Table des matières

Chapitre III

Introduction au langage de programmation Fortrant 77

La première apparition du langage FORTRAN remonte à 1954. Il a su rester depuis, le langage de prédilection, notamment pour les calculs scientifiques. Sa popularité et longévité sont dues à de nombreuses raisons.

· FORTRAN est facile à apprendre ; c'est un langage structuré d'une syntaxe très simple.

·De part sa très longue période d'existence, sa bibliothèque s'est enrichit au fil des ans pour mettre à la disposition de ses utilisateurs tout ce dont ils auront besoin.

· Son omniprésence dans différents domaines qui s'apparentent à la science et à la technologie allant des mathématiques jusqu'à la biologie.

Ci-dessus, nous allons examiner les principales caractéristiques de ce langage au travers de la présentation de quelques applications simples de

Le premier compilateur FORTRAN fut conçu et implémenté par John Backus (IBM Lab, San José Californie) en 1954. L'objectif annoncé était de mettre à la disposition un langage avec une syntaxe proche de celle utilisée dans les formules mathématiques. En effet, le terme FORTRAN est l'acronyme de FORmula TRANslation. La première version fut nommée FORTRAN I, actuellement, on en est à la version FORTRAN 2018. Entre les deux versions, le FORTRAN a connu de nombreuses modifications et améliorations en particulier sur l'aspect syntaxique.

Chapitre I

Vocabulaire du langage de programmation fortran

Pour écrire un programme en FORTRAN, il est impératif d'utiliser uniquement le vocabulaire reconnus par le compilateur qui est:

I.1 Les 36 caractères alphanumériques : Les 10 chiffres décimaux et les 26 lettres de l'alphabet.

I.2 Les caractères spéciaux:

Caractère	Signification	Caractère	Signification
-	Moins	!	Point d'exclamation
+	Plus	«	Guillemet
*	Astérisque	:	Double point
/	Barre oblique	,	Virgule
(	Parenthèse gauche	‘	Apostrophe
)	Parenthèse droite	$	Dollar
.	Point	_	Caractère de concaténation

I.3 Les operateurs

I.3.1 Les operateurs relationnels :

Opérateurs	Equivalent en langage de programmation Fortran
.EQ.	=, égale
.NE.	# , différent
.GT.	>, supérieur
.GE.	>=, supérieur ou égale
.LT.	<, inferieur
.LE.	<=, inferieur ou égalé

I.3.2 Les operateurs logiques

Opérateurs	Equivalent en langage de programmation Fortran
.AND.	Conjonction logique
.OR.	disjonction inclusive
.NOT.	négation logique
.EQV.	Equivalence logique
.NEQV.	Non équivalence logique

I.3.3 Les operateurs arithmétiques

Opérateurs	Equivalent en langage de programmation Fortran
+	L'addition
-	Soustraction
*****	Multiplication
******	Puissance
/	La division

I.5 Les mots-réservés du langage

DATA, READ, WRITE, FORMAT, CONTINUE, GO TO, IF, THEN, ELSE, ENDIF, COMMON, DOWHILE, ENDDO, DO, OPEN, ASSIGN, INTEGER, REAL, CHARACTER, LOGICAL, COMPLEX, IMPLICITE NONE, PARAMETER, PROCEDURE, FUNCTION, STOP, END,GOTO

Le programmeur est libre dans le choix des identificateurs, néanmoins il lui est défendu d'utiliser les mots-réservés du langage qui constituent le lexique reconnu par le compilateur.

I.6 Architecture d'un programme Fortran77

Tout programme FORTRAN se distingue par une structure qui doit être respectée car il commence par un mot clé **PROGRAM** avec le nom du programme et se termine par les mots clés de fin de programme **STOP END**, En plus de cela pour écrire un programme, il est nécessaire de respecter les zones désignées pour les composants de chaque ligne (commentaires, étiquètes, lignes suites, instructions) **Figure 1.**

En plus des choses susmentionnées, il y a une méthodologie qui doit être suivie dans la programmation, où nous commençons par les déclarations qui sont considérées comme l'une des exigences de base du programmeur, de sorte qu'elle nécessite la définition de toutes les choses utilisées (constantes, sous programmes, variables composées et simples)

Ensuite, nous passons en revue la solution au problème en question, en utilisant un ensemble d'**expressions** disponibles dans le langage de programmation Fortran.

