) 

k)

Exercice N°3

1) L'uranium a pour numéro atomique Z = 92. Donner la configuration fondamentale de l'élément en appliquant la règle de Klechkovski. Préciser les électrons de coeur et les électrons de valence. Placer l'élément dans la classification périodique.
2) La configuration réelle est :
   $1s^2\ 2s^2\ 2p^6\ 3s^2\ 3p^6\ 4s^2\ 3d^{10}\ 4p^6\ 5s^2\ 4d^{10}\ 5p^6\ 6s^2\ 4f^{14}\ 5d^{10}\ 6p^6\ 5f^3 6d^1\ 7s^2$
   Commenter.
3) L'uranium se trouve surtout aux degrés d'oxydation +IV et +VI. Interpréter à partir de la configuration électronique.
4) D'autres éléments donnent aussi des composés au degré d'oxydation +VI. Parmi ceux-ci, on trouve :
   - le soufre : donner le numéro atomique du soufre et sa configuration électronique sachant qu'il donne des composés stables au degré d'oxydation +VI et qu'il se situe dans la classification périodique entre le néon ($_{10}Ne$) et l'argon ($_{18}Ar$) ;
   - les éléments de transition de la colonne 6 : chrome Cr, molybdène Mo, tungstène W; expliquer pourquoi ces éléments donnent des composés stables au degré +VI, en traitant uniquement le cas du chrome ($_{24}Cr$).

Citer une cause possible de l'origine des différences observées entre les rayons métalliques de certains éléments de la colonne 6 et celui de l'uranium.

Elément	$_{24}Cr$	$_{42}Mo$	$_{92}U$
Rayon métallique (pm)	129	139	156

Notion de chimie 2 : les hydrogénoïdes et leur énergie d'ionisation.

Chimie 2 : *quantification de l'énergie d'un atome*

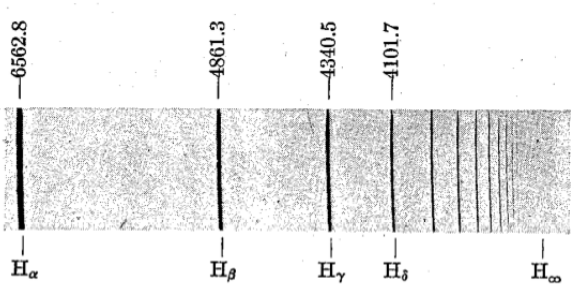
EXERCICE N°1 :

1) Rappeler la définition d'un système hydrogénoïde ainsi que sa formule chimique générale.
2) Les niveaux d'énergie des hydrogénoïdes sont quantifiés par n :

$$E_n = -R_y \frac{Z^2}{n^2}$$, avec R_y= 13,6 eV et Z le numéro atomique.

Représenter schématiquement le diagramme énergétique de l'hydrogène.

3) Indiquer sur le diagramme une absorption, une émission et une ionisation. Préciser le sens physique du niveau électronique de n = ∞.
4) Calculer la différence d'énergie (en eV et en J) entre le niveau n=1 et le niveau n=2.
5) Calculer la longueur d'onde (en m, nm et Å) du rayonnement émis (n=2↘n=1) ou absorbé (n=1↗n=2) lors de la transition entre ces deux niveaux.
6) La figure ci-dessous est un spectre d'émission, dans le domaine visible, de l'hydrogène atomique. Identifier les transitions qui donnent naissance aux raies notées H_α et H_β (les longueurs d'ondes sont indiquées en Å).

