L'Altruiste : Le guide des langages Web

Annexe

Sommaire

1 / La logique

1.1 / Les tables de vérité

Opérateur AND (ET)		
Bit	**Bit**	**Résultat**
0	0	**0**
1	0	**0**
0	1	**0**
1	1	**1**

Opérateur OR (OU)		
Bit	**Bit**	**Résultat**
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**1**

Opérateur XOR (OU EXCLUSIF)		
Bit	**Bit**	**Résultat**
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**0**

Opérateur NOT (NON)	
Bit	**Résultat**
00	**11**
01	**10**
10	**01**
11	**00**

1.2 / Correspondance binaires/décimales

Nombre positif		Nombre négatif	
Code binaire	Nombre décimal	Code binaire	Nombre décimal
0000	0	11111	-1
0001	1	11110	-2
0010	2	11101	-3
0011	3	11100	-4
0100	4	11011	-5
0101	5	11010	-6
0110	6	11001	-7
0111	7	11000	-8
1000	8	10111	-9
1001	9	10110	-10
1010	10	10101	-11
1011	11	10100	-12
1100	12	10011	-13
1101	13	10010	-14
1110	14	10001	-15
1111	15	10000	-16

1.3 / Méthode de conversion binaires/décimaux

1111	$1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 8 + 4 + 2 + 1 =$ **15**
1010	$1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 8 + 0 + 2 + 0 =$ **10**
1010101010	$1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$ $= 128 + 0 + 32 + 0 + 8 + 0 + 2 + 0$ = **170**

1.4 / Méthode de conversion décimaux/binaires

Pour trouver un nombre binaire à partir d'un nombre décimal, il faut successivement diviser par *deux* le nombre décimal puis les quotients résultants jusqu'à obtenir un *zéro* ou un *un*.

Puis il suffit de lire le nombre binaire en prenant en compte le dernier quotient (1 ou 0) de la division puis effectuer une lecture du bas en haut pour les restes.

10 : 2 = 5	reste **0**	**1010**
5 : 2 = 2	reste **1**	
2 : 2 = **1**	reste **0**	
15 : 2 = 7	reste **1**	**1111**
7 : 2 = 3	reste **1**	
3 : 2 = **1**	reste **1**	
14 : 2 = 7	reste **0**	**1110**
7 : 2 = 3	reste **1**	
3 : 2 = **1**	reste **1**	
147 : 2 = 73	reste **1**	**10010011**
73 : 2 = 36	reste **1**	
36 : 2 = 18	reste **0**	
18 : 2 = 9	reste **0**	
9 : 2 = 4	reste **1**	
4 : 2 = 2	reste **0**	
2 : 2 = **1**	reste **0**	

1.5 / Lois

Parfois, il peut être très utile de simplifier des expressions booléennes de sorte à devenir plus compréhensibles et plus maniables dans un programme.

Lois de distributivités

Exp1 OR (Exp2 AND Exp3) = (Exp1 OR Exp2) AND (Exp1 OR Exp3)
Exp1 AND (Exp2 OR Exp3) = (Exp1 AND Exp2) OR (Exp1 AND Exp3)

Lois de De Morgan

non(Exp1 AND Exp2) = NOT Exp1 OR NOT Exp2
non(Exp1 OR Exp2) = NOT Exp1 AND NOT Exp2

Loi de négation

non(non Exp1) = Exp1

Loi du milieu exclu

Exp1 OR NOT Exp1 = vrai

Loi de contradiction

Exp1 AND NOT Exp1 = faux

Lois de simplfication

Exp1 AND Exp1 = Exp1
Exp1 AND vrai = Exp1
Exp1 AND faux = faux
Exp1 AND (Exp1 OR Exp2) = Exp1

Exp1 OR Exp1 = Exp1
Exp1 OR vrai = vrai
Exp1 OR faux = Exp1
Exp1 OR (Exp1 AND Exp2) = Exp1

2 / Les nombres

2.1 / Représentation binaire des nombres décimaux

Les nombres décimaux possèdent une représentation binaire standardisée pour les valeurs de précision simple (32 bits) et double (64 bits) par ANSI/IEEE Standard 754-1985.

Les nombres décimaux simples

La représentation binaire des nombres à virgules flottantes de précision simple requiert un mot de 32 bits, lequel est composé d'un bit de signe *S*, de 8 bits pour l'exposant *E* et de 23 bits pour la fraction décimale.

```
S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
0 1EEEEEEE8 9FFFFFFFFFFFFFFFFFFFFFFF31
```

La valeur exprimée par cette représentation binaire est déterminée selon certaines conditions applicables aux parties *E*, *F* et *S*.

Exposant	Fraction et signe	Valeur
E=255	F != nonzero	NaN
E=255	F = 0 ET S = 1	-Infinity
E=255	F = 0 ET S = 0	Infinity
0 < E < 255		(-1)**S * 2 ** (E-127) * (1.F)
E = 0	F != 0	(-1)**S * 2 ** (-126) * (0.F)
E = 0	F = 0	-0
E = 0	F = 0	0

Les nombres décimaux doubles

La représentation binaire des nombres à virgules flottantes de précision simple requiert un mot de 64 bits, lequel est composé d'un bit de signe *S*, de 11 bits pour l'exposant *E* et de 52 bits pour la fraction décimale.

```
S EEEEEEEEEEE
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0 1EEEEEEEEEE11
  12FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF64
```

La valeur exprimée par cette représentation binaire est déterminée selon certaines conditions applicables aux parties *E*, *F* et *S*

Exposant	Fraction et signe	Valeur
E = 2047	F != 0	NaN
E = 2047	F = 0 ET S = 1	-Infinity
E = 2047	F = 0 ET S = 0	Infinity
0 < E < 2047		(-1)**S * 2 ** (E-1012) * (1.F)
E = 0	F != 0	(-1)**S * 2 ** (-1022) * (0.F)
E = 0	F = 0	-0
E = 0	F = 0	0

Exemples

0 00000000 00000000000000000000000 = 0
1 00000000 00000000000000000000000 = -0
0 11111111 00000000000000000000000 = Infinity
1 11111111 00000000000000000000000 = -Infinity
0 11111111 00000100000000000000000 = NaN
1 11111111 00100010001001010101010 = NaN
0 10000000 00000000000000000000000
= +1 * 2(128-127) * 1.0 = 2**
0 10000001 10100000000000000000000
= +1 * 2(129-127) * 1.101 = 6.5**
1 10000001 10100000000000000000000
= -1 * 2(129-127) * 1.101 = -6.5**
0 00000001 00000000000000000000000
= +1 * 2(1-127) * 1.0 = 2**(-126)**
0 00000000 10000000000000000000000
= +1 * 2(-126) * 0.1 = 2**(-127)**
0 00000000 00000000000000000000001
= +1 * 2(-126) * 0.00000000000000000000001 = 2**(-149)**

2.2 / Représentation binaire des nombres négatifs

Les nombres négatifs possèdent une représentation binaire spécifique afin de les distinguer des nombres positifs.

En fait, **les nombres binaires négatifs sont appelés *nombres signés*** car ils possèdent une longueur fixe et à leur extrême gauche un bit égal à 1.

```
1000 1001 // est égal à -9
0000 1001 // est égal à +9
```

En pratique, **cette représentation n'est guère utilisable** puisque l'ordinateur n'est pas capable de gérer correctement les opérations arithmétiques dans ces conditions.

```
  0000 1110 // est égal à 14
+ 1000 1001 // est égal à -9
  1001 0111 // est égal à -23

  0000 0101 // devrait être égal à 5
```

Devant ce dysfonctionnement patent, il est nécessaire de **représenter différemment les nombres binaires négatifs**.

```
0000 1001 // est égal à 9

1111 1111 // complément à 1
0000 1001
1111 0110
0000 0001 // ajout de 1
1111 0111 // est égal à -9

1111 0111
0000 1001 // ajout de 9
0000 0000 //est bien égal à 0
```

En premier lieu, **le nombre est complémenté à 1 puis la valeur binaire 1 est ajoutée** pour donner finalement une représentation binaire exploitable du nombre négatif correspondant.

```
  0000 1110 // est égal à 14
+ 1111 0111 // est égal à -9
  0000 0101 // est égal à 5
```

Représentation des nombres binaires négatifs

Nombre négatif	
Code binaire	**Nombre décimal**
1111 1111	-1
1111 1110	-2
1111 1101	-3
1111 1100	-4
1111 1011	-5
1111 1010	-6
1111 1001	-7
1111 1000	-8
1111 0111	-9
1111 0110	-10
1111 0101	-11
1111 0100	-12
1111 0011	-13
1111 0010	-14
1111 0001	-15
1111 0000	-16

3 / Les types d'encodage

Le tableau ci-dessous fournit la liste des différents types d'encodage et leur nom utilisable sur Internet et leur disponibilité sur Mac OS.

Les langages d'encodage
Universels
Europe de l'Ouest
Europe centrale
Arabes
Chinois
Cyrilliques
Grecs
Hébraïques
Indiens
Japonais
Coréens
Thaïs
Turcs
Vietnamiens
symboliques

Les langages		
Type d'encodage	**Nom commun sur Internet**	**Information**
▲ Les langages universels		
Unicode 2.0 (16 bit)	*UTF-16*	(voir Unicode.org)
Unicode 2.0 UTF-8	*UTF-8*	(voir Unicode.org)
Unicode 2.0 UTF-7	*UTF-7*	(voir Unicode.org)
Unicode 1.1 (16-bit)	*UNICODE 1-1*	(voir Unicode.org)
Unicode 1.1 UTF-8	*UNICODE-1-1-UTF-8*	(voir Unicode.org)
Unicode 1.1 UTF-7	*UNICODE-1-1-UTF-7*	(voir Unicode.org)
▲ Les langages d'Europe de l'Ouest		
ASCII	*US-ASCII*	
ISO 8859-1 (Latin-1)	*ISO-8859-1*, *latin1*	
ISO 8859-3 (Latin 3)	*ISO-8859-3* , *latin3*	
ISO-8859-15 (Latin 9)	*ISO-8859-15*, *latin9*	*Latin-1* avec le signe EURO et les lettres cp1252
CP 1252 (Windows Latin-1)	***windows-1252***, *cp1252*	*ISO 8859-1*, plus des ajouts dans la zone *C1*
CP 437 (DOS Latin-US)	*cp437*	
CP 850 (DOS Latin-1)	*cp850*	
Mac OS Roman	*mac*, *macintosh*, *x-mac-roman*	
Mac OS Icelandic	*x-mac-icelandic*	Fondé sur *Mac OS Roman*
Mac OS Latin-1, Mac OS Mail	*x-mac-latin1* (communément assimilé à *ISO-8859-1*)	*Mac OS Roman* permuté pour s'aligner à *8859-1*
NextStep Latin		
CP 037 (EBCDIC-US)	*cp037*	*ISO 8859-1* avec différentes dispositions
▲ Les langages d'Europe centrale		
ISO 8859-2 (Latin-2)	*ISO-8859-2*, *latin2*	
ISO 8859-4 (Latin-4)	*ISO-8859-4*, *latin4*	
CP 1250 (Windows Latin-2)	***windows-1250***, *cp1250*	Partiellement *8859-2*, plus des ajouts de *C1*
CP 1257 (Windows Baltique)	***windows-1257***,*cp1257*	
Mac OS Central Roman européen	*x-mac-centraleurroman*	
Mac OS Croatian	x-mac-croatian	Fondé sur *Mac OS Roman*
Mac OS Romanian	*x-mac-romanian*	Fondé sur *Mac OS Roman*
▲ Les langages Arabes		
ISO 8859-6 (Latin/Arabe)	*ISO-8859-6*, arabe	

CP 1256 (Windows Arabe)	***windows-1256***, *cp1256*	Partiellement basé sur *8859-6*, plus des ajouts de *C1*
CP 864 (DOS Arabic)	*cp864*	Encode les formes de présentation Arabe
Mac OS Arabic	*x-mac-arabic*	
Mac OS Farsi	*x-mac-farsi*	
Les langages Chinois		
GB 2312-80		
EUC-CN	*GB2312*, *X-EUC-CN*	*ASCII* et *GB 2312-80* (8-bit)
CP 936 (DOS et Windows simplifiés)		Similaire à *GBK*
Mac OS Chinois simplifié		Fondé sur *EUC-CN*
ISO 2022-CN ("GB")	*ISO-2022-CN*	*ASCII* et *GB 2312-80* (7-bit) (voir RFC 1922)
HZ	*HZ-GB-2312*	*ASCII* et *GB 2312-80* (7-bit) (voir RFC 1842);
GBK (extended GB)		*EUC-CN* et *Unihan* (8-bit)
CNS 11643 plane 1	*x-cns11643-1*	
CNS 11643 plane 2	*x-cns11643-2*	
EUC-TW	*X-EUC-TW*	*ASCII* et *CNS 11643-1992* (8-bit)
Big-5	*Big5*	(8-bit)
CP 950 (DOS et Windows traditionnel)		Fondé sur *Big-5*
Mac OS (Chinois traditionnel)		Fondé sur *Big-5*
CCCII		
EACC		
Les langages cyrilliques		
ISO 8859-5 (Latin/Cyrillique)	*ISO-8859-5*, cyrillique	
KOI8-R	*KOI8-R*	Voir RFC 1489
CP 1251 (Windows Cyrillique)	***windows-1251***, *cp1251*	non fondé sur *ISO 8859-5*
CP 866 (DOS Russe)	*cp866*	
Mac OS Cyrillic	*x-mac-cyrillic*	
Mac OS Ukrainian	*x-mac-ukrainian*	*Mac OS Cyrillic* avec deux remplacements
Les langages Grecs		
ISO 8859-7	*ISO-8859-7*, grec	
ISO 5428	*ISO_5428:1980*	

CP 1253 (Windows Grec)	***windows-1253***, *cp1253*	Quasiment fondé sur *8859-7*, plus des ajouts de *C1*
Mac OS Greek	*x-mac-greek*	
Greek CCITT	*greek-ccitt*	
▲ Les langages Hébraïques		
ISO 8859-8 (Latin/Hébreux)	*ISO-8859-8*, hébreux	
CP 1255 (Windows Hébreux)	***windows-1255***, *cp1255*	Principalement basé sur *8859-8*, plus des ajouts *C1*
Mac OS Hebrew (2 variantes)	*x-mac-hebrew*	
▲ Les langages Indiens		
ISCII-91		Encodages parallèles pour tous les scripts indiens
Mac OS Gujarati		
Mac OS Devanagari		
Mac OS Gurmukhi		
▲ Les langages Japonais		
JIS X0208		
JIS X0212		
EUC-JP	*EUC-JP*, *X-EUC-JP*	*JIS 201*, *JIS 208* et *JIS 212* (8-bit)
ISO 2022-JP ("JIS")	*ISO-2022-JP*	*JIS 201*, *JIS 208* et *JIS 212* (7-bit); RFC 1468
Shift-JIS	*Shift_JIS*, *x-sjis*, *x-shift-jis*	*JIS 201* et *JIS 208* (8-bit)
CP 932 (DOS et Windows)		Fondé sur *Shift-JIS*
Mac OS Japanese		Fondé sur *Shift-JIS*
▲ Les langages coréens		
KSC 5601-1987		
EUC-KR	*EUC-KR*	*ASCII* et *KSC 5601-87* (8-bit); RFC 1557
CP 949 (DOS et Windows)		Code *Hangul* unifié : *EUC-KR* et *Johab*
Mac OS Korean		Fondé sur *EUC-KR*
ISO 2022-KR ("KSC")	*ISO-2022-KR*	*ASCII* et *KSC 5601-87* (7-bit): RFC 1557
KSC 5700		
▲ Les langages Thaïs		
TIS 620-2533		
CP 874 (DOS et Windows)	*cp874*	Fondé sur *TIS 620-2533*

Mac OS Thai	*x-mac-thai*	Fondé sur TIS 620-2533
Les langages Turcs		
ISO 8859-9 (Latin-5)	*ISO-8859*, *latin5*	
ISO 8859-3 (Latin-3)	*ISO-8859-3*	
CP 1254 (Windows Latin-5)	***windows-1254***, *cp1254*	
Mac OS Turkish	*x-mac-turkish*	Fondé sur Mac OS Roman
Les langages Vietnamiens		
VISCII	*VISCII*	RFC 1456
TCVN-n		
CP 1258 (Windows Vietnamien)	***windows-1258***, *cp1258*	
Les langages symboliques		
Adobe Symbol	*Adobe-Symbol-Encoding*	
Mac OS Symbol	*x-mac-symbol*	Fondé sur *Adobe Symbol*
Mac OS dingbats	*x-mac-dingbats*	Fondé sur *Adobe Zapf Dingbats*

4 / Le code ISO-8859-1

Caractères	Codes numériques	Références d'entités	Descriptions
		** **	Espace
!	!	**!**	Point d'exclamation
"	"	**"**	Marque de citation
#	#	**#**	Dièse
$	$	**$**	Dollar
%	%	**%**	Pour-cent
&	&	**&**	Et commercial (esperluette)
`	'	**'**	Apostrophe
(	(	**(**	Parenthèse de Gauche
)	)	**)**	Parenthèse de Droite
*****	*	*****	Astérisque
+	+	**+**	Plus
,	,	**,**	Virgule
-	-	**‐**	Tiret (signe moins)
.	.	**.**	Point
/	/	**/**	Barre de fraction (slash)
0	0		Chiffre zéro
1	1		Chiffre un
2	2		Chiffre deux
3	3		Chiffre trois
4	4		Chiffre quatre
5	5		Chiffre cinq
6	6		Chiffre six
7	7		Chiffre sept
8	8		Chiffre huit
9	9		Chiffre neuf
:	:	**:**	Deux Points
;	;	**;**	Point-virgule
<	<	**<**	Inférieur
=	=	**=**	Egal
>	>	**>**	Supérieur
?	?	**?**	Point d'interrogation
@	@	**@**	Arobase
a	A		A, Majuscule

b	&#66;		B, Majuscule
C	&#67;		C, Majuscule
D	&#68;		D, Majuscule
E	&#69;		E, Majuscule
F	&#70;		F, Majuscule
G	&#71;		G, Majuscule
H	&#72;		H, Majuscule
i	&#73;		I, Majuscule
J	&#74;		J, Majuscule
K	&#75;		K, Majuscule
L	&#76;		L, Majuscule
M	&#77;		M, Majuscule
N	&#78;		N, Majuscule
O	&#79;		O, Majuscule
p	&#80;		P, Majuscule
q	&#81;		Q, Majuscule
R	&#82;		R, Majuscule
S	&#83;		S, Majuscule
T	&#84;		T, Majuscule
u	&#85;		U, Majuscule
V	&#86;		V, Majuscule
W	&#87;		W, Majuscule
X	&#88;		X, Majuscule
Y	&#89;		Y, Majuscule
Z	&#90;		Z, Majuscule
[	&#91;		Crochet Gauche
	&#92;		Barre oblique inversée (Antislash)
]	&#93;		Crochet de droite
^	&#94;		Accent circonflexe
_	&#95;		Souligné
`	&#96;		Accent grave
a	&#97;		a, Minuscule
b	&#98;		b, Minuscule
c	&#99;		c, Minuscule
d	&#100;		d, Minuscule

e	&#101;		e, Minuscule
f	&#102;		f, Minuscule
g	&#103;		g, Minuscule
h	&#104;		h, Minuscule
i	&#105;		i, Minuscule
j	&#106;		j, Minuscule
k	&#107;		k, Minuscule
l	&#108;		l, Minuscule
m	&#109;		m, Minuscule
n	&#110;		n, Minuscule
o	&#111;		o, Minuscule
p	&#112;		p, Minuscule
q	&#113;		q, Minuscule
r	&#114;		r, Minuscule
s	&#115;		s, Minuscule
t	&#116;		t, Minuscule
u	&#117;		u, Minuscule
v	&#118;		v, Minuscule
w	&#119;		w, Minuscule
x	&#120;		x, Minuscule
y	&#121;		y, Minuscule
z	&#122;		z, Minuscule
{	&#123;		Accolade de Gauche
|	&#124;		Barre verticale
}	&#125;		Accolade de Droite
~	&#126;		Tilde
	&#127; ä ™		Inutilisé
™	&#153;	**™**	Signe Marque enregistrée
	š à Ÿ	**¡**	Inutilisé
		** **	Espace insécable
¡	&#161;	**¡**	Exclamation inversée
¢	&#162;	**¢**	Cent (monnaie USA)
£	&#163;	**£**	Livre sterling
¤	&#164;	**¤**	Symbole monétaire général
¥	&#165;	**¥**	Yens

¦	&#166;	**¦ ou &brkbar;**	Barre verticale brisée
§	&#167;	**§**	Section
¨	&#168;	**¨ ou ¨**	Tréma
©	&#169;	**©**	Droit d'auteur
ª	&#170;	**ª**	Ordinal féminin
«	&#171;	**«**	Guillemet français ouvrant gauche
¬	&#172;	**¬**	Symbole "not" (opposé de)
-	&#173;	****	Tiret de césure
®	&#174;	**®**	Marque déposée
¯	&#175;	**¯ ou &hibar;**	Macron
°	&#176;	**°**	Degré
±	&#177;	**±**	Plus ou moins
2	&#178;	**²**	Exposant 2
3	&#179;	**³**	Exposant 3
´	&#180;	**´**	Accent aigu
µ	&#181;	**µ**	Lettre grecque "mu"
¶	&#182;	**¶**	Paragraphe
·	&#183;	**·**	Point médian
ü	&#184;	**¸**	Cédille
1	&#185;	**¹**	Exposant 1
º	&#186;	**º**	Ordinal masculin
»	&#187;	**»**	Guillemet français fermant droit
¼	&#188;	**¼**	Fraction un quart
½	&#189;	**½**	Fraction un demi
¾	&#190;	**¾**	Fraction trois-quarts
¿	&#191;	**¿**	Point d'interrogation inversé
À	&#192;	**À**	A, accent grave
Á	&#193;	**Á**	A, accent aigu
Â	&#194;	**Â**	A, accent circonflexe
Ã	&#195;	**Ã**	A, tilde
Ä	&#196;	**Ä**	A, tréma
Å	&#197;	**Å**	A, anneau
Æ	&#198;	**Æ**	E dans la (ligature)

