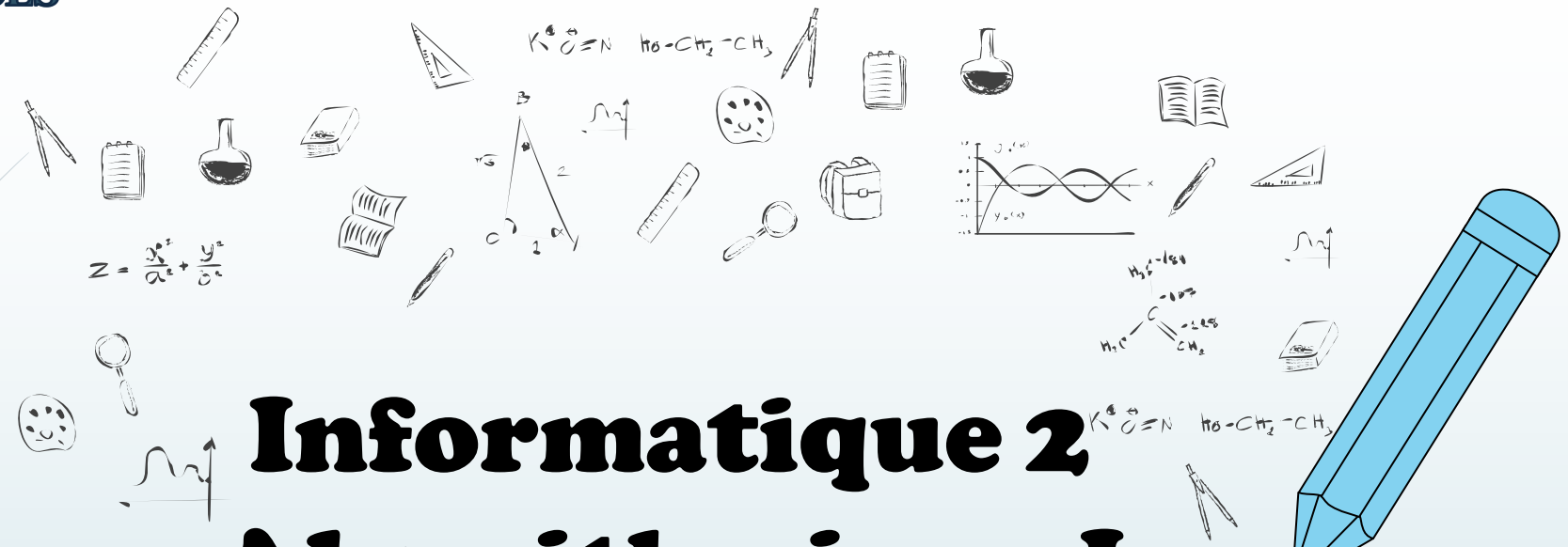
UNIVERSITE IBN ZOHR
FACULTE DES SCIENCES
AGADIR



Informatique 2
Algorithmique I

Filière : SMI2

Année universitaire : 2022/2023

Présentation du module

- Objectif :
  - se familiariser avec les méthodes de résolution de problèmes avec l'outil informatique ;
  - apprendre les principes de l'algorithmique ;
  - acquérir un début de maîtrise des techniques et langages de programmation;
  - Apprendre et maitriser les concepts de base de l'algorithmique et de la programmation;
  - Être capable de mettre en œuvre ces concepts pour analyser des problèmes simples et écrire les algorithmes correspondants
- Pre-requis : Connaissances générales en informatique utiles, mais pas indispensables.
- Professeur :
  - Fadwa Lachhab f.lachhab@uiz.ac.ma

Plan :

1. Introduction à l'algorithmique
2. Les instructions élémentaires
3. Les structures de contrôle conditionnelles
4. Les structures de contrôles répétitives
5. Les tableaux
6. Les Algorithmes de tri et de recherche

Chapitre 1 : Introduction à l'algorithme

- Définition d'un algorithme ;
- Structure d'un algorithme;
- L'exécution d'un algorithme ;
- Les variables : principe, déclaration, affectation ;
- Les instructions d'entrée et de sortie (saisie et affichage);
- Les constante;

Langage Informatique

Un langage informatique est un code de **communication**, permettant un être **humain** de dialoguer avec une **machine** en lui soumettant des instructions et en analysant les données matérielles fournies par le système.

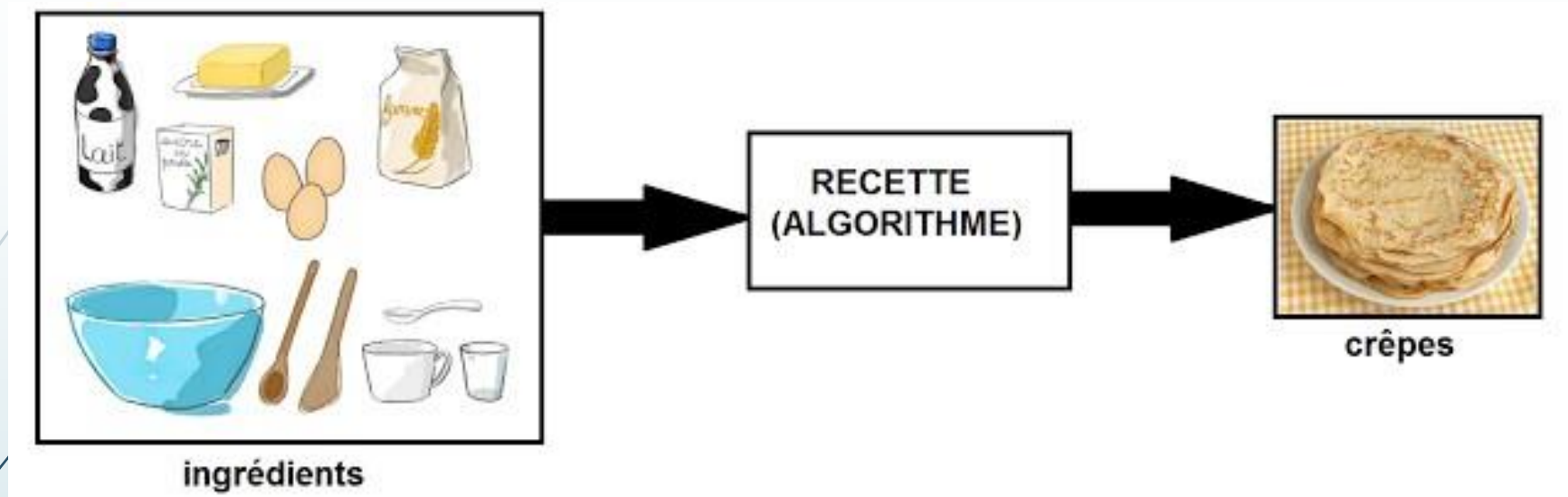
Le langage informatique est l'intermédiaire entre le programmeur et la machine.