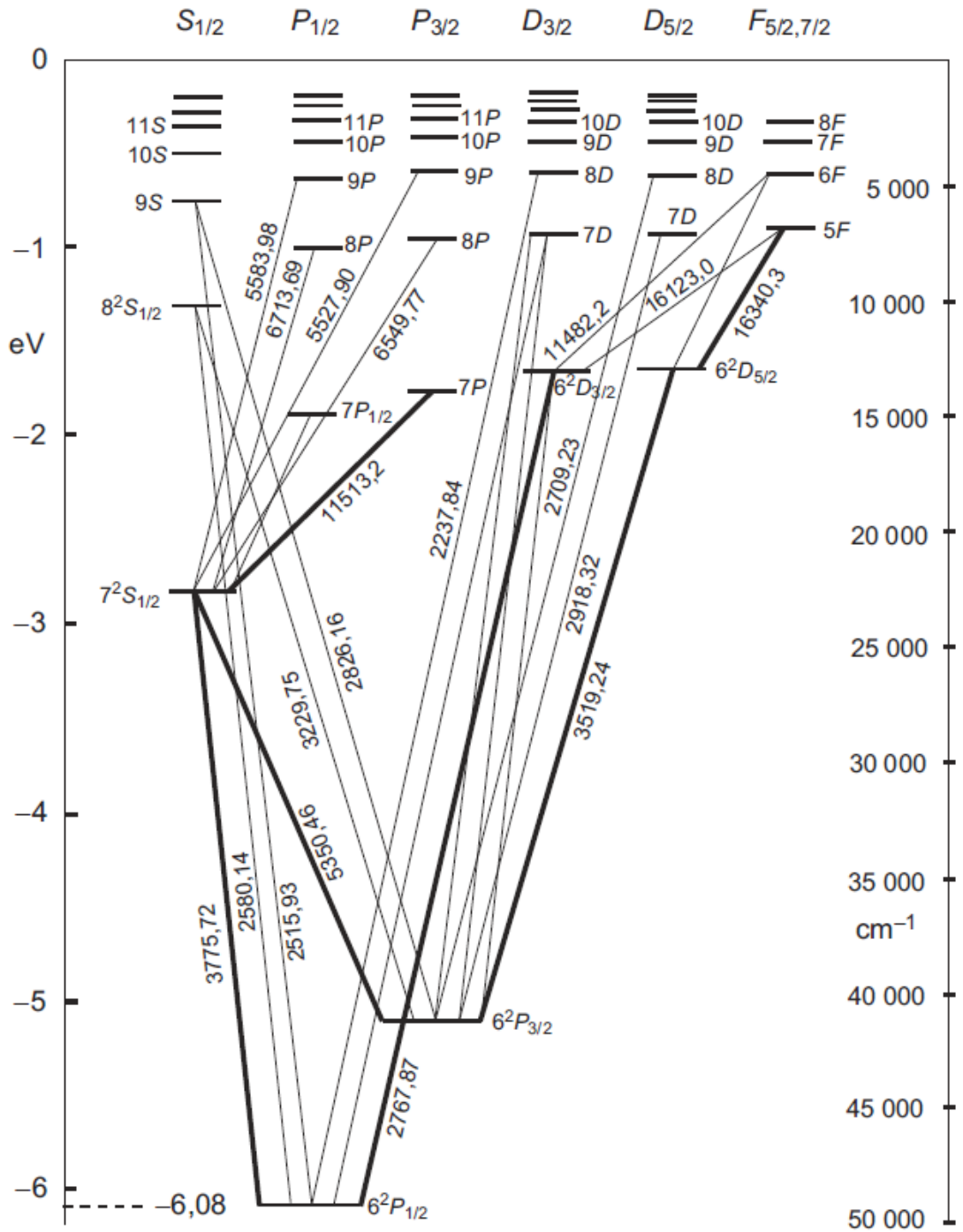


*G. Herzberg, Ann. Physik, **84**, 565 (1927).*

EXERCICE N°2 :

En 1861, en étudiant par spectroscopie les boues résiduelles des chambres de plomb, réacteurs de fabrication de l'acide sulfurique, William Crookes observe la présence d'une raie verte inconnue dans le spectre d'émission. Le nom de thallium est donné en 1862 au nouvel élément responsable de cette émission, du grec thallos = rameau vert. L'élément thallium a pour numéro atomique Z = 81.

1) Quelle est la configuration électronique de l'atome de thallium dans son état fondamental ?
2) Quels sont les degrés d'oxydation les plus probables pour le thallium ?
3) Le schéma représenté sur la figure ci-dessous donne les raies d'émission du thallium. La connaissance de la nomenclature des niveaux d'énergie n'a pas d'importance pour répondre à la question suivante. Quelle est la transition : niveau de départ et niveau d'arrivée, responsable de la raie verte observée par Crookes ?
4) Quelles sont les grandeurs représentées sur les axes verticaux à droite et à gauche ?



Remarques :

• l'intensité des raies est suggérée par l'épaisseur du trait :

• les valeurs indiquées sur les lignes sont les longueurs d'onde exprimées en Ångström (Å) ;

Notion de chimie 3 : la méthode de Slater.