Ç	Ç	**Ç**	C, cédille
È	È	**È**	E, accent grave
É	É	**É**	E, accent aigu
Ê	Ê	**Ê**	E, accent circonflexe
Ë	Ë	**Ë**	E, tréma
Ì	Ì	**Ì**	I, accent grave
Í	Í	**Í**	I, accent aigu
Î	Î	**Î**	I, accent circonflexe
Ï	Ï	**Ï**	I, tréma
Ð	Ð	**Ð ou Đ**	Eth majuscule
Ñ	Ñ	**Ñ**	N, tilde
Ò	Ò	**Ò**	O, accent grave
Ó	Ó	**Ó**	O, accent aigu
Ô	Ô	**Ô**	O, accent circonflexe
Õ	Õ	**Õ**	O, tilde
Ö	Ö	**Ö**	O, tréma
x	×	**×**	Signe "multiplié"
Ø	Ø	**Ø**	O, barré
Ù	Ù	**Ù**	U, accent grave
Ú	Ú	**Ú**	U, accent aigu
Û	Û	**Û**	U, accent circonflexe
Ü	Ü	**Ü**	U, tréma
Ý	Ý	**Ý**	Y, accent aigu
Þ	Þ	**Þ**	Thorn majuscule
ß	ß	**ß**	Ligature de szet
à	à	**à**	a, accent grave
á	á	**á**	a, accent aigu
â	â	**â**	a, accent circonflexe
ã	ã	**ã**	a, tilde
ä	ä	**ä**	a, tréma
å	å	**å**	a, anneau
æ	æ	**æ**	e dans l'a
ç	ç	**ç**	c, cédille
è	è	**è**	e, accent grave

é	&#233;	**é**	e, accent aigu
ê	&#234;	**ê**	e, accent circonflexe
ë	&#235;	**ë**	e, tréma
ì	&#236;	**ì**	i, accent grave
í	&#237;	**í**	i, accent aigu
î	&#238;	**î**	i, accent circonflexe
ï	&#239;	**ï**	i, tréma
ð	&#240;	**ð**	eth minuscule
ñ	&#241;	**ñ**	n, tilde
ò	&#242;	**ò**	o, accent grave
ó	&#243;	**ó**	o, accent aigu
ô	&#244;	**ô**	o, accent circonflexe
õ	&#245;	**õ**	o, tilde
ö	&#246;	**ö**	o, tréma
÷	&#247;	**÷**	Signe "divisé par"
ø	&#248;	**ø**	o, barré
ù	&#249;	**ù**	u, accent grave
ú	&#250;	**ú**	u, accent aigu
û	&#251;	**û**	u, accent circonflexe
ü	&#252;	**ü**	u, tréma
ý	&#253;	**ý**	y, accent aigu
þ	&#254;	**þ**	Thorn minuscule
ÿ	&#255;	**ÿ**	y, tréma
...			
Œ	&#338;	**Œ**	E dans l'O
œ	&#339;	**œ**	e dans l'o
...			
€	&#8364;	**€**	Euro

5 / Les références d'entités HTML

Pour convertir vos références d'entités html, vous pouvez utiliser ce convertisseur de caractères spéciaux, proposé par un fidèle lecteur.

Caractères	Références d'entités	Références décimales	Caractères Unicode
			\u00A0
¡	¡	¡	\u00A1
¢	¢	¢	\u00A2
£	£	£	\u00A3
¤	¤	¤	\u00A4
¥	¥	¥	\u00A5
¦	¦	¦	\u00A6
§	§	§	\u00A7
¨	¨	¨	\u00A8
©	©	©	\u00A9
ª	ª	ª	\u00AA
«	«	«	\u00AB
¬	¬	¬	\u00AC
			\u00AD
®	®	®	\u00AE
¯	¯	¯	\u00AF
°	°	°	\u00B0
±	±	±	\u00B1
²	²	²	\u00B2
³	³	³	\u00B3
´	´	´	\u00B4
µ	µ	µ	\u00B5
¶	¶	¶	\u00B6
·	·	·	\u00B7
¸	¸	¸	\u00B8
¹	¹	¹	\u00B9
º	º	º	\u00BA
»	»	»	\u00BB
¼	¼	¼	\u00BC
½	½	½	\u00BD
¾	¾	¾	\u00BE
¿	¿	¿	\u00BF
À	À	À	\u00C0
Á	Á	Á	\u00C1

Â	Â	Â	\u00C2
Ã	Ã	Ã	\u00C3
Ä	Ä	Ä	\u00C4
Å	Å	Å	\u00C5
Æ	Æ	Æ	\u00C6
Ç	Ç	Ç	\u00C7
È	È	È	\u00C8
É	É	É	\u00C9
Ê	Ê	Ê	\u00CA
Ë	Ë	Ë	\u00CB
Ì	Ì	Ì	\u00CC
Í	Í	Í	\u00CD
Î	Î	Î	\u00CE
Ï	Ï	Ï	\u00CF
Ð	Ð	Ð	\u00D0
Ñ	Ñ	Ñ	\u00D1
Ò	Ò	Ò	\u00D2
Ó	Ó	Ó	\u00D3
Ô	Ô	Ô	\u00D4
Õ	Õ	Õ	\u00D5
Ö	Ö	Ö	\u00D6
×	×	×	\u00D7
Ø	Ø	Ø	\u00D8
Ù	Ù	Ù	\u00D9
Ú	Ú	Ú	\u00DA
Û	Û	Û	\u00DB
Ü	Ü	Ü	\u00DC
Ý	Ý	Ý	\u00DD
Þ	Þ	Þ	\u00DE
ß	ß	ß	\u00DF
à	à	à	\u00E0
á	á	á	\u00E1
â	â	â	\u00E2
ã	ã	ã	\u00E3
ä	ä	ä	\u00E4

å	å	å	\u00E5
æ	æ	æ	\u00E6
ç	ç	ç	\u00E7
è	è	è	\u00E8
é	é	é	\u00E9
ê	ê	ê	\u00EA
ë	ë	ë	\u00EB
ì	ì	ì	\u00EC
í	í	í	\u00ED
î	î	î	\u00EE
ï	ï	ï	\u00EF
ð	ð	ð	\u00F0
ñ	ñ	ñ	\u00F1
ò	ò	ò	\u00F2
ó	ó	ó	\u00F3
ô	ô	ô	\u00F4
õ	õ	õ	\u00F5
ö	ö	ö	\u00F6
÷	÷	÷	\u00F7
ø	ø	ø	\u00F8
ù	ù	ù	\u00F9
ú	ú	ú	\u00FA
û	û	û	\u00FB
ü	ü	ü	\u00FC
ý	ý	ý	\u00FD
þ	þ	þ	\u00FE
ÿ	ÿ	ÿ	\u00FF
ƒ	ƒ	ƒ	\u0192
Α	Α	Α	\u0391
Β	Β	Β	\u0392
Γ	Γ	Γ	\u0393
Δ	Δ	Δ	\u0394
Ε	Ε	Ε	\u0395
Ζ	Ζ	Ζ	\u0396
Η	Η	Η	\u0397

Θ	Θ	Θ	\u0398
Ι	Ι	Ι	\u0399
Κ	Κ	Κ	\u039A
Λ	Λ	Λ	\u039B
Μ	Μ	Μ	\u039C
Ν	Ν	Ν	\u039D
Ξ	Ξ	Ξ	\u039E
Ο	Ο	Ο	\u039F
Π	Π	Π	\u03A0
Ρ	Ρ	Ρ	\u03A1
Σ	Σ	Σ	\u03A3
Τ	Τ	Τ	\u03A4
Υ	Υ	Υ	\u03A5
Φ	Φ	Φ	\u03A6
Χ	Χ	Χ	\u03A7
Ψ	Ψ	Ψ	\u03A8
Ω	Ω	Ω	\u03A9
α	α	α	\u03B1
β	β	β	\u03B2
γ	γ	γ	\u03B3
δ	δ	δ	\u03B4
ε	ε	ε	\u03B5
ζ	ζ	ζ	\u03B6
η	η	η	\u03B7
θ	θ	θ	\u03B8
ι	ι	ι	\u03B9
κ	κ	κ	\u03BA
λ	λ	λ	\u03BB
μ	μ	μ	\u03BC
ν	ν	ν	\u03BD
ξ	ξ	ξ	\u03BE
ο	ο	ο	\u03BF
π	π	π	\u03C0
ρ	ρ	ρ	\u03C1
ς	ς	ς	\u03C2

σ	σ	σ	\u03C3
τ	τ	τ	\u03C4
υ	υ	υ	\u03C5
φ	φ	φ	\u03C6
χ	χ	χ	\u03C7
ψ	ψ	ψ	\u03C8
ω	ω	ω	\u03C9
ϑ	ϑ	ϑ	\u03D1
ϒ	ϒ	ϒ	\u03D2
ϖ	ϖ	ϖ	\u03D6
•	•	•	\u2022
…	…	…	\u2026
′	′	′	\u2032
″	″	″	\u2033
‾	‾	‾	\u203E
⁄	⁄	⁄	\u2044
℘	℘	℘	\u2118
ℑ	ℑ	ℑ	\u2111
ℜ	ℜ	ℜ	\u211C
™	™	™	\u2122
ℵ	ℵ	ℵ	\u2135
←	←	←	\u2190
↑	↑	↑	\u2191
→	→	→	\u2192
↓	↓	↓	\u2193
↔	↔	↔	\u2194
↵	↵	↵	\u21B5
⇐	⇐	⇐	\u21D0
⇑	⇑	⇑	\u21D1
⇒	⇒	⇒	\u21D2
⇓	⇓	⇓	\u21D3
⇔	⇔	⇔	\u21D4
∀	∀	∀	\u2200
∂	∂	∂	\u2202

∃	∃	∃	\u2203
∅	∅	∅	\u2205
∇	∇	∇	\u2207
∈	∈	∈	\u2208
∉	∉	∉	\u2209
∋	∋	∋	\u220B
∏	∏	∏	\u220F
∑	∑	∑	\u2211
−	−	−	\u2212
∗	∗	∗	\u2217
√	√	√	\u221A
∝	∝	∝	\u221D
∞	∞	∞	\u221E
∠	∠	∠	\u2220
∧	∧	∧	\u2227
∨	∨	∨	\u2228
∩	∩	∩	\u2229
∪	∪	∪	\u222A
∫	∫	∫	\u222B
∴	∴	∴	\u2234
∼	∼	∼	\u223C
≅	≅	≅	\u2245
≈	≈	≈	\u2248
≠	≠	≠	\u2260
≡	≡	≡	\u2261
≤	≤	≤	\u2264
≥	≥	≥	\u2265
⊂	⊂	⊂	\u2282
⊃	⊃	⊃	\u2283
⊄	⊄	⊄	\u2284
⊆	⊆	⊆	\u2286
⊇	⊇	⊇	\u2287
⊕	⊕	⊕	\u2295
⊗	⊗	⊗	\u2297

⊥	⊥	⊥	\u22A5
⋅	⋅	⋅	\u22C5
⌈	⌈	⌈	\u2308
⌉	⌉	⌉	\u2309
⌊	⌊	⌊	\u230A
⌋	⌋	⌋	\u230B
〈	⟨	〈	\u2329
〉	⟩	〉	\u232A
◊	◊	◊	\u25CA
♠	♠	♠	\u2660
♦	♣	♣	\u2663
♥	♥	♥	\u2665
♦	♦	♦	\u2666
"	"	"	\u005C
&	&	&	\u0026
<	<	<	\u003C
>	>	>	\u003E
Œ	Œ	Œ	\u0152
œ	œ	œ	\u0153
Š	Š	Š	\u0160
š	š	š	\u0161
Ÿ	Ÿ	Ÿ	\u0178
ˆ	ˆ	ˆ	\u02C6
˜	˜	˜	\u02DC
			\u2002
			\u2003
			\u2009
	‌	‌	\u200C
	‍	‍	\u200D
	‎	‎	\u200E
	‏	‏	\u200F
–	–	–	\u2013
—	—	—	\u2014
‘	‘	‘	\u2018
’	’	’	\u2019

‚	‚	‚	\u201A
“	“	“	\u201C
”	”	”	\u201D
„	„	„	\u201E
†	†	†	\u2020
‡	‡	‡	\u2021
‰	‰	‰	\u2030
‹	‹	‹	\u2039
›	›	›	\u203A
€	€	€	\u20AC

6 / Le code ISO-639

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Code ISO-639	Code ISO-639-2/T	Code ISO-639-2/B	Nom de la langue
aa	aar	aar	**afar**
ab	abk	abk	**abkhaze**
	ace	ace	**atjihais**
	ach	ach	**atcholi (gang)**
	ada	ada	**adangmé**
	afa	afa	**afro-asiatiques (autres langues)**
	afh	afh	**afrihili**
af	afr	afr	**afrikaans**
	ajm	ajm	**aljamia**
	aka	aka	**akan**
	akk	akk	**akkadien**
	ale	ale	**aléoute**
	alg	alg	**langues algonquiennes**
am	amh	amh	**amharique**
	ang	ang	**ancien anglais (vers 450-1100)**
	apa	apa	**langues apaches**
ar	ara	ara	**arabe**
	arc	arc	**araméen**
	arn	arn	**mapoutche (araucan)**
	arp	arp	**arapaho**
	art	art	**artificielles (autres langues)**
	arw	arw	**arawak**
as	asm	asm	**assamais**
	ath	ath	**langues dénées (athasbascanes)**
	aus	aus	**langues australiennes**
	ava	ava	**avar**
	ave	ave	**avestique**
	awa	awa	**awadhi (aoudi)**
ay	aym	aym	**aymará**

az	aze	aze	**azéri**
	bad	bad	**banda**
	bai	bai	**langues bamiléké**
ba	bak	bak	**bachkir**
	bal	bal	**baloutche**
	bam	bam	**bambara**
	ban	ban	**balinais**
	bas	bas	**bassa**
	bat	bat	**baltes (autres langues)**
	bej	bej	**bedja**
be	bel	bel	**biélorusse**
	bem	bem	**bemba**
bn	ben	ben	**bengali**
	ber	ber	**berbères (autres langues)**
	bho	bho	**bhodjpouri**
bh	bih	bih	**bihari**
	bik	bik	**bikol**
	bin	bin	**bini**
bi	bis	bis	**bêche-de-mer**
	bla	bla	**pied-noir (siksika)**
	bnt	bnt	**bantoues (autres langues)**
bo	bod	tib	**tibétain**
	bra	bra	**bradj**
br	bre	bre	**breton**
	btk	btk	**batak (Indonésie)**
	bua	bua	**bouriate**
	bug	bug	**bouguis**
bg	bul	bul	**bulgare**
	cad	cad	**caddo**
	cai	cai	**amérindiennes d'Amérique centrale (autres langues)**
	car	car	**caribe**
ca	cat	cat	**catalan**
	cau	cau	**caucasiennes (autres langues)**
	ceb	ceb	**cébouano**
	cel	cel	**celtes (autres langues)**

cs	ces	cze	**tchèque**
	cha	cha	**chamorro**
	chb	chb	**tchibtcha**
	che	che	**tchétchène**
	chg	chg	**djaghataï**
	chk	chk	**trukais**
	chm	chm	**mari (tchérémisse)**
	chn	chn	**jargon tchinouk (jargon chinook)**
	cho	cho	**choctaw**
	chp	chp	**montagnais de l'Ouest (déné)**
	chr	chr	**chéroki**
	chu	chu	**slavon**
	chv	chv	**tchouvache**
	chy	chy	**cheyenne**
	cmc	cmc	**langues tchames (tiames, chames)**
	cop	cop	**copte**
	cor	cor	**cornique**
co	cos	cos	**corse**
	cpe	cpe	**créoles et pidgins anglais (autres)**
	cpf	cpf	**créoles et pidgins français (autres)**
	cpp	cpp	**créoles et pidgins portugais (autres)**
	cre	cre	**cri**
	crp	crp	**créoles et pidgins (autres)**
	cus	cus	**couchitiques (autres langues)**
cy	cym	wel	**gallois**
	dak	dak	**famille siouse (assiniboine, sioux, dakota)**
da	dan	dan	**danois**
	day	day	**dayak**
	del	del	**delaware (lénappé)**
	den	den	**esclave**
de	deu	ger	**allemand**
	dgr	dgr	**flanc-de-chien**
	din	din	**dinka**
	div	div	**divéhi**
	doi	doi	**dogri (kangri)**

	dra	dra	**dravidiennes (autres langues)**
	dua	dua	**douala**
	dum	dum	**moyen néerlandais (vers 1050-1350)**
	dyu	dyu	**dioula**
dz	dzo	dzo	**bhoutani (dzongkha)**
	efi	efi	**éfik (fi)**
	egy	egy	**ancien égyptien**
	eka	eka	**akadjou (ékadjouk)**
el	ell	gre	**grec moderne (1453-)**
	elx	elx	**élamite**
en	eng	eng	**anglais**
	enm	enm	**moyen anglais (1100-1500)**
eo	epo	epo	**espéranto**
es	esl	spa	**espagnol**
et	est	est	**estonien**
	eth	eth	**éthiopien**
eu	eus	baq	**basque**
	ewe	ewe	**éwé (éhoué)**
	ewo	ewo	**éwondo**
	fan	fan	**fang**
fo	fao	fao	**féroïen**
fa	fas	per	**persan (farsi)**
	fat	fat	**fanti**
fj	fij	fij	**fidjien**
fi	fin	fin	**finnois**
	fiu	fiu	**finno-ougriennes (autres langues)**
	fon	fon	**fon**
fr	fra	fre	**français**
	frm	frm	**moyen français (vers 1400-1600)**
	fro	fro	**ancien français (vers 842-1400)**
fy	fry	fry	**frison**
	ful	ful	**peul**
	fur	fur	**frioulan (rhéto-roman)**
	gaa	gaa	**gan**
	gay	gay	**gayo**

	gba	gba	**gbaya**
	gem	gem	**germanique (autre)**
	gez	gez	**guèze**
	gil	gil	**gilbertais (kiribatien)**
gd	gla	gla	**gaélique écossais (erse)**
ga	gle	gle	**irlandais (gaélique irlandais)**
gl	glg	glg	**galicien**
gv	glv	glv	**mannois**
	gmh	gmh	**moyen haut-allemand (vers 1050-1500)**
	goh	goh	**ancien haut-allemand (vers 750-1050)**
	gon	gon	**gondi**
	gor	gor	**gorontalo**
	got	got	**gotique**
	grb	grb	**grébo**
	grc	grc	**ancien grec (jusqu'à 1453)**
gn	grn	grn	**guarani**
gu	guj	guj	**goudjerati**
	gwi	gwi	**loucheux**
	hai	hai	**haïda**
ha	hau	hau	**haoussa**
	haw	haw	**hawaïen**
he	heb	heb	**hébreu**
	her	her	**héréro**
	hil	hil	**hiligainon**
	him	him	**himachali**
hi	hin	hin	**hindi**
	hit	hit	**hittite**
	hmn	hmn	**hmong (méo)**
	hmo	hmo	**hiri motou**
hr	hrv	scr	**croate**
hu	hun	hun	**hongrois**
	hup	hup	**houpa**
hy	hye	arm	**arménien**
	iba	iba	**iban (néban)**
	ibo	ibo	**igbo (ibo)**

	ijo	ijo	**djo (id'o)**
iu	iku	iku	**inuktitut (esquimau)**
ie	ile	ile	**interlingue**
	ilo	ilo	**iloko**
ia	ina	ina	**interlingua**
in			**indonésien, indicatif désuet, utiliser «id»**
	inc	inc	**Inde (autre langues de l')**
id	ind	ind	**indonésien**
	ine	ine	**indo-europeénnes (autres langues)**
ik	ipk	ipk	**inupiaq (inoupiak)**
	ira	ira	**iraniennes (autres langues)**
	iro	iro	**langues iroquiennes**
is	isl	ice	**islandais**
it	ita	ita	**italien**
iw			**hébreu, indicatif désuet, utiliser «he»**
ji			**yidich, indicatif désuet, utiliser «yi»**
jw	jaw	jav	**javanais**
ja	jpn	jpn	**japonais**
	jpr	jpr	**judéo-perse**
	jrb	jrb	**judéo-arabe**
	kaa	kaa	**kara-kalpak**
	kab	kab	**kabyle**
	kac	kac	**kachin**
kl	kal	kal	**groenlandais**
	kam	kam	**kamba**
kn	kan	kan	**canara (kannada)**
	kar	kar	**karen**
ks	kas	kas	**cachemiri**
ka	kat	geo	**géorgien**
	kau	kau	**kanouri**
	kaw	kaw	**kawi**
kk	kaz	kaz	**kazakh**
	kha	kha	**khasi**
	khi	khi	**khoï-san (autres langues)**
km	khm	khm	**khmer (cambodgien)**

