République algérienne démocratique et populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr. Moulay Tahar de Saïda

Faculté de Sciences

Département de Chimie

Polycopié de cours : Programmation et chimie informatique

Filières : 3ième année licence chimie inorganique

Réalisé par Mr : Hadji Djebar

et Mme Yahia Cherif Fatima

2022-2023

« **Fortran** (*mathematical **FOR**mula **TRAN**slating system*) »

Table des matières

Introduction 8

Chapitre 1 : Introduction au Fortran.....9

1.1 Historique 9

1.2 Définition d'un langage de programmation ……10

1.3 Compilation du programme ……....11

1.4 Le chercheur chef créateur du FORTRAN ……12

1.5 Les caractéristiques majeures du Fortran ……..…..…14

1.6 La différence entre les versions les plus connus de Fortran …….…..15

Chapitre 2 : Notions de base du FORTRAN ……..…17

2.1 Introduction…….17

2.2 Déclaration de données en FORTRAN……..……18

2.2.1 Données numériques ……..……..……18

2.2.2 Données alphanumériques ……..….19

2.2.3 Données logiques ……..……20

2.3 Avantages de Fortran ……..…21

2.4 Inconvénients de Fortran……..…22

2.5 Opérateurs ……..…..22

2.6 Les opérateurs relationnels …….23

2.7 Opérateurs logiques ……...….25

2.8 Relation d'ordre ……..25

2.9 Les variables ……..…26

2.10 Structure d'un programme ……27

2.10.1 Exemple d'application de structure d'un programme ……29

2.11 Les principales commandes en Fortran ……....30

2.12 Les commandes d'intéractions ……....34

2.13 Bibliothèque de fonctions ……..…..36

2.14 Les boucles ……....…38

Chapitre 3 : Applications Fortran……..40

3.1 Application 1 ……40

3.2 Application 2 ……43

3.3 Affichage des erreurs en Fortran ……..45

3.4 Connaissance de la position de l'erreur dans FORTRAN ……46

3.5 Correction de l'erreur dans FORTRAN ……47

3.6 Application 3 ……..…47

3.7 Application 4 ……..…49

3.8 Application 5 ……..……51

Références 54

Liste des tableaux

Tableau 1. Les différents types de données……..21
Tableau 2. Les opérateurs relationnels 23
Tableau 3. Les Opérateurs logiques …….25

Liste des figures

Figure 1. Organisation d'une matrice en mémoire……...…12
Figure 2. Les étapes de la compilation……..12
Figure 3. John Backus (1924-2007), inventeur du FORTRAN……13
Figure 4. Les pionniers du FORTRAN. ……14
Figure 5. Données numériques. ……..…..18
Figure 6. Données alphanumériques. ……..……..…20
Figure 7. Données logiques. ……..……...21
Figure 8. La structure d'un programme Fortran……28
Figure 9. Exemple d'application de la Structure d'un programme……...….30
Figure 10. La commande program……..……...31
Figure 11. La commande END…….32
Figure 12. La commande CONTINUE……33
Figure 13. La commande READ……..……34
Figure 14. La commande Write ……36
Figure 15. Jack Dongarra, créateur de LAPACK. ……37
Figure 16. Des condensateurs qui stockent de l'énergie……40
Figure 17. Programme de l'énergie stockée……..42
Figure 18. La dernière étape du programme de l'énergie stockée……...42
Figure 19. Structure du bleu de bromothymol et en bouteille. ……..…43
Figure 20. Intervalle du PH……...43
Figure 21. Programme PH……44
Figure 22. Exécution du programme PH……..44
Figure 23. Etape « entrer la concentration » du programme PH……..…45
Figure 24. Affichage d'erreur dans un programme Fortran…….46
Figure 25. Indication d'erreur dans un programme Fortran…….46
Figure 26. Correction d'erreur dans un programme Fortran ……47
Figure 27. Niels Bohr (1885- 1962) …….47
Figure 28. Programme Bohr pour n=2……..49
Figure 29. Programme Bohr pour n=3……..49
Figure 30. Programme d'énergie cinétique E_c……..50
Figure 31. Introduction des données au programme……..…51

Figure 32. Programme de l'anisotropie de la polarisabilité α……53
Figure 33. Introduction des données au programme……..53

Introduction générale

Dans ce polycopié de cours, je présente une partie du module de programmation et chimie informatique. Il est destiné aux étudiants des troisièmes années de la licence chimie inorganique de la faculté des sciences. Il est conforme au programme du ministère de l'enseignement supérieur et de la recherche scientifique. Ces cours seront développés et complets par des séries des travaux pratiques et même d'exercices détaillés. Ces TP ont été fait sous Windows pour permettre une bonne assimilation des diverses notions de la programmation avec le langage de programmation Fortran. Les différentes notions ont été présentées dans cette polycopie. Les commandes principales et les plus utilisées au Fortran ont été présentées aussi. Ce polycopié est le fruit de mes efforts, que ce document est personnel, et cite en référence toutes les sources utilisées.

Chapitre 1

Introduction au Fortran

1.1 Historique

Le nom FORTRAN est dérivé de *Formula Translating System* [1], *Formula Translator,* ou *Formula Translation*. **Fortran** (mathematical FORmula TRANslating system) est un langage de programmation généraliste dont les domaines scientifiques comme la

physique, la chimie, et le calcul numérique lourd. On peut utiliser le Fortran sur ordinateur personnel PC que sur les superordinateurs.

Le langage de programmation FORTRAN est destiné à aider les scientifiques comme les physiciens et les chimistes à avoir un moyen simple pour passer de leurs formules mathématiques (algorithmes) jusqu'à un programme effectif. Ce langage de programmation est très efficace dans le domaine du calcul numérique lourd, et offre de nombreuses bibliothèques de programmes d'analyse numérique. Ce langage a eu droit à plusieurs normalisations; FORTRAN 77, FORTRAN 90, FORTRAN 95, et plus récemment FORTRAN 2003 (FORTRAN 03).

Le développement du langage FORTRAN a commencé en 1953 par IBM [2]. John Backus, pionnier de l'informatique dans les années cinquante, publie en 1954 un article titré Preliminary Report Specifications for the IBM Mathematical FORmula TRANslating System, FORTRAN, qui est le premier document de référence au langage FORTRAN est écrit par Jean Sammet [3]. La première version du FORTRAN est apparue après deux ans de travail de l'équipe d'IBM dirigé par John Backus et exactement en 1957. Même si les programmes obtenus à partir de code FORTRAN étaient plus lents que ceux obtenus à partir de codes en langage machine, le FORTRAN s'est imposé auprès de la communauté scientifique.

Après six ans, il y eu une première tentative de normalisation du langage (travaux du Working Group X3.4.3 de l'American Standards Association) mais le groupe de travail n'a pas réussi à s'entendre sur un standard unique. C'est ainsi que sont nées deux normalisations distinctes : FORTRAN et Basic FORTRAN. En 1978, le même groupe de travail s'est de nouveau réuni et les spécifications du FORTRAN 77 furent adoptées. Il est depuis maintenu, de nouvelles spécifications sortant de temps en temps.

1.2 Définition d'un langage de programmation

On voit donc qu'il manque un lien entre l'homme et la machine, un langage commun. C'est le langage de programmation.

Un langage de programmation c'est le lien entre l'homme et la machine. Est un langage informatique destiné à formuler des algorithmes qui sont la résolution mathématique du problème et produire des programmes informatiques qui les appliquent [4]. Un langage de programmation est le moyen de communication par lesquels l'homme communique avec l'ordinateur. D'une manière similaire à une langue naturelle, un langage de programmation est

composé d'un alphabet, d'un vocabulaire, de règles de grammaire, de significations, mais aussi d'un environnement de traduction censée rendre sa syntaxe compréhensible par la machine. L'importe quel langage de programmation donne les mêmes résultats. C'est-à-dire, soit on fait un programme en Fortran, en Pascal, C, ou C++, les résultats sont les mêmes. C'est comme le cas dans les communications, on parle soit en arabe, en français, ou en anglais, le rôle c'est la communication entre nous.

Un langage est constitué par [5]:
– un ensemble de commandes
– un ensemble d'objets manipulables, éventuellement extensible
– des règles de syntaxe
– de structures logiques Programmer, c'est écrire un texte respectant les règles du langage, susceptible de résoudre un problème donné.

1.3 Compilation du programme

Une fois on termine de l'écriture du programme en Fortran. Il reste la compilation de ce programme. Le texte est ensuite vérifié et traduit en une suite de codes machine par l'intermédiaire d'un compilateur. Si le texte est incorrect, le compilateur indique les erreurs de compilation (écrite en rouge dans à la fin de la fenêtre après l'exécution).
Ces fautes d'orthographes et de grammaire peuvent être corrigées facilement à l'aide du Fortran.
L'exécution du programme par exemple en Fortran, c'est faire dérouler par la machine, une séquence de codes machine ainsi créée. Il est malheureusement rare qu'un programme soit en Fortran où d'autre langage de programmation fonctionne du premier coup, et qu'il fournisse exactement le résultat escompté : il présente des dysfonctionnements qu'on appelle des « bugs ». On dispose en général d'un outil appelé debugger (Figure 1 et Figure 2), qui permet de faire tourner le programme par petits bouts, et de donner pour chaque erreur un signe pour faciliter la correction de l'erreur afin de repérer et corriger les erreurs.

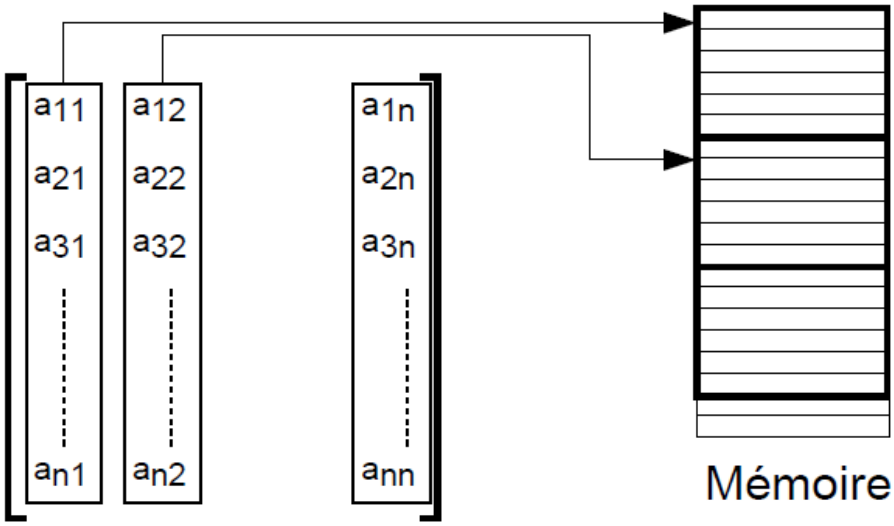


Figure 1. Organisation d'une matrice en mémoire.

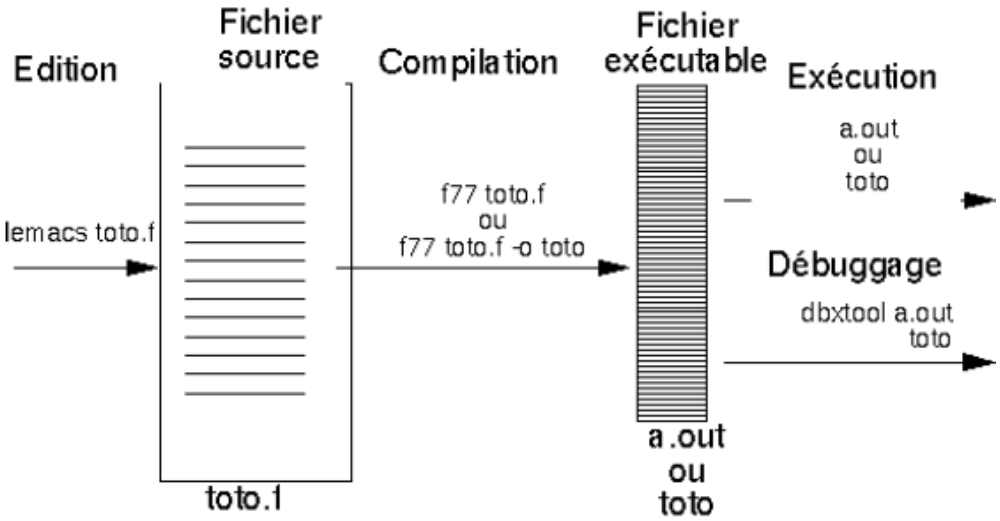


Figure 2. Les étapes de la compilation

1.4 Le chercheur chef créateur du FORTRAN

L'informaticien américain John Warner Backus (Figure 3), né à Philadelphie le 3/11/1924 et mort à Ashland (Oregon) le 17/03/ 2007. Il est directeur de l'équipe qui pour la première fois crée le langage de programmation de haut niveau (le Fortran). John Warner Backus a aussi mené de nombreux travaux de recherche sur la programmation fonctionnelle, qu'il a contribué à populariser.