Chimie 3 : *modèle de Slater*

Exercice N°1 :

Pour les atomes et ions polyélectroniques, le modèle approché de Slater exprime l'énergie d'un électron de la couche n_i, par la formule suivante :

$$E_i = -Ry\frac{Z_i^{*2}}{n_i^{*2}} \quad (n_i^* = n_i \text{ pour } n_i \leq 3)$$

Ce modèle reprend la formule donnée pour les hydrogénoïdes au TE précédent, en assimilant le noyau et l'ensemble des électrons autres que i à un "pseudo-noyau" de charge effective Z_i^*.
$Z_i^* = Z - s_i$ où $s_i = S_j s_{ij}$ intègre l'écrantage s_{ij} du noyau par chaque électron j du cortège électronique situé sur la même couche ou une couche plus profonde que celle de i. L'énergie électronique totale de l'atome est obtenue en sommant les énergies individuelles des électrons.

Pour un électron 1s, les règles empiriques de Slater estiment à 0,31 (s_{ij} = 0,31) "l'écrantage" de la charge du noyau par l'autre électron 1s.

1) Calculer la charge nucléaire effective des électrons de l'hélium (He). En déduire l'énergie de ces électrons.
2) Calculer l'énergie électronique totale de He dans son état fondamental.
3) Calculer l'énergie électronique totale de l'ion hydrogénoïde He^+ dans son état fondamental.
4) En déduire la valeur de l'énergie de première ionisation de l'hélium.
5) Comparer cette valeur à celle de l'hydrogène.

Exercice N°2 :

1) Le zinc a pour numéro atomique *Z* = 30. Donner la configuration électronique de l'élément en rappelant le nom des principes et des règles qui permettent d'obtenir cette configuration.
2) Le seul degré d'oxydation stable du zinc est le degré +II. Donner la configuration électronique de l'ion Zn2+. Le zinc est-il un métal de transition ?
3) Calculer les charges nucléaires effectives des électrons de valence de l'atome de zinc Zn et de l'ion Zn2+ a l'aide du modèle de Slater.
4) Expliquer pourquoi la charge effective ressentie par un électron occupant une orbitale 3*d* est plus forte que celle d'un électron occupant une orbitale 4*s*. Prévoir, parmi les orbitales 3*d* et 4*s*, l'orbitale de rayon atomique le plus grand.

Exercice N°3 :

1) L'élément étain, noté Sn, est caractérisé par le numéro atomique Z=50. Écrire sa configuration électronique. Quel élément chimique très répandu possède la même structure externe ?
2) L'étain conduit aux ions stanneux Sn^{2+} et stannique Sn^{4+}.
    a. Donner leur structure électronique.
    b. Calculer les énergies EI_2 (passage de Sn à Sn^{2+}) et EI_4 (passage de Sn à Sn^{4+}) dans le modèle de Slater.

Notion de chimie 4 : les orbitales moléculaires de H_2

Constantes d'écran de Slater

	1s	2s, 2p	3s, 3p	3d	4s, 4p	4d	4f	5s, 5p
1s	0,31							
2s, 2p	0,85	0,35						
3s, 3p	1,00	0,85	0,35					
3d	1,00	1,00	1,00	0,35				
4s, 4p	1,00	1,00	0,85	0,85	0,35			
4d	1,00	1,00	1,00	1,00	1,00	0,35		
4f	1,00	1,00	1,00	1,00	1,00	1,00	0,35	
5s, 5p	1,00	1,00	1,00	1,00	0,85	0,85	0,85	0,35

Chimie 4 : *les orbitales moléculaires*

Exercice N°1

On donne les énergies des orbitales atomiques des atomes de carbone et d'oxygène exprimées en eV.

Carbone : E_{1s} = -307 eV ; E_{2s} = -19 eV ; E_{2p} = -11,7 eV

Oxygène : E_{1s} = -560 eV ; E_{2s} = -33,7 eV ; E_{2p} = -17,1 eV

1) Construire sur un même graphique les diagrammes énergétiques de C et O.
2) Pour une de ces molécules, il y a une forte interaction sp. Laquelle ?
3) L'interaction sp est négligeable pour l'autre. Laquelle ?
4) Construire le diagramme des Orbitales moléculaires de C_2 et de O_2.
5) Ces deux molécules peuvent donner chacune un cation et un anion. Quand on observe les longueurs de liaisons on trouve :

