MÉDÉE : Une méthode pour construire des algorithmes

Bernard Langer(*)

Nous allons exposer, étape par étape, le principe d'une méthode d'analyse appelée *méthode déductive*, *médée* en abrégé. Médée a été largement utilisé dans le cadre de l'enseignement de l'*Option Informatique* dans les années 80. La méthode est générale et peut être utilisée pour toutes les constructions d'algorithmes. Elle est issue des travaux de Claude Pair sur la *construction systématique des programmes* menés au *CRIN* de Nancy.

Méthode
Marche rationnelle de l'esprit pour arriver à la connaissance ou à la démonstration d'une vérité : La méthode se différencie de la théorie.
Ensemble ordonné de manière logique de principes, de règles, d'étapes, qui constitue un moyen pour parvenir à un résultat : Méthode scientifique.
(Petit Larousse)

Pour illustrer la construction d'*algorithmes* avec Médée, intéressons-nous à un problème particulièrement simple : le calcul de la somme $\frac{n_1}{d_1}+\frac{n_2}{d_2}$ de deux fractions dont il faudra afficher le résultat sous forme de fraction irréductible. Pour simplifier (encore) l'énoncé nous supposerons que n_1, d_1, n_2, d_2 sont des entiers naturels avec bien sûr d_1 et d_2 non nuls.

Le calcul demandé se résume à $\frac{n_1}{d_1}+\frac{n_2}{d_2}=\frac{n_1\times(d\div d_1)+n_2\times(d\div d_2)}{d}$, d étant le *PPCM* de d_1 et d_2.

La fraction obtenue n'étant pas nécessairement irréductible, il faudra simplifier le résultat en divisant numérateur et dénominateur par leur *PGCD*. Par exemple : $\frac{7}{12}+\frac{13}{15}=\frac{35+52}{60}=\frac{87}{60}=\frac{29}{20}$.

L'analyse se déroule en plusieurs étapes et consiste à remplir un unique tableau comportant trois colonnes :

- La première est nommée *lexique*. On y précise le *type* (entier, décimal, caractère, etc.) et la signification des variables utilisées.
- La troisième est intitulée *définitions*. Elle contient les diverses *instructions* de l'algorithme. Pour des raisons qui apparaîtront clairement dans ce qui suit, ces instructions sont énoncées dans le désordre.
- Grâce à une numérotation, la deuxième colonne précisera l'ordre dans lequel les instructions doivent être exécutées.

(*) bernard.langer@laposte.net

Signalons que les instructions utilisées sont liées aux possibilités du *processeur* (humain ou machine) qui les exécutera. On pourrait par exemple imaginer que ce processeur soit en mesure de traiter directement l'addition de fractions en donnant un résultat simplifié. Dans ce cas l'algorithme cherché se réduit trivialement à une seule instruction !
Nous supposerons donc que les fonctions *PGCD* et *PPCM* ne sont pas *primitives*.

Les étapes de la construction de l'algorithme :

Étape 1 :

On commence par énoncer clairement le ***résultat*** à obtenir. Dans notre exemple, le résultat à obtenir consiste à écrire deux entiers : le numérateur de la somme et le dénominateur de la somme. Les noms de ces deux *variables* NS et DS sont recopiés dans le lexique en précisant leur type (ici deux éléments de $\mathbb{N}$) ainsi que leur signification (le mot variable est à prendre au sens de *conteneur* permettant de stocker un élément qui aura été calculé ou saisi manuellement).

Lexique			Définitions
	NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire* *NS*, *DS*
	DS : entier	Dénominateur de la somme simplifiée	

Étape 2 :

À cette étape ainsi qu'à toutes les étapes suivantes, on puise un élément du lexique (ici NS) et on donne sa définition dans la colonne éponyme. Toutes les variables utilisées qui ne figurent pas encore au lexique y sont recopiées (ici N et P).

Lexique			Définitions
✓	NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire* *NS*, *DS* *NS* = *N*÷*P*
	DS : entier	Dénominateur de la somme simplifiée	
	N : entier	Numérateur de la somme	
	P : entier	PGCD(N,D)	

Puisque NS est définie, nous *cochons* (✓) cette variable dans la colonne du lexique.