Le chercheur chef créateur du FORTRAN reçoit le prix Turing en 1977 pour « son influence et sa contribution au développement des langages de programmation de haut niveau, notamment le FORTRAN, et pour les travaux de recherche des procédures formelles servent à la spécification des langages de programmation ». Il prend sa retraite en 1991 et meurt chez lui à Ashland en Oregon le 17 mars 2007. John Warner Backus ne travaille pas seul, mais avec un groupe FORTRAN historique composé des programmeurs (Figure 3) Richard Goldberg, Sheldon F. Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Harold Stern, Lois Haibt et David Sayre [4].

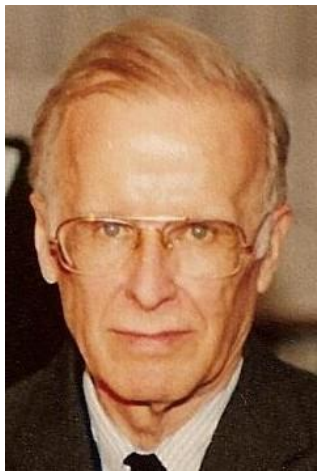


Figure 3. John Backus (1924-2007), inventeur du FORTRAN

Le nom du langage est généralement écrit en majuscules (FORTRAN) pour désigner les versions du langage antérieures à la norme Fortran 90, car à l'époque les lettres minuscules ne font pas partie du jeu de caractères du langage. Par contre, il est toujours écrit avec une majuscule à partir de Fortran 90. Le FORTRAN 77 était la dernière version dans laquelle le jeu de caractères Fortran ne comprenait que des lettres majuscules. Et donc, plusieurs versions du Fortran sont connus, parmi ces versions, les principales sont FII (58), FIII (58), FIV (61), F66, F77, et F90. Il évolue encore : F95, F03 et même F08. On note que le langage de programmation FORTRAN fait l'objet d'un développement actif. La dernière révision du langage est Fortran 2018. La prochaine révision, FORTRAN 2023, devrait être publiée en 2023 selon (https://fortran-lang.org/fr/index).

Figure 4. Les pionniers du FORTRAN.

1.5 Les caractéristiques majeures du Fortran

Le langage de programmation Fortran a des diverses caractéristiques, parmi ces avantages on cite :

- Il était bien plus simple à écrire, en effet, ce langage de programmation a un vocabulaire relativement restreint et est étonnamment facile à apprendre et à utiliser, et à écrire. Comme par exemple, l'écriture de la plupart des opérations mathématiques et arithmétiques (l'addition, la soustraction, la multiplication et d'autres) sur des tableaux de grande taille se fait naturellement, comme sur le papier simple.
- D'autres avantages du Fortran, c'est que ce langage peut être implémenté sur d'autres plateformes que celles d'IBM. Au début des années 1960, est apparue une myriade de compilateurs FORTRAN qui n'obéissaient pas exactement aux mêmes conventions de syntaxe.
- Avec le Fortran, on peut faire des calculs à haute performance. Ce langage a été conçu dès le départ pour des applications de calcul intensif en sciences et ingénierie. Des

compilateurs et des bibliothèques matures et éprouvés vous permettent d'écrire rapidement un code exploitant au mieux la puissance de votre matériel.

- Ce langage de programmation parallèle utilisant une syntaxe intuitive de type tableau pour communiquer les données entre processeurs, qui dite un langage de programmation parallèle. On peut exécuter quasiment le même code que ce soit sur :
    - un seul processeur
    - sur un système multi-cœur à mémoire partagée
    - sur un système HPC à mémoire distribuée,
- ou même sur un système du type Cloud.
- Les concepts de co-tableaux, équipes, événements et sous-routines collectives vous permettent d'utiliser différents types de programmation parallèle adaptés à vos applications
- Le Fortran est l'outil optimal et la solution adéquate pour un programme ou une bibliothèque qui doit effectuer des calculs arithmétiques rapides sur de grands tableaux,
- Ce langage de programmation est principalement utilisé dans des domaines du calcul numérique, c'est-à-dire dans les sciences physiques, chimiques, et l'ingénierie. Par exemple la prévision météorologique et océanique, la mécanique des fluides, les mathématiques appliquées, les statistiques et la finance.
- On note que ce langage de programmation domine dans le calcul haut performance (HPC).

- Le FORTRAN est l'outil optimal et la solution adéquate pour tester la capacité des supercalculateurs les plus puissants au monde où la majorité des librairies de calcul sont écrites avec ce langage ; par exemple, BLAS et LAPACK, qui servent aux tests de supercalculateurs, pour déterminer leur puissance en GFlops. Ces librairies ont été portées vers le langage de programmation C, mais elles étaient prévues, à l'origine, pour le Fortran.

1.6 La différence entre les versions les plus connus de Fortran

Il existe plusieurs versions différées d'une version à une autre, et cette différence entre ces versions c'est le rôle du développement de ces versions du FORTRAN. Dans cette partie, on cite la différence entre FORTRAN II, FORTRAN III, FORTRAN IV, FORTRAN V, FORTRAN 66, FORTRAN 77, FORTRAN 90, FORTRAN 95, FORTRAN 2003, et le FORTRAN 2003.

- En 1956, le **FORTRAN II** n'avait qu'une seule instruction de branchement « IF » à 3 adresses

  Exemple: IF (k-l) 5, 10, 15 indiquaient de sauter aux instructions d'étiquette 5, 10 ou 15 selon que k-l était négatif, nul ou positif.
- En 1958, **FORTRAN III** n'est jamais "sorti" sous forme de produit.
- En 1962, **FORTRAN IV** l'introduction du l'instruction « IF logique », permettant d'écrire

  Exemple :

  IF (k .GE. l) GOTO 2 (aller à 2 si k est supérieur ou égal à l).
- **FORTRAN V** était le nom envisagé au départ pour PL/I, langage de programmation universel d'IBM qui devait réunir les meilleurs aspects de Fortran (pour les applications scientifiques), de COBOL (pour les applications de gestion), avec quelques emprunts à Algol.
- 1966, **FORTRAN 66** est la première version officiellement standardisée (par l'American Standards Association) de FORTRAN. On la confond souvent avec FORTRAN IV.
- 1977, **FORTRAN 77**, cette version facilite la programmation structurée avec des blocs "IF (...) THEN / ELSE / ENDIF". En 78, une extension a été ajoutée DO WHILE / END DO.
- 1990, **FORTRAN 90** : dans cette version, l'introduction de nouveaux types de données, modules, récursivités, et surcharge des opérateurs, … etc. C'est une mise à jour importante pour mettre le langage FORTRAN au niveau des autres langages modernes. Les restrictions concernant la mise en forme des programmes (colonnes 1 à 7, 72 à 80 ...) disparaissent : l'écriture se fait enfin en format libre.
- 1995, **FORTRAN 95**
- 2003, **FORTRAN 2003** : comme le COBOL, Fortran supporte maintenant la programmation orientée objet.
- 2008, **FORTRAN 2008**

Chapitre 2

Notions de base du FORTRAN

2.1 Introduction

Comme la majorité des langages de programmation, le FORTRAN utilise des règles strictes pour fonctionner. Dans cette partie, nous présentons les règles de typage des données (les données numériques, les données alphanumériques, et les données logiques) dans un premier temps.

Ensuite, nous verrons les conventions d'écriture d'un programme FORTRAN nécessaires à sa compilation et à son exécution.

2.2 Déclaration de données en FORTRAN

Ce langage de programmation possède trois types de données :

- Données numériques
- Données alphanumériques
- Données logiques.

On note que chaque type de donnée doit être utilisé avec des règles.

2.2.1 Données numériques : Les données numériques se subdivisent en entiers, rationnels, et complexes (Figure 5)

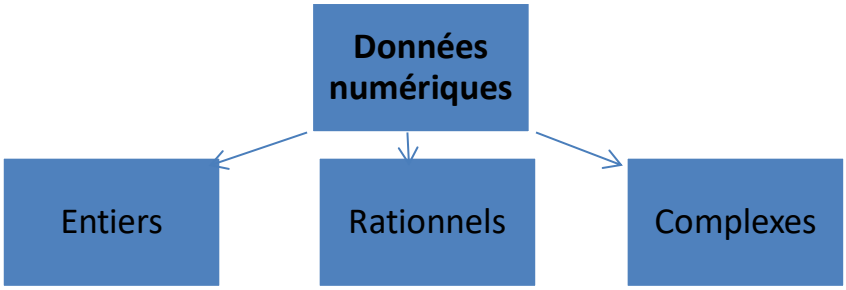


Figure 5. Données numériques.

Le type INTEGER

En Fortran, l'écriture des entiers qui sont les nombres appartenant à Z c'est simple ; il suffit, simplement, d'écrire directement leur valeur avec ou sans signe. Dans ce cas on n'utilise point décimal (les rationnels ne sont pas des entiers).

Exemple :

0	10	4510
-5	-4	1000

Le type REEL

En Fortran, l'écriture des nombres réels (appartenant à Q) ça fait de deux façons : la méthode classique et avec la méthode condensée avec une mantisse et un exposant

Exemple :

1.2514 -1.2 1.4e-18

0. 1.0 1. 3.1415 31415e-4

1.6e-19 1e12 .001 -36.

Le type COMPLEXE

Puisque le langage de programmation Fortran est conçu pour être utilisé en mathématiques et dans les applications de calcul scientifique, il fournit un type standard pour les nombres complexes. En Fortran, l'écriture des nombres complexes est facile et directe. C'est pour cette raison le FORTRAN est conçu pour les scientifiques à cause de leur simplicité. Et donc, pour un nombre complexe, la partie réelle et la partie imaginaire du nombre complexe sont considérées comme deux nombres rationnels assemblés entre parenthèses et séparés par une virgule. Les possibilités de représentation de la partie réelle et de la partie imaginaire du nombre complexe correspondent exactement à celles des données du type real.

Exemple :

(2,1) (5,-1) (0.33,-4)

Et donc, une valeur du type complexe est alors écrite comme suit : (1,2) ce qui correspond au nombre: 1 + 2i

Constantes double précision

Une constante double précision doit obligatoirement être ´écrite en virgule flottante, le e étant remplacé par un d.

Exemples

0d0 0.d0 1.d0 1d0 3.1415d0 31415d-4
1.6d-19 1d12 -36.d0

2.2.2 Données alphanumériques

Utilisant le Fortran, on peut déclarer plusieurs données alphanumériques comme les mots, phrases, paragraphes ou même simples caractères.

Les données alphanumériques s'écrivent de deux façons différentes qui sont équivalentes : la notation de Hollerith et la notation entre quotes (Figure 6).

Notation d'Hollerith : C'est un lointain héritage des premiers compilateurs FORTRAN. On calcul combien de caractères dans la chaine, suivi de la lettre « H », puis la chaîne elle-même est écrite.

Exemple :

10HSOLUTIONS ACID 9HSOLUTIONS 7Hliaison

Dans le premier exemple 10HSOLUTIONS ACID, la chaîne comporte 10 caractères. L'espace au milieu est compris dans le comptage.

Notation entre quotes : cette méthode est facile d'écrire une donnée alphanumérique dans un programme Fortran par rapport à la méthode d'**Hollerith**. Il suffit de mettre la chaîne entre quotes.

Exemple :

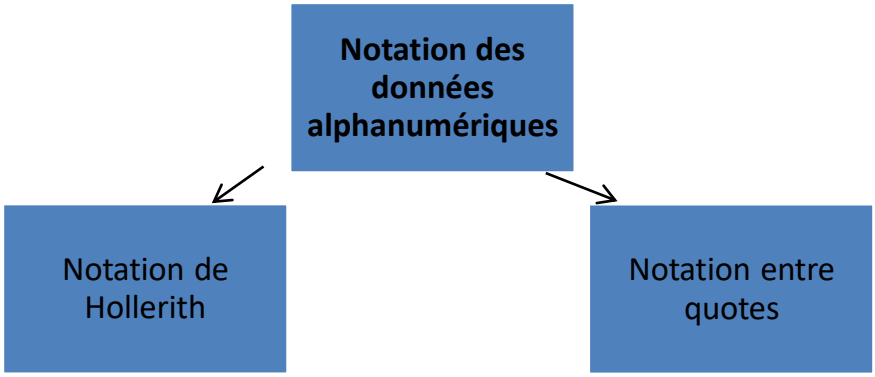


Figure 6. Données alphanumériques.