Il permet d'écrire des **programmes** (suite consécutive d'instructions) destinés à effectuer une tache donnée

Programme

- Un **programme** correspond à la description d'une méthode de résolution pour un **problème donné**.
- Cette **description** est effectuée par une suite **d'instructions d'un langage de programmation**
- Ces instructions permettent de **traiter** et de **transformer** les données (**entrées**) du problème résoudre pour aboutir à des **résultats** (sorties).
- Un **programme** n'est pas une solution en soi mais une **méthode** à suivre pour trouver les solutions.

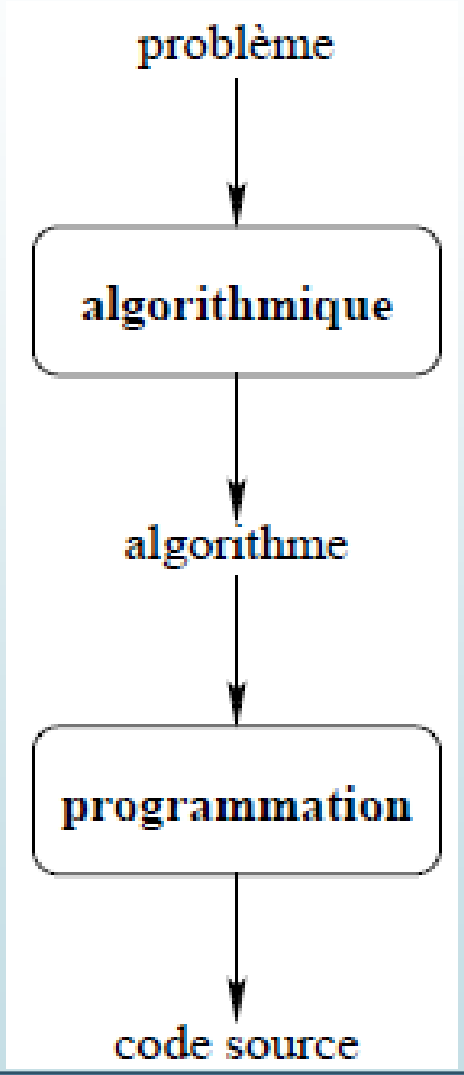
Définition : L'algorithme est une recette



Une définition simple d'un algorithme :
c'est une suite d'instructions qui, quand elles sont exécutées correctement aboutissent au résultat attendu.
C'est un énoncé dans un langage clair, bien défini et ordonné qui permet de résoudre un problème, le plus souvent par calcul.
L'algorithme est donc une recette pour qu'un programme d'ordinateur puisse donner un résultat donné.

Introduction :Définition d'un algorithme

- **Qu'est ce qu'un algorithme ?**
  - Définition informelle
    - Un algorithme est une procédure de **calcul bien définie** qui prend en entrée une valeur, ou un ensemble de valeurs, et qui donne en sortie une valeur, ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforment l'entrée en sortie.
  - Une autre définition
    - Un algorithme est une suite d'instructions, qui une fois exécutée correctement, conduit a un résultat donné.
- **Remarque**
  - On désigne par **algorithmique** l'ensemble des activités logiques qui relèvent des algorithmes.
  - Autrement dit: L'algorithmique s'intéresse à l'art de construire des algorithmes ainsi qu'a caractériser leur validité, leur robustesse, leur réutilisabilité, leur complexité ou leur efficacité.
  - L'algorithmique est la science des algorithmes



Introduction :Définition d'un algorithme

- **Exemples d'algorithmes**
  - indiquer un chemin a un touriste égaré ;
  - rédiger une recette de cuisine ;
  - élaborer un mode d'emploi pour faire fonctionner un magnétoscope ;
  - Etc
- **Important**
  - Pour fonctionner, un algorithme doit contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter..

Introduction :Définition d'un algorithme

Faut-il être matheux pour être bon en algorithmique ?

Non, pas du tout !

- La maitrise de l'algorithmique nécessite **trois qualités** :

1. **Il faut être méthodique :**

Avant d'écrire les instructions d'un algorithme, il faut analyser le problème à résoudre. Il faut ensuite définir les entrées et les sorties de l'algorithme.

Introduction :Définition d'un algorithme

Faut-il être matheux pour être bon en algorithmique ?

Non, pas du tout !

- La maitrise de l'algorithmique nécessite **trois qualités** :

2. **Il faut avoir de l'intuition :**

Aucune recette ne permet de savoir a priori quelles instructions permettront d'obtenir le résultat voulu. Les réflexes du raisonnement algorithmique deviennent spontanés avec l'expérience.