Étape 3 :

On poursuit la démarche entamée au cours des étapes suivantes…

Lexique			Définitions
✓	NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire* *NS*, *DS* *NS* = *N*÷*P* *DS* = *D*÷*P*
✓	DS : entier	Dénominateur de la somme simplifiée	
	N : entier	Numérateur de la somme	
	P : entier	PGCD(N,D)	
	D : entier	Dénominateur commun de fraction 1 et fraction 2	

Étape 4 :

Lexique		Définitions
✓ NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire NS, DS* *NS = N÷P* *DS = D÷P* *N = N1*D÷D1 + N2*D÷D2*
✓ DS : entier	Dénominateur de la somme simplifiée	
✓ N : entier	Numérateur de la somme	
P : entier	PGCD(N,D)	
D : entier	Dénominateur commun de fraction 1 et fraction 2	
N1 : entier	Numérateur de la fraction 1	
D1 : entier	Dénominateur de la fraction 1	
N2 : entier	Numérateur de la fraction 2	
D2 : entier	Dénominateur de la fraction 2	

Étape 5 :

Lexique		Définitions
✓ NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire NS, DS* *NS = N÷P* *DS = D÷P* *N = N1*D÷D1 + N2*D÷D2* *P = PGCD(N,D)*
✓ DS : entier	Dénominateur de la somme simplifiée	
✓ N : entier	Numérateur de la somme	
✓ P : entier	Plus Grand Commun Diviseur de N et D	
D : entier	Dénominateur commun de fraction 1 et fraction 2	
N1 : entier	Numérateur de la fraction 1	
D1 : entier	Dénominateur de la fraction 1	
N2 : entier	Numérateur de la fraction 2	
D2 : entier	Dénominateur de la fraction 2	
PGCD : fonction	Retourne le PGCD de deux entiers.	

Étape 6 :

Lexique		Définitions
✓ NS : entier	Numérateur de la somme simplifiée	*résultat* = *écrire NS, DS* *NS = N÷P* *DS = D÷P* *N = N1*D÷D1 + N2*D÷D2* *P = PGCD(N,D)* *D = PPCM(D1,D2)*
✓ DS : entier	Dénominateur de la somme simplifiée	
✓ N : entier	Numérateur de la somme	
✓ P : entier	Plus Grand Commun Diviseur de N et D	
✓ D : entier	Plus Petit Commun Multiple de D1 et D2	
N1 : entier	Numérateur de la fraction 1	
D1 : entier	Dénominateur de la fraction 1	
N2 : entier	Numérateur de la fraction 2	
D2 : entier	Dénominateur de la fraction 2	
PGCD : fonction	Retourne le PGCD de deux entiers.	
PPCM : fonction	Retourne le PPCM de deux entiers.	

Étape 7 :

Lexique		Définitions
✓ NS : entier	Numérateur de la somme simplifiée	*résultat* = écrire NS, DS
✓ DS : entier	Dénominateur de la somme simplifiée	*NS* = *N*÷*P*
✓ N : entier	Numérateur de la somme	*DS* = *D*÷*P*
✓ P : entier	Plus Grand Commun Diviseur de N et D	*N* = *N1***D*÷*D1* + *N2***D*÷*D2*
✓ D : entier	Plus Petit Commun Multiple de D1 et D2	*P* = *PGCD*(*N*,*D*)
✓ N1 : entier	Numérateur de la fraction 1	*D* = *PPCM*(*D1*,*D2*)
D1 : entier	Dénominateur de la fraction 1	*N1* = *donnée*
N2 : entier	Numérateur de la fraction 2	
D2 : entier	Dénominateur de la fraction 2	
PGCD : fonction	Retourne le PGCD de deux entiers.	
PPCM : fonction	Retourne le PPCM de deux entiers.	

Remarque : une *donnée* est une valeur non calculable qui est en général saisie au clavier…

Étapes 8 – 9 – 10 :

Lexique		Définitions
✓ NS : entier	Numérateur de la somme simplifiée	*résultat* = écrire NS, DS
✓ DS : entier	Dénominateur de la somme simplifiée	*NS* = *N*÷*P*
✓ N : entier	Numérateur de la somme	*DS* = *D*÷*P*
✓ P : entier	Plus Grand Commun Diviseur de N et D	*N* = *N1***D*÷*D1* + *N2***D*÷*D2*
✓ D : entier	Plus Petit Commun Multiple de D1 et D2	*P* = *PGCD*(*N*,*D*)
✓ N1 : entier	Numérateur de la fraction 1	*D* = *PPCM*(*D1*,*D2*)
✓ D1 : entier	Dénominateur de la fraction 1	*N1* = *donnée*
✓ N2 : entier	Numérateur de la fraction 2	*D1* = *donnée*
✓ D2 : entier	Dénominateur de la fraction 2	*N2* = *donnée*
PGCD : fonction	Retourne le PGCD de deux entiers.	*D2* = *donnée*
PPCM : fonction	Retourne le PPCM de deux entiers.	