2.2.3 Données logiques

Les données logiques (Figure 7) en Fortran subdivisent en deux, soit vrai soit faux, et s'écrit soit comme :

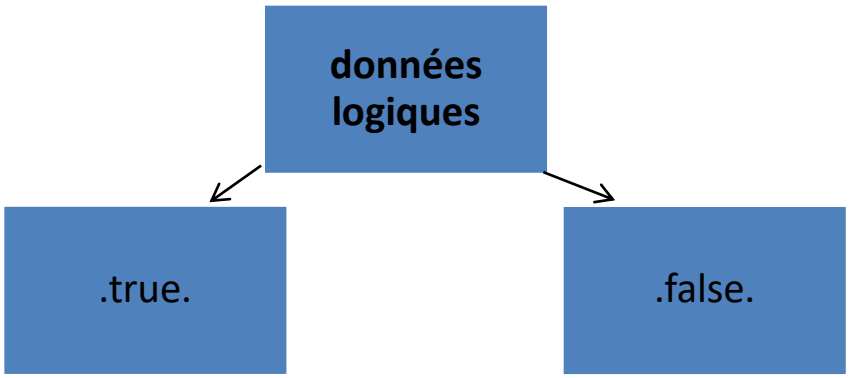


Figure 7. Données logiques.

Les différents types de données sont rassemblés dans ce tableau comme suit :

Tableau 1. Les différents types de données

déclaration	type des variables
integer	entier
real	réel
double precision	réel en double précision
complex	complexe
double complex	complexe en double précision
character	chaîne de caractères
logical	variable logique

2.3 Avantages de Fortran

- Plus facile à apprendre que ses prédécesseurs.
- Il est toujours utilisé comme l'un des langages les plus importants lorsqu'il s'agit d'effectuer des calculs numériques.
- Il est considéré comme une révolution et le début de la programmation moderne.
- Sa mise en œuvre et des années d'utilisation ont abouti à des bibliothèques éprouvées et efficaces qui confirment son efficacité en tant que langage de programmation.

2.4 Inconvénients de Fortran

Ses inconvénients doivent également être pris en compte lors de son utilisation :

- C'est un langage de programmation dans lequel il n'y a pas de classes ou de structures.
- Il rend impossible de faire une réservation dynamique de mémoire.
- Pour le traitement de textes, de listes et de structures de données très complexes, il s'agit d'un langage quelque peu primitif.
- Il s'agit d'un langage compilé **« ahead-of-time »**, ce qui signifie qu'une étape spéciale de compilation doit être effectuée sur le code écrit avant de pouvoir l'exécuter sur un ordinateur. C'est une différence majeure avec les **langages interprétés comme Python et R**, qui s'exécutent par le biais d'un interprète exécutant directement les instructions. L'inconvénient est que l'interprète réduit la vitesse de calcul.

2.5 Opérateurs

Le FORTRAN possède six opérateurs arithmétiques dont il faut connaître leurs différentes priorités.

L'addition : C'est l'utilisation de l'opérateur +

Exemple :

a+b

La soustraction : C'est l'utilisation de l'opérateur _

Exemple :

a-b

La multiplication : C'est l'utilisation de l'opérateur *****

Exemple :

a*b

La division : C'est l'utilisation de l'opérateur /

Exemple :

a/b

Si le numérateur a ou le dénominateur b est un rationnel, le résultat a/b est rationnel. Si ce sont des entiers, c'est en fait une division euclidienne et le résultat un entier.

La puissance : C'est l'utilisation de l'opérateur **

Exemple :

a**b

Il permet de calculer $\mathbf{a^b}$ comme les deux nombres peuvent être des rationnels, il est possible de calculer des exposants fractionnaires. L'opérateur puissance ** a priorité sur l'opérateur produit * et la division / qui ont elles-mêmes priorité sur l'opérateur somme + et l'opérateur différence -.

La négation : C'est l'opérateur unaire -

L'affectation :

Ce n'est pas un opérateur arithmétique mais c'est un opérateur pour le langage FORTRAN. On peut donc écrire :

VAR = NOMBRE

VAR1 = VAR2

Il faut, bien sûr, que les deux membres VAR1 et VAR2 de l'affectation soient de types compatibles.

2.6 Les opérateurs relationnels

Les opérateurs relationnels servent à comparer des variables numériques, et fournissent des résultats du type LOGICAL. Ce sont des opérateurs binaires entre variables numériques ou de type caractère.

Tableau 2. Les opérateurs relationnels

Opération	**opérateurs**
strictement plus petit	.LT. ou <
inférieur ou égal	.LE. ou <=
strictement supérieur	.GT. ou >
supérieur ou égal	.GE. ou >=
égal	.EQ. ou ==
différent	.NE. ou / =

.gt. > (*greater than*)

.lt. < (*less than*)

.ge. ≥ (*greater or equal*)

.le. ≤ (*less or equal*)

.eq. = (*equal*)

.ne. ≠ (*not equal*)

.and. et

.or. ou

.not. contraire d'une expression logique

Dans le cas des opérateurs logiques, ne pas oublier les points de part et d'autre des opérateurs logiques.

Exemple :

x = y en Fortran s'écrit : x.eq.y

x ≠ y en Fortran s'écrit : x.ne.y

x > y en Fortran s'écrit : x.gt.y

x < y en Fortran s'écrit : x.lt.y

x ≥ y en Fortran s'écrit : x.ge.y

x ≤ y en Fortran s'écrit : x.le.y

On peut combiner ces expressions entre elles avec les opérateurs logiques usuels :

Ou .or.

Et .and.

Ou exclusif .xor.

Négation .not.

2.7 Opérateurs logiques

Les opérateurs logiques permettant de combiner des expressions de type LOGICAL entre elles et fournissent des résultats du type LOGICAL. Dans le tableau suivant, on présente les opérateurs les plus connus.

Tableau 3. Les opérateurs logiques

Opération	**opérateurs**
négation	.NOT.
et logique	.AND.
ou logique	.OR.
équivalence logique	.EQV.
non équivalence logique	.NEQV.

2.8 Relation d'ordre

Définition : une relation d'ordre dans un ensemble est une relation binaire dans cet ensemble qui permet de comparer ses éléments entre eux de manière cohérente. Un ensemble muni d'une relation d'ordre est un ensemble ordonné.

En Fortran, ces opérateurs comparent deux données du type INTERGER, REAL ou CHARACTER en fonction d'un certain critère pour déterminer si le résultat est vrai ou faux. Leur syntaxe est :

Operande1 OPERATEUR Operande2

Exemple :

20 .LE. 11.0

L'application de cet opérateur est fausse puisque 20 est supérieur à 11.0. Notons que le FORTRAN a converti l'entier 20 en rationnel pour effectuer la comparaison. De la même façon, il est possible de comparer des chaînes de caractères.

2.9 Les variables

Une variable est une case mémoire référencé par un nom, dans lequel on peut lire et écrire des valeurs au cours du programme.

Les variables en case mémoire permettent entre autres de :

– manipuler des symboles dans un programme, Fortran ;

– programmer des formules simples ou même compliquées.

Avant d'utiliser une variable dans un programme en Fortran, il faut :

– définir son type ;

– lui donner un nom. C'est la déclaration.

Elle doit être écrite dans la première partie (la partie déclaration) d'un bloc fonctionnel (programme principal, subroutine ou fonction) dans lequel intervient la variable.

Dans les langages modernes, toute variable doit être déclarée. En FORTRAN, il y a des exceptions obéissant à des règles bien précises qui sont :

- Seules les lettres (de A à Z) et les chiffres (de 0 à 9) sont autorisés.
- Le premier caractère du nom doit être une lettre.
- La taille des noms de variables ne doit pas dépasser 6 caractères.

2.9.1 Variables typées

Comme a été définie précédemment, une variable en Fortran est un emplacement dans la mémoire vive (RAM) de l'ordinateur, dans laquelle on va pourvoir stocker des valeurs. Pour chacun des types de données il existe un mot-clé permettant de déclarer une variable du type désiré.

En Fortran, On distingue en Fortran essentiellement quatre types de variables :

- les caractères (character),
- les nombres réels (real)
- les nombres entiers (integer),

- et les variables logiques vrai/faux (logical).

Pour les variables entiers :	INTEGER Liste-de-variables
Pour les variables réels :	REAL Liste-de-variables
Pour les variables complexe :	COMPLEX Liste-de-variables
Pour les variables booléen :	CHARACTER a1*b1,a2*b2, . . . ,aN*bN

Si on déclare des variables de type chaîne de caractères. b1,b2, . . . ,bN représentent les nombres de caractères stockables dans chacune des variables. Si toutes les chaînes de caractères doivent avoir la même taille, il est aussi possible d'écrire :

CHARACTER*N Liste-de-variables

En effet, toute variable en Fortran doit être déclarée avant d'être utilisée. Les déclarations doivent être placées immédiatement après la commande 'implicit none'.

Déclaration implicite

D'autre part, il est possible de déclarer un ensemble de noms de variables d'un certain type. L'instruction IMPLICIT permet de déclarer que toutes les variables dont le nom commence par une certaine lettre sont d'un même type. Sa syntaxe est :

IMPLICIT Type (L1-L2)

Avec : L1 et L2 sont les deux lettres extrêmes.

Type, le type commun

2.9.2 Variables non déclarées

En FORTRAN, si une variable est utilisée sans avoir été déclarée, son type dépend de la première lettre de son nom. Si son nom commence par :

I, J, K, L, M ou N donc c'est un entier.

Sinon c'est un rationnel.

Il est toutefois **fortement** recommandé de toujours déclarer les variables de manière explicite.

2.10 Structure d'un programme

Comme pour tout autre langage de programmation soit Delphi, C, ou C++, la structure d'un programme en FORTRAN obéit à certaines règles.

Dans une page Fortran qui comporte 80 caractères (C'était le format des cartes perforées), il existe quatre champs différents, on peut dire qu'elle comprend plusieurs zones (du 1er au 4ème champ) (Figure 8)

- De la 1ère ligne au 5ème : le champ dit de référence. 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. Dans ce champ, les colonnes 1 à 5 contiennent éventuellement des étiquettes, c'est-à-dire un nombre repérant une ligne et qui sert de référence ailleurs.
- Le 6ème : indique que la ligne actuelle n'est que la suite de la ligne précédente
- Du 7ème aux 72 caractères : cette partie contient véritablement les commandes FORTRAN (c t d la solution mathématique du problème) ou contiennent l'instruction proprement dite.
- Du 73ième aux 80 caractères : cette partie contient des commentaires des lignes de commandes. C'est le dernier champ. Ce champ contient des commentaires qui sont ignorés par le compilateur. Vous pouvez y mettre un commentaire; elles servaient autrefois pour numéroter les cartes.

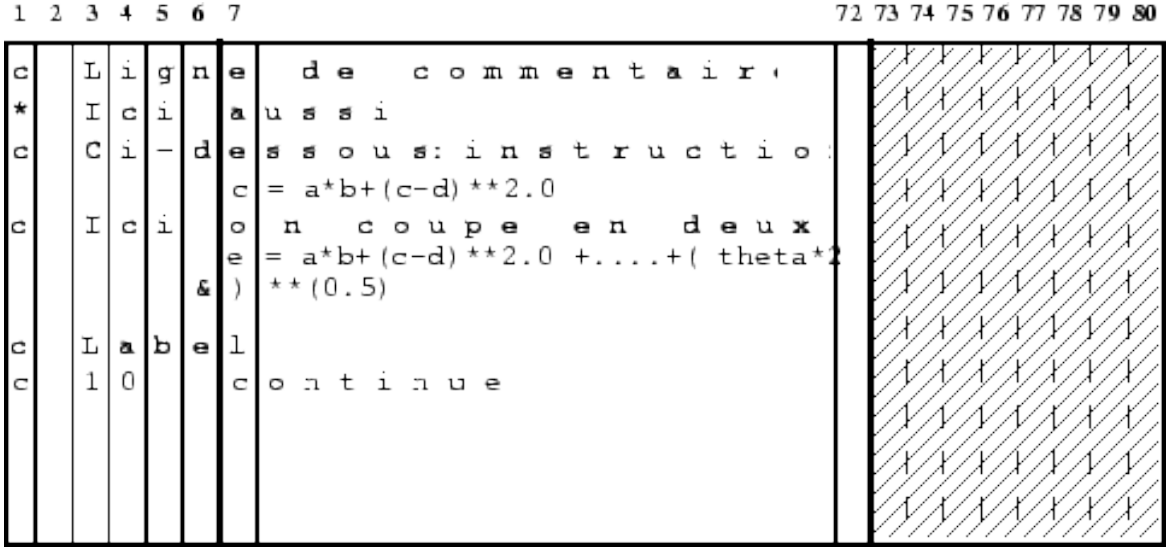


Figure 8. La structure d'un programme Fortran

Le programme est structuré de la manière suivante :

− Le mot PROGRAM suivi d'un titre (cette commande n'est pas obligatoire, c'est-à-dire, on peut ignorer cette étape dans un programme en Fortran

− La partie déclarations des données et des variables. On note que le FORTRAN est un peu neuneu. C'est dans cette partie qu'on définisse les objets (type + nom) qui seront manipulés par le programme. Cette partie est nommée la partie déclarative contient les instructions relatives aux déclarations de variables, de constantes, de paramètres et de formats. La plupart de ces instructions seront développées dans les chapitres qui vont suivent.