Introduction :Définition d'un algorithme

Faut-il être matheux pour être bon en algorithmique ?

Non, pas du tout !

- La maitrise de l'algorithmique nécessite **trois qualités** :

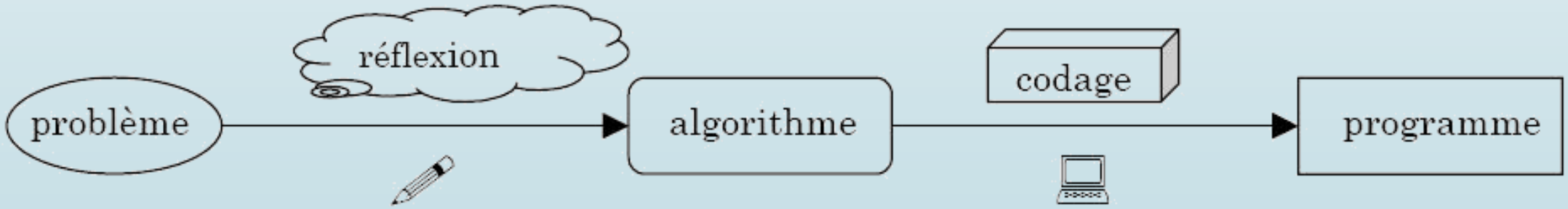
3. **Il faut être rigoureux :**

Chaque fois qu'on écrit une série d'instructions, il faut systématiquement se mettre mentalement à la place de la machine qui va les exécuter. Si nécessaire, il faut avoir recours à une simulation sur papier.

Pourquoi utiliser un algorithme ?

Un algorithme bien établi et qui fonctionne pourra être directement réécrit dans un langage de programmation évolué comme le C, Python, Java ou PHP.

Malheureusement, en programmation c'est souvent à l'homme de se mettre au niveau de la machine.



Le formalisme

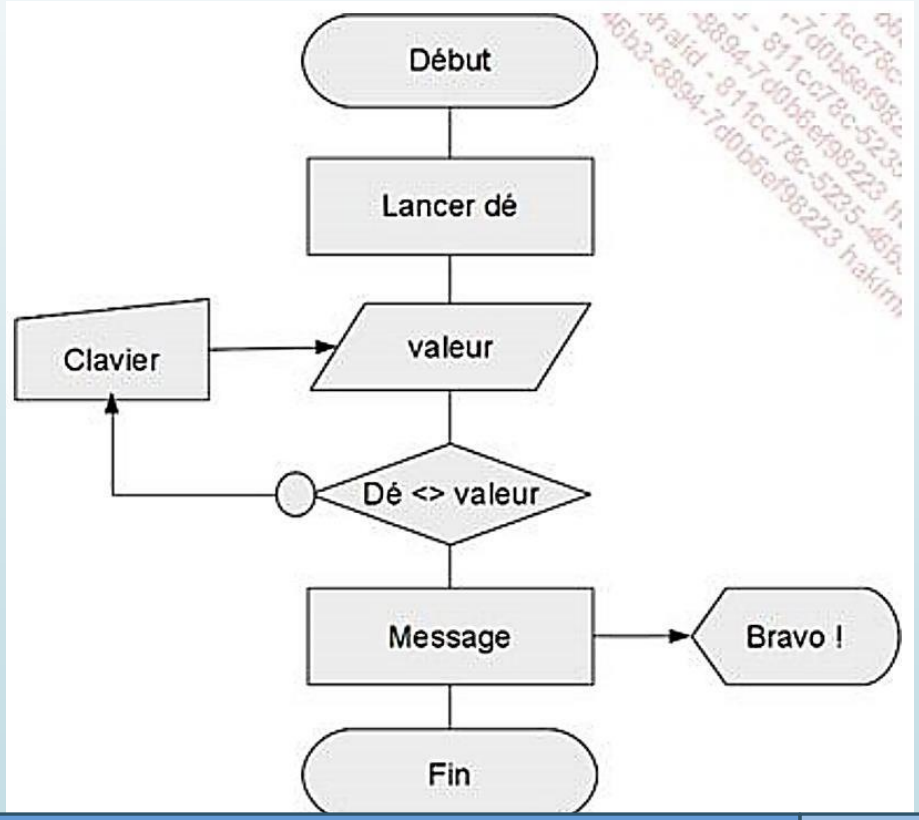
- Le but d'un algorithme étant de décrire un traitement informatique dans quelque chose de compréhensible par l'humain (et facilement transposable vers la machine), pour qu'un algorithme soit compréhensible, il faut qu'il soit clair et lisible. Dans ce cas, il existe deux moyens efficaces :
- soit d'écrire l'algorithme sous forme de texte simple et évident (faire ceci, faire cela),
- soit de faire un schéma explicatif avec des symboles.

Le formalisme

Les traitements sont dans des rectangles, les prises de décision dans des losanges, les affectations dans des parallélogrammes, et les flèches représentent les décisions ou des entrées/sorties. Le programme se déroule de haut en bas. La flèche précédée d'un cercle indique un « non ». Les décisions et les flèches permettent de décrire des boucles.

Les algorigrammes

Les algorithmes peuvent être construits à l'aide de symboles d'organigrammes. On appelle cette représentation des algorigrammes, organigrammes de programmation ou encore logigrammes.