	kho	kho	**khotanais**
	kik	kik	**kikouyou**
	kim	kim	**kimboundou**
rw	kin	kin	**kinyarouanda**
ky	kir	kir	**kirghiz**
	kmb	kmb	**kimboundou**
	kok	kok	**konkani**
	kom	kom	**komi (zyriène)**
	kon	kon	**kikongo**
ko	kor	kor	**coréen**
	kos	kos	**kosraéen**
	kpe	kpe	**kpèllé**
	kro	kro	**krou**
	kru	kru	**kouroukh**
	kua	kua	**ochikouanyama**
	kum	kum	**koumyk (koumouk)**
ku	kur	kur	**kurde**
	kus	kus	**kousaïe**
	kut	kut	**koutenai**
	lad	lad	**ladino**
	lah	lah	**lahnda**
	lam	lam	**lamba**
lo	lao	lao	**lao**
la	lat	lat	**latin**
lv	lav	lav	**letton (lette)**
	lez	lez	**lezguien**
ln	lin	lin	**lingala**
lt	lit	lit	**lituanien**
	lol	lol	**mongo**
	loz	loz	**lozi**
lb	ltz	ltz	**luxembourgeois**
	lua	lua	**luba-louloua**
	lub	lub	**louba-katanga**
	lug	lug	**louganda (ganda)**
	lui	lui	**luiseño**

	lun	lun	**lounda**
	luo	luo	**louo (Kénya et Tanzanie)**
	lus	lus	**lousaï**
	mad	mad	**madourais**
	mag	mag	**magahi**
	mah	mah	**marshallais**
	mai	mai	**maithili**
	mak	mak	**macassar**
ml	mal	mal	**malayalam**
	man	man	**mandingue**
	map	map	**austronésiennes (autres langues)**
mr	mar	mar	**marathe**
	mas	mas	**massaï**
	mdr	mdr	**mandar**
	men	men	**mendé**
	mga	mga	**irlandais, moyen (900-1200)**
	mic	mic	**micmac (souriquois)**
	min	min	**minangkabao**
	mis	mis	**diverses (autres langues)**
mk	mkd	mac	**macédonien**
	mkh	mkh	**môn-khmer (autres langues)**
mg	mlg	mlg	**malgache**
mt	mlt	mlt	**maltais**
	mni	mni	**manipouri**
	mno	mno	**langues manobo**
	moh	moh	**mohawk**
mo	mol	mol	**moldave**
mn	mon	mon	**mongol**
	mos	mos	**mossi**
mi	mri	mao	**maori**
ms	msa	may	**malais**
	mul	mul	**multiples langues**
	mun	mun	**langues mounda**
	mus	mus	**creek (mouskogui)**
	mwr	mwr	**marvari**

my	mya	bur	**birman**
	myn	myn	**langues maya**
	nah	nah	**aztèque (nahuatl)**
	nai	nai	**amérindiennes de l'Amérique septentrionale (autres langues)**
na	nau	nau	**nauri**
	nav	nav	**navaho**
	nbl	nbl	**ndébélé du Sud**
	nde	nde	**ndébélé du Nord**
	ndo	ndo	**ndonga**
ne	nep	nep	**népalais**
	new	new	**néwari**
	nia	nia	**niassais**
	nic	nic	**nigéro-kordofaniennes (autres langues)**
	niu	niu	**niouéen**
nl	nld	dut	**néerlandais**
	non	non	**norrois**
no	nor	nor	**norvégien**
	nso	nso	**sotho du Nord (pédi)**
	nub	nub	**langues nubiennes**
	nya	nya	**nyandja**
	nym	nym	**nyamwézi**
	nyn	nyn	**nkolé (lounyankolé)**
	nyo	nyo	**nyoro**
	nzi	nzi	**nzéma**
oc	oci	oci	**occitan ou langue d'oc (après 1500)**
	oji	oji	**ojibwa (saulteaux)**
or	ori	ori	**oriya**
om	orm	orm	**oromo**
	osa	osa	**osage**
	oss	oss	**ossète**
	ota	ota	**turc ottoman (osmanli) (1500-1928)**
	oto	oto	**langues otomangues**
	paa	paa	**papou-australiennes (autres langues)**
	pag	pag	**pangasinan**
	pal	pal	**pèhlevî**

	pam	pam	**pampangan (pampangueño)**
pa	pan	pan	**pendjabi**
	pap	pap	**papiamento**
	pau	pau	**palauan**
	peo	peo	**perse (vieux-) (vers 600-400 av. J.-C.)**
	phi	phi	**philippines (autres langues)**
	phn	phn	**phénicien**
	pli	pli	**pali**
pl	pol	pol	**polonais**
	pon	pon	**ponapien**
pt	por	por	**portugais**
	pra	pra	**prâkrit**
	pro	pro	**ancien provençal (jusqu'à 1500)**
ps	pus	pus	**pachto (pachtou)**
	qaa-qtz	qaa-qtz	**réservé pour utilisation locale**
qu	que	que	**quéchua (kitchoua)**
	raj	raj	**radjasthani**
	rap	rap	**rapanoui (pascuan)**
	rar	rar	**rarotongan**
	roa	roa	**romanes (autres langues)**
rm	roh	roh	**rhéto-roman**
	rom	rom	**romani (gitan)**
ro	ron	rum	**roumain**
rn	run	run	**roundi**
ru	rus	rus	**russe**
	sad	sad	**sandawé**
sg	sag	sag	**sango**
	sai	sai	**amérindiennes (Amérique du Sud, autres langues)**
	sal	sal	**langues salish**
	sam	sam	**samaritain (araméen)**
sa	san	san	**sanscrit**
	sas	sas	**sassak**
	sat	sat	**santali**
	sco	sco	**écossais**
	sel	sel	**selkoup**

	sem	sem	**sémitiques (autres langues)**
	sga	sga	**irlandais, ancien (jusqu'à 900)**
sh			**serbo-croate**
	shn	shn	**chan**
	sid	sid	**sidama**
si	sin	sin	**singhalais**
	sio	sio	**langues siouses**
	sit	sit	**sino-tibétaines (autres langues)**
	sla	sla	**slaves (autres langues)**
sk	slk	slo	**slovaque**
sl	slv	slv	**slovène**
se	smi	smi	**langues laponnes (samé)**
sm	smo	smo	**samoan**
sn	sna	sna	**chona**
sd	snd	snd	**sindhi**
	snk	snk	**soninké**
	sog	sog	**sogdien**
so	som	som	**somali**
	son	son	**songoï (songhaï)**
st	sot	sot	**sotho du Sud (sésoutho)**
sq	sqi	alb	**albanais**
	srd	srd	**sarde**
sr	srp	scc	**serbe**
	srr	srr	**sérère**
	ssa	ssa	**nilo-sahariennes (autres langues)**
ss	ssw	ssw	**swazi (siswati)**
	suk	suk	**soukouma**
su	sun	sun	**soundanais**
	sus	sus	**soussou**
	sux	sux	**sumérien**
sw	swa	swa	**souahéli**
sv	sve	swe	**suédois**
	syr	syr	**syriaque**
	tah	tah	**tahitien**
	tai	tai	**taï (autres langues)**

ta	tam	tam	**tamoul**
tt	tat	tat	**tatar**
te	tel	tel	**télougou**
	tem	tem	**timné**
	ter	ter	**térêna (téréno)**
	tet	tet	**tétun**
tg	tgk	tgk	**tadjik**
tl	tgl	tgl	**tagal**
th	tha	tha	**thaï**
	tig	tig	**tigré**
ti	tir	tir	**tigrigna**
	tiv	tiv	**tiv**
	tkl	tkl	**tokélaouén**
	tli	tli	**tlingit**
	tmh	tmh	**tamacheq**
to	tog	tog	**kitonga**
	ton	ton	**tonga (îles Tonga)**
	tpi	tpi	**tok pisine**
	tru	tru	**truk**
	tsi	tsi	**tsimchiane**
tn	tsn	tsn	**setchwana**
ts	tso	tso	**tsonga**
tk	tuk	tuk	**turkmène**
	tum	tum	**tchitoumbouka**
tr	tur	tur	**turc**
	tut	tut	**altaïques (autres langues)**
	tvl	tvl	**touvalouén**
tw	twi	twi	**tchi**
	tyv	tyv	**touvine (touvain)**
	uga	uga	**ougaritique**
ug	uig	uig	**ouïgour**
uk	ukr	ukr	**ukrainien**
	umb	umb	**oumboundou**
	und	und	**indeterminée**
ur	urd	urd	**ourdou**

uz	uzb	uzb	**ouzbek**
	vai	vai	**vaï**
	ven	ven	**venda**
vi	vie	vie	**vietnamien**
vo	vol	vol	**volapük**
	vot	vot	**vote (votiak)**
	wak	wak	**langues wakaches**
	wal	wal	**walamo**
	war	war	**waray**
	was	was	**washo**
	wen	wen	**sorabe (wende)**
wo	wol	wol	**ouolof**
xh	xho	xho	**xhosa**
	yao	yao	**yao**
	yap	yap	**yap**
yi	yid	yid	**yidich**
yo	yor	yor	**yorouba**
	ypk	ypk	**langues youpikes**
	zap	zap	**zapotèque**
	zen	zen	**zénaga**
za	zha	zha	**tchouang (zhuang)**
zh	zho	chi	**chinois**
	znd	znd	**zandé**
zu	zul	zul	**zoulou**
	zun	zun	**zuñi**

7 / Les identificateurs locaux

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Langue	Abréviation	Code hexadécimal	Code décimal
Africaans	af	0x0436	1078
Albanais	sq	0x041C	1052
Allemand - standard	de	0x0407	1031
Allemand - Autriche	de-at	0x0C07	3079
Allemand - Liechtenstein	de-li	0x1407	5127
Allemand - Luxembourg	de-lu	0x1007	4103
Allemand - Suisse	de-ch	0x0807	2055
Anglais - Australie	en-au	0x0C09	3081
Anglais - Belize	en-bz	0x2809	10249
Anglais - Canada	en-ca	0x1009	4105
Anglais - Irlande	en-ie	0x1809	6153
Anglais - Jamaïque	en-jm	0x2009	8201
Anglais - Nouvelle-Zélande	en-nz	0x1409	5129
Anglais - Afrique du Sud	en-za	0x1C09	7177
Anglais - Trinidad	en-tt	0x2C09	11273
Anglais - Royaume-Uni	en-gb	0x0809	2057
Anglais - États-Unis	en-us	0x0409	1033
Arabe - Émirats Arabes Unis.	ar-ae	0x3801	14337
Arabe - Bahreïn	ar-bh	0x3C01	15361
Arabe - Algérie	ar-dz	0x1401	5121
Arabe - Égypte	ar-eg	0x0C01	3073
Arabe - Iraq	ar-iq	0x0801	2049
Arabe - Jordanie	ar-jo	0x2C01	11265
Arabe - Koweït	ar-kw	0x3401	13313
Arabe - Liban	ar-lb	0x3001	12289
Arabe - Libye	ar-ly	0x1001	4097
Arabe - Maroc	ar-ma	0x1801	6145
Arabe - Oman	ar-om	0x2001	8193
Arabe - Qatar	ar-qa	0x4001	16385

Arabe - Arabie Saoudite	**ar-sa**	0x0401	1025
Arabe - Syrie	**ar-sy**	0x2801	10241
Arabe - Tunisie	**ar-tn**	0x1C01	7169
Arabe - Yémen	**ar-ye**	0x2401	9217
Basque	**eu**	0x042D	1069
Biélorusse	**be**	0x0423	1059
Bulgare	**bg**	0x0402	1026
Catalan	**ca**	0x0403	1027
Chinois - Rép. Populaire de Chine	**zh-cn**	0x0804	2052
Chinois - Hong Kong	**zh-hk**	0x0C04	3076
Chinois - Singapour	**zh-sg**	0x1004	4100
Chinois - Taïwan	**zh-tw**	0x0404	1028
Coréen	**ko**	0x0412	1042
Croate	**hr**	0x041A	1050
Danois	**da**	0x0406	1030
Espagnol - standard	**es**	0x040A	1034
Espagnol - Argentine	**es-ar**	0x2C0A	11274
Espagnol - Bolivie	**es-bo**	0x400A	16394
Espagnol - Chili	**es-cl**	0x340A	13322
Espagnol - Colombie	**es-co**	0x240A	9226
Espagnol - Costa Rica	**es-cr**	0x140A	5130
Espagnol - République dominicaine	**es-do**	0x1C0A	7178
Espagnol - Équateur	**es-ec**	0x300A	12298
Espagnol - Guatemala	**es-gt**	0x100A	4106
Espagnol - Honduras	**es-hn**	0x480A	18442
Espagnol - Mexique	**es-mx**	0x080A	2058
Espagnol - Nicaragua	**es-ni**	0x4C0A	19466
Espagnol - Panama	**es-pa**	0x180A	6154
Espagnol - Pérou	**es-pe**	0x280A	10250
Espagnol - Puerto Rico	**es-pr**	0x500A	20490
Espagnol - Paraguay	**es-py**	0x3C0A	15370
Espagnol - El Salvador	**es-sv**	0x440A	17418
Espagnol - Uruguay	**es-uy**	0x380A	14346
Espagnol - Venezuela	**es-ve**	0x200A	8202
Estonien	**et**	0x0425	1061

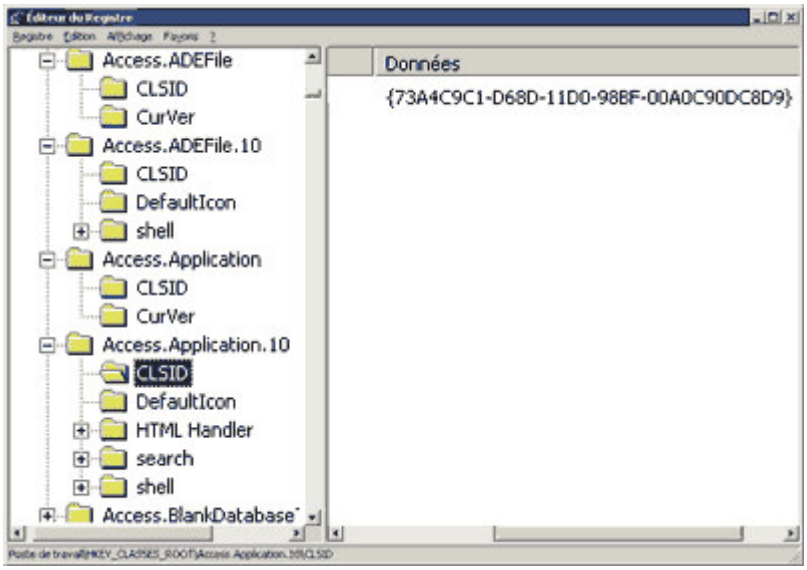
Farsi (Persan)	**fa**	0x0429	1065
Finnois	**fi**	0x040B	1035
Féroen	**fo**	0x0438	1080
Français - standard	**fr**	0x040C	1036
Français - Belgique	**fr-be**	0x080C	2060
Français - Canada	**fr-ca**	0x0C0C	3084
Français - Luxembourg	**fr-lu**	0x140C	5132
Français - Suisse	**fr-ch**	0x100C	4108
Gaélique - Écosse	**gd**	0x043C	1084
Grec	**el**	0x0408	1032
Hébreu	**he**	0x040D	1037
Hindi	**hi**	0x0439	1081
Hongrois	**hu**	0x040E	1038
Islandais	**is**	0x040F	1039
Indonésien	**in**	0x0421	1057
Italien - standard	**it**	0x0410	1040
Italien - Suisse	**it-ch**	0x0810	2064
Japonais	**ja**	0x0411	1041
Letton	**lv**	0x0426	1062
Lituanien	**lt**	0x0427	1063
Macédonien	**mk**	0x042F	1071
Malais - Malaisie	**ms**	0x043E	1086
Maltais	**mt**	0x043A	1082
Néerlandais	**nl**	0x0413	1043
Néerlandais - Belgique	**nl-be**	0x0813	2067
Norvégien - Bokmaal	**no**	0x0414	1044
Ourdou - Pakistan	**ur**	0x0420	1056
Polonais	**pl**	0x0415	1045
Portugais - standard	**pt**	0x0816	2070
Portugais - Brésil	**pt-br**	0x0416	1046
Rhéto-roman	**rm**	0x0417	1047
Roumain	**ro**	0x0418	1048
Roumain - Moldavie	**ro-mo**	0x0818	2072
Russe	**ru**	0x0419	1049
Russe - Moldavie	**ru-mo**	0x0819	2073

Serbe - Cyrillique	**sr**	0x0C1A	3098
Setswana	**tn**	0x0432	1074
Slovène	**sl**	0x0424	1060
Slovaque	**sk**	0x041B	1051
Sorbe	**sb**	0x042E	1070
Sutu	**sx**	0x0430	1072
Suédois	**sv**	0x041D	1053
Suédois - Finlande	**sv-fi**	0x081D	2077
Tchèque	**cs**	0x0405	1029
Thaï	**th**	0x041E	1054
Turc	**tr**	0x041F	1055
Tsonga	**ts**	0x0431	1073
Ukrainien	**uk**	0x0422	1058
Vietnamien	**vi**	0x042A	1066
Xhosa	**xh**	0x0434	1076
Yiddish	**ji**	0x043D	1085
Zoulou	**zu**	0x0435	1077

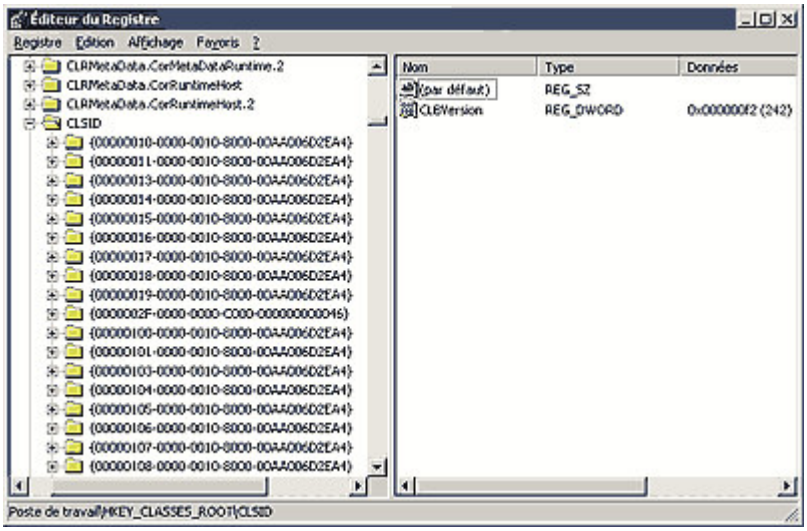
8 / Les identifcateurs d'application windows

Les identificateurs *CLSID* (CLasse IDentifier) sont utilisés pour appeler des applications Microsoft Windows dans des prorammes écrits dans divers langages tels que l'ASP, le PHP ou d'autres.

La liste des identificateurs *CLSID* présents sur un serveur se trouvent dans le fichier *regedit.exe*, dans le répertoire *HKEY_CLASSES_ROOT* classés par application.



Une liste complète classée par identificateurs, se trouve également sous le dossier *CLSID* du même répertoire.



9 / Les pages de code

Les pages de code (CodePages) standards supportés par Windows sont citées dans le tableau ci-dessous avec le ou les jeux de caractères et les langues correspondants.

