ALGO

Ch 2 – Bases d'un Langage algorithmique

I. INTRODUCTION 1

A. STRUCTURE GENERALE D'UN ALGORITHME DE FONCTION 1
B. STRUCTURE GENERALE D'UN ALGORITHME PRINCIPAL 2
C. LES COMMENTAIRES : % … % 2
D. IDENTIFIER ET DECRIRE L'ALGORITHME 2
E. DECLARER DES DONNEES : DECLARATION 3
F. DECRIRE LES INSTRUCTIONS : DEBUT … FIN 3

II. DECLARATION DES DONNEES : VARIABLES ET CONSTANTES 3

A. TYPES DE DONNEES DE BASE 3
B. VARIABLES 4
C. CONSTANTES 5
D. AU SUJET DES CONVENTIONS DE NOMMAGE 5

III. INSTRUCTION D'AFFECTATION, OPERATEURS ET EXPRESSIONS 6

A. VALEURS LITTERALES, OU LITTERAUX 6
B. EXPRESSIONS ET OPERATEURS 6
1. Expressions numériques 6
2. Expression alphanumérique 7
3. Expressions logiques et opérateurs relationnels 7
4. Expressions logiques et opérateurs logiques 8
C. L'AFFECTATION 8
D. CONSEIL AU SUJET DES VARIABLES 9

IV. INSTRUCTIONS D'ENTREE-SORTIE 9

A. AFFICHER DES INFORMATIONS : ECRIRE 9
B. RECUPERER DES DONNEES : LIRE 9

V. PREMIER TEST D'UN ALGORITHME 10

A. ALGORITHME 10
B. TRACE D'EXECUTION 11

I. Introduction

A. Structure générale d'un algorithme de fonction

Dans le cadre d'une fonction prise isolément :

- L'identification de la fonction (son identificateur, les paramètres formels (attendus) et le type de donnée retourné) :

```
FONCTION estMajeur(age : ENTIER) : LOGIQUE
```

- L'indication du rôle de l'algorithme :

```
ROLE détermine la notion de majorité
```

- L'identification des données qui seront utilisées :

```
DECLARATIONS
CONSTANTES
     AGE_MAJORITE = 18
VARIABLES
     résultat : LOGIQUE ← FAUX
```

- La suite d'actions à exécuter pour obtenir le résultat :

```
DEBUT
    SI (age >= AGE_MAJORITE)
       |  ALORS
       |      resultat ← VRAI
    finSI
    RETOURNER resultat
FIN
```

B. Structure générale d'un algorithme principal

Un algorithme exprimé en pseudo code comporte les éléments suivants :

- L'identification de l'algorithme (son nom) :

```
ALGO TesterMajorite
```

- L'indication du rôle de l'algorithme :

```
ROLE Afficher « mineur » ou « majeur » en fonction de l'age
```

- L'identification des éléments qui seront utilisés :

```
DECLARATIONS
    FONCTION estMajeur(age : ENTIER) : LOGIQUE
    VARIABLES
       unAge : ENTIER
```

- La suite d'actions à exécuter pour obtenir le résultat :

```
DEBUT
    AFFICHER ("donnez votre âge")
    SAISIR(unAge)
    SI (estMajeur(unAge))
       |  ALORS
       |      AFFICHER ("majeur")
       |  SINON
       |      AFFICHER ("mineur")
    finSI
FIN
```

C. Les commentaires : % … %

Les commentaires sont des explications textuelles ajoutées à l'algorithme afin d'apporter des informations supplémentaires au sujet de données utilisées ou de traitements effectués.

Syntaxe retenue :

```
% . . . n'importe quel texte . . . %
```

D. Identifier et décrire l'algorithme

Afin de préparer la phase de programmation, il est utile de donner un nom à l'algorithme afin de l'identifier de manière unique. La précision de son rôle est purement descriptif mais apporte une information sommaire sur son objectif.