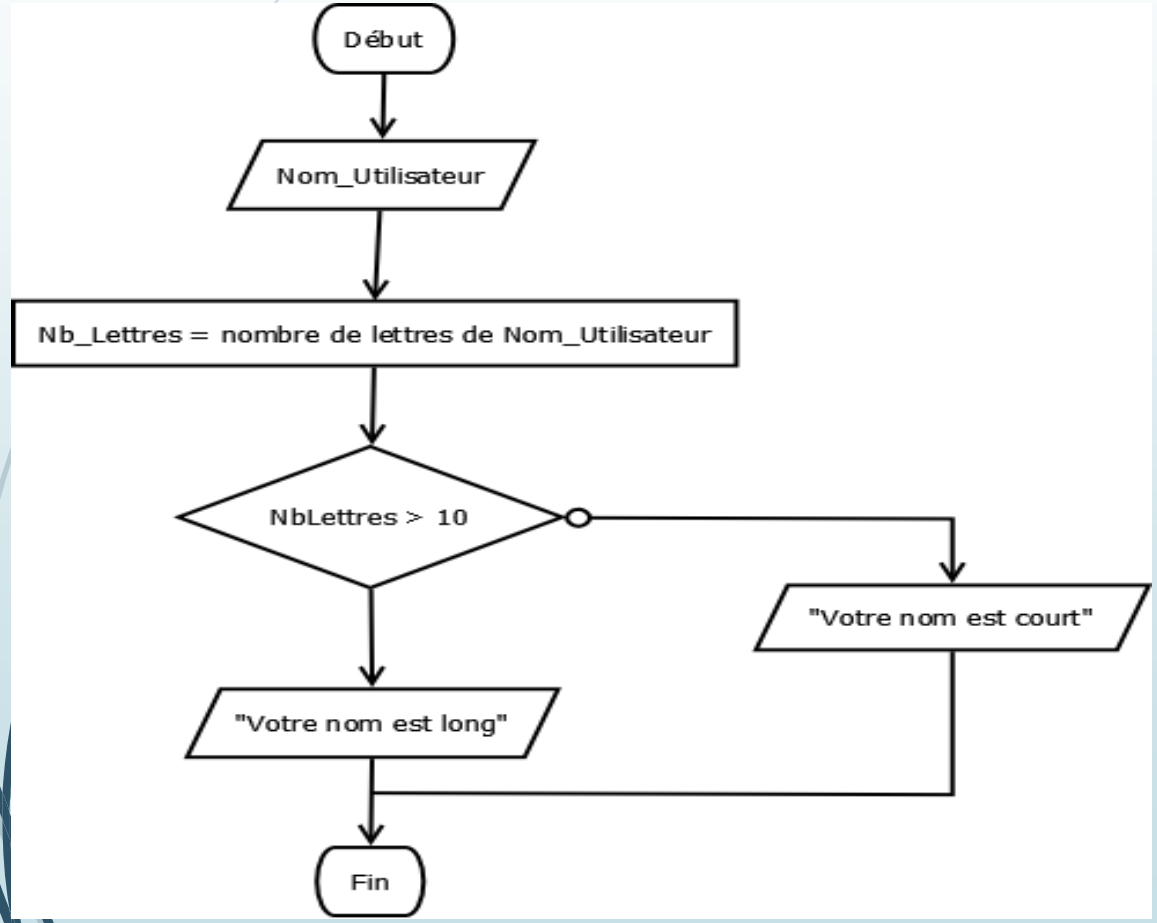
L'algorithme sous forme de texte

Prenez le même énoncé du lancé de dé. Celui-ci pourrait être écrit ainsi en français correct :

- Première étape : lancer le dé
- Deuxième étape : saisir une valeur
- Troisième étape : si la valeur saisie est différente de la valeur du dé, retourner à la deuxième étape, sinon continuer
- Quatrième étape : afficher "bravo".