Alors il faut lui dire clairement au début du programme le nom et le type des variables que l'on va utiliser

− La partie lecture des données et du traitement des données : cette étape consiste à vérifier que la source respecte les règles de syntaxe du langage Fortran. L'exécution d'un programme FORTRAN consiste à dérouler dans l'ordre toutes les instructions de la partie exécutable du programme principal. Certaines instructions déroutent le pointeur de programme vers d'autres blocs fonctionnels (subroutines ou fonctions). Cette partie est nommée la partie fonctionnelle contient véritablement l'algorithme du programme.

− Le mot END pour signaler la fin du programme

2.10.1 Exemple d'application de la Structure d'un programme :

Dans l'exemple suivant (Figure 9), on présente un programme qui calcul l'énergie cinétique E_c.

Le programme est structuré de la manière suivante :

1- Le nom du programme Program EMC2

2- La partie **déclaration**

```
C Declaration:
      REAL    E,M,C
      C=3.E08
```

3- C Corps du programme: c'est la partie lecture des données et du traitement des données :

```
      Write(*,10)
10    Format('    Entrez la masse :')
      Read(*,*) M
```

E=M*C*C

Write(*,*)'E=',E

4- En fin, la commande END

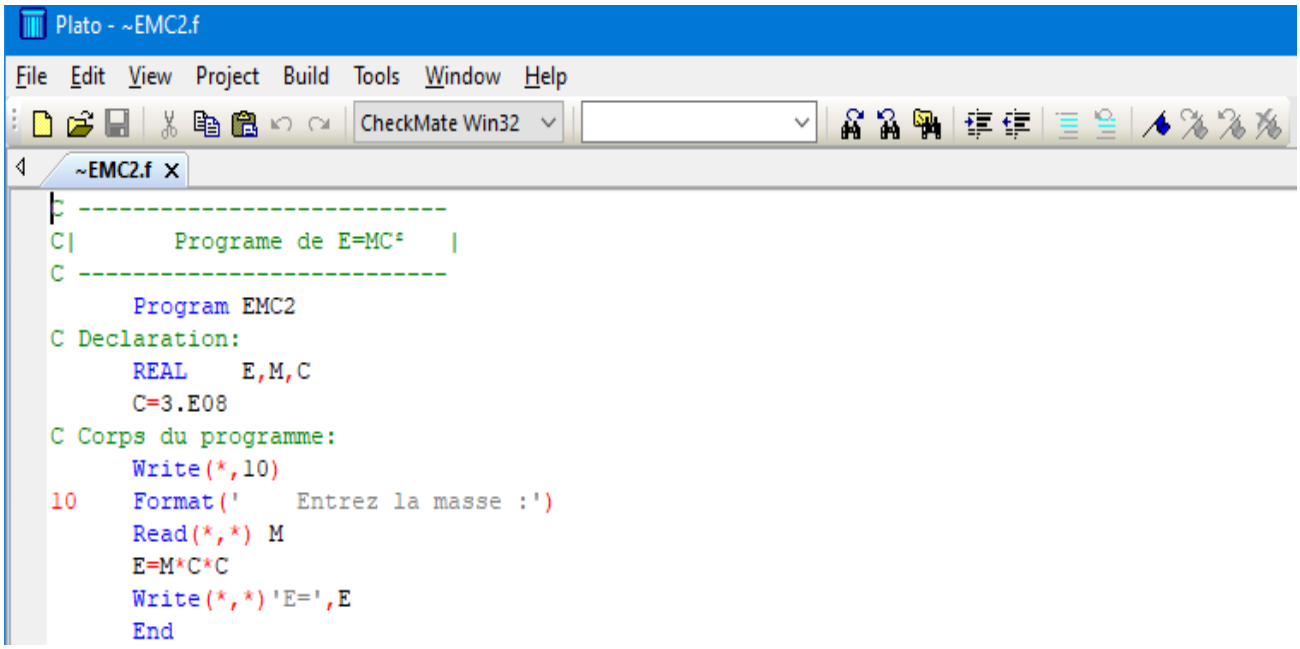


Figure 9. Exemple d'application de la Structure d'un programme

2.11 Les principales commandes en Fortran

La commande PROGRAM

La commande PROGRAM, si elle existe, c'est la première commande dans un programme en Fortran. Cette commande n'est pas obligatoire en FORTRAN mais elle permet de nommer un programme. Sa syntaxe est [6]:

PROGRAM Name

Exemple :

Pour nos programmes en chimie, par exemple, on peut nommer un programme qui calcul le PH d'une solution par :

PROGRAM calculPH

Ou

PROGRAM Energie

Qui calcul l'énergie totale ou d'autre énergie d'système moléculaire.

La commande PROGRAM c'est une aide à la programmation. Et donc, il est recommandé de commencer tout programme écrit en Fortran par cette ligne. Cette commande 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, ou pour la déclaration d'une donnée ou une variable).

Exemple :

Voici dans le programme qui calcul le PH d'une solution (Figure 10). Dans la première ligne, on écrit la première commande qui est :

program PHsolution

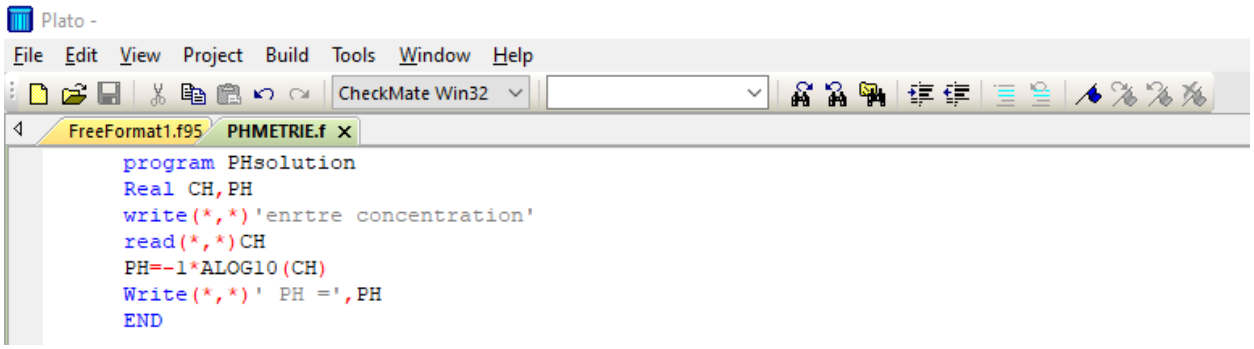


Figure 10. La commande program

La commande END

Au contraire de la commande program, la commande END, c'est la dernière commande de tout programme FORTRAN, et aussi au contraire de la commande program, celle-ci, est obligatoire. Sa syntaxe est [6] :

END

Et donc, le compilateur Fortran n'est pas pris en compte toute ligne écrite au-delà de la commande END, ou bien, celle-ci plante le compilateur (cela dépend de l'implémentation de ce dernier).

Exemple :

Voici dans le programme qui calcule le PH d'une solution (Figure 11). Dans la dernière ligne, on écrit la dernière commande qui est :

END

Remarque : en Fortran, les commandes sont écrites soit en majuscule ou minuscule. Pas comme par exemple en système d'exploitation Linux, qui a ait la déférence entre majuscule et minuscule

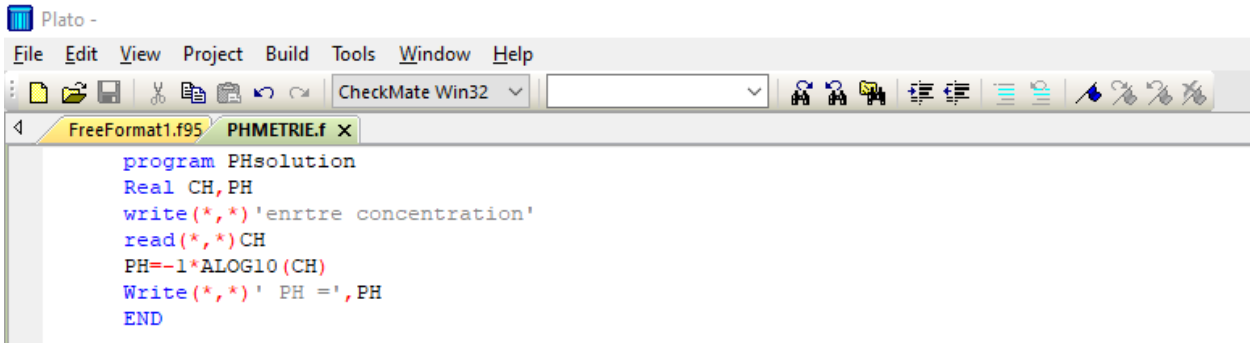


Figure 11. La commande END

La commande STOP

Cette commande est équivalente à la commande END, mais elle est souple d'utiliser puisqu'elle permet d'avoir plusieurs points de sortie dans un même programme, et qu'elle permet de renvoyer un code de référence.

La commande STOP permet de sortir d'un programme Fortran sans toutefois avoir atteint la dernière ligne du programme. Sa syntaxe

est :

STOP

La commande GO TO

Cette instruction permet d'effectuer un branchement à un endroit particulier du code. Elle permet de se déplacer dans un programme sans exécuter d'instruction. Cette instruction est à éviter, car elle peut générer des programmes illisibles et difficiles à corriger. Sa syntaxe est :

GO TO N

Dans cette commande, N indique le numéro de ligne où le programme doit reprendre son exécution.

La commande CONTINUE

En Fortran, cette commande joue le rôle d'une séparation entre deux parties dans un programme. En effet, cette commande ne fait rien. Elle peut être utile pour remplir des lignes de programme dont le principal intérêt est leur numéro de référence.

Dans l'exemple suivant, on utilise la commande CONTINUE (Figure 12) dans un programme qui calcul l'énergie E d'un condensateur. On remarque que cette commande ne fait rien.

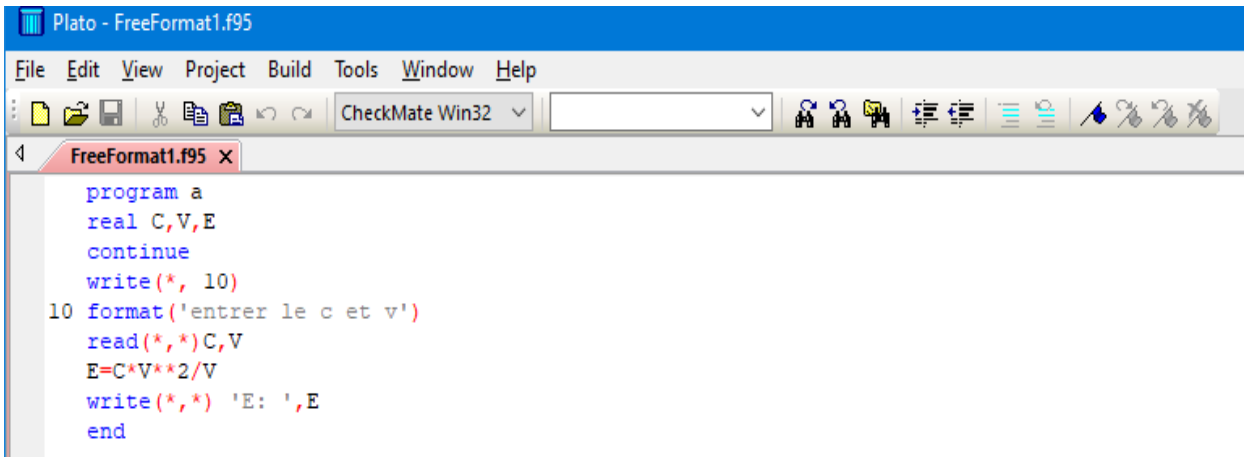


Figure 12. La commande CONTINUE

La commande DATA

L'instruction data permet de stocker des valeurs dans des variables sans faire d'opération d'affectation. Elle n'est pas exécutée par le programme mais directement interprétée par le compilateur Fortran qui, au lieu d'initialiser des variables à zéro, les initialisera à la valeur souhaitée. **La** syntaxe de la **commande DATA** est [6]:

DATA Liste_de_variables/Liste_de_valeur/

Exemple :

On peut également affecter des valeurs à une série de variables au moyen de l'instruction DATA placé après les déclarations :

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

Dans cette exemple, et avec la commande DATA, on a affecté des valeurs à l'énergie E, le nombre d'Avogadro, et pour la constante de Planck:

1.6E-19……E

AVOG……..6.023E23

PLANC……6.62E-34

à une série de variables utilisant l'instruction DATA.

La syntaxe de la liste de valeur utilisant la commande DATA obéis a des règles sont les la suivantes :

- Chacune des valeurs est séparée par une virgule. (val1,val2,val3)
- Si *N* variables doivent être initialisées à la même valeur, il est possible de ne pas répéter *N* fois cette valeur et d'écrire Valeur*N.