C1	C2->C5	C6	C7----------→C72	C73---→C80
c c	Ce ci	Est -	Programme test ---------- **Program** <nom du programme> [déclaration des constantes] [déclaration des procédures] [déclaration des fonctions] [déclaration des variables] [<les actions de traitements >] **<stop>** **<end>**	C H A M P N O N **U** **T** **I** **S** **E**

Figure 1: Architecture un programme fortran 77

Description des champs :

La colonne **1** est utilisée pour mentionner un **commentaire** en mettant un **c** suivi du commentaire.
Les commentaires sont l'un des moyens explicatifs pour le lecteur de lui permettre de comprendre la solution

Les colonnes de 1 à **5** sont utilisées pour référencer les **étiquettes** qui sont des valeurs numériques absolues exprimant la forme de lecture ou d'écriture des données et peuvent être utilisées pour le branchement lors de l'utilisation de l'instruction **goto**.

Le champ de la colonne **7** à **72** est réservé pour les instructions du programme, cependant, nous pouvons commencer à écrire des actions de traitement **à partir** de la position 7.

Le champ de **73** à **80** est utilisé pour numéroter les lignes d'un programme, nous pouvons également le laisser vide et ne va pas être pris en compte lors de la mise en œuvre du programme

Chapitre II

Les déclarations :

Les déclarations (spécification des données) sont considérées comme un type d'instructions que nous utilisons pour identifier les choses utilisées d'une part et pour demander l'allocation de places vacantes en mémoire de l'ordinateur (réservation de l'espace mémoire sur la RAM) pour les paramètres et les variables utilisés dans un programme.

II.1 Déclaration des constantes

Les constantes sont des éléments dont les valeurs ne changent pas pendant les étapes de l'exécution du programme.

Syntaxe de déclaration :

C1	2345	6	789 72	73 80
			PARAMETER < Liste des constantes >	

Exp :

C1	2345	6	789 72	73 80
			PARAMETER n=10, pi=3.14	

Dans l'exemple précédent, nous avons défini et fixé les constantes n, pi avec des valeurs respectivement 10 et 3.14.

II.2 Déclaration des variables

Définition d'une variable : Une variable est chaque composant dont la valeur change pendant l'exécution d'un programme, et nous pouvons distinguer deux types de variables simples et composées

Définition: **une variable composée** est une variable formée de un a plusieurs éléments de même type.

Exp 1 : A (2,3,-5,10) : A est un tableau(vecteur) de 4 éléments de type entier.

Exp 2 : T(AL,CL,PL) : T est un tableau de 3 éléments de type caractère.

Exp 3 : C(4,6) est une matrice (tableau) composée de 4 lignes et 6 colonnes d'éléments de type réel.

4	6	-7	8	4	6
6	1.00	12.25	7	3	1
15	12	0	9	23	1
5	4.78	8	89	0	1

Figure 2 : M(4,6)

Définition d'une variable simple: une variable simple est une variable composée d'un seul élément de type quelconque (entier, réel, …. . .).

Exp : Si nous utilisons les variables M, S dans un programme elles sont définies spontanément simple entière et simple réel respectivement.

II.2.1 Actions de déclaration des variables simples et composées

Avant de commencer à définir les processus utilisés dans actions de déclaration des variables, nous devons définir la convention utilisée dans le langage **Fortran** qui stipule que toutes les variables dont les noms qui commencent par des lettres I,J,K,L,M,N sont déclarée implicitement des entiers, mais pour éliminer la confusion avec le programmeur, nous devons exclure ces principes et adhérer à déclarer toutes les variables en fonction de leurs types si nous utiliserons l'expression du langage fortran **IMPLICIT NONE.**.