Étape 11 : la numérotation

D'une manière évidente, les diverses instructions ne peuvent pas être exécutées dans l'ordre où elles ont été écrites.
La numérotation des instructions se fait en recherchant dans la colonne des définitions la première instruction dont tous les paramètres nécessaires à son exécution sont connus.
Ainsi *P* ne peut être calculé que lorsque *N* et *D* l'ont été. Mais *N* ne sera calculable qu'après avoir déterminé *D*, etc.
La numérotation des instructions n'est en général pas unique et il est d'usage de démarrer par la saisie des données.

Lexique				Définitions
			10	*résultat* = *écrire NS, DS*
✓	NS : entier	Numérateur de la somme simplifiée	9	*NS* = *N*÷*P*
✓	DS : entier	Dénominateur de la somme simplifiée	8	*DS* = *D*÷*P*
✓	N : entier	Numérateur de la somme	6	*N* = *N1***D*÷*D1* + *N2***D*÷*D2*
✓	P : entier	Plus Grand Commun Diviseur de N et D	7	*P* = *PGCD*(*N*,*D*)
✓	D : entier	Plus Petit Commun Multiple de D1 et D2	5	*D* = *PPCM*(*D1*,*D2*)
✓	N1 : entier	Numérateur de la fraction 1	1	*N1* = *donnée*
✓	D1 : entier	Dénominateur de la fraction 1	2	*D1* = *donnée*
✓	N2 : entier	Numérateur de la fraction 2	3	*N2* = *donnée*
✓	D2 : entier	Dénominateur de la fraction 2	4	*D2* = *donnée*
	PGCD : fonction	Retourne le PGCD de deux entiers.		
	PPCM : fonction	Retourne le PPCM de deux entiers.		

Étape 12 : les fonctions

Il reste à « fabriquer » les fonctions *PPCM* et *PGCD*.
Puisque $PPCM(a,b) \times PGCD(a,b) = a \times b$, il suffira de connaître la valeur prise par l'une des fonctions pour en déduire immédiatement la valeur prise par l'autre. Notre choix se portera sur le calcul du *PPCM*. Il s'agit d'un choix arbitraire qui n'est sans doute pas le meilleur mais il évite d'écrire une n$^{\text{ème}}$ fois l'algorithme d'Euclide ! L'idée retenue est la suivante :

Parmi les multiples de a, on cherche le premier qui est aussi multiple de b.

Lexique				Fonction *PPCM*(*a*,*b*)
			3	*résultat* = m_f
✓	a : entier	Premier argument de la fonction	2	*répéter tant que* *m* *mod* *b* ≠ 0
✓	b : entier	Deuxième argument de la fonction		$m = @m + a$
✓	m : entier : liste	élément générique de la liste des multiples de a	1	$m_i = a$

Quelques explications concernant les listes (suites finies) sont nécessaires :
L'algorithme du *PPCM* ci-dessus construit la liste des multiples de a inférieurs ou égaux à b. Cette liste est construite grâce à une *itération* qui est essentiellement une « fabrique de listes ». De fait une liste engendrée par itération apparaît toujours sous quatre occurrences distinctes dans l'analyse :

- La *valeur initiale* m_i (lire « *m initial* »).
- L'*ancienne valeur* de m notée $@m$.
- La *valeur courante* m.
- La *valeur finale* m_f (lire « *m final* »).

Cette approche a le mérite de clarifier ce qui ne l'est plus dans un *programme* où m, m_i, $@m$, m_f se fondent en une désignation unique. L'écriture $m = @m + a$ a le mérite de clarifier le processus d'affectation.