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Jeu de caractères	Page de code (CodePage)	Langue
ansi_x3.4-1968	1252	Europe de l'Ouest
ansi_x3.4-1986	1252	Europe de l'Ouest
ascii	1252	Europe de l'Ouest
big5	950	Chinois traditionnel (BIG5)
chinese	936	Chinois simplifié
cp367	1252	Europe de l'Ouest
cp819	1252	Europe de l'Ouest
csascii	1252	Europe de l'Ouest
csbig5	950	Chinois traditionnel (BIG5)
cseuckr	949	Coréen
cseucpkdfmtjapanese	CODE_JPN_EUC	Japonais (EUC)
csgb2312	936	Chinois simplifié (GB2312)
csiso2022jp	CODE_JPN_JIS	Japonais (JIS-Allow 1 byte Kana)
csiso2022kr	50225	Coréen (ISO)
csiso58gb231280	936	Chinois simplifié (GB2312)
csisolatin2	28592	Europe Centrale (ISO)
csisolatinhebrew	1255	Hébreux (ISO-Visual)
cskoi8r	20866	Cyrillique (KOI8-R)
csksc56011987	949	Coréen
csshiftjis	932	Shift-JIS
euc-kr	949	Coréen
extended_unix_code_packed_format_for_japanese	CODE_JPN_EUC	Japonais (EUC)
gb2312	936	Chinois simplifié (GB2312)
gb_2312-80	936	Chinois simplifié (GB2312)
hebrew	1255	Hébreux
hz-gb-2312	936	Chinois simplifié (HZ)
ibm367	1252	Europe de l'Ouest
ibm819	1252	Europe de l'Ouest
ibm852	852	Europe Centrale (DOS)
ibm866	866	Cyrillique (DOS)
iso-2022-jp	CODE_JPN_JIS	Japonais (JIS)
iso-2022-kr	50225	Coréen (ISO)
iso-8859-1	1252	Europe de l'Ouest

iso-8859-2	28592	Europe Centrale (ISO)
iso-8859-8	1255	Hébreux (ISO-Visual)
iso-ir-100	1252	Europe de l'Ouest
iso-ir-101	28592	Europe Centrale (ISO)
iso-ir-138	1255	Hébreux (ISO-Visual)
iso-ir-149	949	Coréen
iso-ir-58	936	Chinois simplifié (GB2312)
iso-ir-6	1252	Europe de l'Ouest
iso646-us	1252	Europe de l'Ouest
iso8859-1	1252	Europe de l'Ouest
iso8859-2	28592	Europe Centrale (ISO)
iso_646.irv:1991	1252	Europe de l'Ouest
iso_8859-1	1252	Europe de l'Ouest
iso_8859-1:1987	1252	Europe de l'Ouest
iso_8859-2	28592	Europe Centrale (ISO)
iso_8859-2:1987	28592	Europe Centrale (ISO)
iso_8859-8	1255	Hébreux (ISO-Visual)
iso_8859-8:1988	1255	Hébreux (ISO-Visual)
koi8-r	20866	Cyrillique (KOI8-R)
korean	949	Coréen
ks-c-5601	949	Coréen
ks-c-5601-1987	949	Coréen
ks_c_5601	949	Coréen
ks_c_5601-1987	949	Coréen
ks_c_5601-1989	949	Coréen
ksc-5601	949	Coréen
ksc5601	949	Coréen
ksc_5601	949	Coréen
l2	28592	Europe Centrale (ISO)
latin1	1252	Europe de l'Ouest
latin2	28592	Europe Centrale (ISO)
ms_kanji	932	Shift-JIS
shift-jis	932	Shift-JIS
shift_jis	932	Shift-JIS
us	1252	Europe de l'Ouest

us-ascii	1252	Europe de l'Ouest
windows-1250	1250	Europe Centrale (Windows)
windows-1251	1251	Cyrillique (Windows)
windows-1252	1252	Europe de l'Ouest
windows-1253	1253	Grec (Windows)
windows-1254	1254	Turc (Windows)
windows-1255	1255	Hébreux
windows-1256	1256	Arabe
windows-1257	1257	Balte (Windows)
windows-1258	1258	Vietnamese
windows-874	874	Thai
x-cp1250	1250	Europe Centrale (Windows)
x-cp1251	1251	Cyrillique (Windows)
x-euc	CODE_JPN_EUC	Japonais (EUC)
x-euc-jp	CODE_JPN_EUC	Japonais (EUC)
x-sjis	932	Shift-JIS
x-x-big5	950	Chinois traditionnel (BIG5)

10 / Les entêtes MIME

Les entêtes des messages Internet sont normalisés par les RFC (Requests For Comments) 822, 2045, 2046, 2047, 2048, 2049 relatif au extensions polyvalentes des messages internet (MIME : Multipurpose Internet Mail Extensions).

Les champs d'entête
From: expediteur@email.com [, ...] CRLF
représente la liste des auteurs du courrier.
Sender: expediteur@email.com CRLF
représente l'adresse de l'expéditeur du courrier.
Reply-To: adresse_reponse@email.com [, ...] CRLF
représente l'adresse de réponse au courrier électronique.
To: destinataire@email.com [, ...] CRLF
représente la liste d'adresses des destinataires du courrier.
Cc: destinataire_copie@email.com [, ...] CRLF
représente la liste des destinataires d'une copie du courrier.
Bcc: destinataire_copie@email.com [, ...] CRLF
représente les destinataires non-visible d'une copie du courrier.
Message-ID: code_message CRLF
représente un code unique d'identification du courrier.
In-Reply-To: message_id [, ...] CRLF
est utilisé pour identifier le (ou les) courriers pour lequel il en est un nouveau.
References: message_id CRLF
est utilisé pour identifier le fil de la conversation.
Subject: [Re:] Sujet... CRLF
représente le sujet du courrier électronique avec optionnellement le suffixe *Re:* pour une réponse.
Comments: Commentaire... CRLF
représente un commentaire à propos du courrier.
Keywords: Mot-clé [, ...] CRLF
représente des mots-clés relatifs au courrier.
Date: date CRLF
représente des mots-clés relatifs au courrier.
MIME-Version: 1.0 CRLF
représente la version MIME du courrier.
Content-Type: type/sous-type; {charset = encodage} | {boundary = délimiteur} CRLF
représente le type et le sous-type (text/plain, image/jpeg, audio/basic, application/postscript, etc.) et l'encodage (US-ASCII ou ISO-8859-X) du contenu d'un courrier. Si le couple type/sous-type possède la valeur *multipart/mixed* ou *multipart/alternative*, l'attribut *boundary* permet de délimiter les parties encodées différemment par une chaîne de caractères spéciale.
Content-transfer-encoding: 7bit | 8bit | binary | quoted-printable | base64 CRLF
définit un mécanisme d'encodage du contenu d'un courrier.

Content-ID: message_id CRLF
représente une référence à un contenu d'un autre courrier.
Content-Description: texte... CRLF
représente une information descriptive à propos du contenu d'un courrier.

From: Jacques Crenca <j_c@domaine.net>
To: Jean Jean <jean2@dom.com>
Reply-To: "Jacques Crenca" <message@domaine.net>
Subject: Re: Bonjour
Date: Mon, 25 Mar 2002 09:18:52 -0200
Message-ID: <1255388558@domaine.net>
In-Reply-To: <20012500365485@domaine.net>
References: <20012500365485@domaine.net>
MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Le second exemple fait appel à un contenu mixte en assemblant deux messages à un courrier électronique.

From: Jacques Crenca <j_c@domaine.net>
To: Jean Jean <jean2@dom.com>
Bcc: Direction <pdg@domaine.net>
Date: Mon, 25 Mar 2002 09:18:52 -0200
Subject: Critique du rapport n°10254365
MIME-Version: 1.0
Message-ID: <1255388558@domaine.net>
Content-Type: multipart/mixed; boundary="/-----10254365-----/"
Content-ID: <id53464631236546@site.com>

--/-----10254365-----/

Première partie du message...

--/-----10254365-----/
Content-Type: multipart/digest; boundary="/-----suite du courrier-----/"

--/-----suite du courrier-----/

From: Emile Ntamack <emile@domaine.net>
Date: Mon, 25 Mar 2002 09:18:52 -0200
Subject: Remarque d'un interlocuteur

Seconde partie du message...

--/-----suite du courrier-----/

From: Jean-Pierre Rives <jp.rives@domaine.net>
Date: Mon, 25 Mar 2002 09:18:52 -0200
Subject: Remarque d'une autre personne

Troisième partie du message...

--/-----suite du courrier-----/

--/-----10254365-----/--

Un message l'attribut *content-type* égal à *multipart/alternative* possède plusieurs parties proposant un contenu identique mais accessible par différent mécanisme. Dans l'exemple ci-dessous, un contenu spécial est proposé selon trois méthodes alternatives.

From: Jacques Crenca <j_c@domaine.net>
To: Jean Jean <jean2@dom.com>
Date: Mon, 25 Mar 2002 09:18:52 -0200
Subject: Sujet du courrier
MIME-Version: 1.0
Message-ID: <123486709786768@domaine.net>
Content-Type: multipart/alternative; boundary=216878686686346458

```
Content-ID: <id53464631236546@site.com>

--216878686686346458
Content-Type: message/external-body; name="fichier.ps";
        site="laltruiste.com"; mode="image";
        access-type=ANON-FTP; directory="fichier/rapport";
        expiration="Fri, 14 Jun 1991 19:13:14 -0400 (EDT)"

Content-type: application/postscript
Content-ID: <id216878686686346458@laltruiste.com>

--216878686686346458
Content-Type: message/external-body; access-type=local-file;
        name="/doc/sujet/fichier.ps";
        site="laltruiste.com";
        expiration="Fri, 14 Jun 1991 19:13:14 -0400 (EDT)"

Content-type: application/postscript
Content-ID: <id216878686686346458@laltruiste.com>

--216878686686346458
Content-Type: message/external-body;
        access-type=mail-server
        server="laltruiste@server.net";
        expiration="Fri, 14 Jun 1991 19:13:14 -0400 (EDT)"

Content-type: application/postscript
Content-ID: <id216878686686346458@laltruiste.com>

get fichier.rtf

--216878686686346458--
```

11 / Les requêtes HTTP

Les échanges entre un serveur et un client sur Internet doivent se conformer à des règles communes, afin que chacun puisse se comprendre. Les hommes (et femmes) communiquent entre eux par l'intermédiaire d'une langue commune comme le français, l'anglais, ou encore le mandarin. De la même façon, les différents intervenants du web se conforment à un protocole de communication appelé HTTP (HyperText Tranfer Protocol 1.0 ou 1.1) dont les normes sont édictées dans la RFC 1945 et RFC 2616.

Ainsi, lorsqu'un client à l'instar des navigateurs Internet, demande l'affichage d'une page Web, en fait une requête HTTP est transmise à un serveur qui retournera une réponse HTTP adéquate qui sera visualisable dans le navigateur.

Les requêtes et les réponses HTTP sont composées d'un entête et éventuellement d'un corps.

Le corps HTTP est soit une combinaison de paire clé/valeur pour une requête soumise par un client, soit un contenu (une page HTML par exemple) pour une réponse envoyée par un serveur.

```
domaine=laltruiste.com&titre=Le+guide+des+langages+web
//ou
<html>
   <body>
      <p>Un contenu quelconque...</p>
   </body>
</html>
```

La combinaison de paires clé/valeur résulte des champs d'un formulaire HTML que l'utilisateur aurait rempli par le biais de son navigateur Internet.

```
<form method="POST"
            name="formulaire"
            action="http://serveur.com/script.cgi">
   <label>Domaine :</label>
      <input type="text" name="domaine" value="laltruiste.com"/>
   <label>Titre :</label>
      <input type="text" name="titre" value="Le guide des langages web"/>
   <input type="submit" name="soumission" value="Soumettre"/>
</form>
```

Après soumission d'un tel formulaire, l'ensemble des champs sont combinées de telle manière que les attibuts *name* et *value* forment des paires séparées par un caractères *&*.

Cette combinaison peut être :

- soit concaténée à l'adresse URL de l'attribut *action* si la méthode de transmission du formulaire (attribut *method*) est *GET*, dans ce cas, la combinaison se dénomme une chaîne de requêtes,
- soit composée le corps d'une requête HTTP si la méthode de transmission est *POST*.

```
Méthode GET (chaîne de requêtes)
http://serveur.com/script.cgi?domaine=laltruiste.com
      &titre=Le+guide+des+langages+web&soumission=Soumettre

Méthode POST (corps HTTP) :
domaine=laltruiste.com&titre=Le+guide+des+langages+web&soumission=Soumettre
```

Un entête HTTP est composée d'un certain nombre de lignes.

La première ligne, appelée parfois prologue, d'un entête HTTP pour une requête, regroupe la méthode de transmission (ex. *GET* et *POST*), le chemin de la ressource sollicitée sur le serveur (ex.: *http://site.com/script.cgi* ou */script.cgi*), et la version du protocole(ex.: *HTTP/1.0*).

```
POST http://site.com/script.php HTTP/1.1
```

En ce qui concerne une réponse HTTP, le prologue est constitué de la version du protocole (ex.: *HTTP/1.x*), du code de statut (ex.: *200* ou *404*) et de sa decription (ex.: *OK* ou *NOT FOUND*).

```
HTTP/1.x 200 OK
```

Les lignes suivantes sont décomposées en un nom d'entête HTTP et une valeur associée. Les valeurs des

noms d'entêtes sont définies par rapport aux caractéristiques du client HTTP ou du serveur (ex.: encodage de caractères, agent utilisateur, langages acceptés) et celles du corps HTTP (ex.: type MIME, encodage, longueur du contenu).

```
//Requête HTTP
GET /accueil.php?compteur=1&evolution=5 HTTP/1.1
Host: www.laltruiste.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; fr) Gecko/20040113
Accept: text/xml,application/xml,application/xhtml+xml,
        text/html;q=0.9,text/plain;q=0.8,image/png,
            image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: fr,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.laltruiste.com/accueil.php

//Réponse HTTP
HTTP/1.x 200 OK
Date: Tue, 01 Mar 2005 11:42:27 GMT
Server: Apache/1.3.33 (Unix) mod_ssl/2.8.22 OpenSSL/0.9.7e DAV/1.0.3
        mod_gzip/1.3.26.1a AuthMySQL/2.30 PHP/4.3.10
X-Powered-By: PHP/4.3.10
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
Content-Encoding: gzip
Content-Length: 5357

//Contenu HTML...
```

Les lignes d'un message HTTP son systématiquement terminées par un caractère de retour charriot (*Carriage Return :* \r) suivi d'un caractère de fin de ligne (*Line Feed :* \n). De plus, l'entête est séparé du corps par un double saut de lignes (*CRLF + CRLF*).

11.1 / Les champs HTTP

Champs d'entête d'une requête HTTP

Nom	Description
Exemple	
Accept	propose une liste de types de contenu acceptés par le navigateur.
Accept: text/xml,application/xml,application/xhtml+xml,text/html	
Accept-Charset	propose une liste de jeux de caractères acceptés par le navigateur.
Accept-Charset: ISO-8859-1,utf-8	
Accept-Encoding	désigne l'encodage de données accepté par le navigateur.
Accept-Encoding: gzip,deflate	
Accept-Language	propose une liste de langues acceptés par le navigateur.
Accept-Language: fr,en	
Authorization	contient des informations d'dentification du client auprès du serveur distant.
Authorization: Basic JKhJfrNdiiz1qOs02	
Connection	indique le type de persistance de la connexion pour le transfert de plusieurs documents.
Connection: Keep-Alive	
Cache-Control	permet de gérer la manière dont la ressource concernée doit être partagée entre plusieurs utilisateurs.
Cache-Control: max-age=0	
Content-Encoding	précise le type d'encodage du corps de la requête.
Content-Encoding: gzip	
Content-Language	indique le type de langage du corps de la requête.
Content-Length	indique la longueur du corps de la requête.
Content-Length: 159	
Content-Type	désigne le type de contenu du corps de la requête.
Content-Type: application/x-www-form-urlencoded	
Date	fournit la date et l'heure de l'émission de la requête.
Date: Tue, 01 Mar 2005 14:21:12 GMT	
From	indique le courriel du client.
From: webmaster@laltruiste.com	
If-Modified-Since	demande une mise à jour de la ressource concernée, s'il a été modifié depuis cette date.
If-Modified-Since: Thu, 26 Apr 2001 23:27:22 GMT	
If-None-Match	demande une mise à jour si aucune ressource ne correspond à l'identifiant (*Etag*) spécifié.
If-None-Match: "6530b-952-3f7d76b8"	

Keep-Alive	conserve la connexion persistante avec le serveur pendant un certain temps.
Keep-Alive: timeout=15, max=100	
Host	désigne le serveur hôte auquel la requête s'adresse.
Host: www.laltruiste.com	
Last-Modified	indique la date et l'heure à laquelle la ressource concernée a été probablement modifiée.
Last-Modified: Thu, 26 Apr 2001 23:27:22 GMT	
Location	indique la localisation exacte de la ressource concernée.
Location: http://www.site.com/rep/page.html	
Referer	indique la dernière adresse URL visitée.
Referer: http://www.laltruiste.com/document.php	
Set-Cookie	contient des informations à propos du cookie affecté au client.
Set-Cookie: ban_1188=%23%2F221862%23%2F4256648; expires=Wed, 02-Mar-05 16:51:06 GMT	
User-Agent	contient des informations à propos du client HTTP (noms et versions du navigateur Internet et du système d'exploitation, ...).
Mozilla/5.0 (Windows; U; Windows NT 5.0; fr; rv:1.6) Gecko/20040113	

Champs d'entête d'une réponse HTTP

Nom	Description
Exemple	
Connection	fournit le type de persistance de la connexion pour le transfert de plusieurs documents.
Connection: close	
Content-Encoding	indique le type d'encodage du corps de la réponse.
Content-Encoding: gzip	
Content-Language	indique la langue contenu du corps de la réponse.
Content-Language: ru	
Content-Length	donne la longueur du contenu du corps de la réponse.
Content-Length: 2454	
Content-Type	indique le type du contenu du corps de la réponse.
Content-Type: text/html	
Date	fournit la déte de l'émision de la réponse.
Date: Tue, 01 Mar 2005 16:51:05 GMT	
Etag	correspond à un identifiant unique de la ressource envoyée.
Etag: "6530b-952-3f7d76b8"	
Expires	fournit une date d'expiration de la ressource concernée.
Expire: Thu, 26 Apr 2001 23:27:22 GMT	
Location	contient une adresse URL de redirection vers une nouvelle ressource.
Location: http://www.site.com/dir/page.html	
Server	donne des informations à propos du serveur concerné.
Server: Microsoft-IIS/5.0	
Transfer-Encoding	indique le type d'encodage utilisé pour l'émission de la réponse.
Transfer-Encoding: chunked	

Méthode de transmission HTTP

Nom	Description
CONNECT	est destiné à être utilisé avec un proxy.
DELETE	entraîne la suppression de la ressource désignée par l'adresse URL suivant cette commande.
GET	engendre l'émission d'une requête dont les éventuelles données sont stockées dans l'adresse URL.
HEAD	Requête de la ressource située à l'URL spécifié
OPTIONS	permet d'obtenir des informations à propos du serveur.
POST	provoque la soumission de données contenu dans le corps de la requête, au serveur chargé de les traiter.
PUT	provoque l'émission de données au serveur à l'adresse URL spécifiée.
TRACE	permet de récupérer une trace de ce qui a été reçu à l'autre extrêmité de la chaîne de transmission.

11.2 / Les codes HTTP

La transmission d'une requête HTTP (Hyper Text Transfer Protocole) sur Internet provoque une réponse du serveur sollicité. Ces réponses sont toujours accompagnées d'un code HTTP qui permet de savoir comment la requête soumise a été gérée par le serveur.

Cette page fournit la liste des codes HTTP renvoyés par un serveur. Les codes HTTP sont divisés en quatre catégories :

1. Informationnel 1xx
2. Opération réussie
3. Erreur du client 4xx
4. Erreur du serveur 5xx

Code	Message	Description
	Information 1xx	
100	Continue	La partie initiale de la requête a bien été reçue et le serveur poursuit le traitement.
101	Switching Protocols	Le serveur est disposé à se conformer au changement de protocole, demandé par la requête.
	Opération réussie 2xx	
200	OK	Le traitement de la requête s'est accompli avec succès.
201	Created	Une nouvelle ressource a été créée.
202	Accepted	La requête a été acceptée pour un traitement, mais celui-ci n'est pas encore achevé.
203	Non-Authoritative Information	Les metainformations retournées dans l'entête ne sont pas définies par le serveur d'origine, mais ont été récupérées à partir d'une copie locale ou d'un tiers.
204	No Content	Le serveur n'a pas besoint suite au traitement de la requête d'envoyer un réponse au client.
205	Reset Content	Le serveur a achevé le traitement de la requête mais le client réinitialise le document provoquant la transmission d'une de la requête.
206	Partial Content	Le serveur a accompli une requête GET partielle pour la ressource.
	Redirection 3xx	
300	Multiple Choices	Le client doit sélectionner le choix approprié car il existe plusieurs choix de retour.
301	Moved Permanently	La ressource demandée a été déplacée de façon permanente.
302	Found	La ressource demandée existe temporairement à un emplacement différent.
303	See Other	La ressource demandée peut être trouvée sous une URI différente et devrait pouvoir être obtenue en utilisant un méthode GET sur cette ressource.
304	Not Modified	La ressource accédée par le client n'a pas été modifiée.
305	Use Proxy	La ressource requise doit être accédée au travers d'un proxy.
306	Inutilisé	Inutilisé
307	Temporary Redirect	La ressource demandée est temporairement sous une URI différente.
	Erreur client 4xx	
400	Bad Request	La requête soumise a été mal formulée.
401	Unauthorized	L'utilisateur ne possède pas d'autorisation pour accéder à la ressource demandée.
402	Payment Required	Le paiement est requis.
403	Forbidden	L'accès au serveur est interdit.
404	Not Found	La page demandée n'a pu être trouvée.
405	Method Not Allowed	La méthode spécifié n'est pas autorisée pour la ressource identifiée par l'URI de la requête.
406	Not Acceptable	La requête ne peut être acceptée par le serveur.
407	Proxy Authentication Required	L'utilisateur doit s'authentifier auprès du proxy.
408	Request Timeout	Le temps d'accès à la ressource demandée a expiré.

409	Conflict	La requête a échoué en raison d'un conflit avec l'état courant de la ressource. L'utilisateur doit être capable de résoudre le conflit et de soumettre à nouveau la requête.
410	Gone	La ressource attendue n'est plus disponible.
411	Length Required	Le serveur a refusé la requête car la longueur du message n'a pas été définie par le champ *Content-Length*.
412	Precondition Failed	Les préconditions spécifiées dans un ou plusieurs champs de la requête ont échoué.
413	Request Entity Too Large	L'entité de la requête est trop grande.
414	Request-URI Too Long	L'URI de la requête est trop longue.
415	Unsupported Media Type	Le type de média n'est pas supporté.
416	Requested Range Not Satisfiable	La valeur du champ *Content-Range* n'est pas approprié.
417	Expectation Failed	L'attente spécifiée dans le champ *Expect* ne peut être satisfaite.
Erreur serveur 5xx		
500	Internal Server Error	Le serveur a produit une erreur interne.
501	Not Implemented	Le serveur ne supporte pas la fonctionnalité requise.
502	Bad Gateway	La passerelle d'accès ou le proxy est mauvais.
503	Service Unavailable	Le service est indisponible.
504	Gateway Timeout	Le temps d'accès à la passerelle a expiré.
505	HTTP Version Not Supported	La version HTTP n'est pas gérée par le serveur.

12 / Les modes d'accès aux fichiers et répertoires

Il existe différents modes permettant d'affecter des droits d'accès aux fichiers ou aux répertoires.