II.2.1.1 Actions de déclaration des variables entières

Pour déclarer les variables composées, telles que les matrices et les tableaux, nous utilisons la l'expression **DIMENSION** qui s'écrit selon la règle suivante :

C1	2345	6	789 72	73 80
			DIMENSION < Liste de variable composée>	

Remarque : le nom d'une variable est un identificateur alphanumérique dont son type doit être déclaré en avance et on distingue les types suivants : entier, réel, caractère, logique, complexe.

Pour déclarer les **variables entières** nous utiliserons l'action de déclaration INTEGER qui s'écrit selon la syntaxe suivante :

C1	2345	6	789 72	73 80
			INTEGER < Liste de variable simples>	

Exp :

C1	2345	6	789 72	73 80
			INTEGER A,B,C,K(10)	

Dans cet exemple A, B, C sont des variables simples entières, mais K(10) est une variable composée de 10 éléments entiers.

II.2.1.2 Actions de déclaration des variables composées réels

Pour déclarer les **variables réels** nous utiliserons l'action de déclaration REAL qui s'écrit selon la syntaxe suivante :

C1	2345	6	789 72	73 80
			REAL < Liste de variable réels >	

Exp :

C1	2345	6	789 72	73 80
			REAL K22,A(5,5),B,ISOM	

Dans la déclaration cité précédemment K22, B,ISOM sont des variables simples définies réels, A(5,5) est une matrice de 5 lignes et 5 colonnes d'éléments réels.

II.2.1.13 Actions de déclaration des variations littérales (caractères)

- La déclaration des **variations littérales** (caractères) diffère de ses prédécesseurs de variables car nous avons plusieurs règles pour les écrire :

1. Première forme de syntaxe :

C1	2345	6	789 72	73 80
			CHARACTER < Liste de variable de type caractère >	

Cette règle nous permet de déclarer un ensemble de variables littérales (caractères) consistant en un seul caractère.

Exp :

C1	2345	6	789 72	73 80
			CHARACTER A,B,C,M	

L'écriture précédente nous permet de déclarer un ensemble de variables littérales constituées chacune d'un seul caractere.

2. Deuxième forme de syntaxe :

C1	2345	6	789 72	73 80
			CHARACTER#<n>< Liste de variable de type caractère >	

Exp :

C1	2345	6	789 72	73 80
			CHARACTER#5 K, L, T, D	

Cette forme d'écriture précédente nous permet de déclarer un ensemble (K, L, T, D) de variables littérales constituées de 5 caractères chacune.

II.2.1.4 Actions de déclaration des variations logiques (boolean)

- La déclaration des **variations logiques** (Boolean) :

Une variable logique est une variable **bistable** soit VRAI OU FAUX (TRUE OR FALSE) utilise dans les actions logiques d'un programme :

C1	2345	6	789 72	73 80
			LOGICAL < Liste de variables logiques>	

Exp :

C1	2345	6	789 72	73 80
			LOGICAL U, V, E	

Dans cet exemple **U**, **V, E** sont des variables logiques initialisées a FAUX(FALSE) par le compilateur du langage Fortran.

Chapitre III : Les actions de traitement

III.1 L'expression d'action conditionnelle IF

Cette expression(**IF**) nous permet d'effectuer une ou plusieurs opérations si une condition ou plusieurs conditions sont remplies (vérifies).

Syntaxe d'écriture :

C1	2345	6	789 72	73 80
			IF < (< conditions logiques)> **<THEN>** [< Action 1>] [<Action 2>] . . . [<Action n>] **<ENDIF>**	

Exp1 :

C1	2345	6	789 72	73 80

			A=4
			B=2
			IF (A . GT. B) **THEN**
			M= A+B
			ENDIF	

Dans l'exemple précédent, la condition (A est supérieure à B) est remplie, nous pouvons donc effectuer l'action M=A+B , et à partir d'elle M←6.

Exp 2 :

C1	2345	6	789 72	73 80

			A=4
			B=2
			C=5
			IF (A . GT. B .AND. B .LT. C) **THEN**
			M= A+B+1
			ENDIF	

Dans l'exemple précédent, la condition (A est supérieure à B et B est inferieure à C) est remplie, nous pouvons donc effectuer l'action M=A+B +1, et à partir d'elle M←7.