Représentation d'un Algorithme

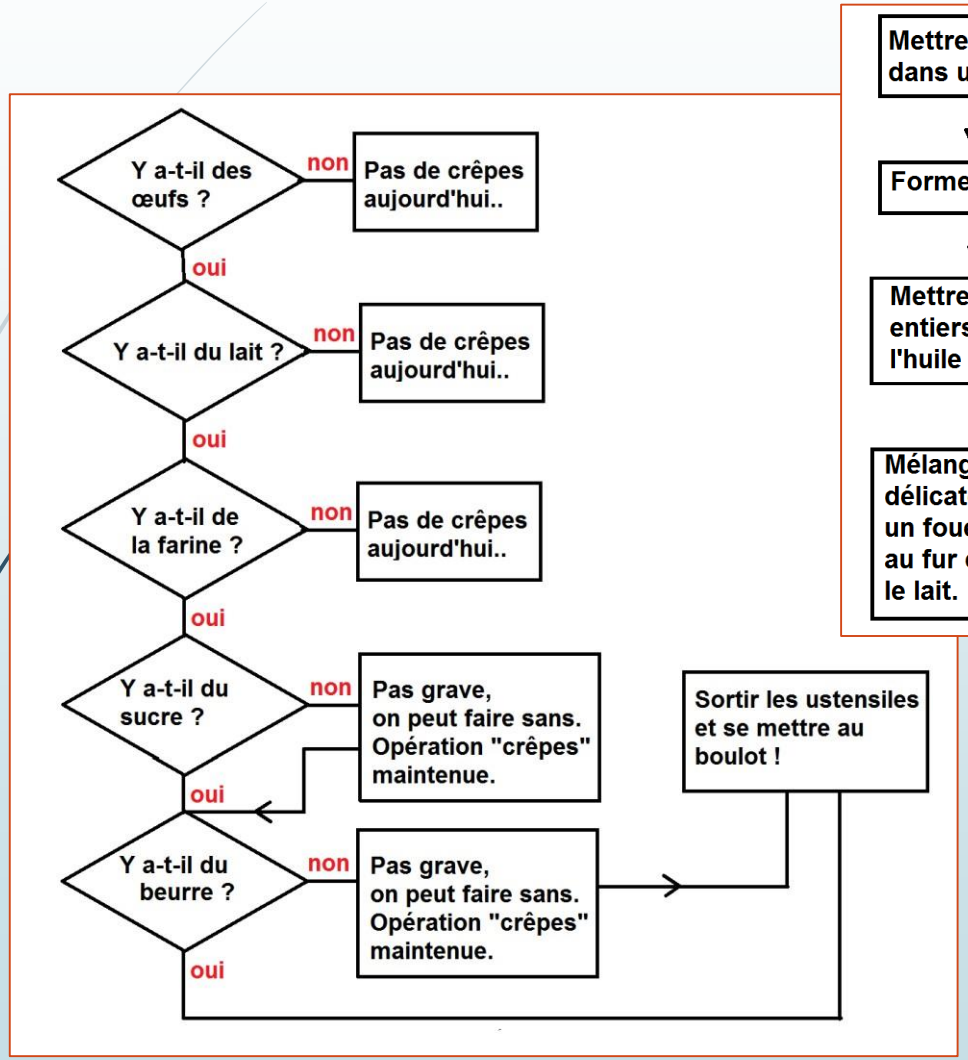
Les algorigrammes

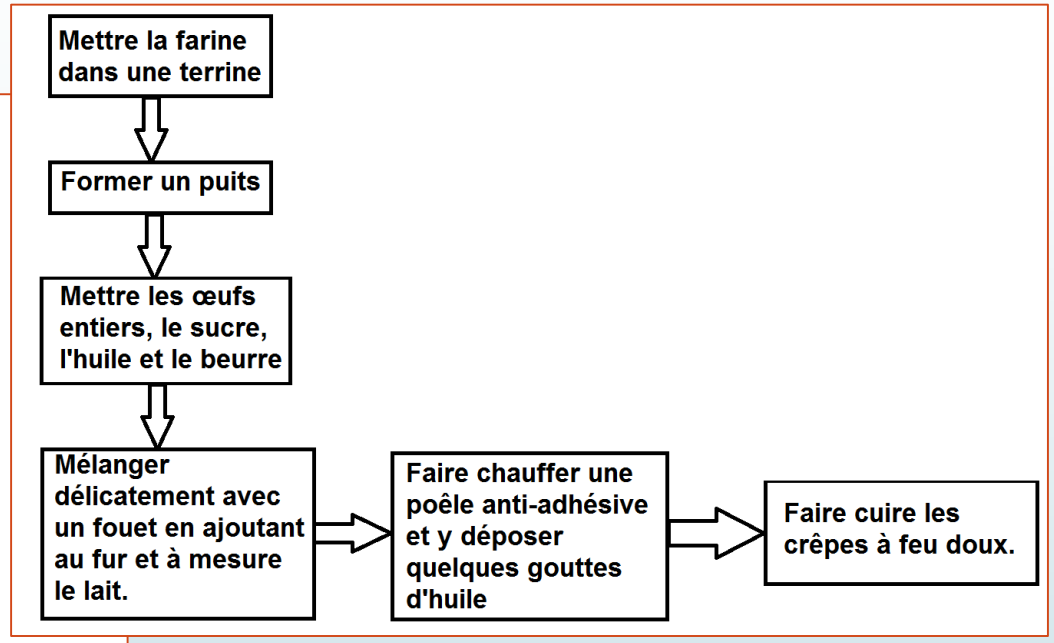


Les pseudo-code

Algorithme longueur_nom
Variables nom: chaine de caractère
taille: entier
Début
Ecrire ("Saisir votre nom: ")
Lire (nom)
taille ← Longueur(nom)
si (taille> 10) alors
Ecrire ("Votre nom est long")
sinon
Ecrire ("Votre nom est court")
Fin

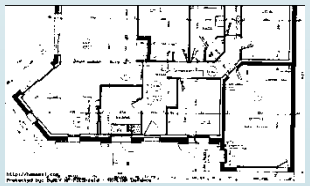
Les algorigrammes





Environnement algorithmique
L'algorithmique et la programmation ...

- **Quelle est la différence entre l'algorithmique et la programmation ?**
- **Réponse**
  - L'écriture d'un programme dans un langage de **programmation** n'est que l'**étape finale** d'un développement qui se déroule en trois phases : l'analyse, l'algorithmique et la programmation.
  - En d'autre terme : Un algorithme est un maillon de la chaîne de développement d'un programme. Il est le lien indispensable entre l'analyse et la programmation.
- **En utilisant des images :**
  - Si un programme était une construction, l'algorithme serait le plan
  - Si un programme était une toile de peinture, l'algorithme serait l'esquisse



Environnement algorithmique
Niveau logique du développement

- Apprendre l'algorithmique, c'est apprendre a manier la structure logique d'un programme informatique.
- L'algorithmique exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage.
- Lorsqu'on programme dans un langage (en C, C++, phyton en Visual Basic, etc.) on doit, en plus de la structure logique, prendre en considération les problèmes de syntaxe et les types d'instructions propres a ce langage.
- Niveaux de développement :
  1. Analyse : niveau conceptuel ;
  2. Algorithmique : niveau logique ;
  3. Programmation : niveau physique.

Environnement algorithmique
Représentation d'un algorithme

Exemple de conventions :

- exprimer les actions avec des verbes à l'infinitif ;
- numéroter les instruction dans l'ordre séquentiel en commençant par 1 ;
- exprimer le nom de l'acteur dans les instructions ;
- exprimer le lien entre l'acteur et l'action par le symbole $\longrightarrow$;
- encadrer les instructions de l'algorithme ;
- exprimer, en en-tête, le nom de l'algorithme.