2.12 Les commandes d'intéractions

Dans tout programme écrit en Fortran, il est nécessaire d'avoir des intéractions entre ce dernier et l'utilisateur. C'est-à-dire, permettre à l'utilisateur d'entrer des données et au programme d'afficher des résultats.

La commande READ

La commande READ permet de lire des données entrées via un périphérique (essentiellement le clavier ou un _chier). Sa syntaxe est [6]:

READ*, Liste_de_variables

READ(N,*) Liste_de_variables

La commande READ demande la lecture au clavier d'une donnée. La variable qui suit La commande READ prendra la valeur qui sera saisie au clavier lors de l'exécution du programme Fortran.

Exemple :

Dans l'exemple du calcul de PH d'une solution quelconque, la commande read sélectionnée dans l'exemple présenté (Figure 13) est utilisée pour entrer la valeur de la concentration noté (CH) de la solution :

read(*,*)CH

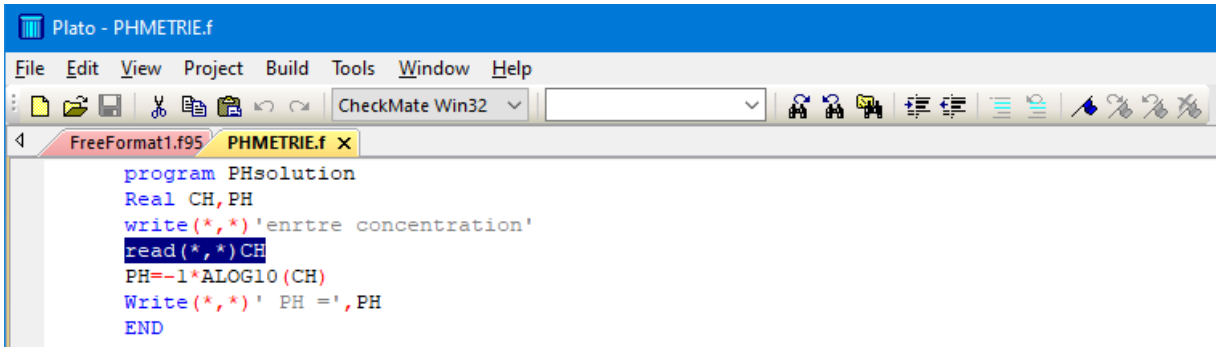


Figure 13. La commande READ

Lors de l'exécution du programme Fortran, N représente l'identificateur d'un flux de donnée. Pour le clavier, c'est généralement 5. Quelle que soit la forme utilisée, dans un programme en Fortran, il faut, bien sûr, respecter le type des données.

La commande PRINT

La commande PRINT demande l'affichage à l'écran de données, elle permet d'afficher des informations ou des variables à l'écran. Sa syntaxe est [6] :

```
PRINT*, Liste_a_afficher
```

On peut donc écrire :

```
REAL a,b
PRINT*, 'Entrer deux nombres'
READ*, a,b
PRINT*, 'Le resultat de',a,'*',b,' est :',a×b
```

La commande WRITE

En fortran, la commande PRINT et La commande WRITE ne diffèrent pas, la différence est dans la syntaxe de ces deux commandes.

```
PRINT*, Liste_a_afficher
WRITE(N,*) Liste_a_afficher
```

Lors de l'exécution du programme Fortran, N représente l'identificateur d'un flux de donnée. Pour le clavier, c'est généralement 6.

Exemple :

Dans l'exemple du calcul de PH d'une solution quelconque, la commande WRITE sélectionnée dans l'exemple présenté dans la Figure 14 est utilisée pour afficher la valeur de PH noté (PH) de la solution :

```
Write(*,*)' PH =',PH
```

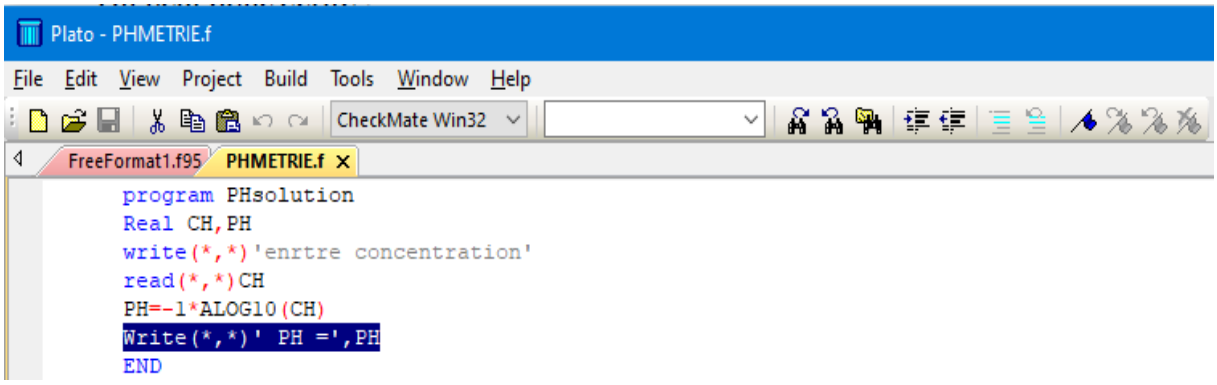


Figure 14. La commande Write

La commande RETURN

Cette commande permet de sortir d'un bloc fonctionnel sans aller jusqu'à la dernière commande dans un programme Fortran 'END'. Il faut toutefois que la variable de retour ait été remplie. On peut aussi définit cette commande qu'on retourne au programme principal. Il peut éventuellement y en avoir plusieurs dans le même sous-programme, sous différentes options contrôlées par une instruction 'if' par exemple

La commande CALL

Cette commande permet d'appeler une procédure. Sa syntaxe est :

CALL Nom(Arg1,Arg2, . . . ArgN)

Durant les séances de TP de licence chimie inorganique, on n'utilisa pas tous ces commandes, on utilise juste les commandes simples comme la commande PROGRAM, READ, WRIT, GO TO, CONTINUE, … etc.

Il existe aussi d'autres commandes utilisées en Fortran n'est pas cité dans ce polycopie de cours comme les commandes : la commande COMMON, EQUIVALENCE, EXTERNAL, PARAMETER, OPEN, REWIND, CLOSE, FORMAT, … etc.

2.14 Bibliothèque de fonctions

Dans le langage de programmation FORTRAN, il existe des bibliothèques de fonctions (appelées couramment *libraries*) pour calculer, par exemple :

- des racines carrées,

- Déterminer la valeur absolue d'un nombre.
- Arrondir par défaut un rationnel à l'entier le plus proche
- Calculer la valeur absolue d'un rationnel
- Déterminer le maximum des entiers
- Déterminer le minimum des entiers
- Génère un rationnel à partir de l'entier
- Calculer le reste de la division euclidienne
- Calculer l'exponentielle d'un rationnel
- Déterminer la taille d'une chaîne de caractères

Pour exploiter ces fonctions, il suffit de taper le nom de celle que l'on veut utiliser.

LINPACK (Linear **A**lgebra **Pack**age**)** est une bibliothèque de fonctions en Fortran parmi les bibliothèques du fortran les plus connus. Cette bibliothèque est utilisée pour l'algèbre linéaire, et notamment la résolution numérique de systèmes d'équations linéaires. Avec cette bibliothèque pour le calcul des valeurs et vecteurs propres, elle est à l'origine de MATLAB dont le but premier était de permettre son utilisation sans programmation en Fortran.

On note que LINPACK a été supplantée par LAPACK. Cette dernière est un test de performance servant à classer les plus puissants superordinateurs du monde dans le TOP500.

La LAPACK est créé par Jack Dongarra (Figure 15), il mesure le temps mis par un ordinateur pour résoudre un système linéaire dense de n équations à n inconnues.

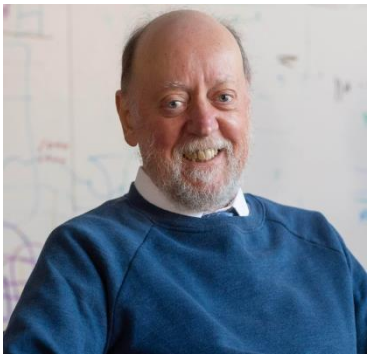


Figure 15. Jack Dongarra, créateur de LAPACK.

Exemples :

En voici quelques fonctions utilisées en Fortran :

R = SQRT(R1)

Cette fonction est utilisée pour le calcul de la racine carrée d'un rationnel, le résultat est aussi un rationnel.

I= INT(R)

Cette fonction est utilisée pour arrondit par défaut un rationnel à l'entier le plus proche.

R = ABS(R1)

Cette fonction est utilisée pour le calcul de la valeur absolue d'un rationnel, le résultat est aussi un rationnel.

I = MAX0(I1,I2, . . . ,IK)

Cette fonction est utilisée pour le calcul du maximum des entiers I1, I2 . . .IK.

I = MIN0(I1,I2, . . . ,IK)

Cette fonction est utilisée pour le calcul du minimum des entiers I1, I2 . . .IK.

R = FLOAT(I)

Cette fonction est utilisée pour générer un rationnel à partir de l'entier I.

R = EXP(R1)

Cette fonction est utilisée pour le calcul de l'exponentielle d'un rationnel.

I = LEN(CH)

Cette fonction est utilisée pour déterminer la taille d'une chaîne de caractères.

Les boucles

Les boucles WHILE-DO et IF-GO TO sont des boucles conditionnelles, leurs structures permettent de répéter un ensemble d'instructions en fonction d'une condition. Elles sont strictement équivalentes. Toutefois, il est préférable d'utiliser la seconde qui est bien plus lisible.

La boucle do while

En fortran, la boucle do while permet des logiques plus compliquées, sa syntaxe est :

```
while (condition) do
ensemble_de_commandes
endwhile:
```

Exemple

```
open(10,file='donnees')
read(10,*) x
do while ( x .ge. 0. )
  print *, 'La racine de',x,' est', sqrt(x)
  read(10,*) x
enddo
end(10)
```

Dans cet exemple, le Fortran va lire des valeurs de x dans le fichier tant que la dernière valeur de x reste positive, et à la première valeur négative trouvée, l'on passera à l'instruction end.

La boucle if go to

Sa syntaxe est :

```
10 IF (.NOT. Condition) GO TO 20
  Ensemble_de_commandes
  GO TO 10
20 commande_quelconque
```

L'ensemble de commandes est exécuté tant que la condition est vraie.

Pour plus de détail sur les cours en fortran voir aussi le lien suivant : http://www.softndesign.org/?page=manuels/fortran.php.

Chapitre 3

Applications Fortran:

3.1 Application 1

Un condensateur stocke de l'énergie sous forme électrique. On remarque que cette énergie est toujours positive (ou nulle) et qu'elle croît comme le carré de la charge ou de la tension. Ces propriétés sont analogues à celles de l'énergie cinétique d'une masse m animée d'une vitesse v. Un condensateur (Figure 16) est donc un réservoir d'énergie électrique noté E.

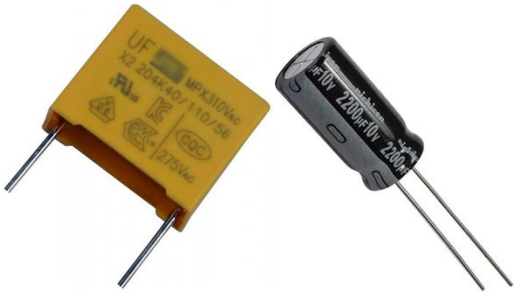


Figure 16. Des condensateurs qui stockent de l'énergie

L'énergie stockée sur un condensateur noté E est :

$E = CV^2/2$

Où

C : la capacité

V : la différence de potentiel.