Syntaxe retenue :

```
ALGO nom de l'algorithme
ROLE commentaire explicatif
```

E. Déclarer des données : DECLARATION

Toutes les données nécessaires à un algorithme doivent être connues : il est indispensable de les recenser et de :

- les identifier par un nom unique,
- et, pour chacune, de décrire la nature des valeurs qu'elle peut prendre (son domaine de valeurs, ou type de données).

La **DECLARATION** déclare l'existence des données qui seront utilisées dans l'algorithme (variable, constante) en leur attribuant un **identificateur** (nom de la donnée ou identifiant de la donnée), ainsi que le **type de cette donnée** (nature ou domaine de définition).

Lorsque l'algorithme sera devenu programme, la déclaration d'une variable va réserver un emplacement mémoire pour y stocker une valeur ; cet emplacement sera accessible grâce à l'identifiant de la donnée.

Un **TYPE DE DONNEE** détermine les valeurs autorisées pour le contenu d'une variable (nombres, chaine, etc.) et donc les opérations réalisables sur cette variable.

On parle aussi de **domaine de définition** : l'ensemble des valeurs que peut prendre une donnée, peut être défini

- en extension : liste de toutes les valeurs possibles,
- ou en compréhension : avec une définition de l'ensemble (ex. : les nombres entiers)

Les types de données généralement utilisés sont les suivants :

- caractères seuls : les lettres de l'alphabet, les chiffres, etc.
- textes (chaines de caractères) : assemblage de caractères qui forment un mot, une phrase
- nombres :
  - Entiers : nombres sans décimale
  - Réels : nombres avec décimale
- valeurs logiques VRAI et FAUX

F. Décrire les instructions : DEBUT … FIN

Les mots-clefs DEBUT et FIN vont délimiter la partie de l'algorithme qui va décrire la séquence d'actions à réaliser.
Ces actions vont utiliser et modifier le contenu des données déclarées.

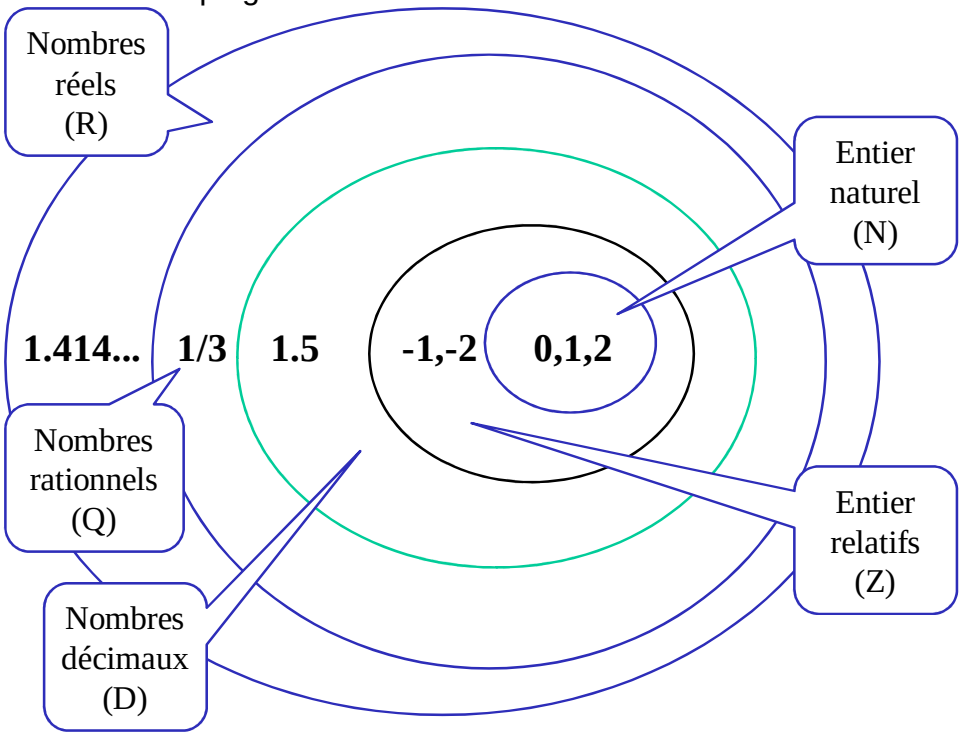
II. Déclaration des données : variables et constantes