Les droits d'accès peuvent se faire **en lecture** (*r : read*), **en écriture** (*w : write*) et **en exécution** (*x : execute*) ou bien en une quelconque combinaison de ces trois derniers.

droits d'accès : **rwx r-x r--** *égale à* **7 5 4**

Les permissions peuvent être **accordées différemment à trois types d'utilisateurs** : les propriétaires, les utilisateurs faisant partis d'un groupe et les tous les autres.

Chaque type est représenté par un nombre octal résultant d'une addition des valeurs de bit provenant de chacun de leurs attributs.

Valeurs des attributs

Attribut	Action	Valeur
r	Aucun	0
x	eXecute	1
w	Write	2
r	Read	4

Droits d'accès au maximum

Type	Attributs	Valeur
Propriétaire	rwx	7
Groupe	rwx	7
Autres	rwx	7

Combinaisons possibles

Valeur	Attributs	Description
7	rwx	procure des droits en lecture, écriture et en exécution.
6	rw-	procure des droits en lecture et en écriture.
5	r-x	procure des droits en lecture et en exécution.
4	r--	procure des droits en lecture seule.
3	-wx	procure des droits en écriture et en exécution.
2	-w-	procure des droits en écriture seule.
1	--x	procure des droits en exécution seule.
0	---	Aucune permission n'est accordée.

13 / Les algorithmes

13.1 / Notions

Les types

- **entier** : -limite_système, -124, -45, -2, 0, 14, 78, 325, limite_système.
- **réel** : -limite_système, -43E+541, -15.62, -0.08502, 0.0, 25.36, 9778.03, 759E-95, limite_système.
- **caractère** : caractères Unicode : [a-zA-Z0-9] et n'importe quels caractères accentués (é, à, ö, etc..) ou spéciaux ($, #, ?, etc..).
- **chaîne** : n'importe quelle suite de caractères "\tUne brebis égarée dans la forêt\n".
- **booléen** : vrai (true : 1) ou faux (false : 0)
- **énuméré** : représente une collection de constantes énumérées possibles pour une variable.

```
# Création #
  type mois : chaîne : ("janvier","février","mars",
                        "avril","mai","juin",
                        "juillet","août","septembre",
                        "octobre","novembre","décembre",);
  type num_mois : entier : (1,2,3,4,5,6,7,8,9,10,11,12);
```

- **intervalle** : représente un intervalle de valeur possible pour une variable.

```
# Création #
  type num_mois : entier : 1..12;
  type alphabet : caractère : 'A'..'Z';
  type été : mois : "juillet".."septembre";
```

Les variables

```
# Création #
var identificateur : type;

var identificateur : type := valeur_initialisatrice;

var i : entier;

var j : entier := 0;

# Affectation #
variable := valeur;

phrase := "Une chaîne de caractères...";
```

Les constantes

```
# Création #
const IDENTIFICATEUR : type := valeur_constante;

const PI : réel := 3.14;
```

Les expressions

```
variable : type := expression;

nom : chaîne := "Jean-Pierre" + " " + "RODERER";

surface : réel := PI * Rayon ** 2;

reussir : booléen := vrai et faux ou vrai et non faux;
```

Les opérateurs arithmétiques

1. + (unaire), - (unaire),
2. ** (puissance),
3. *, /, div (division d'entier), modulo,
4. + (binaire), - (binaire).

```
var resultat : réel := 4 * -5 / -2 + 20 - 5 ** 3 / 10;
resultat := ((((4 * (-5)) / (-2)) + 20) - ((5 ** 3) / 10));
resultat := (((-20 / -2) + 20) - (125 / 10));
```

```
resultat := 30 - 12.5;
resultat := 17.5;
```

Les opérateurs booléens

1. non
2. et
3. ou

```
non Exp1 et Exp2 ou Exp3 et non Exp4 ou Exp5
((((non Exp1) et Exp2) ou (Exp3 et (non Exp4))) ou Exp5)
```

Les opérateurs de comparaisons

- = : égalité,
- <> : différence,
- >= : supérieur ou égal,
- <= : inférieur ou égal,
- > : supérieur,
- < : inférieur.

```
variable : booléen := expression_comparative;

réussite : booléen := i < 10;

réussite : booléen := (i < 10) et (j <> 0);

L'opérateur de concaténation
var resultat : chaîne := chaîne + chaîne2 + caractère + ... + chaîneN;

resultat : chaîne := "Bonjour " + 'à' + " tous";
resultat := "Bonjour à tous";
```

Les opérateurs sont associatifs à gauche hormis l'opérateur de puissance **.

La mise entre paranthèses d'une expression contraint à une évaluation prioritaire de cette dernière.

Les procédures

```
# Création #
proc identificateur[([val param_val: type,...,param_valN: type][;
                      valres param_val_res: type,...,param_val_resN: type][;
                      res param_res: type,...,param_résN: type])];
 [variable_locale : type := valeur;
  ...
  variable_localeN : type := valeur;]

  début
  Instructions...
fproc

proc compteur;
  var cpt : entier := 0;
     i : entier;
  début
  pour i := 0 jusqu'à 10 faire
     cpt := cpt + 1;
     écrire(i, " ", cpt);
  fpour;
fproc

proc tri(val nb : entier; valres un_tableau :
                              tableau[1..MAX]; res reussie : boolean)
```

Les fonctions

```
# Création #
fonction identificateur([paramètre: type,...,paramètreN: type]): type;
 [variable_locale : type := valeur;
  ...
  variable_localeN : type := valeur;]
```

```
  début
  Instructions...
  résultat valeur;
ffonction

fonction aire_cercle(diamètre : réel) : réel;
  var PI : réel := 3.14;

  début
  aire : réel := (PI * (diamètre ** 2)) / 2;
  resultat aire;
ffonction

# Appel de fonction #
variable : type_fonction :=
                identificateur_fonction([argument,..., argumentN]);

surface : réel := aire_cercle(15);
```

Les structures

```
# Création #
type identificateur : struct
  variable : type,
  ...,
  variableN : type
  fstruct;
...
  identificateurN struct
  variable : type,
  ...,
  variableN : type
  fstruct;

type personnel : struct
  identifiant : entier,
  nom, prénom : chaîne,
  date_naissance : chaîne(10),
  adresse : chaîne,
  code_postal : entier,
  ville : chaîne,
  pays : chaîne,
  telephone : entier
  fstruct;

  société : struct
  num_siret : entier,
  nom : chaîne,
  dirigeant : personnel,
  nb_personnel : entier
  fstruct;

# Déclaration de variable structure #
var nom_variable : identificateur_structure;

var dirigeant : personnel;

var entreprise : société;

# Affectation #
identificateur_structure.nom_champs := valeur;

entreprise.dirigeant.nom := "MAGNARD";

entreprise.nom := "ALAPAGE";

entreprise.num_siret := 41426553800010;

# Lecture #
variable : type_champs := identificateur_structure.nom_champs;

pdg : chaîne := entreprise.dirigeant.nom
          + " " + entreprise.dirigeant.prénom

# Utilisation avec avec #
```

```
avec identificateur_structure faire
  nom_champs := valeur;
  variable : type_champs := nom_champs;
favec;

avec entreprise faire
  nom := "ALAPAGE";
  dirigeant.pays := "France";

  avec dirigeant faire
  nom := "MAGNARD";
  prénom := "Patrice";
  favec;
favec;
```

Les tableaux

```
# Création #
identificateur : tableau[début..fin{, ..., débutN..finN}] de type

un_tableau : tableau[1..10,1..10] de entier;
#Création d'un tableau bidimensionnel de 100 cellules#
# Affectation #
tableau[index{, indexN}] := valeur;

un_tableau[1,1] := 12;

# Lecture #
tableau[index{, indexN}] := valeur;

type_tableau : variable := un_tableau[1,1] := 12;
```

Ecriture à l'écran

```
écrire(valeur);

écrire("Notions d'algorithme");
```

La structure conditionnelle *si...alors...sinon*

```
si expression_booléenne
alors instructions...;
[sinon si seconde_expression_booléenne
alors instructions...;]
[...]
[sinon si Nième_expression_booléenne
alors instructions...;]
[sinon Instructions...;]
fsi

si variable < 10 alors
  écrire("La variable est supérieure à 10 :", variable);
sinon si variable > 10 alors
  écrire("La variable est inférieure à 10 :", variable););
sinon si variable = 10 alors
  écrire("La variable est égale à 10 :", variable););
fsi
```

La structure conditionnelle *choix...cas*

```
choix
  expression_booléenne -> cas
  Instructions...;
  fcas
  ...
  expression_booléenneN -> cas
  Instructions...;
  fcas
fchoix;

choix
  mois = "janvier"
  ou mois = "février"
```

```
  ou mois = "mars" -> cas;
  écrire("Nous sommes dans la saison hivernale.");
  fcas
  mois = "avril"
  ou mois = "mai"
  ou mois = "juin" -> cas;
  écrire("Nous sommes dans la saison du printanière.");
  fcas
  mois = "juillet"
  ou mois = "août"
  ou mois = "septembre" -> cas;
  écrire("Nous sommes dans la saison estivale.");
  fcas
  mois = "octobre"
  ou mois = "novembre"
  ou mois = "décembre" -> cas;
  écrire("Nous sommes dans la saison automnale.");
  fcas
fchoix;
```

La structure itérative *répéter...jusqu'à*

```
répéter
  Instructions...
jusqu'à expression_booléenne

i : entier := 0;
répéter
  écrire("Contenu cellule ", i, " : ", tableau[i], "\n");
  i := i + 1;
jusqu'à i >= tableau.longueur()
```

La structure itérative *tant que...*

```
tant que expression_booléenne
  Instructions...
fin tant que

i : entier := 0;
tant que i < tableau.longueur()
  écrire("Contenu cellule ", i, " : ", tableau[i], "\n");
  i := i + 1;
fin tant que
```

La structure itérative *pour...*

```
pour expression_d'initialisation
[1(défaut) | pas expression_d'incrémentation]
jusqu'à expression_finale
faire
  Instructions...
fin pour

num : entier := 0;
pour i := 0 pas 2 jusqu'à 10 faire
  num := num + 1;
  écrire("Numéro de boucle et valeur de i : ", num, " et ", i);
fin pour
```

Lecture des saisies au clavier

```
lire(variable);

lire(nom);

algorithme
  var nom, prénom : chaîne;
début
  écrire("Saisissez vos nom et prénom ",
              "séparés par une virgule : ");
  lire(nom, prénom);
  écrire("Bienvenue dans ce programme ",
               prénom, " ", nom);
```

```
fin
```

13.2 / Création

Déclaration d'algorithme

```
# Création #
algorithme
  déclaration des objets
début
  instructions...
fin
```

Précondition et Postcondition

Une précondition représente les états initiaux possibles des objets énoncés dans un problème à résoudre par un algorithme.
Tandis qu'une postcondition représente les états finaux de ces mêmes objets.

```
P : prédicat_préconditionnel,
Q : prédicat_postconditionnel

# a et b sont des chaînes à comparer lexicographiquement #

P: a <> b,
Q: c = max(a, b)

# signifie que a et b sont strictement différents et qu'il faut écrire un algorithme pour déterminer le plus
grand d'entre eux. #

(c = a et c <> b) ou (c = b et c <> a)

algorithme
var a, b, c : chaîne;

début
  si a > b alors
  c := a;
  sinon si b > a alors
  c := b;
  fsi;
fin

P: a >= b
Q: c = max(a, b)

algorithme
var a, b, c : chaîne;

(c = a et c >= b) ou (c = b et c < a)


début
  si a >= b alors
  c := a;
  sinon
  c := b;
  fsi;
fin

P: n > 2,
Q: factorielle = f(n)

(f = 1 et n <= 2) ou (f = x et n > 2)

fonction factorielle(n : entier) : entier;
  entier : i, f := 1;

début
  pour i := 2 jusqu'à n faire
  f = f * i;
  fpour

  resultat f;
```

ffonction

13.3 / Les fonctions prédéfinies

Fonctions et procédures relatives aux chaînes de caractères

```
fonction longueur(val chaîne_source : chaîne) : entier;

fonction morceau(val var_chaîne : chaîne,
                        pos_début, pos_fin : entier) : chaîne;

fonction index(val chaîne_source,
                    chaîne_recherchée : chaîne) : entier;

proc insérer(val chaîne_inseree : chaîne, position : entier,
                valres chaîne_source : chaîne);

proc effacer(val pos_début, pos_fin : entier,
                valres chaîne_source : chaîne);

# Exemples #
chaîne_source : chaîne := "Microsoft ajoutera un outil "
                            + "XML à sa suite Office en 2003.";
taille : entier := longueur(chaîne_source);
taille := 58;

insérer("peut-être ", 19, chaîne_source);
chaîne_source : chaîne := "Microsoft ajoutera peut-être un outil "
                            + "XML à sa suite Office en 2003.";
```

Fonctions et procédures relatives aux tableaux

	TABLEAU symbolisant la relation PERSONNEL		
	id	NOM	PRENOM
L T	001357	LECLERC	Aurore
I U	1013056	CONTINE	Magalie
G o p	1036412	ASUR	Aurianne
N u L	1055848	ALHEUR	Jean-Christophe
E E	1103256	PATTRON	Delphine
S S	1123480	MILLAUD	Marc

C O L O N N E S ou C H A M P S

```
fonction compter_ligne(val rel : relation) : entier;

fonction énumérer_ligne(val rel : relation) : entier;

fonction énumération_terminée(val numéro_énumération : entier)
                                                    : booléen;

proc fermer_énumération(val numéro_énumération : entier);

fonction ligne_suivante(val numéro_énumération : entier) : ligne;

proc ajouter_ligne(val rel : relation, ligne : t_ligne,
                    res ok : booléen);

proc supprimer_ligne(val rel : relation, clé : t_clé,
                     res ok: booléen);

proc modifier_ligne(val rel : relation, ligne : t_ligne,
                   res ok booléen);

proc rechercher_ligne(val rel : relation, clé : t_clé,
                    res ligne : t_ligne, trouvé : booléen);

# Exemples #
```

```
var num_lignes : entier := compter_ligne(PERSONNEL);

ajouter_ligne(PERSONNEL, "1032578 MOULET Isabelle", reussi);

rechercher_ligne(PERSONNEL, 1036412, enregistrement, existe);

algorithme
 type opération : (1, 2, 3);
 type tuple : struct
 ID : entier,
 NOM : chaîne,
 PRENOM : chaîne
 fstruct;

 var action : opération;
      tableau : relation;
      ligne : tuple;
      ident : entier;
      nom, prénom : chaîne;

 proc affichage(val rel : relation);
 var i : entier := 0;

 début
 i := énumérer_ligne(rel);
 écrire("\tID \tNom \tPrénom");
 tant que non(énumération_terminée(i))
    ligne := tuple_suivant(i);
    avec ligne faire
      écrire("\t", ID, "\t", NOM, "\t", PRENOM);
    favec;
 fin tant que;
 fin

 proc modification(rel : relation);
 var réussi : boolean;

 début
 choix
    action = 1 -> cas
      écrire("fournissez les données (ID, NOM, PRENOM) :");
      lire(ident, nom, prénom);
      avec ligne faire
      lire(ID, NOM, PRENOM);
      favec;
      ajouter_ligne(rel, ligne, réussi);
      si réussi alors écrire("Ajout réussi !");
      sinon écrire("Echec de l'ajout !");
      fsi
    fcas
    action = 2 -> cas
      écrire("fournissez la clé de l'enregistrement (ID) :");
      lire(ident);
      supprimer_ligne(rel, ident, réussi);
      si réussi alors écrire("Suppression réussie !");
      sinon écrire("Echec de la suppression !");
      fsi
    fcas
    action = 3 -> cas
      écrire("fournissez les données (ID, NOM, PRENOM) :");
      avec ligne faire
      lire(ID, NOM, PRENOM);
      favec;
      modifier_ligne(rel, ligne, réussi);
      si réussi alors écrire("Modification réussie !");
      sinon écrire("Echec de la modification !");
      fsi
    fcas
 fchoix
 fin

 proc recherche(val rel : relation)
 var réussi : boolean;

 début
```

```
  écrire("fournissez la clé de l'enregistrement (ID):");
  lire(ident);
  rechercher_ligne(rel, ident, ligne, reussi);
  si réussi alors
     écrire("La recherche a réussi :\n");
     écrire(ligne);
  sinon écrire("La recherche a échoué !");
  fsi
  fin

début
  tableau := PERSONNEL;
  écrire("1 - Ajout d'un enregistrement,\n",
         "2 - Modification du tableau,\n",
         "3 - Recherche d'un enregistrement.\",
         "choix : ");
  lire(action);
  choix
  action = 1 -> cas
     affichage(tableau);
  fcas
  action = 2 -> cas
     modification(tableau);
  fcas
  action = 3 -> cas
     recherche(tableau);
  fcas
  fchoix
fin
```

13.4 / Les types abstraits

Les types abstraits introduisent la notion de méthodologie objet dans les algorithmes.

Un type abstrait est constitué de plusieurs constantes, variables et méthodes (fonctions ou procédures) dont certaines constituent une interface par laquelle un objet pourra être manipulé sans pour autant connaître sa mise en oeuvre (ou implémentation).

```
# Création d'un type abstrait #
type abstrait : identificateur
 interface
 # Déclaration des méthodes #
 {fonction* | proc} nom_méthode([arguments]){: type*};
 ...
 {fonction* | proc} nom_méthodeN([arguments]){: type*};

 représentation
 # Déclaration des variables et constantes #
 {var | const} nom : type;
 ...
 {var | const} nomN : type;

 algorithmes
 # Définitions des méthodes #
 {fonction* | proc} nom_méthode([arguments]){: type*};
    ...
 début
    ...
 fin
 ...
 {fonction* | proc} nom_méthodeN([arguments]){: type*};
    ...
 début
    ...
 fin

ftypeabstrait

# Création d'un objet #
var identificateur_objet : identificateur_type_abstrait;

# Utilisation des méthodes #
identificateur_objet.nom_méthode([arguments]);
```

Utilisation des variables et constantes # Une variable ou une constante déclarée dans *représentation* peut être accédé dans toutes les méthodes du type abstrait.
Les variables peuvent également être modifiées à partir du bloc d'instructions de ces mêmes méthodes.

```
# Exemple #
type abstrait : Calculatrice
 interface
 proc saisie(res valA, valB : réel, valOP : opération);
 proc calcul(val valA, valB : réel, valOP : opération, res resultat: réel);
 proc affichage(val valA, valB, resultat : réel, valOP : opération);

 représentation
 type opération : énumération : ('+','-','*','/');
 var valA, valB : réel, valOP : opération

 algorithmes
 proc saisie();
 début
    écrire("Entrez un calcul simple
                [Nombre, (+ | - | * | /), Nombre (ex.:10.2, *, 5)] :");
    lire(valA, valOP, valB);
 fproc

 proc calcul(res result: réel);
 début
    choix
```

```
        type_calcul = '+' -> cas
        result := addition(valA, valB);
        fcas
        type_calcul = '-' -> cas
        result := soustraction(valA, valB);
        fcas
        type_calcul = '*' -> cas
        result := multiplication(valA, valB);
        fcas
        type_calcul = '/' -> cas
        result := division(valA, valB);
        fcas
      fchoix
  fproc

  proc affichage(val result : réel);
  début
     écrire("Opération : ", valA, " ", valOP, " ", valB, " = ", result);
  fproc

  fonction addition(val a, b) : réel;
  début
     résultat (a + b);
  ffonction

  fonction soustraction(val a, b) : réel;
  début
     résultat (a - b);
  ffonction

  fonction multiplication(val a, b) : réel;
  début
     résultat (a * b);
  ffonction

  fonction division(val a, b) : réel;
  début
     résultat (a / b);
  ffonction
ftypeabstrait

algorithme
  var calc : Calculatrice;
        valeur_resultat : réel;
début
  calc.saisie();
  calc.calcul(valeur_resultat);
  calc.affichage(valeur_resultat);
fin
```

14 / La programmation orientée objet

La programmation a évolué selon plusieurs étapes successives, en passant de l'assembleur à des langages spécialisés tels que *FORTRAN* (1957) et *COBOL* (1959), puis aux langages procéduraux comme *BASIC* (1964), *PASCAL* (1968) et *C* (1973) et enfin vers des langages orientées objets à l'image du *C++* (1983), *Java* (1995) et *C#* (2000).

Une des étapes principales fut la programmation structurée, appelée également procédurale ou impérative, dont le principe fondamental était la décmposition d'un programme en de multiples sous-programmes ou modules effectuant chacun une certaine action non élémentaire, et partant, permettant de décomplexifier une application de grande taille.

Les langages orientés objets sont une nouvelle méthode de programmation qui tend à se rapprocher de notre manière naturelle d'appréhender le monde. Les L.O.O. se sont surtout posé la question "Sur quoi porte le programme ?". En effet, un programme informatique comporte toujours des traitements, mais aussi et surtout des données. Si la programmation structurée s'intéresse aux traitements puis aux données, la conception objet s'intéresse d'abord aux données, auxquelles elle associe ensuite les traitements. L'expérience a montré que les données sont ce qu'il y a de plus stable dans la vie d'un programme, il est donc intéressant d'architecturer le programme autour de ces données.

La programmation orientée objet se différencie radicalement de celle fondée sur des procédures. La programmation procédurale (ou impérative) met en oeuvre des fonctionnalités écrites sous la forme d'une liste d'instructions à exécuter progressivement, soit les unes après les autres et éventuellement avec des appels de procédures ou de fonctions effectuant chacune une certaine action non élémentaire.