Écrire un programme pour calculer l'énergie pour quelques exemples de valeurs de C et V

Solution de l'application 1 (Figure 17)

```
PROGRAM Energy
REAL C, E, V
READ*, C, V
E = C * V ** 2 / 2
PRINT*, "Stored energy:", E
END PROGRAM Energy
```

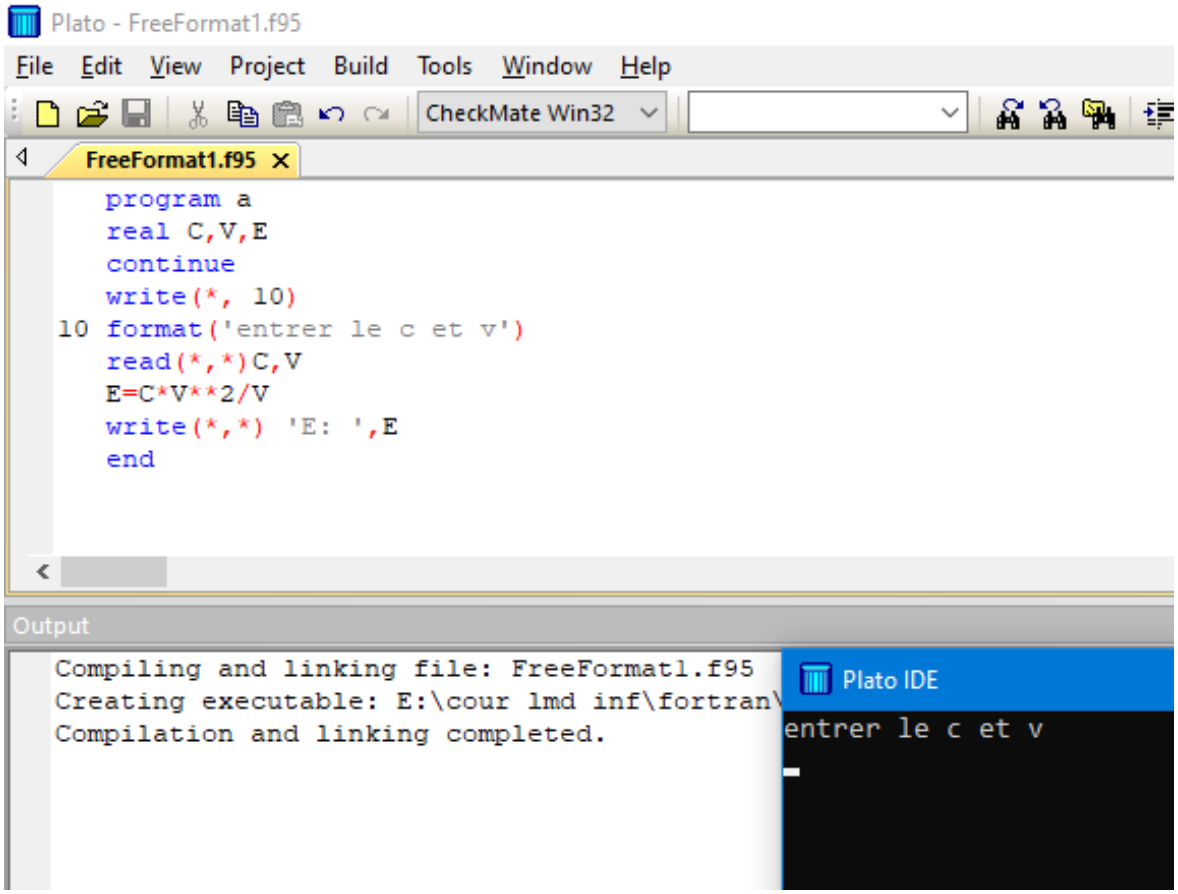


Figure 17. Programme de l'énergie stockée

Maintenant la dernière étape (Figure 18), c'est on introduit au fortran la valeur de C et V qui sont respectivement par exemple, 1 et 2

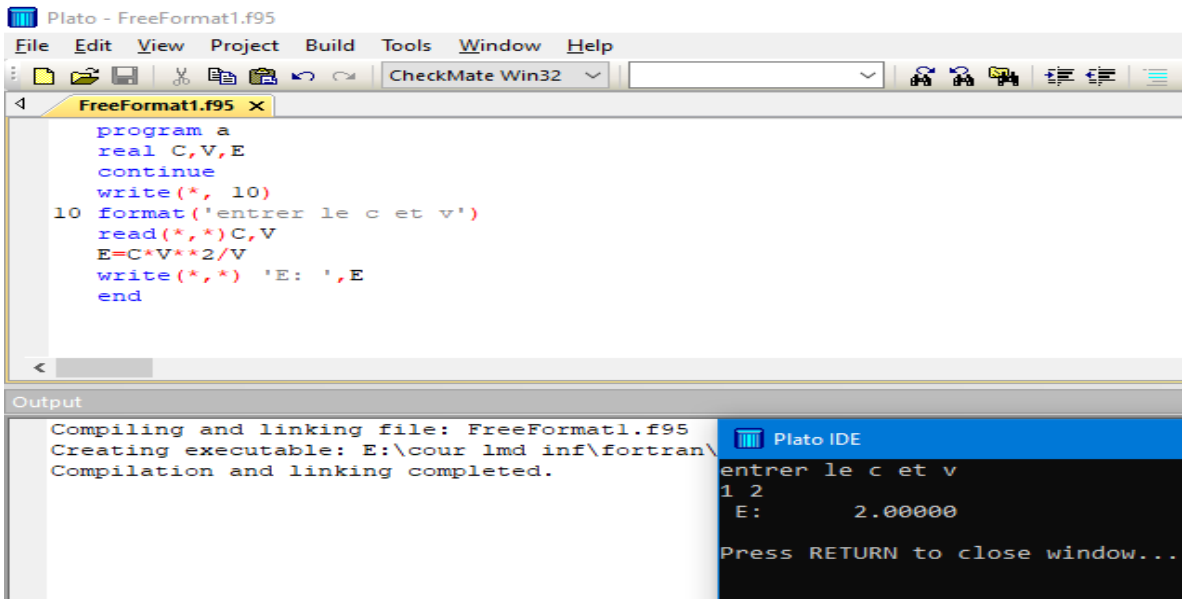


Figure 18. La dernière étape du programme de l'énergie stockée

Donc le Fortran affiche la valeur de E = 2

3.2 Application 2

Ecrire un programme en Fortran qui calcul du PH d'une solution

Définition : Le pH d'une solution aqueuse est un nombre qui exprime son acidité (ou sa basicité). Il est compris entre 0 et 14 (Figure 20). Si le pH est compris entre 0 et 7, on a une solution acide. Si le pH vaut 7, on a une solution neutre.

Le pH est l'opposé du logarithme décimal de la concentration en ions H_3O^+ :

PH = -Log[H_3O^+]. Le pH se mesure avec un pH-mètre, du papier pH ou s'évalue grossièrement avec des indicateurs colorés comme le bleu de bromothymol (Figure 19).

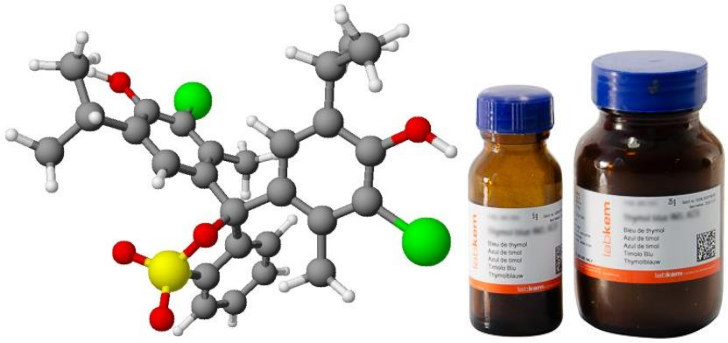


Figure 19. Structure du bleu de bromothymol et en bouteille.

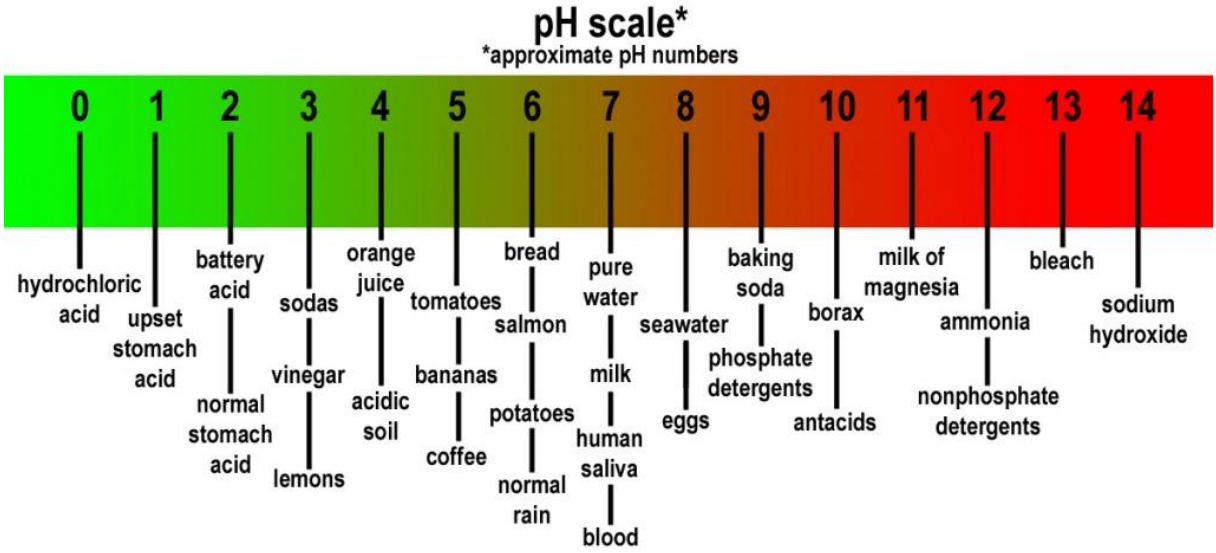


Figure 20. Intervalle du PH

Solution de l'application 2

```
Program PHsolution
Real CH,PH
write(*,*)'enrtre la concentration'
read(*,*)CH
PH=-1*ALOG10(CH)
Write(*,*)' PH =',PH
END
```

Voici la fenêtre Fortran (Figure 21), dans cette fenêtre on écrit le programme PH

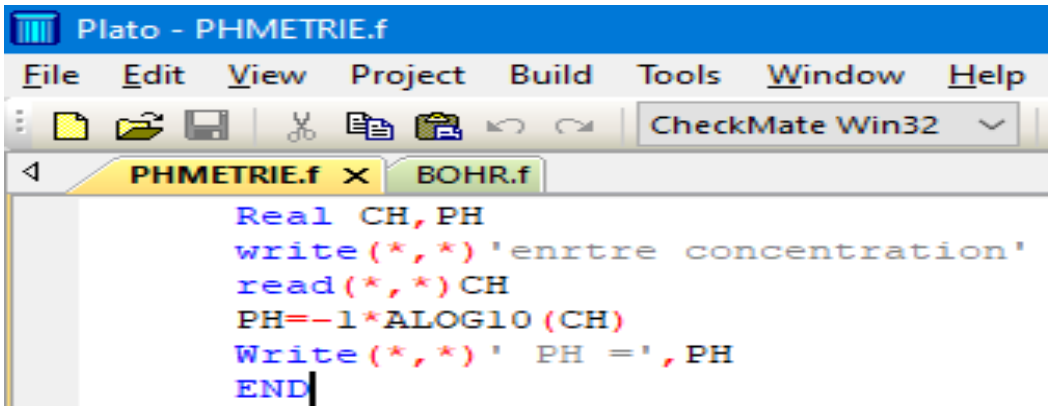


Figure 21. Programme PH

Après la vérification du programme Fortran et la compilation du Fortran (Figure 22), on obtient cette fenêtre :

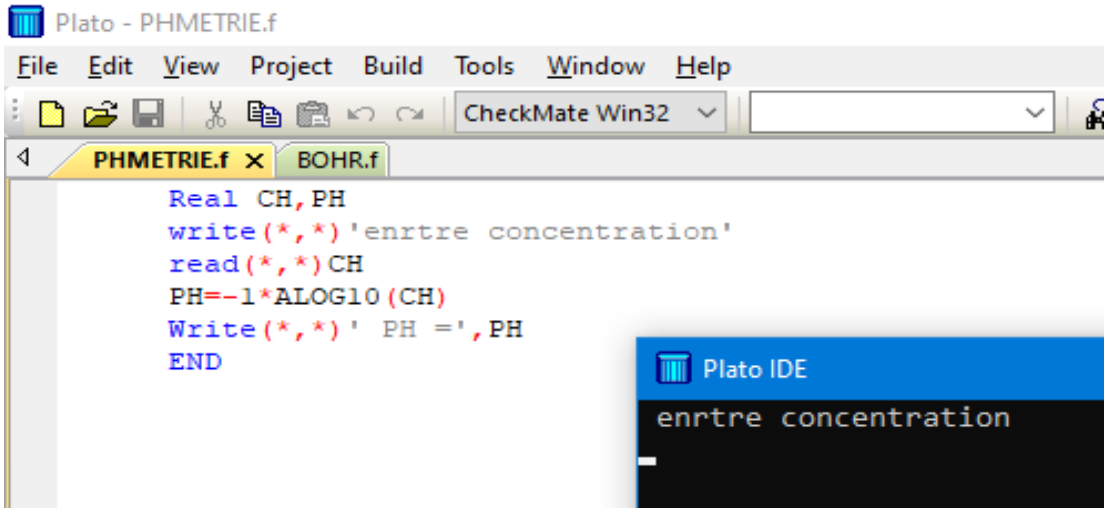


Figure 22. Exécution du programme PH

Maintenant la dernière étape (Figure 23), c'est on donne au Fortran la valeur de la concentration CH, qui est par exemple CH = 0.01 mol/l