Chaque donnée manipulé dans un algorithme doit comporter un **identificateur** (ou identifiant) associé aux éléments qui décrivent sa structure : nature (ou domaine de définition), valeur initiale.

A. Types de données de base

	Nombres		Textes		Logique
Types de de base	**ENTIER**	**REEL**	**CARACTERE**	**CHAINE**	**LOGIQUE (BOOLEEN)**
En Anglais	*integer*	*float*	*character*	*string*	*boolean*
Contenu	Nombre sans décimales	Nombre avec décimales	1 seul caractère	Plusieurs caractères	2 valeurs possibles
Exemples	0, -125, 10, 3256, 2654980	123.56 -0.56 3.1415927	'a' 'b' '-'	"bonjour" "bonne annee"	VRAI FAUX

Concernant les nombres (simplifié ici en ENTIER et REEL), il est parfois nécessaire d'être plus précis. L'utilisation des ensembles peut être alors utilisée. Parfois même, il sera nécessaire de réduire ces ensembles à des plages de valeurs.



B. Variables

Une **VARIABLE** est une donnée déclarée dont **LE CONTENU PEUT ETRE MODIFIE** au cours de l'exécution d'un algorithme (= son type de donnée est fixe, son contenu variable)

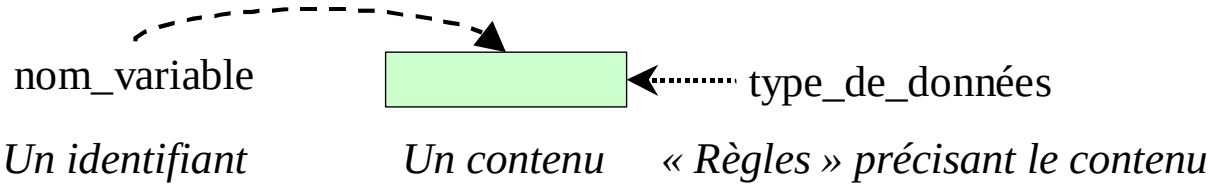


Figure 1 : la déclaration d'une variable associer un identifiant à un contenu, ce dernier devant respecter des règles (nombre, texte, etc.)

Syntaxe retenue:

```
VARIABLE   nom_variable : type_de_données [← valeur_initiale]
```

- **nom_variable**
  - identificateur unique de la variable
  - composé de lettres et chiffres et du caractère « _ » (convention retenue)
- **type_de_données** est l'un des types défini ci-dessus
- **valeur_initiale** (optionnel) est une valeur affectée à la variable avant de début de l'algorithme

Exemples :

```
VARIABLES
  age : ENTIER
  prime : REEL
  prime : REEL
```

```
  lettre : CARACTERE
  titre : CHAINE
  estMajeur : BOOLEEN
```

C. Constantes

Une **CONSTANTE** est une donnée dont **LE CONTENU NE PEUT PAS ETRE MODIFIE** au cours de l'exécution d'un algorithme.