Les structures algorithmiques

Tout raisonnement pouvant être décrit sous forme d'algorithme va être décomposé en structures de base :

- les structures de contrôle :
  - séquences (blocs d'instructions, fonctions, procédures, etc.).
  - conditionnelles (conditions, expressions booléennes, etc.).
  - boucles (itérations).
- les structures de données :
  - constantes.
  - variables.
  - tableau.
  - structures récursives.

Les structures algorithmiques

- un algorithme est un texte très structuré,
- Tout algorithme commence par le mot ***Programme*** suivi du *nom de l'algorithme et se termine par le mot* ***Fin***
- L'algorithme lui-même se subdivise en deux parties : la partie **déclarations des variables et la partie liste d'instructions délimitées respectivement par les** mots ***Variable et Début*****.**

Programme EXEMPLE

Variables

déclarations des variables

Début

liste des instructions

Fin

Les structures algorithmiques

Exemple : Soit un algorithme effectuant le calcul de Surface d'un disque

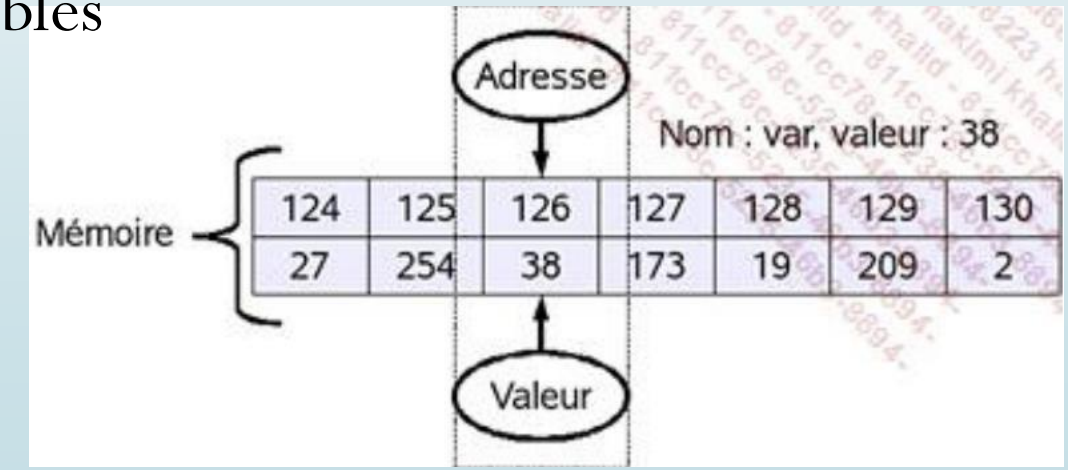
```
Algorithme Surface d'un disque
Variables Rayon, Surface : Réels
Constante Pi = 3.14
Début
   Ecrire ("Donner le rayon du disque ")
    Lire (Rayon)
    Surface ← (Rayon ^ 2)* Pi
    Ecrire ("La surface du disque est : ", Surface)
Fin
```

La variable

- la mémoire de l'ordinateur, composée de cases, peut contenir des informations.
- En programmation, il faut quelque chose de simple, pratique et souple à manipuler pour représenter ces informations. Chaque case de la mémoire est numérotée.
- Si la mémoire fait, disons, 256 octets, et que chaque case peut contenir un octet, alors il y a 256 cases numérotées de 0 à 255.
- Avec 1 Go de mémoire, vous avez 1073741824 cases pouvant contenir chacune un octet. Comment voulez-vous vous souvenir de chaque numéro de case ? C'est bien entendu impossible.
- Si par contre vous donnez un nom ou une étiquette à chaque valeur contenue dans la case, ou à une suite de valeurs de plusieurs cases, pour vous en rappeler plus facilement, cela devient bien plus évident. C'est ce qu'on appelle une variable.

La variable

- En informatique, une variable est l'association d'une étiquette à une valeur.
- Vous nommez la valeur. La variable représente la valeur et se substitut à elle.
- La variable est donc la valeur. Mais comme son nom l'indique, cette valeur peut changer dans le temps, soit que la variable ne représente plus la (ou les) même(s) case (s) mémoire, soit que la valeur de la case a changé.
- Voici quelques exemples de noms de variables
  - a
  - var
  - titre
  - Total
  - Somme_globale



Déclaration de la variable

- Les variables doivent être **déclarées** avant d'être utilisées
- Une variable est caractérisée par :
  - ✓ **Un nom :** qui sert à la désigner
  - ✓ **Un type :** définit la nature de l'information qui sera représentée dans la variable (numériques, caractères…)
  - ✓ **Une valeur** : à un instant donné, une variable ne peut contenir qu'une seule valeur

La déclaration d'une variable se fait comme suit :

VAR ou VARIABLE nom : TYPE

ou

VAR ou VARIABLES nom1, nom2, … : TYPE

Identificateur d'une variable

- Doit commencer par une lettre alphabétique
  exemple valide: **A1** exemple invalide: **1A**

- Doit être constitué uniquement de lettres, de chiffres et du soulignement _
  valides: **SMI2023, SMI_2023**
  invalides: **SMI 2023, SMA-2023, SMI;2023**

- Doit être différent des mots réservés du langage (par exemple en C: **int, float, long, else, for, if, return**, …)

- La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisé

Déclaration

- Pour exister, une variable doit être déclarée, c'est-à-dire que vous devez indiquer au début de l'algorithme comment elle s'appelle et ce qu'elle doit contenir.
- Il ne s'agit pas ici de définir la valeur de la variable, vous pourrez le faire dans la suite de l'algorithme en lui affectant une valeur.
- Il s'agit de donner **son nom** et de préciser **le type** de valeur qu'elle peut contenir. Les variables se déclarent au début de l'algorithme, avant le programme lui-même.