Exp 3 :

C1	2345	6	789 72

			M=0
			A=4
			B=2
			C=1
			IF (A . GT. B .AND. B .LT. C) **THEN**
			M= A+B+1
			ENDIF

Dans l'exemple précédent, la condition (A est supérieure à B et B est inferieure à C) est n'est pas remplie(la deuxième condition B < C est fausse) , nous ne pouvons pas donc effectuer l'action M=A+B +1, et à partir d'elle M←0.

III.2 L'expression d'action conditionnelle IF alternée

Ce deuxième type d'expression de processus conditionnel, qui est appelé **expression d'action conditionnelle alternée**, nous permet d'effectuer un certain ensemble d'opérations si une **condition** ou **plusieurs conditions** sont vérifies **sinon** nous effectuerons d'autres actions.

Syntaxe d'écriture de l'expression IF :

C1	2345	6	789 72	73 80
			IF < (conditions logiques)> <**THEN**> **[< Action 1>]** **[<Action 2>]** . . . **[<Action n>]** **<ELSE>** **[< Action A>]** **[<Action B>]** . . . **[<Action Z>]** **<ENDIF>**	

Exp1:

C1	2345	6	789 72

			M=0
			A=4
			B=2
			C=1
			IF (A . GT. B) **THEN**
			M= A+B+1
			ELSE
			C=C+1
			ENDIF

Dans l'exemple précédent, la condition **(A> B)** est vérifié l'action **M=A+B+1** sera exécutée et le l'action **C=C+1** ne sera pas prise en considération.

Exp2 :

C1	2345	6	789 72

			M=0
			A=4
			B=2
			C=1
			IF((A –B) . EQ . 0) **THEN**
			M= A+B+1
			ELSE
			C=C+1
			ENDIF

Dans l'exemple précédent, la condition (**(A- B)** = 0) est n'est pas vérifié l'action **M=A+B+1** ne sera pas prise en considération, l'autre action **C=C+1** sera exécutée.

III.3 L'expression d'actions répétitives DO

Cette expression permet un ou plusieurs actions dans un intervalle défini

Syntaxe d'écriture de l'expression :

C 1	2345	6	789 72
			DO < variable contrôlée entière > =< valeur initiale entière>, <valeur finale entière>, [<pas>] [< ACTIONS>] **ENDDO**

Exp 1 : Calculer la somme des termes de 1 a 100

C1	2345	6	789 72	73 80
			ISOMME=0 **DO** I= 1 , 100,1 ISOMME= ISOMME + I **ENDDO**	

Remarque :

- I : est une variable entière contrôlée par l'expression **DO** (elle est incrémentée une manière **implicite** par l'expression DO)
- 1 c'est la valeur entière initiale
- 100 est la valeur entière finale
- 1 est le **PAS** d'incrémentation de la variable contrôlée I
- Les actions se trouvant entre **DO** et **ENDDO** constituent le corps de la boucle **DO**
- Il ne faut jamais mettre la variable contrôlée a gauche d'une expression arithmétique dans le corps de la boucle **DO exp (I=I+1 ca c'est faux).**

Processus de déroulement du fragment de programme précédant

ISOMME	I
0	1
1	2
3	3
6	4
	.
	.
	.
	100

III. 4 L'expression d'actions répétitifs DOWHILE

L'expression **DOWHILE** crée une boucle qui exécute un ensemble d'action jusqu'à ce qu'une condition de test ne soit plus vérifiée.

L'expression nous permet de répéter un certain nombre d'action **tant qu'une condition** est vérifie.