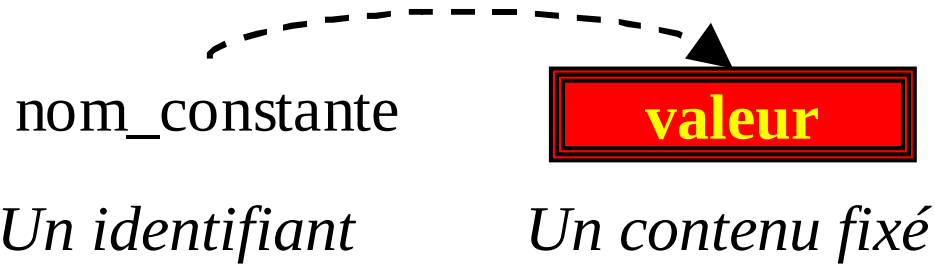


Figure 2 : une constante associe un identifiant à un contenu fixé, qui ne pourra changer au cours du déroulemen de l'algorithme

Syntaxe retenue:

```
CONSTANTES NOM_CONSTANTE : type_de_données ← valeur
```

- **NOM_CONSTANTE**
  - identificateur unique de la constante, composé de lettres majuscules et de chiffres et du caractère « _ » (convention retenue)
- **valeur :**
  - valeur du contenu que prendra cette constante tout au long de l'algorithme

Exemples :

```
CONSTANTES
  PI : REEL ← 3.1415927
  TAUX_EURO : REEL ← 6.55957
  MOT_ACCUEIL : CHAINE ← "Bonjour !"
  NOMBRE_DEBUT : ENTIER ← 1, NOMBRE_FIN : ENTIER← 10
  DEBUT_ALPHABET : CARACTERE ← 'a'
```

D. Au sujet des conventions de nommage

Une **CONVENTION DE NOMMAGE** détermine les règles (caractères utilisées et agencement de ces caractères) qu'on se fixe pour attribuer des noms aux variables, constantes, nom d'algorithmes, de programmes, etc.

Important :

Les conventions de nommage participent à la qualité des algorithmes et des programmes : il est important d'en définir … et les respecter !

III. Instruction d'affectation, opérateurs et expressions

A. Valeurs littérales, ou littéraux

Une **VALEUR LITTERALE**, ou un **LITTERAL**, désigne **UNE VALEUR D'UN CERTAIN DOMAINE DE VALEURS** (nombre, caractère, texte, etc.), **SANS IDENTIFICATEUR**.

On peut dire qu'un littéral est un contenu sans identifiant.

Exemple de littéraux : 2010, -100, 1, 3.14, 'z', "bonjour"

Un signe – (moins) précède une valeur numérique (ENTIER ou REEL) pour indiquer qu'elle est négative.

B. Expressions et opérateurs

Une **EXPRESSION** désigne le calcul d'une valeur à partir d'autres valeurs (littéraux, constantes et variables) et d'opérateurs. Les opérateurs utilisés dans une expression déterminent le domaine de valeur du résultat.
L'EVALUATION D'UNE EXPRESSION consiste à effectuer le calcul contenu dans cet expression afin d'obtenir la valeur résultante.

La valeur résultant de l'évaluation d'une expression est perdue, à moins d'être conservée comme contenu d'une variable: c'est l'objet de l'instruction d'affectation (voir plus bas).

1. Expressions numériques

Une **EXPRESSION NUMERIQUE** est une expression utilisant des valeurs de type **ENTIER** ou **REEL** et des opérateurs arithmétiques, et qui a pour valeur un nombre

Opérateur arithmétique	Signification	Exemple avec des valeurs littérales	Résultat de l'évaluation de l'expression (= la valeur de l'expression)
+	Addition	2 + 3	5
-	Soustraction	2 – 3	-1
*****	Multiplication, produit	3 * 2	6
/	Division, quotient	3 / 2	1.5
MOD	Modulo, reste de la division	3 MOD 2	1

Dans une expression numérique comportant plusieurs opérateurs, les calculs intermédiaires sont effectués selon la priorité des opérateurs : (idem. Mathématiques) : EMDAS (Exponentiation, Multiplication et Division et Modulo, Addition et Soustraction), puis, au même niveau de priorité, de la gauche vers la droite

- Par exemple : 2 + 3 * 5 ➔ effectuer 3 * 5, soit 15, puis ajouter 2 ➔ 17

L'usage des parenthèses est fortement conseillé afin de définir le calcul sans ambiguïté (mais aussi d'améliorer la lisibilité des expressions) :

- Par exemple : (2 + 3) * 5 ➔ effectuer (2 + 3), soit 5, puis 5 * 5 ➔ 25

2. Expression alphanumérique

Une **EXPRESSION ALPHANUMERIQUE** est une expression combinant des valeurs de type CARACTERE ou CHAINE et un opérateur alphanumérique.