La programmation orientée objet (POO) est basée sur la représentation par des objets, des entités définies par l'étude du problème à réaliser.

La POO décompose un programme en de multiples classes, soit des modules autonomes accomplissant certaines tâches. Ces classes permettent de créer des objets par l'instanciation de ces classes.

Les procédures ou fonctions effectuent. En général, une telle action ne peut pas se programmer de la même façon pour tous les types de données (ou objets).

La POO permet de développer des applications à partir d'objets, c'est-à-dire, d'entités possédant des propriétés, des méthodes et des événements, ces objets pouvant être une page Web, un formulaire, un tableau, une chaîne de caractères, un nombre, une image, etc..

Les propriétés sont des méthodes fournissant un accès à des valeurs simulant celles des champs. **Les propriétés (ou attributs)d'un objet contiennent les valeurs relatives à cet objet**, telles que la taille d'un tableau, le type de données d'un nombre, la contenu d'une chaîne de caractères, etc..
Les propriétés peuvent être simplement lues pour une affectation ou pour un affichage, ainsi que modifiées pour un paramétrage de l'objet à certaines valeurs.

Les méthodes d'un objet représentent les opérations que cet objet peut accomplir comme l'ajout d'une valeur dans un tableau, la concaténation d'une chaîne de caractères, la conversion du type de données d'un nombre.

Les événements d'un objet consistent en la capture d'une action sur cet objet, à l'image d'un clic de souris sur un bouton, la soumission d'un formulaire, le chargement et déchargement d'une page Web.

Les objets sont issus d'une instanciation de classe, ces dernières lui fournissant des propriétés, des méthodes et des événements, soit les membres de cette classe.

Une classe est un type référence encapsulant des données (champs et constantes), ainsi qu'un comportement (méthodes, propriétés, indexeurs, événements, opérateurs, constructeurs et destructeurs), pouvant contenir des types imbriqués.

Les champs représentent des variables contenant des valeurs.

Les constantes représentent des champs contenant des valeurs fixes.

Les indexateurs permettent d'indexer un objet par l'intermédiaire des méthodes d'accès *get* et *set*.

Les opérateurs permettent d'inclure les opérations mathématiques de base dans les classes par

l'intermédiaire de la surcharge des opérateurs.

14.1 / Les concepts fondamentaux de la POO

Les langages orientés objets assimilent trois concepts fondamentaux :

1. l'encapsulation,
2. l'héritage,
3. et le polymorrphisme.

Encapsulation

L'encapsulation consiste en la capacité que possède un objet, de masquer certains de ses éléments tout en fournissant un accès à d'autres éléments constituant l'interface utilisateur.

L'encapsulation permet donc de cacher des propriétés et des méthodes internes à l'utilisateur, et autorise ce dernier à accéder aux membres prévus de l'objet par le programme.

Les méthodes et les propriétés d'un objet peuvent être divisées en membres visibles nécessaires au paramètrage et aux opérations de l'objet exploité par l'utilisateur, et en membres invisibles participant à l'implémentation de l'objet.

L'utilisation des modificateurs d'accès permet d'appliquer les principes d'encapsulation, en définissant des niveaux de visibilité (privé, public, protégé) des membres d'une classe.

Abstraction

L'abstraction permet d'accroître la concentration des programmeurs sur la modélisation d'un problème par du code plutôt que sur la manière dont les résultats seront obtenus.

Un nom clair, exprimé en langage naturel, pour une classe permettra de connaître avec certitude quelles fonctionnalités accomplissent cette classe.

Ainsi, l'utilisation d'une classe sera d'autant plus intuitive si son nom est explicite, tout comme ceux de ses interfaces et de ses membres.

Héritage

L'héritage consiste en la création d'une classe à partir d'une classe existante, permettant de construire une hiérarchie de classes.

La nouvelle classe hérite automatiquement de l'ensemble des membres de la classe de base et par conséquent, de toutes les fonctionnalités de cette dernière.

Polymorphisme

Le polymorphisme caractérise la possibilité de définir plusieurs méthodes avec un nom identique, mais des paramètres dont le types références et l'ordre sont différents. La méthode attendue est, alors, choisie en fonction des arguments fournis lors de son appel.

Le polymorsphisme permet d'appeler précisément la méthode de la classe instanciant l'objet en cours, dans une hiérarchie de classes. Ainsi, un objet issu d'une classe dérivée pourra appeler la méthode de cette dernière, tout en laissant de côté les autres méthodes de même nom dans la hiérarchie de classes du programme.

Le polymorphisme constitue, également, un moyen d'extension d'un programme sans nécessiter la modification du code existant.

15 / Installation sur Windows du Trio PHP/MySQL/PHPMyAdmin

Le couple PHP/MySQL associe un langage de script multi-plateformes à un système de base de données efficace, dans le but de créer des applications Web dynamiques. PHPMyAdmin est une interface graphique écrite en PHP et destinée à la manipulation et à la gestion de base de données MySQL.

phpMyAdmin possède les fonctions suivantes :

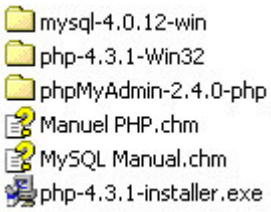
- Création et suppression des bases de données,
- Gestion des privilèges d'accès aux bases de données,
- Ajout, édition et suppression d'enregistrements et de champs
- Exécution de requêtes SQL,
- Gestion des clés et index sur les données,
- Optimisation des tables de données,
- Importation ou exportation des données à partir d'archives compressées ou de fichiers texte, CVS ou XML
- Administration d'un ou plusieurs serveurs de base de données et d'uniques base de données

En premier lieu, il faut télécharger les dernières versions stables de ces outils.

- PHP : http://www.php.net/downloads.php
- MySQL : http://www.mysql.com/downloads/index.html
- PHPMyAdmin : http://www.phpmyadmin.net/

En ce qui concerne PHP, il est plus simple de télécharger la version *PHP XXX installer* permettant une installation automatisée à l'aide d'un exécutable *.exe* et également la version compressée au format ZIP contenant plusieurs fichiers importants.

Une fois en possession de ces derniers, une décompression des fichiers s'impose afin d'obtenir les fichiers installables.



A ce stade, vous lancez l'application *PHP XXX installer*, en repérant le répertoire d'installation, en général *C:\PHP*, dans lequel vous collerez ensuite, le contenu du dossier *php-XXX-Win32*.

L'installation de MySQL ne nécessite qu'un simple double-clic sur le fichier *setup.exe* présent dans le répertoire *mysql-XXX-win*.

Le contenu du dossier *phpMyAdmin-XXX* doit être recopié à la racine du site Web par défaut, soit dans le répertoire *www* ou *wwwroot*. Par ailleurs, il peut être préférable de renommer le dossier *phpMyAdmin-XXX* en *phpMyAdmin*, permettant un accès simple et court à l'interface d'administration (*http://www.site.net/phpmyadmin/*).

Désormais, il faut configurer l'ensemble de ces outils afin que chacun fonctionne dans la plénitude de leurs moyens et se reconnaisse sans problème.

Dans un premier temps, il faut copier tous les fichiers *.dll* et autres présents dans les répetoires *c:\php\dlls* et *c:\php\extensions* dans le dossier *system32* ou *system* (si le premier n'existe pas) de Windows.

L'édition par le bloc-notes du fichier *php.ini* localisé dans le répertoire de Windows, en général *c:\windows* ou encore *c:\winnt*, doit permettre de configurer les extensions PHP.

```
;Extrait du fichier php.ini
;Windows Extensions
;Note that MySQL and ODBC support is now built in,
;so no dll is needed for it.
;
```

```
extension=php_bz2.dll
extension=php_cpdf.dll
extension=php_crack.dll
extension=php_curl.dll
extension=php_db.dll
extension=php_dba.dll
extension=php_dbase.dll
extension=php_dbx.dll
extension=php_domxml.dll
extension=php_exif.dll
;extension=php_fbsql.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
extension=php_gd.dll
;extension=php_gd2.dll
extension=php_gettext.dll
;extension=php_hyperwave.dll
extension=php_iconv.dll
;extension=php_ifx.dll
;extension=php_iisfunc.dll
extension=php_imap.dll
;extension=php_interbase.dll
extension=php_java.dll
extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_mcrypt.dll
extension=php_mhash.dll
extension=php_mime_magic.dll
extension=php_ming.dll
extension=php_mssql.dll
extension=php_msql.dll
;extension=php_oci8.dll
extension=php_openssl.dll
;extension=php_oracle.dll
extension=php_pdf.dll
extension=php_pgsql.dll
extension=php_printer.dll
extension=php_shmop.dll
extension=php_snmp.dll
extension=php_sockets.dll
;extension=php_sybase_ct.dll
extension=php_w32api.dll
extension=php_xmlrpc.dll
extension=php_xslt.dll
extension=php_yaz.dll
extension=php_zip.dll
```

Le point-virgule permet de désactiver l'extension attenante. Chacune des extensions actives dans PHP doit obligatoirement se trouver dans le répertoire *c:\php\extensions*, sinon un message d'erreur apparaitra lors du lancement de PHP.

La vérification de la bonne installation des extensions PHP, peut s'effectuer par l'intermédiaire de l'instruction *phpinfo()*.

Pour cela, il suffit de créer un document PHP, lequel sera dénommé *phpinfo.php* et enregitré sous la racine de votre site Web par défaut.

```
<?php
  phpinfo();
?>
```

Le chargement de ce fichier affichera toutes les informations relatives à PHP. Si des erreurs d'installation subsistent, des messages d'erreurs seront affichés.

Si votre système utilise le kit de développement Java (JDK), vous devrez modifier en conséquence le fichier *php.ini*.

```
;Extrait du fichier php.ini
[Java]
java.class.path = .\php_java.jar
java.home = C:\j2sdk
java.library = C:\j2sdk\jre\bin\hotspot\jvm.dll
java.library.path = .\
```

Vous pouvez de même modifier la variable *register_globals* afin d'autoriser l'utilisation des variables globales.

```
register_globals = On
```

De même, plusieurs autres domaines peuvent être configurés dans ce fichier de configuration de PHP, tels que les chemins et répertoires (*Paths and Directories*), les sessions, le téléchargement de fichiers (*File Uploads*), etc..

Les accès aux bases de données peuvent également être configurés dans le fichier php.ini, mais cela n'est pas recommandé pour des raisons de sécurité.

La configuration de PHPMyAdmin s'effectue par l'entremise du fichier *config.inc.php* situé dans le répertoire *phpMyAdmin*.

```
//Extrait du fichier config.inc.php
...
$cfg['Servers'][$i]['host']          = 'localhost';
// MySQL hostname
$cfg['Servers'][$i]['port']          = '';
// MySQL port - leave blank for default port
...
$cfg['Servers'][$i]['auth_type']     = 'config';
// Authentication method (config, http or cookie based)?
$cfg['Servers'][$i]['user']          = 'root';
// MySQL user
$cfg['Servers'][$i]['password']      = '';
// MySQL password (only needed with 'config' auth_type)
...
```

Lors de l'installation de MySQL, un compte par défaut est automatiquement créé, il s'agît de l'utilisateur *root* sans mot de passe.

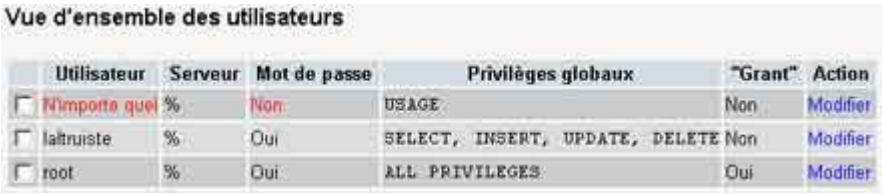
Donc, il est possible d'ouvrir l'interface d'administration PHPMyAdmin avec la configuration par défaut, mais cela devra être immédiatement rectifié pour des raisons de sécurité évidentes.

L'ouverture de l'interface s'accomplit en lançant son navigateur Web, puis en saisissant dans la barre d'adresse le chemin adéquat.

```
http://www.site.com/phpMyAdmin/index.php
ou
http://localhost/phpMyAdmin/index.php
```

A partir de là, vous cliquez sur le lien *Privilèges* présent sur la page de droite.

La table *user* contient les comptes d'utilisateurs du système de gestion de base de données de MySQL.



Modifiez le compte d'utilisateur *root* en spécifiant un mot de passe dans les champs appropriés.

En rechargeant PHPMyAdmin, on pourra remarquer que l'accès à l'interface graphique est refusée puisque la configuration initiale du fichier *config.inc.php* n'est plus valable.

Dans ce cas, il suffit d'éditer le fichier précité et d'affecter le mot de passe correspondant à la variable *$cfg['Servers'][$i]['password']*.

16 / Création de fichiers PDF

Le documents PDF (Portable Document Format) possèdent la particularité de pouvoir être lus indépendamment de toutes plateformes (Windows, Mac, Linux, etc.) tout en conservant rigoureusement la mise en page.

Néanmoins, **le format PDF a été conçu par l'éditeur de logiciels graphiques, Adobe** (Acrobat Reader, Photoshop, Premiere, Golive, etc.) et demeure donc **un format propriétaire** que la plupart des logiciels d'édition ne gère pas.

Heureusement, il n'est pas utile de s'équiper du très coûteux logiciel, Adobe Acrobat. **De simples utilitaires souvent gratuits où d'un coût modique suffiront à créer des documents PDF à l'allure professionnelle.**

Il est possible d'opter pour différentes solutions, telles que le couple Ghostview et Ghostscript (Windows, Mac, Linux), HTMLDOC (Windows, Linux), PDF995 (Windows) ou encore d'autres.

Dans un premier temps, il vous faudra **télécharger les logiciels choisis et les installer** sur votre configuration.

- En ce qui concerne Windows, il suffit de **lancer les fichiers exécutables et suivre la procédure d'installation**.
- Pour Linux, vous avez parfois le choix entre des **fichiers *.rpm*** se comportant comme des fichiers exécutables Windows ou les **archives *.tar.gz***.

```
Décompression de l'archive Archive.source.tar.gz
% tar xzf Archive-source.tar.gz
Installation du logiciel
% cd Archive
% ./configure
% make
```

- Au niveau du Mac, il suffit de **double-cliquez sur les fichiers d'installation**.

Il est souhaitable également de **télécharger et installer l'utilitaire Adobe Acrobat reader** afin de vérifier d'une manière optimum vos travaux.

Maintenant, il est nécessaire d'**installer un pilote d'impression postscript** sur votre configuration Windows. En principe, la plupart des distributions Linux, installe directement un pilote d'impression Postscript, voire même un pilote PDF.

- Cliquez sur le menu *Démarrer -> Paramètres -> Imprimantes*,
- Double-cliquez sur *Ajout d'imprimante*,
- A l'ouverture de l'*Assistant Ajout d'imprimante*, cliquez sur *Suivant*,
- Si besoin est, cochez *Imprimante locale*, puis cliquez sur *Suivant*,
- Sélectionnez le port imprimante, en ganéral LTP1, puis cliquez sur *Suivant*,
- Sélectionnez une imprimante Laser Postscrit de préférence comme l'*HP Color LaserJet 8500 PS*, puis cliquez sur *Suivant*,
- Donnez un nom à l'imprimante, puis cliquez sur *Suivant*,
- Donnez éventuellement un nom de partage à cette imprimante, puis cliquez sur *Suivant*,
- Cliquez sur *Suivant* jusqu'à la fin de l'*Assistant Ajout d'imprimante*, et enfin cliquez sur *Terminer* pour valider l'installation.

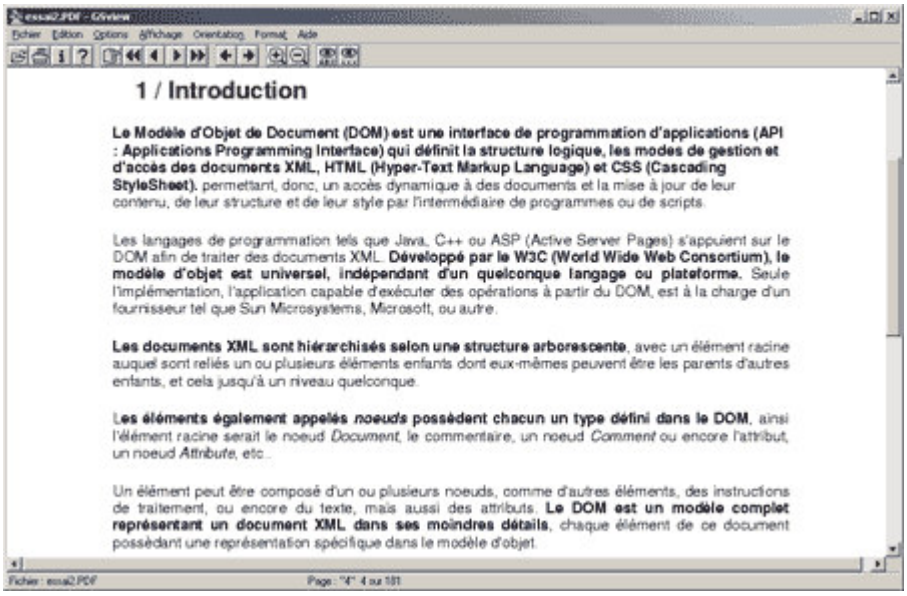
Pour **vérifier le bon fonctionnement de cet installation**, ouvrez un document textuel ou une page html quelconque, puis cliquez sur *Fichier -> Imprimer*. Sélectionnez l'imprimante Postscript (par ex.: *HP Color LaserJet 8500 PS*) et validez l'impression. Normalement une boîte de dialogue devrait vous demander de saisir le nom du fichier de sortie.

Lorsque vous imprimez dans un tel fichier, **l'extension de ce dernier doit donc être généralement *.ps***.

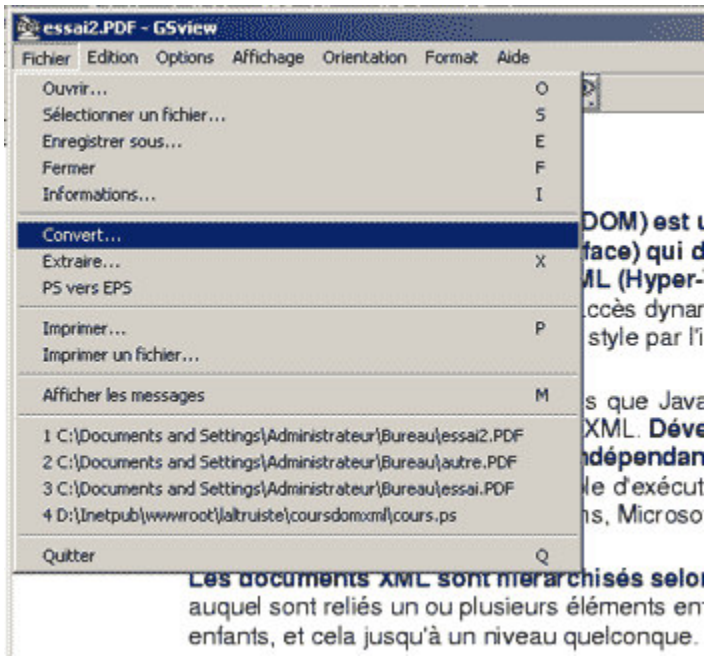
```
nom_du_fichier_dimpression.ps
```

A partir de ce genre de fichier, **Ghostview peut créer très facilement un fichier PDF**. Il suffit d'ouvrir le fichier Postscript avec le logiciel, puis de le convertir en cliquant sur *File -> Convert...*. Dans la boîte de dialogue, il faut choisir le type *pdfwrite* et une résolution. Après la validation, le nom du fichier de sortie est

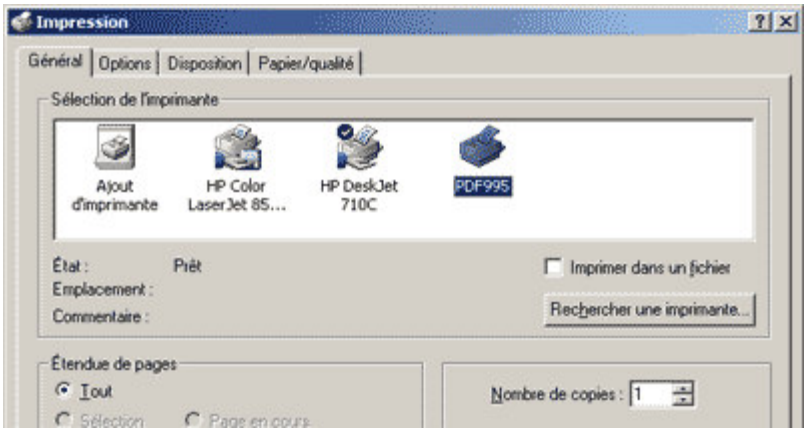
demandé avant d'obtenir un document PDF.



La génération d'un fichier postscript est possible à partir de n'importe quelle application en effectuant une impression vers un fichier. Ensuite **Ghostview se charge de convertir le document postscript dans le format PDF**.