Syntaxe d'écriture :

C1	2345	6	789 72	73 80
			DOWHILE (< conditions>) [<actions>] **ENDDO**	

Exp :

C1	2345	6	789 72	73 80
			J=1 **DOWHILE** (J .LE. 10) ISOM=ISOM+J J=J+1 **ENDDO**	

Processus de déroulement de l'exemple :

J	Condition	Actions
1	**Vrai**	ISOM=0+ 1, J=1+1
2	**Vrai**	ISOM=1 + 2, J=2+1
3	**Vrai**	ISOM=3+ 3, J=3+1
4	**Vrai**	ISOM=6 + 4 J=4+1
5	**Vrai**	ISOM=10+ 5 , J=5+1
6	**Vrai**	ISOM=15 + 6 J=6+1

7	**Vrai**	ISOM=21 + 7, J=7+1
8	**Vrai**	ISOM=28+ 8, J=8+1
9	**Vrai**	ISOM=36 + 9, J=9+1
10	**Vrai**	ISOM=45 + 10, J=10+1
11	**Faux**	

III.4 Les expressions de lecture et écriture (READ, WRITE)

Les deux expressions nous permettent d'introduire les données à la l'ordinateur et d'écrire les résultats à partir de la mémoire centrale de l'ordinateur.

Syntaxe d'écriture de l'expression READ :

C1	2345	6	789 72	73 80
	M		**READ(**N,M) < Liste de variables) **FORMAT**(<Type de format>)	

N : C'est le numéro logique de l'unité utilisée pour la l'introduction des données à la mémoire de l'ordinateur par défaut est égale à **5** lorsque il s'agit du clavier.

M : C'est un numéro sur cinq positions pour designer l'étiquette du format de la lecture (1 à 99999)

Type de format: le type de format des données à lire :

Le format **In** : **I** pour les données **entières, n** est le nombre de position dans le nombre.

Exp : le format de lecture de la valeur 654789 est **I6**

Le format Fm,n : F pour la lecture et écriture des données réels , **m** est le total de chiffre dans le nombre y compris la virgule points et **n** est le nombre de chiffre après la virgule.

Exp : le format d'écriture de -34265.456 est **F10.3**

Le format At : A pour la lecture et écriture des données de type **caractère**, **t** est le total de caractère dans la chaine de caractère.

Exp : le format d'écriture de la chaine de caractère '**universite'** est **A10**

Exp : k=13, ib= - 6543, betta = 3455098, r= 3675.33,s= -13.0,nom='rtabc',prenom='medali'

L'expression de lecture des variables k,ib,betta,r,s,nom,prenom est

C1	2345	6	789 72	73 80
	231564		**READ**(5,231564) K,IB,BETTA,R,S,NOM,PRENOM **FORMAT**(**I2,I5,I7,F7.2,F4.1,A5,A6**)	

Syntaxe d'écriture de l'expression WRITE:

C1	2345	6	789 72	73 80
	M		**WRITE**(N,M) < Liste de variables) **FORMAT**(<Type de format>)	

N : C'est le numéro logique de l'unité utilisée pour la visualisation des résultats à partir de la mémoire de l'ordinateur par défaut est égale a **6** lorsque il s'agit de l'**écran**..

M : C'est un numéro sur cinq positions pour designer l'étiquette du format de l'écriture (1 à 99999)

Type de format : le type de format des résultats à écrire :

Le format **In** : **I** pour les données entières, **n** est le nombre de position dans le nombre.

Le format Fm,n : **F** pour la lecture et écriture des données réels, **m** est le total de chiffre dans le nombre y compris la virgule points et **n** est le nombre de chiffre après la virgule.

Le format At : **A** pour la lecture et écriture des données de type caractère, **t** est le total de caractère dans la chaine de caractère.