Opérateur alphanumérique	Signification	Exemple avec des valeurs littérales	Résultat de l'évaluation
&	concaténation	″ta″ & ″tou″	″tatou″

Exemple :

```
"ta" & "tou" & "12"  % va donner la valeur "tatou12" %
```

3. Expressions logiques et opérateurs relationnels

Une **EXPRESSION LOGIQUE ou BOOLEENNE** est une expression dont le résultat est une valeur logique ou booléenne, soit VRAI ou FAUX.
Elle peut résulter de la combinaison de valeurs (littéraux, variables, constantes, expressions arithmétiques, etc.) et d'opérateurs relationnels.

Les opérateurs relationnels mettent en relation 2 valeurs pour les comparer :

Opérateurs relationnels	Signification	Exemple	Résultat de l'évaluation
=	égal	2 = 3	**F**(AUX)
!=	différent	2 != 3	**V**(RAI)
<	Inférieur	2 < 3	**V**(RAI)
<=	Inférieur ou égal	2 <= 3	**V**(RAI)
>	Supérieur	2 > 3	**F**(AUX)
>=	Supérieur ou égal	2 >= 3	**F**(AUX)

Exemple :

```
(age >= 18) ➔ VRAI si age est >= 18, FAUX sinon
(1 = 1) ➔ toujours VRAI
(1 = 2) ➔ toujours FAUX
( ( age + 2) < (3 * 6) ) ➔ VRAI si age est inférieur à 16
```

4. Expressions logiques et opérateurs logiques

Un **EXPRESSION LOGIQUE** peut aussi résulter de la combinaison de valeurs logiques et d'opérateurs logiques.

Les opérateurs logiques ET, OU et NON permettent la construction d'expressions logiques plus élaborées :

Expression logique A	Expression logique B	**A ET B**	**A OU B**	**NON A**
F	F	F	F	V
F	V	F	V	V
V	F	F	V	F
V	V	V	V	F

Exemple :

`(age >= 18) ET (note > 10)` ➔ VRAI si les 2 expressions sont vraies - FAUX sinon
`(age >= 18) OU (note > 10)` ➔ VRAI si au moins des deux expressions logique est VRAI - FAUX sinon

C. L'affectation

L'**AFFECTATION** est l'opération qui REMPLACE le contenu d'une variable avec une nouvelle valeur.

La nouvelle valeur doit être de même nature que le variable affectée.
L'affectation modifie l'état des variables de l'algorithme et le fait progresser vers le résultat.