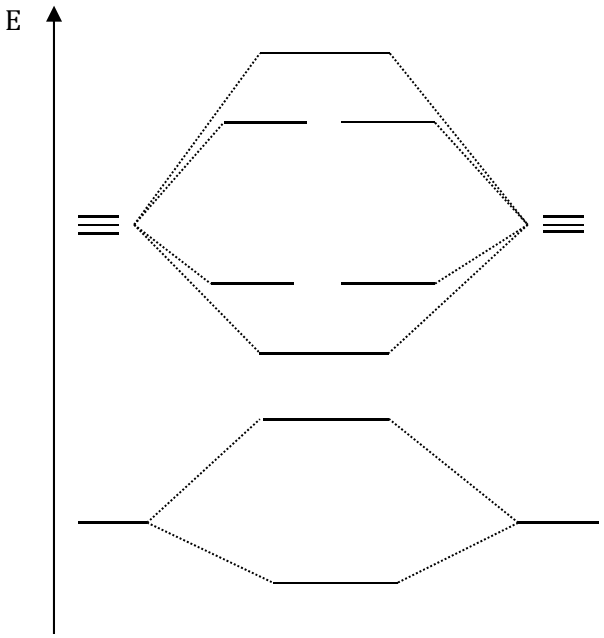
$$d(O_2^-) > d(O_2) > d(O_2^+) \text{ et } d(C_2^+) > d(C_2) > d(C_2^-)$$

Justifier ces résultats.

6) Dans la série O_2, O_2^+ et O_2^-, quelle est la molécule dont l'énergie de dissociation est la plus élevée ? Pourquoi ?

Exercice N°2

Un diagramme d'orbitales moléculaires simplifié de la molécule S_2 est proposé ci-dessous :



1) Donner la structure électronique du soufre (Z = 16) dans son état fondamental.
2) Quelles sont les OA qui participent à la construction du diagramme d'orbitales moléculaires ? Justifier.
3) Rappeler les règles d'interaction qui régissent la construction d'un tel diagramme.
4) Indiquer la nature des OM (σ, π, liantes, antiliantes, non liantes) sur le diagramme d'orbitales moléculaires de S_2 et placer les électrons dans ce diagramme.
5) S_2 est-elle une molécule diamagnétique ou paramagnétique ? Justifier.
6) Calculer l'indice (ordre) de liaison de S_2.

7) Des calculs récents sur la molécule de S2 et ses ions ont conduit aux résultats suivants pour la distance interatomique S-S :

	S_2^{2+}	S_2^+	S_2	S_2^-	S_2^{2-}
d (pm)	172	179	188	200	220

Interpréter l'évolution de cette distance.

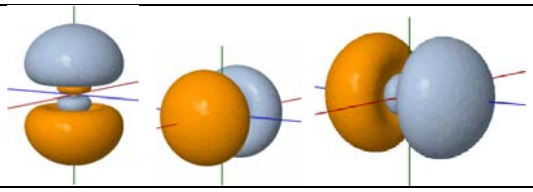
8) Classer par ordre croissant les énergies de la liaison S-S dans S_2^{2+}, S_2^+, S_2, S_2^- et S_2^{2-}. Justifier.

Chimie 5 TP OrbiMol : de l'atome à la molécule, de l'OA à l'OM.

(2 heures)

Partie I : le chrome

1) Donner la configuration électronique de l'atome de $_{24}Cr$ dans son état fondamental
2) Représenter la couche de valence avec le formalisme des cases quantiques.
3) En chimie théorique, la multiplicité de spin mesure la dégénérescence des fonctions d'onde électroniques, c'est-à-dire le nombre de celles qui ne diffèrent que par l'orientation de leur spin. Si S représente la somme des spins des électrons d'un atome, la multiplicité vaut 2S+1. Calculer cette multiplicité pour l'atome de chrome.
4) L'état d'oxydation le plus commun du chrome est +3. Donner la configuration électronique de cet ion.
5) Donner la formule chimique de l'hydrogénoïde du chrome ainsi que sa configuration électronique de son état fondamental.
6) En vous aidant de la représentation tridimensionnelle du chrome dans OrbiMol (métaux d), dessiner des orbitales atomiques du chrome par ordre décroissant d'énergie, indiquer les énergies correspondantes ainsi que les occupations de OA.
7) L'orbitale atomique vide de plus basse énergie (BV) est représentée ci-dessous. Compléter le tableau.