Déclaration

```
Programme HEURSEC
    Variables
        HH : numérique
        MM : numérique
        SS : numérique
        TOTALSEC : numérique
    Début
        Lire (HH)
        Lire (MM)
        Lire (SS)
        TOTALSEC⟵ HH*3600 + MM*60 + SS
        Ecrire (TOTALSEC)
    Fin
```

Déclaration des variables

Liste des instructions

Déclaration

Quelques exemples de noms de variables correctement écrits :

- Nombre, Numero, NoteAnglais, Total, Moyenne, Poids, Heure, Minute, Seconde, PlusPetit, MontantInitial

Quelques exemples de noms de variables incorrectement écrits :

- 40Voleurs, A+B, Tempér@ture, Montant Initial, Fin, Debut, Var

Les Types

- Une case mémoire contient généralement un octet, c'est-à-dire une valeur de 0 à 255. Mais une variable peut très bien contenir le nombre 214862, le réel 3,1415926, le texte "bonjour", etc.
- Donc une variable n'est pas uniquement définie par la valeur qu'elle contient, mais aussi par la place que cette valeur occupe et par la manière dont l'algorithme va la représenter et l'utiliser : nombre, texte, etc. C'est le type de la variable.
- vous devez préciser quel type la valeur représente. Est-ce un nombre ?
- Si oui, un entier (sans virgule) ou un réel (avec virgule) ? Est-ce du texte ? Est-ce un tableau ? Et ainsi de suite. selon le type de la valeur, celle-ci est encodée de manière différente dans la mémoire, utilisant plus de cases.

Les types : nombres

Type numérique	Plage de valeurs possibles
Byte (char)	0 à 255
Entier simple signé (int)	-32 768 à 32 767
Entier simple non signé	0 à 65535
Entier long signé (long)	-2 147 483 648 à 2 147 483 647
Entier long non signé	0 à 4294967295
Réel simple précision (float)	Négatif : $-3,40x10^{38}$ à $-1,40x10^{45}$ Positif : $1,40xl0^{-45}$ à $3,40x10^{38}$
Réel double précision (double)	Négatif : $-1,79x10^{308}$ à $-4,94x10^{-324}$ Positif : $4,94x10^{-324}$ à $1,79x10^{308}$

D'une manière générale, deux types numériques sont utilisés en algorithmique

- Les **entiers** : nombres sans virgule, négatifs ou positifs ;
- Les **réels** : nombres à virgule, positifs ou négatifs.

Les types : caractères

- Une variable peut aussi contenir des caractères. Si vous devez représenter un seul caractère, utilisez le type "caractère". Pour une chaîne, utilisez le type "chaîne de caractère".

```
VAR
texte : chaîne
car : caractère
```

- En principe, un caractère occupe un octet. À chaque valeur comprise entre 0 et 255 est associé un caractère. C'est le principe de l'ASCII (American Standard Code for Information Interchange), norme de codage des caractères la plus connue et la plus utilisée.

Les types : ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Les types : Chaine de caractère

Points délicats:

- caractère blanc : "-"
- caractère " (double quote):
  - ✓ La convention est simple : à chaque fois que l'on veut mettre un caractère " dans une chaîne de caractères, on le double, ce qui revient à le mettre deux fois, l'un derrière l'autre.

Exemples :

- "X""Y" est composée de trois caractères : X suivi de " suivi de Y.
- "X""Y""Z" est composée de cinq caractères : X suivi de " suivi de Y suivi de " suivi de Z.
- "X""""Z" est composée de quatre caractères : X suivi de " suivi de " suivi de Z.
- """" est composée d'un seul caractère, le caractère ".
- "" désigne la chaîne vide.
- "Y"Z" est mal écrite car le caractère " entre le Y et le Z n'est pas doublé

Les types : booléen

- Pour déterminer si une affirmation est vraie ou fausse, l'ordinateur doit se baser sur le résultat de cette affirmation. En informatique, on emploie plus volontiers la notion d'expression et d'évaluation de cette expression.
- "a>b" est une expression qui vaut soit vrai, soit faux.
- Un caractère, ou une chaîne de caractères, peut aussi être vraie ou fausse. On considère alors qu'un caractère vide ou qu'une chaîne vide "" sont faux, tandis que tout autre contenu est vrai.
- Pour représenter les valeurs vrai et faux, il suffit de deux chiffres, 0 et 1.
- Dans la pratique, ces langages proposent des constantes (des variables qui prennent une valeur une fois pour toute) spéciales pour représenter les valeurs vrai et faux :
  - **TRUE** pour vrai
  - **FALSE pour faux**

Affectation

- Pour donner une valeur à une variable, il faut passer par un processus d'affectation à l'aide d'un opérateur.
- En pseudo-code, on utilise le symbole d'affectation «← ».
- À gauche de ce symbole, vous placez le nom de la variable, à droite la valeur.
- Voici quelques exemples d'affectations en pseudo-code.

```
PROGRAMME AFFECTATION
VARIABLES
   a : entier
   b,c: réel
   titre : chaîne de caractère
   vrai : booléen
DEBUT
   a ← 10
   b ← 3,1415927
   c ← 12345
   titre ← "ma première affectation"
   vrai ← TRUE
FIN
```

Affectation

L'instruction d'affectation, notée ←

- Syntaxe 1 : nom d'une variable numérique ← expression arithmétique
  - ✓ Exemple 1 : TOTALSEC ← HH*3600 + MM*60 + SS
- Syntaxe 2 : nom d'une variable chaîne de caractères ← expression chaîne de caractères
  - ✓ Exemple 2 : MESSAGE ← "Le total est égal à "
- Sémantique : quelle que soit la syntaxe, l'exécution est la même.
  - ✓ Le processeur calcule d'abord le résultat de l'expression à droite des signes ←
  - ✓ Puis il met le résultat trouvé dans la variable indiquée à gauche des signes ←.