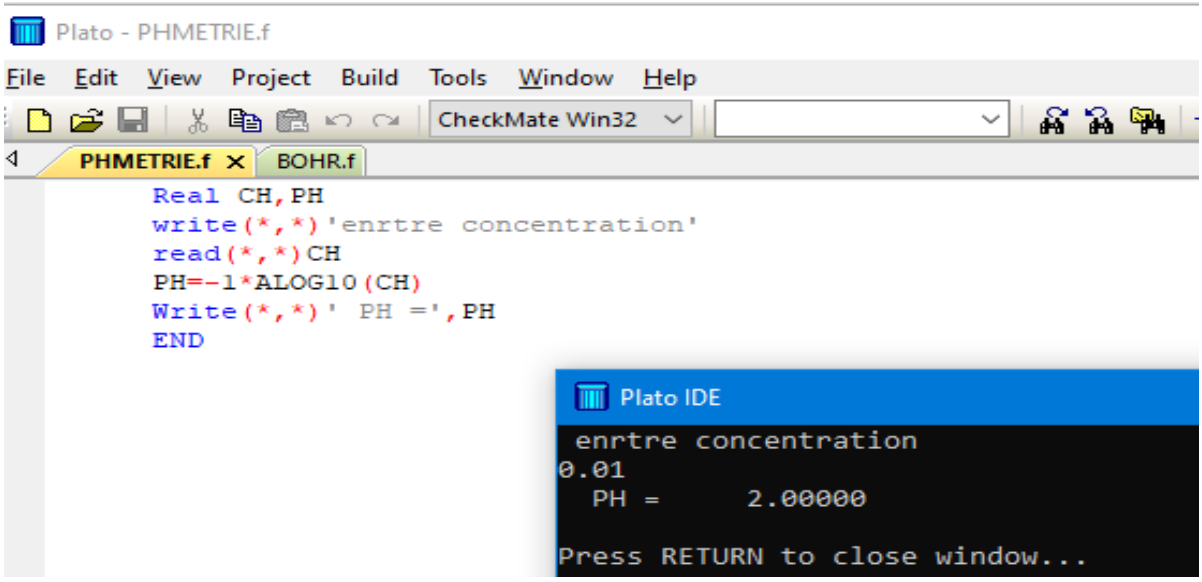


Figure 23. Etape « entrer la concentration » du programme PH

Et donc le Fortran affiche la valeur de PH = 2.00000

3.3 Affichage des erreurs en Fortran

La fenêtre du Fortran affiche à la fin de la fenêtre de l'exécution Fortran des erreurs en rouge (Figure 24), chaque type d'erreur à un numéro. Cette fenêtre aide l'utilisateur ou le programmeur en Fortran pour corriger ces erreurs facilement.

Exemple :

Dans le programme de PH, on crée des erreurs dans la première ligne du programme correct, et on va voir l'affichage à la fin de la fenêtre de cette erreur en rouge **:**

Voici la réponse du FORTRAN en rouge: "error 489 - 'PH' cannot be in a declaration since this is the name of the PROGRAM"

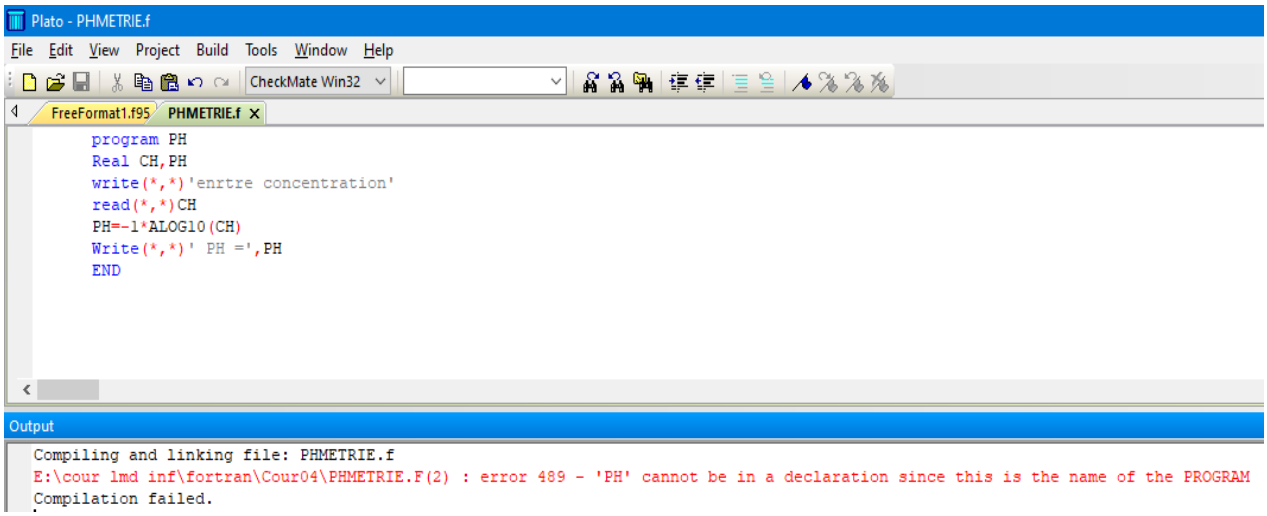


Figure 24. Affichage d'erreur dans un programme Fortran

Compilation failed, c'est-à-dire échec de la compilation, et donc il faut corriger cette erreur.

- Et donc dans la première ligne, on a nommé le programme PH
- Dans le deuxième ligne, on déclare les données : Real CH,PH
- Donc, le *nom du programme* PH ainsi que la *déclaration de PH* sont les mêmes, et automatiquement le Fortran déclare l'erreur.

3.4 Connaissance de la position de l'erreur dans FORTRAN

Pour connaitre la place de l'erreur dans le programme (Figure 25), il suffit de cliquer sur l'erreur en rouge, et le Fortran donne un Flèche qui montre la position de l'erreur comme ça :

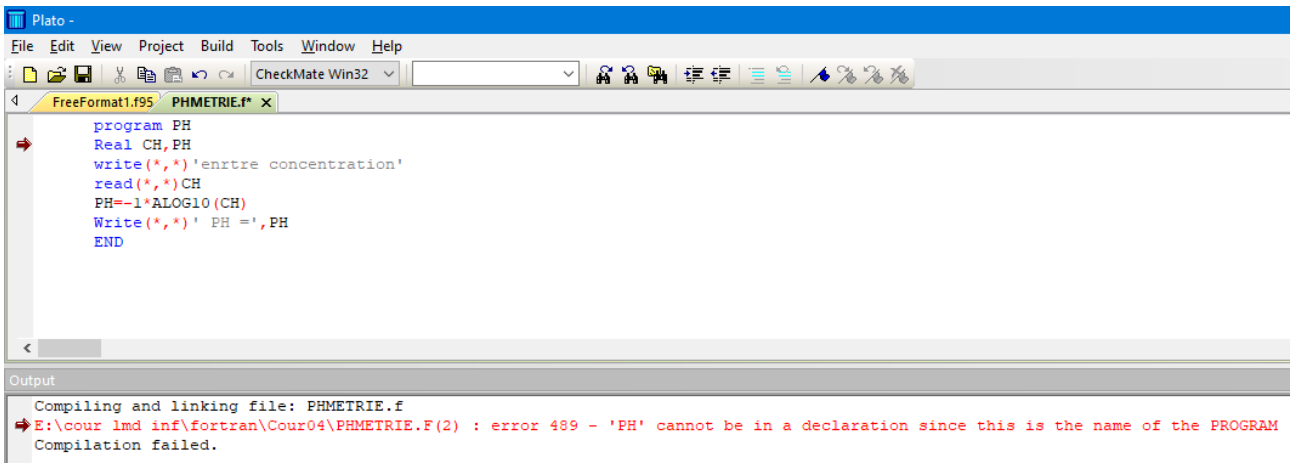


Figure 25. Indication d'erreur dans un programme Fortran

3.5 Correction de l'erreur dans FORTRAN

Après l'affichage de l'erreur, il faut faire la correction dans la commande program (Figure 26), sinon, on modifier le mot PH dans la deuxième ligne.

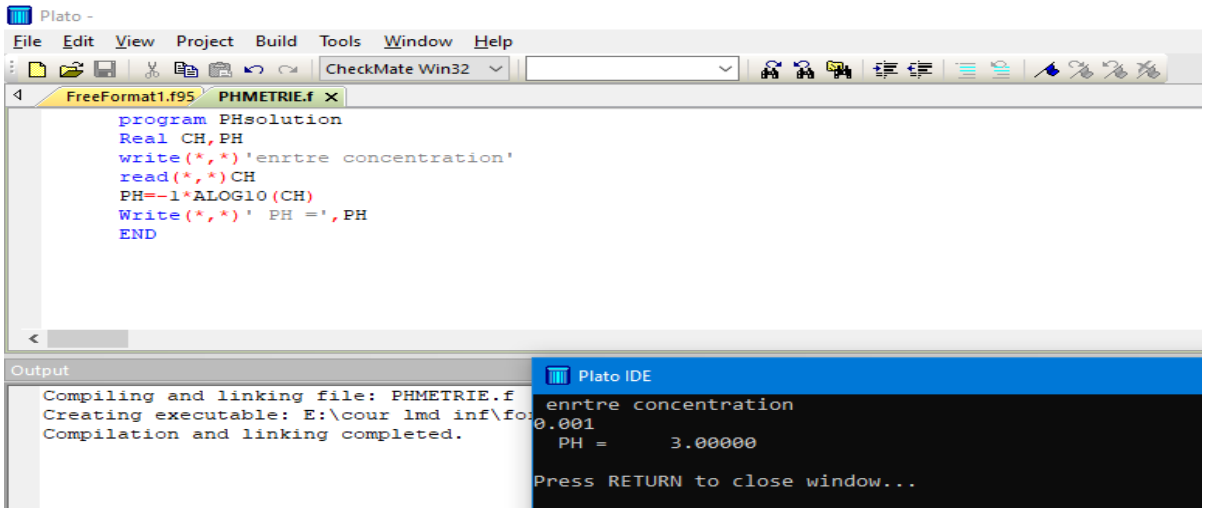


Figure 26. Correction d'erreur dans un programme Fortran

3.6 Application 3 :

Le modèle de Bohr est une description simplifiée de l'atome. Elle stipule, entre autres, que les électrons dans les atomes occupent des orbites circulaires autour du noyau, de façon semblable aux planètes qui tournent autour du soleil. Par un calcul classique, Bohr (Figure 27) établit une relation entre la fréquence orbitale de l'électron et l'énergie E qu'il faut lui fournir pour le placer sur une orbite de rayon infini. Puis, en tenant compte de la condition de quantification, il détermine les énergies des différents états stationnaires,

$$E_n = -2\pi^2 me^4/n^2 h^2$$

où

m : la masse de l'électron

qe : charge électrique de l'électron

Figure 27. Niels Bohr (1885- 1962)

Ecrire un programme en FORTRAN qui calcul l'énergie correspondante à l'orbite caractérisée par le nombre entier n. avec E :

$$E_n = \frac{-2\pi^2 m Z^2 e^4}{n^2 h^2}$$

Et donc la solution mathématique du problème est connue, qui la formule pour le calcul de E.