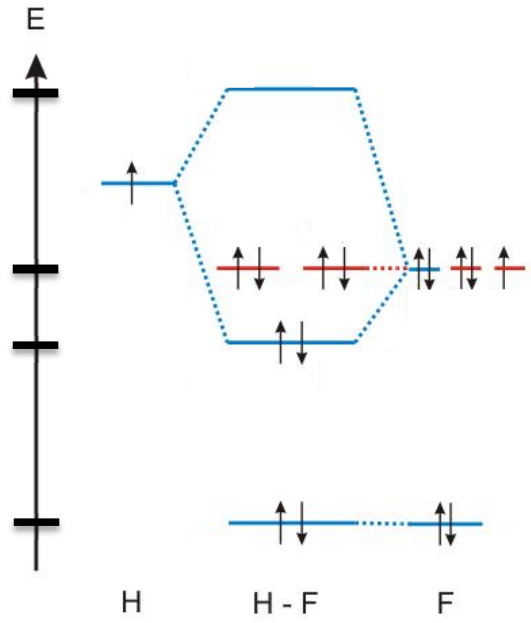
Energie (eV)	O.A.	Dessins	remplissage
			

8) Afficher la HO de la molécule $Cr(CO)_6$ (métaux d). Décrire cette OM à partir de la combinaison linéaire (liante ou antiliante) des OA du chrome, du carbone ($_6C$) et de l'oxygène ($_8O$).

Partie II : le fluor

1) À l'aide du modèle de Slater, estimer l'énergie électronique du Fluor.
2) En vous aidant de la représentation tridimensionnelle du difluor (diatomiques),
    a. Justifier que le diagramme des OM est non corrélé.
    b. Dessiner le diagramme des OM en faisant apparaître les noms des OA, les noms des OM, les énergies des OM, la distribution des électrons sur les OA et les OM.
3) Dessiner et décrire la BV de F_2 à partir de la combinaison linéaire (liante ou antiliante) des OA.
4) Indiquer la configuration électronique fondamentale de cette molécule.
5) Calculer l'énergie de première ionisation et expliquer la différence entre la valeur trouvée et la valeur expérimentale qui est de 15,7eV.
6) Expliquer pourquoi la distance internucléaire F-F varie ainsi $d(F_2^*) \approx d(F_2^-) > d(F_2) > d(F_2^+)$? où F_2^* représente la molécule de F_2 dans son premier état électronique excité..

7) Calculer l'indice de liaison il(F_2) la comparer à l'indice de liaison de O_2. Justifier ces résultats à l'aide de structures de Lewis.
8) En vous aidant de la représentation tridimensionnelle de la molécule OF_2 (triatomiques), dessiner sa HO et expliquer comment construire cette OM à partir des OA de O et des deux F. Est-elle liante ou antiliante ? favorise-t-elle les liaisons O-F ?
9) La molécule FH est une diatomique hétéronucléaire. Dans ce cas, quelles différences y-a-t-il pour les contributions des OA dans les OM ?
10) Compléter le diagramme des OM qui est donné ci-dessous. Que pouvez-vous dire sur la première OM ?



11) Que pouvez-vous dire sur la différence d'électronégativité au sein de cette molécule ? Expliquer cette différence à partir des coefficients des OM liantes et antiliantes ($c_H \pm c_F$) de la molécule HF donnés dans le tableau 3.

<u>Données :</u>

Tableau 1: Énergies calculées de quelques OA (en eV)[1]

	H	Li	B	C	N	O	F
1s	-13,606	-67,424	-209,40	-308,18	-425,28	-562,42	-717,92
2s		-5,3416	-13,461	-19,200	-25,720	-33,859	-42,793
2p			-8,4328	-11,791	-15,448	-17,195	-19,864

Tableau 2 : constantes d'écran de Slater

[1] D. B. Lawson et J. F. Harrison, « Some Observations on Molecular Orbital Theory », Journal of Chemical Education, vol. 82, no 8, 2005, p. 1205

	1s	2s, 2p	3s, 3p	3d	4s, 4p	4d	4f	5s, 5p
1s	0,31							
2s, 2p	0,85	0,35						
3s, 3p	1,00	0,85	0,35					
3d	1,00	1,00	1,00	0,35				
4s, 4p	1,00	1,00	0,85	0,85	0,35			
4d	1,00	1,00	1,00	1,00	1,00	0,35		
4f	1,00	1,00	1,00	1,00	1,00	1,00	0,35	
5s, 5p	1,00	1,00	1,00	1,00	0,85	0,85	0,85	0,35

Tableau 3 : coefficients des OA $1s_H$ et $2p_{z,F}$ dans les OM liante (2σ) et antiliante ($2\sigma^*$) de la molécule HF.

	c_H	c_F
2σ	0,19	0,98
$2\sigma^*$	0,98	-0,19