Exp : le format d'écriture de la valeur 654789 est **I6**

Le format d'écriture de la valeur – 5643.78 est **F8.2**

Le format de la lecture de la chaine de caractère ' annaba' est **A6**

Exp : k=13, ib= - 6543, betta = 3455098,nom='univers',adresse= 'bloc 10 annaba'

L'expression de lecture des variables k, ib, betta, adresse est :

C1	2345	6	789 72	73 80
	204		**WRITE**(6,**204**) K,IB,BETTA,NOM,ADRESSE **FORMAT(I2,I5,I7,A4,A14**)	

Notion de sous-programmes

En programmation il est préférable de découper le programme en plusieurs fragments de programme qu'on appel sous-programmes, les avantages de ce principe de fragmentation en programmation est:

- La vérification de chaque fragment est très facile, car il est facile de connaitre le tampon de l'erreur dans le programme.
- Facilite d'ajustement en cas de besoin
- Pour empêcher la répétition (favoriser la réutilisation), si nous devions répéter un ensemble d'instructions pour effectuer une tache spécifique plusieurs fois il vaut mieux le mettre dans un sous programme, et nous l'appellerons que lorsque cela est nécessaire.
- La facilité de compréhension du code lorsqu'il est affiché et descriptive.

On langage de programmation fortran on peut distinguer deux types de sous programmes :

SUBROUTINE et FUNCTION : Qui sont des programmes partiels contenant un ensemble de commandes et instructions de programmation effectuant une tache spécifique.

Lorsque nous appelons la procédure(**SUBROUTINE)** ou bien la fonction, l'ensemble des instructions contenues dans cette procédure ou bien fonction(**FUNCTION)** est exécuté.

Après la fin de la dernière instruction de la procédure ou bien la fonction, le contrôle sera transféré au programme appelant.

Syntaxe d'écriture de **SUBROUTINE** et **FUNCTION** :

C1	2345	6	789 72	73 80
			SUBROUTINE < nom de la procédure><(<paramètres >) [< Actions>] **RETURN** **END** <Type de la function> **FUNCTION**<paramètres> [< Actions>] **RETURN** **END**	

Exemples d'illustration de PROCEDURE et FUNCTION

C1	2345	6	789 72
			SUBROUTINE ADDITION(A,B,C) REAL C C=A+B **RETURN** **END**

C1	2345	6	789 72
			REAL **FUCTION** POLYNOME(X) REAL X POLYNOME=X**2 + 2*X – 1.0 **RETURN** **END**

Remarque :

Nous utiliserons les instructions CALL ADDITION (A, B, C) et Y= POLYNOME(X) pour appeler respectivement les deux sous programmes **SUBROUTINE** et **FUNCTION.**

Annexe :

Questions

Exercice 1) Donnez le format de lecture des variables suivantes :

3425, -7685, 675784, 786 , - 4.243 , ALPHA23, ANNABA_23

Exercice 2) Ecrire le programme qui fait la résolution de N équations seconde degré

Exercice 3) écrire le programme qui calcul la somme des 10 premiers nombres

Exercice 4) Ecrire le programme qui fait la lecture d'un table de 10 éléments entiers.

Exercice 5) Ecrire le Program qui chercher l'existence d'une valeur VAL dans un tableau

Exercice 6) Ecrire sous forme de SUBROUTINE PUIS FUNCTION le Programme qui calcul la somme de A+B+C

Solutions :

Exercice 1) Le format de lecture des variables : I4, I5, I6, I3, F6.3, A7, A9

Exercice 2) Résolution de N équations seconde degré :

C1	2345	6	789 72	73 80
C	**Ce**		Fait la résolution de N équations seconde degré	
			PROGRAM RESOL_SEC_DEGRE	
C			Déclaration des variables A,B,C,DELTA,X1,X2,NB	
C			I**mplicitemen**t A,B,C,DELTA SONT DES **REELS**	
C			NB : Le nombre d'équations à résoudre est **entier**	
C			Lecture de NB	
			READ(5,12) NB	
	12		**FORMAT**(I3)	
			DOWHILE (NB .GT. 0)	
C			LECTURE DE A,B,C	
			READ(5,6) A,B,C	
	6		FORMAT (F6.2,1X,F6.2,1X,F6.2)	
C			RQ : 1X : c'est laisser un blanc entre chaque variable	
C			CALCUL DE **DELTA**	
			DELTA = B*B – 4*A*C	
C			Testez le signe de DALTA	
			IF(DELTA .GE. 0) **THEN**	
			X1=(- B + SQRT(DELTA))/(2.0 * A)	
			X2=(- B - SQRT(DELTA))/(2.0 * A)	
C			SQRT : C'est une fonction bibliothèque **racine carré**	
C			Ecriture des résultats X1,X2	
			WRITE(6,2222) X1,X2	
	2222		FORMAT(' X1 = ', F6.2 , 1X, ' X2= ', F6.2)	
			ELSE	
			WRITE(6,45)	
	45		FORMAT(' PAS DE SOLUTION ')	
			NB = NB -1	
			ENDDO	
			STOP	
			END	