L'écriture du programme est simple maintenant. Voici le programme Fortran (Figure 28) qui calcul cette énergie :

```
    Program Bohr
    Real*8 E,R,Pi,M,Qe,h
    Integer N,I
    Parameter (Pi=3.1415,M=9.108 E-31,Qe=1.602 E-19,h=6.62 E-34)
    Write(*,10)
10  Format('                MODELE DE BOHR')
    write(*,*)'Entrez n:'
    Read(*,*) n
    Do n=1,10,1
      E=-1*(2*pi*pi*m*qe*qe*qe*qe)/(n*n/h*h)
      Write(*,*)'E=',E
    ENDDO
    END
```

Le Fortran affiche après la compilation :

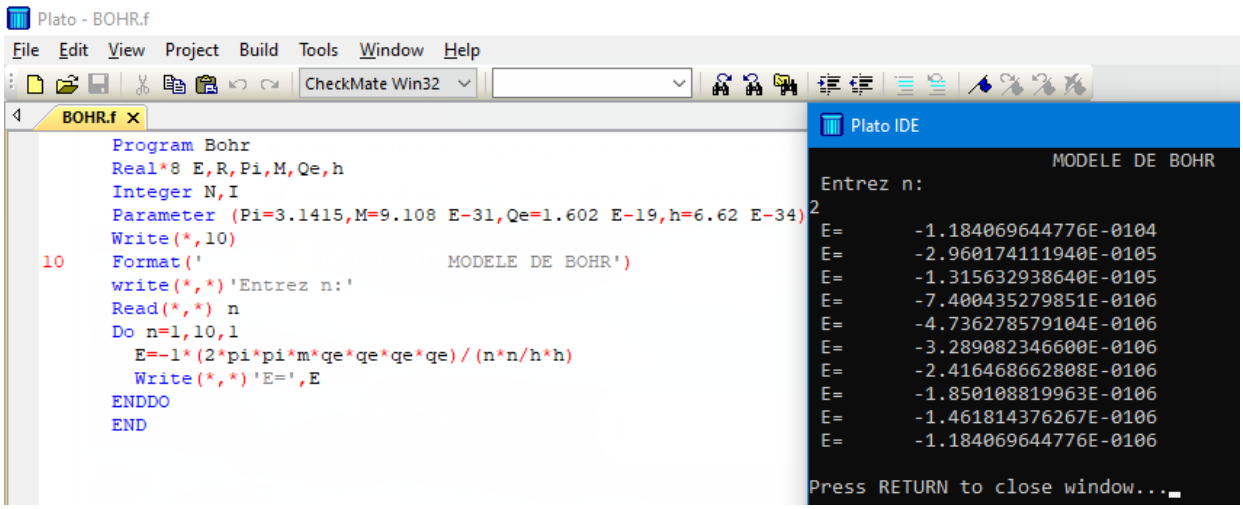


Figure 28. Programme Bohr pour n = 2

Si on doit modifier le nombre n, les énergies sont changées (Figure 29)

Exemple n = 3

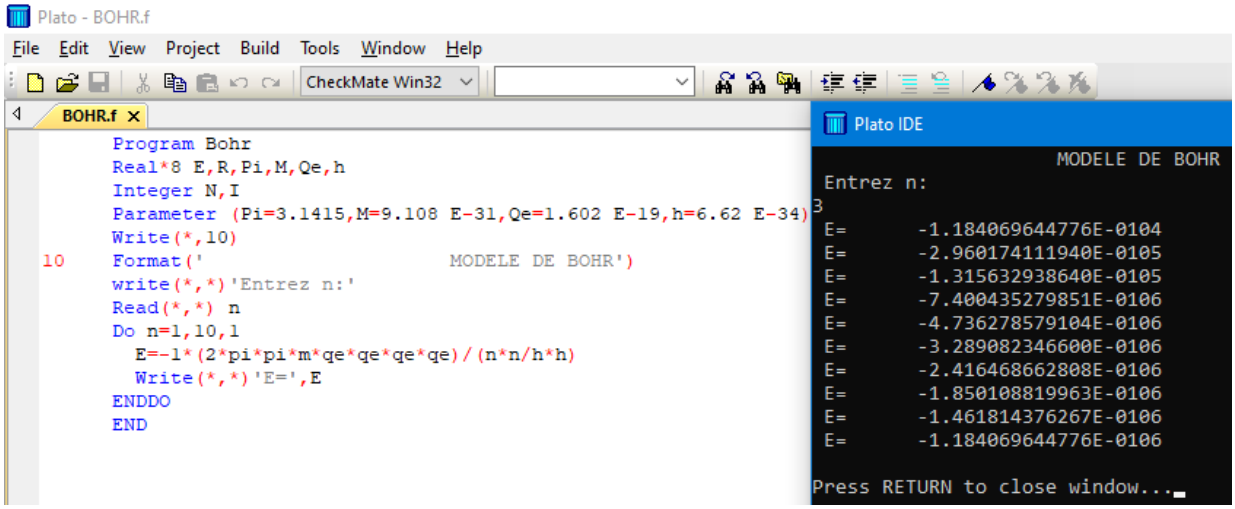


Figure 29. Programme Bohr pour n = 3

3.7 Application 4

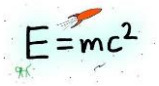
Ecrire un programme en Fortran qui calcul l'énergie cinétique E_c.

L'équation $E_c = mc^2$ (lire « E égale m c carré » ou « E égale m c deux ») est une formule d'équivalence entre la masse et l'énergie, rendue célèbre par Albert Einstein dans une publication en 1905 sur la relativité restreinte [7,8].

Cette relation signifie qu'une particule de masse m isolée et au repos dans un référentiel possède, du fait de cette masse, une énergie E appelée énergie de masse, dont la valeur est donnée par le produit de m par le carré de la vitesse de la lumière dans le vide (c).

Définition : en mécanique classique, l'énergie cinétique E_c d'un objet quelconque est l'énergie qu'il possède en raison de son mouvement. Elle est proportionnelle à la masse m du système ainsi qu'au carré de sa vitesse v. Cette propriété physique peut définir aussi comme suit : c'est la quantité de travail nécessaire pour accélérer un corps d'une masse donnée m au repos à son mouvement. Cette énergie sera constante tant que la vitesse du corps est maintenue.

La formule d'énergie cinétique est la suivante:



Le programme écrit en Fortran pour cette énergie est présenté dans la Figure 30.

Figure 30. Programme d'énergie cinétique E_c

Dans ce programme, en présente deux partie :

Partie declaration

Dans cette partie on présente, ou on déclare les paramètres utilisées dans ce programme qui sont E, M, et C, avec c'est la vitesse de la lumière : C = 3.E08

On écrit donc :

REAL   E,M,C

    C=3.E08

Corps du programme ou la partie fonctinnelle

Cette partie rassemble le corps du programme, ctd c'est la traduction de la sulution mathématique du problème $E_c = mc^2$

On écrit donc :

```
Write(*,10)
10    Format('    Entrez la masse :')
      Read(*,*) M
      E=M*C*C
      Write(*,*)'E=',E
      End
```

Après l'exécution du programme nommée EMC2, le Fortran affiche la fenetre suivante (Figure 31):

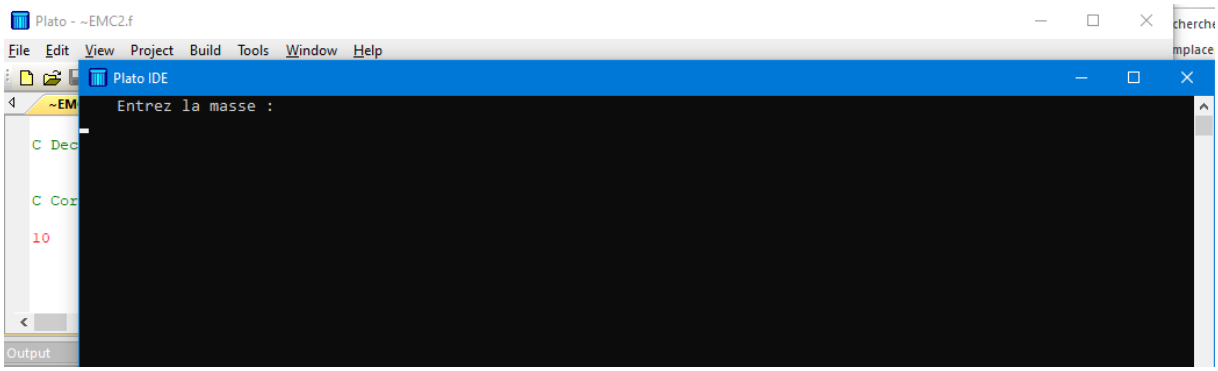


Figure 31. Introduction des données au programme

On donne des valeurs à chaque fois, et le Fortran afffiche des valeurs d'énergie

Exemples :

Pour m = 10 E= 9.000000E+17

Pour m = 100 E= 9.000000E+18

Pour m = 1000 E= 9.000000E+19

3.8 Application 5

Ecrire un programme en Fortran qui calcul l'anisotropie de la polarisabilité.

Définition : La polarisabilité notée α défini l'aptitude d'un objet à se déformer sous l'action d'un champ électromagnétique *E*. Elle se mesure par la diffusion d'un faisceau de photons. Ce processus appelé diffusion Compton, est un des phénomènes les plus caractéristiques de l'aspect corpusculaire du rayonnement électromagnétique.

Le programme est très simple, on écrit donc :

```
program anisotropie
read(*,*)a,b,c;
write(*,*)sqrt(((a-b)**2+(a-c)**2+(b-c)**2)/2);
end
```

Dans ce programme (Figure 32) qui calcul l'anisotropie de la polarisabilité α d'un système moléculaire, on a :

Partie declaration:

Dans cette partie, on déclare les contributions α_{xx}, $\alpha_{yy}, et\ \alpha_{zz}$ noté a, b, et c, respectivement :

On écrit donc :

```
read(*,*)a,b,c;
```

Partie fonctionnelle:

Dans cette partie, on écrit la formule qui calcul l'anisotropie de la polarisabilité :

La formule qui calcul l'anisotropie de la polarisabilité est :

$$\Delta\alpha = \sqrt{\frac{1}{2}\left(\left(\alpha_{xx} - \alpha_{yy}\right)^2 + (\alpha_{xx} - \alpha_{zz})^2 + \left(\alpha_{yy} - \alpha_{zz}\right)^2\right)}$$

On écrit donc :

```
write(*,*)sqrt(((a-b)**2+(a-c)**2+(b-c)**2)/2);
end
```

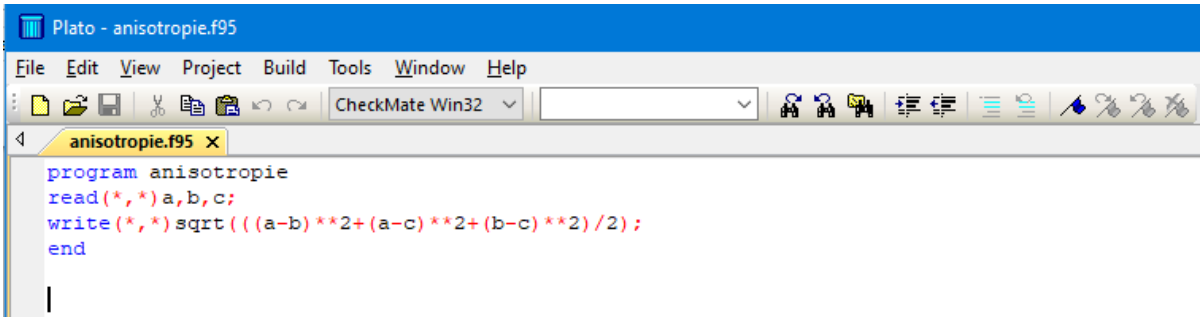


Figure 32. Programme de l'anisotropie de la polarisabilité α

Après l'exécution du programme nommée l'anisotropie, le Fortran affiche la fenetre suivante (Figure 33) :

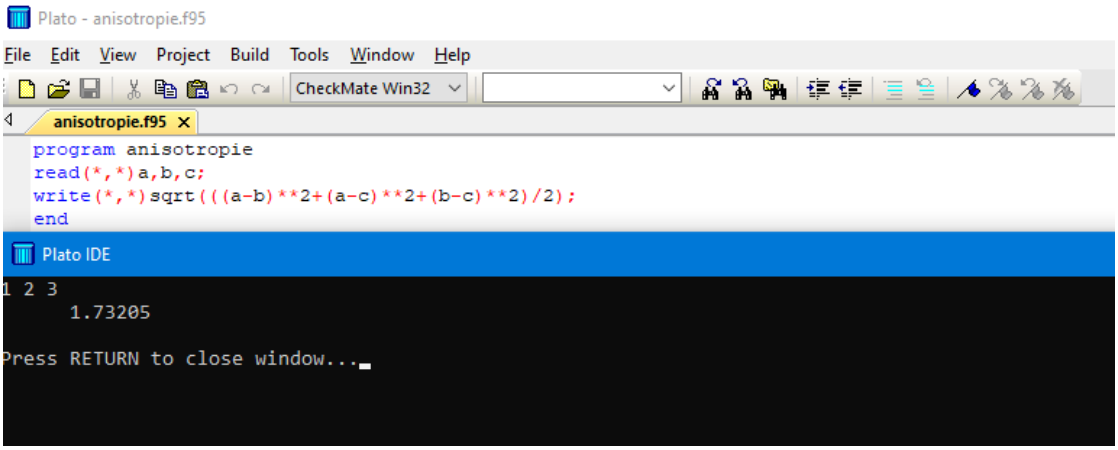


Figure 33. Introduction des données au programme

Références

[1] J. Backus, L'histoire de FORTRAN, 2014 .

[2] J. Sammet, Programming Languages: History and Fundamentals (Englewood Cliffs, NJ: Prentice-Hall, 1974.

[3] J. W. Backus, R. J. Beeber, S. Best, R. Goldberg, L. M. Haibt, The FORTRAN automatic coding system, 1957, 188-198.

[4] M. Gabbrielli, S. Martini, Programming Languages: Principles and Paradigms, Springer, 2010.

[5] O. Louisnard, J. Letourneau, P. Gaborit, Initiation au Fortran, Ecole des mines, 1997.

[6] L. Mazet, Fortran 77 Langage d'un autre age, 2004.

[7] Diu et Leclercq , s.v. E = mc2 : défaut de masse, 2005, 167.

[8] Fink, Le Bellac et Leduc , chap. 5, 2016, 74.