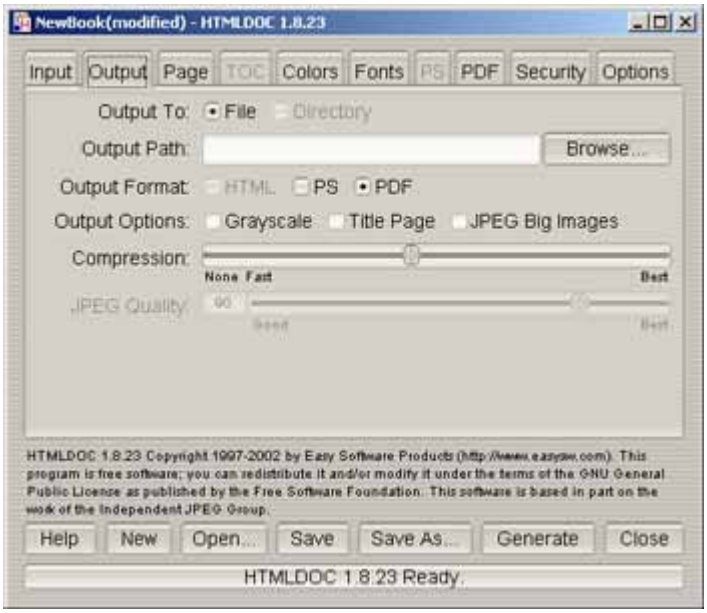
Avec PDF995, la procédure demeure semblable à l'exception du format du document résulant de l'impression vers un fichier. En effet, **PDF995 génére directement le fichier PDF** et donc ne nécessite pas de passer par une phase de conversion. Egalement, lors de l'installation du logiciel, **PDF995 installe sa propre imprimante postcript locale** portant d'ailleurs le nom de l'application.



Quant à HTMLDOC, la procédure est différente :

- Dans ce cas, il faut ouvrir le logiciel,
- Ajouter soit des fichiers, soit des adresses URL et sélectionner l'option *Web Page* à partir de l'onglet *Input*,

- Puis sélectionner le format PDF et saisir le nom du fichier de sortie à partir de l'onglet *Output*.



- Enfin, un clic sur *Generate* permet de générer le document PDF.

HTMLDOC autorise l'enchaînement de plusieurs documents distincts afin de ne produire qu'un seul et unique fichier PDF. De plus, il est le seul des trois logiciels à **créer automatiquement des liens** au sein des documents PDF résultants. En outre, diverses options permettent de régler plus précisément les marges, les couleurs, ainsi que l'encryptage.

Il reste donc simple et rapide de créer des documents PDF à partir de quasiment n'importe quelle source, puisqu'il suffit dans la plupart des cas d'imprimer vers un fichier postscript puis de convertir ce dernier dans le format PDF.

17 / Les lignes de commandes

Les lignes de commande utilisables aussi bien sur Windows que Linux ou d'autres systèmes d'exploitation, permettent de lancer des actions monotâches à partir de l'invite de commandes MS-DOS ou de la console Linux.

Chaque commande MS-DOS ou Linux propose une aide en ligne en utilisant respectivement les options /? ou --h et --help. Il est possible également d'utiliser les commandes spécifiques d'aide, c'est-à-dire *help* pour MS-DOS et *man* pour Linux.

Windows
mkdir /**?**
help shift

Linux
tty **--help**
man ls

17.1 / Les commandes MS-DOS

Les commandes MS-DOS permettent d'exécuter des tâches à partir de l'invite de commandes, parfois très utiles dans le cadre de développement avec des langages comme Java ou C#.

Commande	Description
ASSOC	Affiche ou modifie les applications associées aux extensions de fichiers.
AT	Planifie l'exécution de commandes ou programmes sur un ordinateur.
ATTRIB	Affiche ou modifie les attributs d'un fichier.
BREAK	Active ou désactive le contrôle étendu de CTRL+C.
CACLS	Affiche ou modifie les listes de contrôles d'accès aux fichiers.
CALL	Appelle un fichier de commandes depuis un autre fichier de commandes.
CD	Modifie le répertoire ou affiche le répertoire en cours.
CHCP	Modifie la page de code active ou affiche son numéro.
CHDIR	Modifie le répertoire ou affiche le nom du répertoire en cours.
CHKDSK	Vérifie un disque et affiche un relevé d'état.
CLS	Efface l'écran.
CMD	Lance une nouvelle instance de l'interpréteur de commandes de Windows 2000.
COLOR	Modifie les couleurs du premier plan et de l'arrière plan de la console.
COMP	Compare les contenus de deux fichiers ou groupes de fichiers.
COMPACT	Modifie ou affiche la compression des fichiers sur une partition NTFS.
CONVERT	Convertit des volumes FAT en volumes NTFS. Vous ne pouvez pas convertir le lecteur en cours d'utilisation.
COPY	Copie un ou plusieurs fichiers.
DATE	Affiche ou modifie la date.
DEL	Supprime un ou plusieurs fichiers.
DIR	Affiche la liste des fichiers et des sous-répertoires d'un répertoire.
DISKCOMP	Compare les contenus de deux disquettes.
DISKCOPY	Copie le contenu d'une disquette sur une autre.
DOSKEY	Modifie les lignes de commande, rappelle des commandes Windows 2000, et permet de créer des macros.
ECHO	Affiche des messages à l'écran ou active/désactive l'affichage des commandes.
ENDLOCAL	Stoppe la localisation des modifications de l'environnement dans un fichier de commandes.
ERASE	Supprime un ou plusieurs fichiers.
EXIT	Quitte l'interpréteur de commandes (CMD.EXE).
FC	Compare deux fichiers ou groupes de fichiers, et affiche les différences entre eux.
FIND	Cherche une chaîne de caractères dans un ou plusieurs fichiers.
FINDSTR	Cherche des chaînes de caractères dans un ou plusieurs fichiers.
FOR	Exécute une commande sur chaque fichier d'un groupe de fichiers.
FORMAT	Formate un disque pour utilisation avec Windows 2000.
FTYPE	Affiche ou modifie les types de fichiers utilisés dans les associations d'extensions.

GOTO	Poursuit l'exécution d'un fichier de commandes à une ligne identifiée par une étiquette.
GRAFTABL	Permet à Windows 2000 d'afficher un jeu de caractères en mode graphique.
HELP	Affiche des informations sur les commandes de Windows 2000.
IF	Effectue un traitement conditionnel dans un fichier de commandes.
LABEL	Crée, modifie ou supprime le nom de volume d'un disque.
MD	Crée un répertoire.
MKDIR	Crée un répertoire.
MODE	Configure un périphérique du système.
MORE	Affiche la sortie écran par écran.
MOVE	Déplace un ou plusieurs fichiers d'un répertoire à un autre.
PATH	Affiche ou définit le chemin de recherche des fichiers exécutables.
PAUSE	Interrompt l'exécution d'un fichier de commandes et affiche un message.
POPD	Restaure la valeur précédente du répertoire courant enregistré par PUSHD.
PRINT	Imprime un fichier texte.
PROMPT	Modifie l'invite de commande de Windows 2000.
PUSHD	Enregistre le répertoire courant puis le modifie.
RD	Supprime un répertoire.
RECOVER	Récupère l'information lisible d'un disque défectueux.
REM	Insère un commentaire dans un fichier de commandes ou CONFIG.SYS.
REN	Renomme un ou plusieurs fichiers.
RENAME	Renomme un ou plusieurs fichiers.
REPLACE	Remplace des fichiers.
RMDIR	Supprime un répertoire.
SET	Affiche, définit ou supprime des variables d'environnement Windows 2000.
SETLOCAL	Commence la localisation des changements de l'environnement dans un fichier de commandes.
SHIFT	Modifie la position des paramètres remplaçables dans un fichier de commandes.
SORT	Trie les éléments en entrée.
SUBST	Affecte une lettre de lecteur à un chemin d'accès.
START	Lance une fenêtre pour l'exécution du programme ou de la commande.
TIME	Affiche ou définit l'heure de l'horloge interne du système.
TITLE	Définit le titre de la fenêtre pour une session CMD.EXE.
TREE	Représente graphiquement l'arborescence d'un lecteur ou d'un chemin.
TYPE	Affiche le contenu d'un fichier texte.
VER	Affiche le numéro de version de Windows 2000.

VERIFY	Indique à Windows 2000 s'il doit ou non vérifier que les fichiers sont écrits correctement sur un disque donné.
VOL	Affiche le nom et le numéro de série du volume.
XCOPY	Copie des fichiers et des arborescences de répertoires.

17.2 / Les commandes Linux

Les lignes de commandes sont particulièrement importantes dans l'environnement Linux. Elles permettent d'exécuter des tâches parfois très utiles dans le cadre de développement d'application, de maintenance ou encore de configuration du système.

Commande	Description
alias	définit des abréviations pour les appels de commandes.
at	exécute une commande à un moment précis.
awk (gawk)	une implémentation GNU du langage *awk* permettant le traitement de fichiers.
banner	imprime une bannière (sortie de caractères en majuscule).
basename	extrait le nom de fichier d'un chemin d'accès.
bg	place un processus en arrière plan.
break	termine une boucle.
cal	affiche le calendrier.
case	une structure de contrôle à choix multiples.
cat	affiche le contenu d'un fichier.
cd	change de répertoire actif.
chgrp	change l'affectation de groupe pour des fichiers.
chmod	change les droits d'accès des fichiers.
chown	change le propriétaire d'un fichier.
chroot	change le répertoire racine pour l'exécution d'une commande.
cmp	compare deux fichiers.
continue	reprend une boucle interrompue avant son terme.
cp	copie des fichiers.
cpio	copie un fichier d'archive pour la sauvegarde.
crontab	exécute des commandes à intervalles réguliers.
cut	découpe des morceaux de lignes.
date	retourne et régle la date système.
dd	copie et convertit des données.
df	affiche l'espace disponible sur un support de données.
diff	détermine les différences entre les fichiers.
du	détermine l'espace disque utilisé.
echo	affiche une ligne de texte.
egrep	recherche à l'aide d'une expression régulière étendue.
env	modifie l'environnement d'une commande.
eval	évalue une commande shell.
exit	quitte le shell courant.
export	exporte les variables du shell.
expr	utilise et calcule des expressions.
false	exprime la valeur de retour standard des shelles scripts.

fc	rappelle une ligne de commande.
fg	place une commande d'arrière-plan au premier plan.
fgrep	recherche sans expression réguliere.
file	affiche le type de fichier.
find	recherche récursivement des fichiers.
for	ireprésente une structure de controle.
gcc	représente le compilateur C GNU.
grep	recherche avec des expressions régulières.
id	affiche des identificateurs d'utilisateurs et de groupes.
if	représente une condition dans un script shell.
jobs	affiche des processus d'arrière plan en cours.
join	joint deux fichiers.
kill	envoie un signal à un processus.
let	affectate arithmétiquement dans le shell.
ln	affecte un lien à un fichier.
logname	afficher le nom d'utilisateur.
lpq	détermine l'état des files d'attentes d'impression.
lpr	imprime des fichiers.
lprm	annule une requête d'impression.
ls	liste les fichiers d'un répertoire.
mail	lit et envoie des messages.
man	appelle de l'aide en ligne.
mesg	fournit des accès aux terminaux.
mkdir	crée un répertoire.
mknod	crée des fichiers de périphérique et de FIFOs.
more	affiche des fichiers et données page par page.
mv	déplace des fichiers.
newgrp	modifie l'appartenance à un groupe.
nice	lance une commande avec des priorités modifiés.
nohup	ignore les signaux dans le cadre d'une commande.
od	affiche des données dans le format interne.
passwd	modifie le mot de passe utilisateur.
pg	visualise les fichiers et les données page par page.
pr	formate des données et des fichiers.
ps	affiche des informations sur l'etat des processus en cours.

pwd	affiche le répertoire actif.
read	lit des valeurs.
readonly	protége des variables du shell contre la réécriture.
return	retourne une valeur à partir d'une fonction du shell.
rm	supprimr un fichier.
rmdir	supprime un répertoire.
sed	représente un éditeur de texte batch.
select	représente une sélection de menu simple dans le shell.
set	fixe des options et des paramètres de position.
shift	convertit des paramètres de position.
sleep	représente une interruption du traitement pendant un certain temps.
sort	trie des données et des fichiers ligne par ligne.
stty	configure une interface série.
su	change de numéro d'utilisateur.
sync	sauvegarde de la mémoire tampon d'entrées/sorties.
tail	affiche la fin d'un fichier ou d'un ensemble de données.
tar	sauvegarde et archive des fichiers.
tee	duplique un flux de données.
test	effectue un test sur une condition.
time	calcule la durée d'exécution d'une commande.
touch	modifie la date d'accés ou de modification.
tr	convertit des caractères.
trap	gère des réactions aux signaux.
true	représente la valeur standard pour un shell script.
tty	affiche le nom des terminaux.
typeset	modifie les valeurs d'attributs des variables du shell.
ulimit	fixe la taille maximale d'un fichier.
umask	définit des droits d'accès prédéfinis.
unalias	supprime un nom d'alias.
uname	demande le nom du système.
unset	supprime des définitions de variables et de fonctions.
until	représente une structure de contrôle de boucles.
vi	appelle l'éditeur orienté écran.
wait	temporise un processus en arrière-plan.
wall	envoie un message à tous les utilisateurs.

wc	compte des caractères, des mots et des lignes.
while	représente une structure de contrôle de boucles.
who	affiche la liste des utilisateurs connectés.
write	écrit un message à d'autres utilisateurs.
xargs	combine des lignes de commandes et de saisie de clavier.

18 / Les fichiers batch

Les fichiers *batch* permettent d'exécuter une série d'instructions passées en ligne de commandes. Cela fonctionne sur les invites de commandes proposée par Windows (MS-DOS) ou Linux (Terminaux).

Les fameux écrans noirs invitent l'utilisateur à saisir des instructions afin d'effectuer des tâches précises.

```
Affichage de la date courante
c:\>date /T [Validez]

Affichage de l'heure courante
c:\>time /T [Validez]

Changement de répertoire
c:\>cd .\travail\trimestre2\ [Validez]

Edition d'un fichier
c:\travail\trimestre2\>edit cours.txt [Validez]
```

Parfois, il peut être utile de **lancer plusieurs instructions les unes à la suite des autres** afin d'atteindre un but précis.

```
c:\>d: [Validez]
d:\>cd travail [Validez]
d:\travail\>cd cours [Validez]
d:\travail\cours\>cd temporaire [Validez]
d:\travail\cours\temporaire\>del *.* [Validez]
d:\travail\cours\temporaire\>cd .. [Validez]
d:\travail\cours\>rd temporaire [Validez]
```

Dans ce cas, plutôt que de taper cette suite de commandes, surtout si elle est destinée à être répétée, il sera préférable d'utiliser un fichier batch.

```
rem Fichier : effacement.bat rem Empeche l'inscription des lignes de commandes !
@echo off
rem Efface la console
cls
rem Affiche du texte a l'ecran
echo Effacement du repertoire temporaire et de ses fichiers
rem Marque une pause
pause
d:
cd travail
cd cours
cd temporaire
del *.*
cd ..
rd temporaire
echo Les fichiers temporaires et le répertoire ont bien ete effaces !
```

L'enregistrement de ce genre de texte dans **un fichier portant l'extension *.bat* pour Windows et *.sh* pour Linux**, puis l'appel de ce fichier à partir de la console, permettront de lancer automatiquement le traitement attendu.

```
c:\>effacement.bat [Validez]
```

Une particularité importante des fichiers batch est qu'ils peuvent recevoir de **un à neuf arguments**.

```
rem Fichier : bienvenue.bat
@echo off
echo Bienvenue %1 !

c:\>bienvenue Frederic [Validez]
Bienvenue Frederic !
```

En fait, il existe dix arguments possible, il s'agît du nom du fichier batch lui même, portant la référence %0.

```
rem Fichier argument.bat
@echo off
cls
```

```
echo %0

C:\argument [Validez]
c:\argument
```

De cette manière, **l'utilisateur du fichier batch pourra saisir des informations** particulières au traitement attendu.

```
rem Fichier : suppression.bat
@echo off
cls
%1:

cd %2
del *.*
echo Effacement des fichiers !
cd \

rem Si le troisieme argument est egal a n alors allez a fin
if "%3"=="n" goto fin
rd %2
echo Effacement du repertoire %2 !

rem Etiquette de fin
:fin

C:\>suppression d repertoire o [Validez]
D:\dd\*.*, Êtes-vous sûr (O/N) ? o
Effacement des fichiers !
Effacement du repertoire dd !
D:\>
```

Dans cet exemple, **une instruction conditionnelle *if*** est utilisé afin de tester un argument et en fonction de ce dernier un certain traitement sera ou ne sera pas appliqué. En l'occurrence, le troisième argument détermine si le répertoire cible doit être effacé après les fichiers.

Il est possible d'**utiliser plus de neuf arguments** à l'aide d'une commande de décalage : *shift*.

```
rem Fichier : arguments.bat
@echo off
:continue
if "%1"=="" goto fin
echo %1
shift
goto continue
:fin
echo Lecture des arguments terminee !

C:\>arguments 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 [Validez]
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Lecture des arguments terminee !
```

Les fichiers batch peuvent également utiliser des variables, initialisées à l'aide de la commande *set*.

```
rem Fichier variable.bat
@echo off
set repertoire=%1
set fichier=%2

if exist %repertoire% goto suite
echo Le repertoire %repertoire% est introuvable !
goto fin

:suite
echo Le repertoire %repertoire% a ete trouve !
cd %repertoire%
if exist %fichier% goto suivant
echo Le fichier %fichier% est introuvable !
goto fin


:suivant
echo Le fichier %fichier% a ete trouve !
```

```
type %fichier%
echo .../...
echo Traitement termine !

:fin

C:\>variable repertoire fichier.txt [Validez]
Le repertoire repertoire a ete trouve !
Le fichier fichier.txt a ete trouve !
Le contenu du fichier .../...
Traitement termine !
E:\repertoire\>
```

Les variables ne peuvent délivrer leur contenu qu'à condition d'encadrer leur identificateur avec des signes de pourcentage %, sinon sera utilisé dans une expression, l'identificateur de la variable, et non sa valeur.

Une autre commande intéressante n'existe malheureusement plus dans les versions récentes de Windows. Il s'agît de **la commande *choice*** procurant une structure de choix particulièrement utile et puissante.

```
rem Fichier : choix.bat
:menu
echo A - Compilez un fichier source
echo B - Exécutez un fichier classe
echo C - Consultez un fichier source
echo D - Quittez le menu
echo E - Fermez la console

choice /c:abcde
if "%errorlevel%"=="A" goto compil
if "%errorlevel%"=="B" goto exec
if "%errorlevel%"=="C" goto edite
if "%errorlevel%"=="D" goto fin
if "%errorlevel%"=="D" goto quit

:compil
javac %1.java
goto menu

:exec
java %1
goto menu

:edite
edit %1.java
goto menu

:quit
exit

:fin

C:\>choix fichier
```

Les fichiers batch proposent donc une solution simple pour automatiser des tâches répétitives et plus ou moins complexes. Par exemple dans le cas des compilations Java, il peut être souhaitable d'initialiser certaines variables d'environnement nécessaires.

```
rem Fichier : compilexec.bat
@echo off
set path=c:\j2sdk\bin\;%PATH%

if not "%3"=="" goto init
set repertoire=%1
set fichier=%2
goto suite

:init
set repertoire=%2
set fichier=%3
:suite
set classpath=c:\%repertoire%\classes\
```

```
set source=c:\%1\sources\
set direction=c:\%1\classes\
javac -classpath %classpath%
                                -d %direction% %source%%fichier%.java
echo La compilation a ete accomplie avec succes !

java -classpath %classpath% %fichier%
echo L'execution s'est deroulee avec succes !
```

```
C:\compilexec projet repertoire fichier.java [Validez]
La compilation a ete accomplie avec succes !
...
L'execution s'est deroulee avec succes !
```

19 / Le système Informatique

Un système informatique (*SI*) est composé de deux parties : le matériel (hardware) et le logiciel (software). Un SI permet de traiter des informations reçues en entrée, et retourne d'autres informations en sortie.

Par exemple, deux nombres et un opérateur d'addition sont fournis par un opérateur via un clavier et le SI retourne sur un écran, le résultat d'un traitement applicatif qui est la somme des deux nombres. Dans ce cas, en parfaite symbiose, le logiciel et le matériel ont concouru à la saisie de données d'entrée, au traitement des informations et à l'affichage de données de sortie.

Le matériel est un ensemble de composants destiné à acquérir des informations extérieures (clavier, souris, webcam, scanner, etc.), **à accomplir des traitements internes** (processeur, CPU : Central Processing Unit, etc.), **à stocker des informations dans des mémoires de masse** (disque dur, disquette, cédérom, DVD ROM, etc.) et **à restituer le résultat des traitements** (écran, imprimante, carte son, etc.).
Le matériel est une machine possédant des entrées et des sorties, permettant le traitement de données.

Les données convergeant vers des entrées et divergeant vers des sorties sont des informatons compréhensibles par un être humain. Par exemple, un paragraphe saisi au clavier, ou une page numérisée sont directement interprétables par l'homme, de même que des paroles émises par l'intermédaire de moyens phoniques ou un document web affiché sur un moniteur. Par contre, ces informations sont converties dans **un langage compréhensible par le matériel** et en particulier le processeur ou unité de traitement. En l'occurrence, **ces informations sont des données binaires**, soit une suite de bits dont chacun peut prendre la valeur 0 ou 1 (false ou true, faux ou vrai, fermé ou ouvert) que l'ensemble du système périphérique convertit au gré des besoins.