Exercise 3)

C1	2345	6	789 72	73 80
C	**Ce**		Programme calcul 1+2+3+4+5+6+7+8+9+10	
			PROGRAM SOMME	
C			Déclaration de la variable SOMME	
			INTERGER SOMME	
C			I**mplicitemen**t I est entier : l'indice qui parcourt l'intervalle de 1 → 10	
C			Initialisation de la SOMME ←0	
			SOMME = 0	
			DO I = 1 , 10 , 1	
C			Le **PAS d'incrémentation** est égale 1	
			SOMME = SOMME + I	
			ENDDO	
C			Ecriture du résultat	
C			==========	
			WRITE(6,145)	
	145		FORMAT (' La somme = ', I4)	
			STOP	
			END	

Exercise 4)

C1	2345	6	789 72	73 80
C	**Ce**		Programme qui fait la lecture d'un Tableau de 10 éléments entiers	
			PROGRAM LECTURE	
C			Déclaration Du Tableau	
			DIMENSION ITAB(10)	
C			**K** est l'indice qui parcourt le Tableau	
C			Lecture des éléments du Tableau ITAB	
			DO K = 1 , 10 ,1	
			READ(5, 100)(ITAB(K),K=1,10)	
	100		**FORMAT(**10(I3,1X))	
			ENDDO	
			STOP	
			END	

Exercise 5)

C1	2345	6	789 72	73 80
			PROGRAM LECTURE	
			DIMENSION ITAB(10)	
			LOGICAL TROUVE	
			READ(5, 100)(ITAB(K),K=1,10) , VAL	
	100		**FORMAT(**10(I3,1X) , I4)	
			K=1	
			DOWHILE (K .LE. 10 .AND. TROUVE .NE. .TRUE)	
			IF (ITAB(K) .EQ. VAL) THEN	
			TROUVE = .TRUE.	
			ELSE	
			K=K+1	
			ENDIF	
			ENDDO	
			IF (TROUVE) **THEN**	
			WRITE(6,1) K	
	1		**FORMAT** (' La valeur existe ','la position := ' , I2)	
			ENDIF	
			STOP	
			END	

Exercise 6)

C1	2345	6	789 72
			PROGRAM LECTURE
			READ(5,1) A,B,C
	1		FORMAT(3(F4.2,2X)
C			Déclaration du sous programme SOM
			SUBROUTINE SOM(A,B,C,Y)
			REAL Y
			Y= A+B+C
			RETURN
			END
C			Déclaration du sous programme SOMME
			REAL FUNCTION SOMME(A,B,C,Y)
			REAL Y
			Y= A+ B+C
			RETURN
			END
C			APPEL DE LA PROCEDURE(SUBROUTINE) SOM
			CALL SOM(A,B,C,Y)
C			ECRITURE DU RESULTAT
			WRITE(6,2) Y
	2		**FORMAT** (' La somme = ', F6.2)
C			APPEL DE LA FONCTION (FUNCTION) SOMME
			RESUL = SOMME(A,B,C)
C			ECRITURE DU RESULTAT
			WRITE(6,3) RESUL
	3		**FORMAT**(' La Somme = ' , F6.2)
C			
			STOP
			END

References bibliographiques

http://freddy.univ-tln.fr/enseignement/POLY_FORTRAN_FG.pdf

ECOLE D ' INGENIEURS DE FRIBOURG SECTION DE MECANIQUE COURS D ' INFORMATIQUE : SECONDE ANNEE ÉLÉMENTS DE PROGRAMMATION EN FORTRAN - 77