Étape 13 :

Lexique		Fonction *PGCD*(*a*,*b*)
✓ a : entier — Premier argument de la fonction ✓ b : entier — Deuxième argument de la fonction	1	*résultat* = *a**b÷*PPCM*(*a*,*b*)

À ce stade l'algorithme est complet. Nous le reproduisons sous sa forme définitive ci-dessous :

Lexique		Définitions
✓ NS : entier — Numérateur de la somme simplifiée ✓ DS : entier — Dénominateur de la somme simplifiée ✓ N : entier — Numérateur de la somme ✓ P : entier — Plus Grand Commun Diviseur de N et D ✓ D : entier — Plus Petit Commun Multiple de D1 et D2 ✓ N1 : entier — Numérateur de la fraction 1 ✓ D1 : entier — Dénominateur de la fraction 1 ✓ N2 : entier — Numérateur de la fraction 2 ✓ D2 : entier — Dénominateur de la fraction 2 ✓ *PGCD* : fonction — Retourne le PGCD de deux entiers. ✓ *PPCM* : fonction — Retourne le PPCM de deux entiers.	10 9 8 6 7 5 1 2 3 4	*résultat* = *écrire NS, DS* *NS* = *N*÷*P* *DS* = *D*÷*P* *N* = *N1***D*÷*D1* + *N2***D*÷*D2* *P* = *PGCD*(*N*,*D*) *D* = *PPCM*(*D1*,*D2*) *N1* = *donnée* *D1* = *donnée* *N2* = *donnée* *D2* = *donnée*
		Fonction *PPCM*(a,b)
✓ a : entier — Premier argument de la fonction ✓ b : entier — Deuxième argument de la fonction ✓ m : entier — élément générique de la liste des multiples de a	3 2 1	*résultat* = m_f *répéter tant que m mod b* ≠ 0 *m* = @*m* + *a* $m_i = a$
		Fonction *PGCD*(a,b)
✓ a : entier — Premier argument de la fonction ✓ b : entier — Deuxième argument de la fonction	1	*résultat* = *a***b*÷*PPCM*(*a*,*b*)

L'algorithme obtenu grâce à cette démarche n'a pas la prétention d'être optimisé et il arrive parfois qu'une mise en œuvre sur un exemple permette de découvrir des améliorations simples à mettre en place.

Examinons par exemple ce qui se passe dans le calcul du *PPCM*.

Déterminons le *PPCM* de 35 et de 28, i.e. *PPCM*(35,28).

Instructions	*a*	*b*	*m*	*m mod b*
	35	28		
$m_i = a$			35	35 *mod* 28 =7
m = @*m* + *a*			70	70 *mod* 28 =14
m = @*m* + *a*			105	105 *mod* 28 =21
m = @*m* + *a*			140	140 *mod* 28 =0

Le *PPCM* de 35 et 28 est $m_f = 140$.

Nous aurions également pu calculer *PPCM*(28,35). Les itérations ne sont plus les mêmes pour un résultat bien sûr identique :

Instructions	***a***	***b***	***m***	***m mod b***
	28	35		
$m_i = a$			28	28 *mod* 35 =28
$m = @m + a$			56	56 *mod* 35 =21
$m = @m + a$			84	84 *mod* 35 =14
$m = @m + a$			112	112 *mod* 35 =7
$m = @m + a$			140	140 *mod* 35 =0

Le *PPCM* de 28 et 35 est $m_f = 140$.

Sur cet exemple nous constatons que si l'on choisit comme premier argument le plus grand des deux entiers les itérations sont moins nombreuses. Il est donc judicieux d'ordonner au préalable les deux entiers. D'où une variante plus performante, bien que plus longue, du même algorithme :

Lexique				**Fonction *PPCM*(a,b)**
✱	a : entier	Premier argument de la fonction	4	*résultat* = m_f
✱	b : entier	Deuxième argument de la fonction	3	*répéter tant que* m *mod* $b \neq 0$
✱	m : entier	élément générique de la liste des multiples de a		$m = @m + a$
✱	aux : entier	Variable auxiliaire servant à échanger a et b	2	$m_i = a$
			1	*si* a<b *alors*
				$aux = a$
				$a = b$
				$b = aux$

Une fois l'algorithme obtenu il reste à l'exécuter ou *le faire exécuter*. L'exécuter à la main consiste en général à remplir des tableaux tels ceux présentés ci-dessus pour le calcul du *PPCM*. Néanmoins cette tâche devient vite fastidieuse, voire impossible à réaliser du fait de sa complexité et du temps nécessaire !
C'est là où l'ordinateur intervient. Mais, avant de pouvoir tester l'algorithme sur l'ordinateur, il faut le coder dans un *langage de programmation*. Cette tâche de codage est une tâche mécanique, peu gratifiante et qui doit absolument être précédée par une phase d'analyse telle que celle présentée ci-dessus. Ne pas se livrer à cette phase d'analyse conduit à de grandes pertes de temps et à des programmes *spaghettis* dont on ne maîtrise pas les méandres et qui sont impossibles à maintenir.