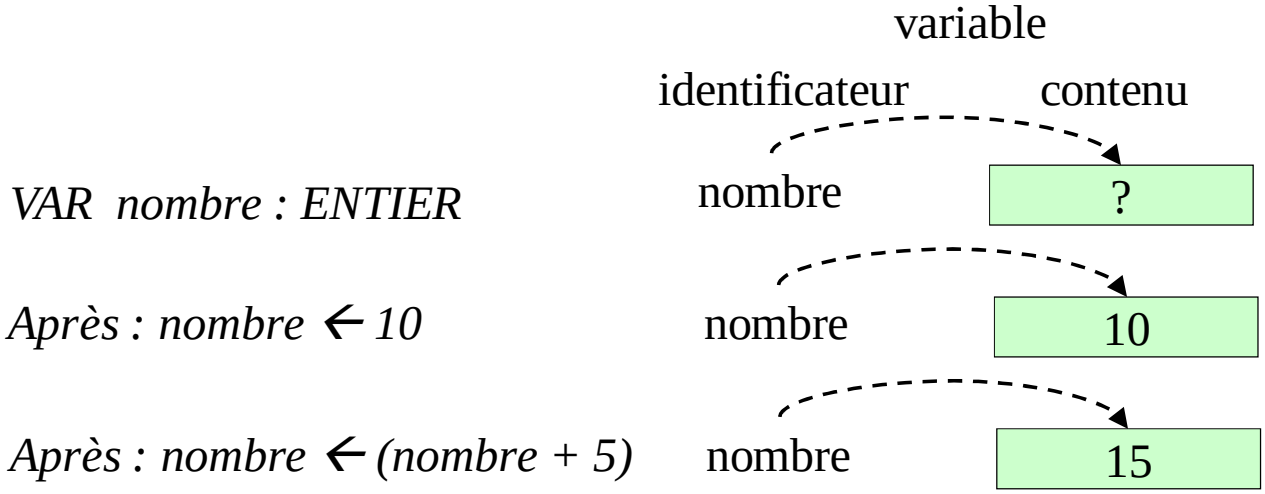


Figure 3 : évolution de la valeur (= du contenu) de nombre après chaque affectation

Syntaxe retenue :

`nom_variable ← littéral_ou_expression`

- **Nom_variable**
  - nom d'une variable préalablement déclarée
- **littéral_ou_expression**
  - valeur à affecter à la variable, valeur littérale , résultat de l'évaluation d'une expression, etc.

Exemples avec des valeurs littérales :

```
age ← 20
prime ← 120.5
```

```
estMajeur ← VRAI
titre ← 'titre du programme'
lettre ← 'a'
```

Exemples avec des expressions :

```
age ← (20 + 5)
prime ← (120.5 * (1 + 0.50))
estMajeur ← (age > 17)
titre ← "titre" & "du" & "programme"
```

D. Conseil au sujet des variables

L'INITIALISATION D'UNE VARIABLE consiste à lui affecter une valeur initiale en fonction de son domaine de valeurs avant le début de l'algorithme ←→ TOUTE VARIABLE DOIT SUBIR UNE AFFECTATION AVANT D'ÊTRE UTILISEE

En général, à une variable

- caractère, on affecte la valeur ' ' (un caractère vide),
- chaîne, on affecte la valeur " " (un texte vide),
- numérique, on affecte la valeur 0,
- logique, on affecte la valeur FAUX.

IV. Instructions d'Entrée-sortie

Les actions d'entrée-sortie vont permettre de définir des possibilités simples d'interactions entre l'algorithme et son utilisateur.
Des messages pourront lui être envoyés, et il pourra intervenir dans le déroulement de l'algorithme en fournissant des données nécessaires à son fonctionnement.

A. Afficher des informations : ECRIRE

L'action **ECRIRE** consiste à envoyer une information (un texte, une valeur calculée, etc.) à un utilisateur.

Autre syntaxe possible : AFFICHER, SORTIR

Syntaxe retenue :

ECRIRE (exp1, exp2, . . . , expN)

- **exp1, exp2, expN** représentent
  - l'expression ou les expressions dont on commande l'envoi
  - un code particulier permettant de sauter une ligne : NL (pour Nouvelle Ligne)

Exemple :

```
ECRIRE('bonjour')
ECRIRE('bonjour ',nomEtudiant)
```

B. Récupérer des données : LIRE

L'action **LIRE** consiste à récupérer une information fournie par l'utilisateur afin de l'affecter à une variable.

Autre syntaxe : SAISIR, ENTRER

Syntaxe retenue :

LIRE (nom_variable1, nom_variable2, ...)

- **nom_variable1, nom_variable2 :**
  - Nom d'une ou plusieurs variables qui vont recevoir les données fournies par l'utilisateur.