Exercice 1

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Programme Calcul
Variables
    A, B : Entiers
Début
    A ← 1
    B ← A + 3
    A ← 3
Fin
```

Exercice 2

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

Programme Calcul
Variables
A, B, C : Entiers
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B – A
Fin

Exercice 3

1. Quelles seront les valeurs des variables A et B après l'exécution des instructions suivantes ?

```
VARIABLES  A, B : Entier
     Début
   A ← 1
   B ← A + 3
   A ← 3
     Fin
```

Solution

Les valeurs des variables:	
A=1	B=?
A=1	B=4
A=3	**B=4**

2. Quelles seront les valeurs des variables A, B et C après l'exécution des instructions ?

```
VARIABLES  A, B , C: Entier
     Début
   A ← 5
   B ← 3
   C ← A + B
   A ← 2
   C ← B – A
     Fin
```

Solution

Les valeurs des variables:		
A=5	B=?	C=?
A=5	B=3	C=?
A=5	B=3	C=8
A=2	B=3	C=8
A=2	**B=3**	**C=1**

Exercice 4

3. Quelle est la valeur de C?

```
VARIABLES A, B, C : CHAINE
    Début
    A ← "423"
    B ← "12"
    C ← A & B
  Fin
```

Solution

Il ne peut produire qu'une erreur d'exécution, puisqu'on ne peut pas additionner des caractères

4

```
VARIABLES A, B, C : CHAINE
    Début
    A ← "423"
    B ← "12"
    C ← A + B
  Fin
```

Solution

On peut concaténer ces variables. A la fin de l'algorithme, C vaudra donc **"42312".**

Exercice 5

5. Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quels que soient leurs contenus préalables

Solution

```
VARIABLES A, B, C : Entier
Début
C ← A
A ← B
B ← C
Fin
```

6. On dispose de trois variables A, B et C. Ecrire un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

Solution

```
VARIABLES A, B, C : Entier
Début
D ← C
C ← B
B ← A
A ← D
Fin
```

Commentaire

En Algorithmique, un commentaire est une chaîne de caractères délimitée de part et d'autre, soit par :

- Les accolades ainsi,

  {Ceci est un commentaire}.

- Le signe %, plus facile et plus rapide à calligraphier, ainsi,

  % Ceci est un commentaire %

Affichage

Pour simuler l'affichage d'un texte ou d'une valeur sur l'écran, il faut utiliser la pseudo-instruction "**Ecrire**" qui prend à sa suite une chaîne de texte ou une variable.

Si vous mélangez du texte et des variables, séparez ceux-ci par des virgules. À l'affichage, les virgules seront remplacées par des espaces.

```
PROGRAMME AFFICHE
VARIABLES
    a : entier
    texte : chaîne
DEBUT
    a ←10
    texte ←"HelloWorld"
    Ecrire (a)
    Ecrire (texte)
    Ecrire ("Bonjour les amis", texte)
    FIN
```

Saisie

- Pour inviter un utilisateur à rentrer au clavier une valeur utilisez le mot **Lire**.
- L'algorithme attendra alors une entrée au clavier qui sera validée avec la touche d'entrée.
- La valeur que vous saisissez sera placée dans la variable indiquée à la suite de "Saisir".

```
PROGRAMME SAISIE
VARIABLE
   réponse : chaîne
DEBUT
    Ecrire ("Quel est votre nom ?")
    Lire (réponse)
    Ecrire ("Vous vous appelez",réponse)
FIN
```

La constante

- Vous pouvez décider de donner une valeur à une variable et de rendre celle-ci invariable : elle doit rester fixe dans le temps et inaltérable, pour toute la durée du programme.
- Sa valeur doit rester constante ; d'où son nom.
- Une constante est une valeur, représentée tout comme une variable par un nom, qui ne peut pas être modifiée après son initialisation. Elle est immuable.
- Un exemple de constante pourrait être la valeur de PI.
- Une constante se déclare généralement avant clé CONST.
- Elle est aussi d'un type donné.
-

```
PROGRAMME CONSTANTE
 CONSTANTE
PI (réel)=3.1415927
VARIABLES
  R : entier
  Aire : réel
   DEBUT
  Aire ← 2*PI*R
  Ecrire (Aire)
   FIN
```

Trace d'exécution d'un algorithme

- Faire la trace d'exécution d'un algorithme consiste à suivre pas à pas (instruction par instruction), sur une feuille de papier, le contenu des variables et les valeurs entrées et sorties. Faire la trace d'exécution d'un algorithme est un excellent moyen voire indispensable pour :
  - ✓ Comprendre la fonction opérée par l'algorithme.
  - ✓ Tester sur quelques exemples que cette fonction est programmée correctement.
  - ✓ Apprendre la syntaxe et assimiler la sémantique des instructions. Il est à noter qu'on ne peut faire la trace d'un algorithme que si sa syntaxe est correcte.
- La trace d'exécution est un exercice qui demande beaucoup de rigueur et d'attention. Savoir réussir une trace d'exécution sans fautes est un exercice d'algorithmique indispensable. Il est à la portée de tous mais ne s'improvise pas.

Trace d'exécution d'un algorithme

```
Programme HEURSEC
    Variables
        HH : numérique
        MM : numérique
        SS : numérique
        TOTALSEC : numérique
    Début
        Lire (HH)
        Lire (MM)
        Lire (SS)
        TOTALSEC ← HH*3600 + MM*60 + SS
        Ecrire (TOTALSEC)
    Fin
```

Faites la trace d'exécution de

l'algorithme

HEURSEC pour l'exécution :

6, 30, 15—> exécution de

HEURSEC — > ***Valeur en sortie ?***