Nous donnons en annexe les codages en ALGOBOX, PYTHON et SCRATCH de l'algorithme précédent. Les remarques critiques qui précédent chaque codage n'engagent que leur auteur.

ANNEXE 1 : CODAGE ALGOBOX

Algobox est un excellent outil d'initiation qui ne permet cependant pas de développer des algorithmes après la classe de seconde.
La syntaxe très lourde le rend fastidieux à utiliser au-delà la phase d'initiation.
Écrire « m prend la valeur de $m + D1$ » ne résout pas la difficulté de l'instruction d'affectation.
Il n'est pas possible de coder des fonctions, ce qui oblige à écrire deux fois les instructions du calcul du *PPCM* (lignes 17-21 puis 24-28)

Codage

```
1     VARIABLES
2       NS EST_DU_TYPE NOMBRE
3       DS EST_DU_TYPE NOMBRE
4       N EST_DU_TYPE NOMBRE
5       P EST_DU_TYPE NOMBRE
6       D EST_DU_TYPE NOMBRE
7       N1 EST_DU_TYPE NOMBRE
8       D1 EST_DU_TYPE NOMBRE
9       N2 EST_DU_TYPE NOMBRE
10      D2 EST_DU_TYPE NOMBRE
11      m EST_DU_TYPE NOMBRE
12    DEBUT_ALGORITHME
13      LIRE N1
14      LIRE D1
15      LIRE N2
16      LIRE D2
17      m PREND_LA_VALEUR D1
18      TANT_QUE (m% D2 != 0) FAIRE
19        DEBUT_TANT_QUE
20        m PREND_LA_VALEUR m+D1
21        FIN_TANT_QUE
22      D PREND_LA_VALEUR m
23      N PREND_LA_VALEUR N1*D/D1+N2*D/D2
24      m PREND_LA_VALEUR N
25      TANT_QUE (m%D!=0) FAIRE
26        DEBUT_TANT_QUE
27        m PREND_LA_VALEUR m+N
28        FIN_TANT_QUE
29      P PREND_LA_VALEUR N*D/m
30      NS PREND_LA_VALEUR N/P
31      DS PREND_LA_VALEUR D/P
32      AFFICHER "Résultat : "
33      AFFICHER N1
34      AFFICHER "/"
35      AFFICHER D1
```

ANNEXE 2 : CODAGE PYTHON

Python est un vrai langage de programmation largement utilisé par la communauté scientifique.
Cependant les diverses versions ne sont pas francisées et interdisent en général l'affichage d'un texte comportant des caractères accentués. Cette situation nous fait malheureusement revenir une trentaine d'années en arrière.
Le typage dynamique des variables (le type d'une variable est défini par sa première affectation) n'est pas très pédagogique et peut être source de nombreuses confusions pour le débutant.
En revanche Python permet une programmation professionnelle (y compris la programmation de type *objet*) et offre une bibliothèque gigantesque, en particulier concernant les applications mathématiques.

Codage

```
# ***************
# Fonction PPCM *
# ***************
def ppcm(a,b) :
    if a<b:
        aux=a
        a=b
        b=aux
    m=a
    while m%b!=0:
        m=m+a
    return m
# ***************
# Fonction PGCD *
# ***************
def pgcd(a,b):
    return a*b/ppcm(a,b)
# ********************
# Début du programme *
# ********************
n1=input('Numerateur de la fraction 1 ?')
d1=input('Denominateur de la fraction 1 ?')
n2=input('Numerateur de la fraction 2 ?')
d2=input('Denominateur de la fraction 2 ?')
d=ppcm(d1,d2)
n=n1*d/d1+n2*d/d2
p=pgcd(n,d)
ns=n/p
ds=d/p
print n1,"/",d1,"+",n2,"/",d2,"=",ns,"/",ds
```

ANNEXE 3 : CODAGE SCRATCH

Un langage surprenant et novateur.
Il est cependant très déroutant lors d'une première utilisation.
Les expressions algébriques sont très lourdes à construire et il ne semble pas adapté au codage d'algorithmes tels celui présenté dans cet article.
La boucle universelle *répéter tant que* est absente.
On ne peut pas construire de fonctions.