Exemple :

```
LIRE(age)
LIRE(nomEtudiant)
```

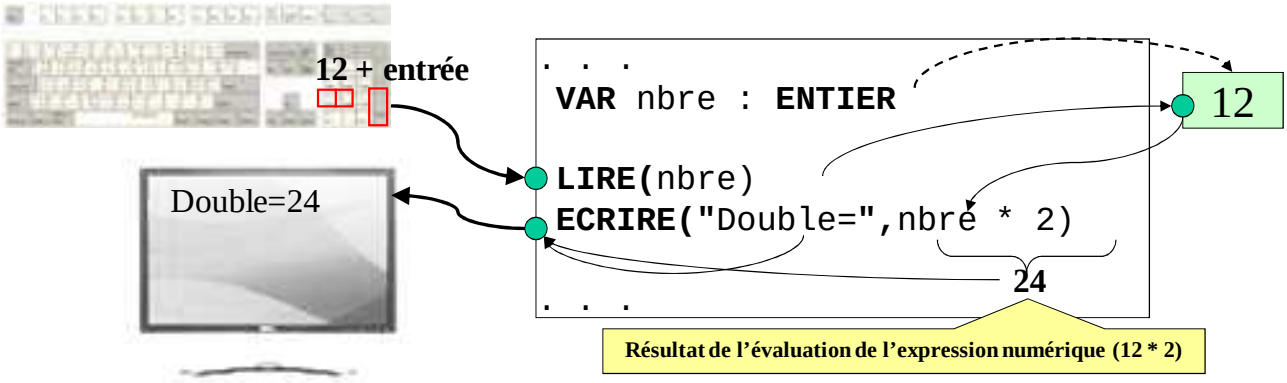


Figure 4 : LIRE et ECRIRE, les 2 instructions permettant le dialogue avec l'utilisateur

V. Premier test d'un algorithme

A. Algorithme

La numérotation des instructions va permettre le suivi du déroulement de l'algorithme :

```
% But de l'algo : calculer et afficher un montant TTC à partir de la saisie d'un nombre de produits et du prix HT du produit %
% Nom : Moi %
% Date de création : 04/09/2012 %

ALGO CalculerTTC
ROLE calculer un montant TTC à partir d'un nombre de produits
DECLARATIONS
CONSTANTES
  TAUX_TVA = 19.6
VARIABLES
  nombre : ENTIER
  prixUnitHT, montantHT, montantTVA, montantTTC : REEL
DEBUT
1.    ECRIRE ("donnez le nombre de produits")
2.    LIRE(nombre)
3.    ECRIRE ("donnez le prix unitaire")
4.    LIRE(prixUnitHT)
5.    montantHT ← nombre * prixUnitHT
6.    montantTVA ← montantHT * 19.6 / 100
7.    montantTTC ← montantHT + montantTVA
8.    ECRIRE ("TTC : ", montantTTC)
FIN
```

B. Trace d'exécution

La **TRACE D'EXECUTION** permet de **suivre le déroulement d'un l'algorithme** pas à pas (action après action) afin de s'assurer de son bon fonctionnement.
L'exécution est manuelle (VOUS êtes le processeur)

Exemple :

pas	*	Saisie	CONST	VAR					Ecran
			TAUX_TVA	nombre	prixUnitHT	montantHT	montantTVA	montantTTC	
DEB			19.6						
1									Donnez le nombre de produits
2		5 →	→	5					
3									Donnez le prix unitaire
4		10 →	→	→	10				
5						50			
6							9.8		
7								59.8	
8								● →	TTC : 59.8

(*) Cette colonne nous servira pour évaluer les expressions logiques

Les traces d'exécution permettent d'effectuer une première vérification de la validité de l'algorithme.
Cependant, on peut déjà noter la difficulté à assurer une validité complète d'un algorithme Celle-ci qui nécessite en effet la vérification de nombreux cas de tests, pour différentes valeurs saisies, les unes correctes, les autres non...
Le programme issu de l'algorithme devra aussi faire l'objet de tests sévères pour s'assurer de son bon fonctionnement.