La plus petite information binaire est un bit. Les bits sont regroupés dans des octets (bytes), soit des groupes de huit bits. Les octets constituent des mots de un, deux, quatre ou huit octets directement soumis au processeur. Un processeur 32 bits pourra traiter un mot de 64 bits en utilisant deux temps d'horloge. En effet, l'unité de traitement synchronise ses opérations par rapport à une horloge (quartz) envoyant des signaux à intervalles réguliers, il s'agît en fait, de **la fréquence du processeur**. Plus, cette fréquence est élevée, plus rapidement seront effectués les calculs et les échanges d'informations entre le système périphérique et l'unité de traitement. **Différentes mémoires sont mises à la disposition du CPU** afin de stocker les différentes informations relatives au fonctionnement du système informatique (ROM : Read Only Memory, mémoire permanente) et de conserver des informations provenant de différentes sources comme le système périphérique ou le processeur lui-même (RAM : Random Access Memory, mémoire volatile) d'accélérer le traitement des données (mémoire cache) et de charger des mots binaires avant traitement (registres intégrés au processeur).

Le CPU commande tout le système informatique via un circuit électrique appelé bus système. Toutes les mémoires sont connectées à ce bus possédant également une certaine fréquence. **L'ensemble des périphérique est également relié à ce circuit par l'intermédiaire de dispositifs d'intégration appelés interfaces.** Ces dernières fonctionnent comme des mémoires pour le processeur, dans lesquelles il peut lire et écrire des données binaires.

Le CPU est capable de faire des opérations élémentaires telles que l'addition, la soustraction, la multiplication, les décalages, des opérations logiques ET, OU, NON, etc.. Lorsqu'un calcul est éffectué par le processeur, plusieurs informations binaires lui sont d'abord transmises, comme des opérandes droite et gauche ainsi qu'une instruction d'addition (par exemple), puis en fonction de ces données, un résultat est produit et est retourné à l'utilisateur.

Toutes les opérations élémentaires précitées que le processeur est capable de faire, sont désignées chacune par une information binaire spécifique stockée en mémoire centrale. **Ces instructions constituent le langage machine, en quelque sorte le logiciel de programmation le plus prôche du processeur.** Chacune de ces instructions déclenche une commande de base du processeur comme dans l'exemple précedent : rechercher des valeurs dans la mémoire pour charger des registres, puis additionner les deux registres et stocker le résultat dans un autre registre avant utilisation.
Le langage machine n'est pas universel puisque des systèmes informatiques Macintosh ou PC (Personal Computer) ne gère pas les instructions de la même façon si bien que ces machines sont incompatibles entre elles à moins de posséder un émulateur manquant souvent de performances.

Le système d'exploitation (OS : Operating System) de tout système informatique s'appuie sur le

langage machine pour accomplir la moindre de ses actions. L'OS est un logiciel chargé des opérations de bas niveau, c'est-à-dire au plus près du matériel. Les données et les instructions sont décomposées en unité élémentaires par le système d'exploitation avant de les soumettre à l'unité de traitement.

Il existe **différents types d'OS** sur le marché comme DOS, Unix, Mac OS, Windows, OS/2 et Linux. Tous n'ont pas les mêmes caractéristiques, et **ne sont pas tous employés dans les mêmes domaines**. Par exemple la famille Unix (AIX, Digital Unix, HP-UX, NetBSD, OpenBSD, SCO, Solaris, SunOS), sans conteste la plus mature, est utilisée dans le domaine des serveurs professionnels, Mac OS se positionne dans la sphère de la PAO (Publication Assistée par Ordinateur), du graphisme ou du multimédia, et Windows s'éparpillant dans tous les secteurs, équipe les particuliers comme les entreprises.

Les systèmes d'exploitation se démarquent les uns des autres par certaines capacités de fonctionnement interne de leur noyau (kernel). Par exemple, DOS est mono-utilisateur et monotâche, la gamme des Windows XX destinée aux particuliers sont de types mono-utilisateur et multitâches, tandis qu'un Unix est multi-utilisateurs et multitâches. **Les termes mono/multi-utilisateurs et mono/multitâches signifient respectivement que l'OS est capable d'accepter sur un même SI un ou plusieurs utilisateurs et exécuter une ou plusieurs opérations simultanément.** D'autre part, certains OS seront excellents dans des secteurs précis car leur jeu d'instructions élémentaires et leur architecture seront plus ou moins bien adaptés à l'utilisation qui en sera faite.

Les interfaces utilisateurs selon les systèmes d'exploitation diffèrent entre eux. Acutellement, **l'interface graphique utilisateur (GUI : Graphic User Interface)** propose une ergonomie sans conteste efficace et conviviale. Quasiment tous les OS possèdent une inteface fenêtrée pilotable à la souris, permettant une navigation visuelle jusqu'au coeur du système. Néanmoins, **le mode console (ligne de commandes)** des anciens Unix, DOS ou Linux demeurent encore aujourd'hui incontournable dans le secteur des réseaux et de la sécurité informatique.

Divers programmes exécutables intégrés dans les systèmes d'exploitation permettent de configurer l'OS et le matériel (make, lsmod, menuconfig sous Linux, regedit, ipconfig sous Windows), de renseigner les utilisateurs (help, perfmon, setver sous Windows, history, diff, hdparm sous Linux,) ou d'effectuer diverses tâches (edit, notepad, calc sous Windows, tar, tty, sh sous Linux).

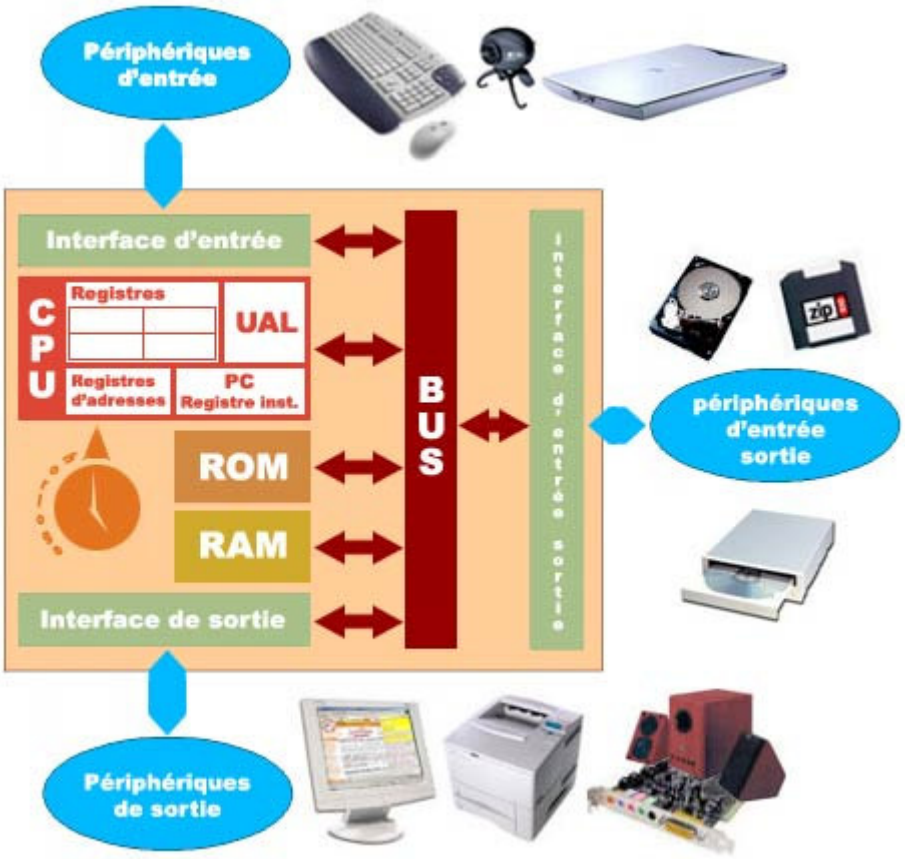
Le système de fichiers est un composant extrêmement important d'un OS puisque c'est à lui que revient l'organistation des mémoires de masses. Dans un premier temps, les différents supports physiques tels que les disques durs ou les disquettes doivent subir un formatage spécifique déterminant la manière dont sera structurée les données binaires. Un disque dur sera ainsi divisé en secteurs, pistes et cylindres et une table d'allocation de fichiers située en général dans le premier secteur servira à repérer la localisation de chaque fichier et dossier.

Les fichiers sont une collection d'octets avec une structure particulière dépendant de son type (Feuille de calcul, document texte, image, programme source, exécutable, bibliothèque, etc.). Leur taille dépend de leur contenu, un fichier texte comprend des caractères pesant chacun un octet, tandis qu'un exécutable peut contenir des instructions binaires pesant chacune de un à huit octets. **L'utilisation des fichiers passent obligatoirement par leur chargement en mémoire centrale** (RAM). Seulement à ce moment, ils pourront être exécutés, consultés, ou encore modifiés. Les fichiers possédant en général un nom et une extension (*identificateur.ext*) sont regroupés dans des répertoires, lesquels peuvent être également placés dans d'autres répertoires, formant ainsi **une arborescence parfaitement connue par la table d'allocation**. Tous les fichiers y compris les fichiers systèmes sensibles peuvent être exploitables par des utilisateurs quelconques comme sous Windows 9X ou une segmentation des mémoires de masse est créée afin que les utilisateurs n'accèdent qu'aux fichiers autorisés par rapport à leurs droits d'accès à l'image de Unix, Linux, Windows NT, 2000...

Un système informatique est une combinaison hautement interdépendante de matériels (processeur, périphériques, etc.) et de logiciels (langage machine, système d'exploitation). Les OS proposent une interface en mode graphique ou parfois en ligne de commandes garantissant une compréhensibilité entre l'utilisateur et la machine pour finalement exploiter au mieux l'ensemble des possiblités qu'offrent un système informatique. A partir de l'interface utilisateur (UI : User Interface), **le développeur sera en mesure de programmer des applications qui viendront se greffer sur le SI** et ainsi enrichir ce dernier d'une nouvelle couche applicative améliorant substantiellement l'UI.

19.1 / Architecture d'un Système Informatique

Un système informatique est composé de plusieurs éléments matériels dont la synergie permet la manipulation ou la visualisation d'informations par rapport aux qualités réceptives de l'homme.



Le processus de traitement de l'information semble transparente pour l'utilisateur car chaque action entrepris par l'utilisateur est transformée en une suite de données binaires que les différents composants seront alors capable de comprendre et d'interpréter.

Le CPU ou l'unité centrale de traitement (Central Processing Unit) est souvent connu sous le nom de processeur. Ce composant est **au coeur du système informatique**, quasiment toutes les informations binaires sont traitées par le processeur. Ce-dernier peut selon son architecture, faire des **calculs sur des mots binaires** de un octet (Intel 8088, Commodore, Apple II), deux octets (Intel 8086, i286), quatre octets (i386, AMD 386), huit octets (Intel Pentium, AMD Athlon, PowerPC) et plus pour les gros systèmes informatiques. **La fréquence et la densité des composants élémentaires** sont également des caractèristiques importantes du processeur. La loi de Moore datant de 1966 (Gordon E. Moore était alors directeur de la recherche chez Fairchild) prévoyait un doublement des capacités d'un processeur tous les dix-huit mois. Ainsi, en moins de deux décennies, de 16 mégahertz avec 275 000 transistors pour l'intel i386 en 1985, les processeurs du millénaire comme l'Intel Pentium IV dispose de 42 millions de transistors pour une fréquence de 1,5 gigahertz.

L'UAL ou Unité Arithmétique et Logique (ALU : Arithmetic and Logical Unit) permet de faire des **calculs mathématiques** tels que l'addition ou la multiplication, et des **opérations booléennes** telles que le ET, OU et NON. Ainsi, à partir de chaque combinaison binaire (0 et 1) en entrée, l'UAL détermine l'opération à accomplir et produit un résultat binaire en sortie.L'UAL fonctionne dans le mode combinatoire, à l'aide de portes logiques.

Les registres sont des mémoires directement intégrées dans le processeur. Un registre permet de **stocker une collection d'octets** telle que le résultat de la dernière opération de l'UAL ou la donnée du prochain calcul. Le nombre de registres est limité dans les processeurs, c'est pourquoi les informations binaires n'y font qu'un bref passage, le temps d'un ou plusieurs coups d'horloge.
PLusieurs registres spécifiques servent à accueillir un certain type d'information binaires.
Les registres d'adresses permettent de référencer les adresses des programmes et des données dans les mémoires ROM, RAM et les interfaces. Ces adresses, en fait des nombres binaires, permettent au processeur de récupérer les informations nécessaires à un endroit précis de la mémoire centrale (ROM

et RAM) pour l'accomplissement de ses tâches.
Le registre d'instruction sert à stocker l'information binaire désignée par un *code opération* dédiée à la réalisation d'un calcul arithmétique ou logique prédéfini.

La PC (Partie Commande) pilote le processeur en opérant des instructions de branchement entre les registres. A chaque coups d'horloge, la PC ouvre et ferme des portes dans le but d'alimenter l'UAL en donnée et instruction et d'inderdire la collision d'information. Par cet intermédiaire, le processeur traite une opération à la fois dans un intervalle donnée par l'horloge.

Le système informatique est entièrement commandé par l'intermédaire de la PC. Les actions que ce composant est capable d'accomplir sont désignées par des nombres binaires appelés *codes opérations*, lesquels forment **le langage machine** propre au fabricant du processeur. Un processeur Intel Pentium IV ne possède pas le même langage machine que son homologue Power Mac G4. **Le langage compréhensible par l'homme au plus près du matériel s'appelle l'*assembleur*** regroupant des commandes sous forme de mots clés de trois lettres maximum, comme *MOV* (déplacement vers un registre), *ADD* (addition), *INC* (incrémentation), etc.. Les programmes informatiques, à commencer par le système d'explotation s'appuie sur cette forme primitive de langage pour exécuter des applications plus ou moins évoluées.

20 / Les cycles de développement logiciel

Avant la livraison à un client, le développement d'un logiciel passe par plusieurs étapes définies au cours des deux dernières décennies :

- **l'analyse préalable d'un cahier des charges,**
- **la conception,**
- **le codage,**
- **les tests, *la validation,***
- **l'intégration,**
- **la mise en production,**
- **la recette* du système et sa validation,**
- **la maintenance du système.**

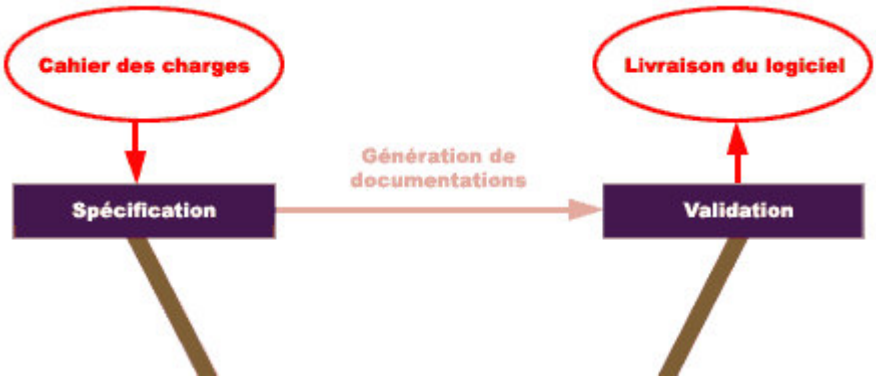
* La recette est une vérification de conformité du logiciel par rapport aux spécifications théoriques définies au début du projet, avant son déploiement final.

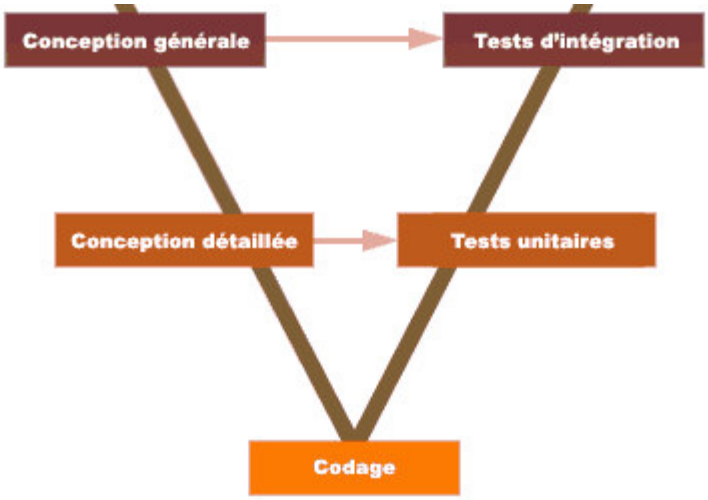
Le développement d'un produit applicatif doit prendre en compte des risques inhérents à cette activité (développement d'interfaces utilisateurs, ou de fonctions inappropriées, non-respect du cahier des charges, problèmes de performance, etc.) et à l'évolution d'une entreprise (calendrier irréalisable, budget inadapté, défection des personnels, etc.).

C'est pourquoi, une société de service en ingénierie informatique doit **se conformer à des règles intangibles de développement d'application**. A cet effet, il existe plusieurs **modèles de cycle de vie** d'un logiciel :

- **Le modèle en cascade** consiste en une succession de phases dont chacune est méthodiquement vérifié avant de passer à l'étape suivante :
  1. faisabilité et analyse des besoins : validation,
  2. conception générale et détaillée : vérification,
  3. intégration : tests d'intégration et tests d'acceptation,
  4. installation : tests du système.
- **Le modèle en V** repose sur une étroite interdépendance des étapes soumises à une validation avant la prochaine étape et une vérification anticipatoire du produit final :
  1. spécification textuelle,
  2. conception générale,
  3. conception détaillée,
  4. codage,
  5. tests unitaires,
  6. tests d'intégration,
  7. validation.
- **Le modèle en spirale** s'appuie sur une succession de cycles dont chacun se déroule en quatre phases :
  1. analyse initiale des besoins et des objectifs du cycle (solutions et contraintes) ou analyse à partir du cycle précèdent,
  2. étude des risques, évaluation des solutions de remplacement et éventuellement conception,
  3. développement et vérification de la solution résultant de l'étape précédente,
  4. examen du produit et projection vers le cycle suivant.
- **Le modèle par incrément** propose un développement du logiciel par morceaux, lesquels sont livrés successivement au client, en venant se greffer à un noyau logiciel.

Le modèle en V demeure actuellement le cycle de vie le plus connu et certainement le plus utilisé.





La première étape, appelé spécification ou analyse des besoins, a pour but de dégager du cahier des charges, toutes les contraintes nécessaires à l'élaboration du logiciel. Trois sortes de contraintes logicielles sont à prendre en considération :

- **Les contraintes externes** définissent les caractéristiques d'entrée/sortie du logiciel attendues par le client (données à utiliser et à afficher, les exigences ergonomiques, le parc informatique, etc.).
- **Les contraintes fonctionnelles** caractérisent le fonctionnement interne du logiciel, c'est-à-dire, quels moyens seront mis en oeuvre pour traiter les informations d'entrée/sortie du logiciel (méthode de calcul, intervalle des données, etc.).
- **Les contraintes de performances** indiquent la vitesse d'exécution du logiciel ou de ses modules, la résolution d'affichage, la précision des données, etc..

Les documents produits (plan de développement du logiciel, spécifications des besoins du logiciel et cahier de recette) au cours de cette phase **permettent de passer à l'étape suivante** et également **de préparer les vérifications de conformité du logiciel**.

Cette phase constitue environ **15 pourcents du temps total du cycle de développement**.

La conception générale (ou analyse organique générale)a pour objectif de déduire de la spécification, l'architecture du logiciel. Lors de cette phase, plusieurs solutions peuvent être envisagées afin d'en étudier leur faisabilité. A l'issue, **un document de conception générale du logiciel est réalisé** afin de décrire la structure générale de l'alternative approuvée.

Lors de cette phase, il peut être décider de **découper le logiciel en plusieurs modules distincts** afin de les sous-traiter par plusieurs équipes de développement. **Un module possède une interface** permettant son intégration au logiciel global ou à d'autres modules, **et un corps** pour son fonctionnement interne. **Ils sont hiérarchisés** de telle façon que des modules de bas niveau s'emboîtent dans des modules intermédiaires, lesquels s'intègrent à un module de haut niveau (noyau logiciel).

Un autre découpage peut être aussi utilisé pour **scinder le logiciel en tâches distinctes**. L'application contient alors plusieurs sous-ensembles ayant en charge des traitements spécifiques (tâches externes et tâches internes au logiciel), lesquels s'interconnectent entre eux

La phase de conception détaillée (ou analyse organique détaillée) en s'appuyant sur le document de conception générale, énumère l'architecture approfondie du logiciel jusqu'à parvenir à une description externe de chaque sous-ensemble et information utilisable dans le futur logiciel. A partir de cette étape, **seront connus toutes les données** (variables, constantes, attributs, champs, etc.) **et fonctions** (procédures, méthodes, etc.) de l'application vue de l'extérieur. Le logiciel peut être entièrement écrit en algorithme. Un langage de programmation est en général validé lors de cette phase. **Un document de conception détaillée, ainsi qu'un manuel d'utilisation sont édités** afin de respectivement de décrire l'architecture détaillée et la mise en oeuvre du logiciel. Les phases de conception doivent prendre environ 25 pourcents du temps total du cycle de développement.

Des spécifications de tests d'intégration et unitaire sont également produites, à l'issue des deux phases de conception. Elles permettront de confronter le fonctionnement de l'application à son architecture générale et détaillée.

Le codage consiste à écrire avec un langage de programmation chacune des sous-programmes du logiciel. Le développement peut être confié à une seule personne dans le cas d'une application simple ou

divisé entre plusieurs équipes de développeurs dans le cas de projets importants. Cette phase durant environ 15 pourcents du temps total du cycle de vie se termine par la **production d'un code source**.

Les tests unitaires ont pour objectif de vérifier individuellement la conformité de chaque élément du logiciel (fonctions et variables) par rapport aux documents de conception détaillée. Toutes les fonctionnalités internes et externes de chaque sous-programme sont contrôlées méthodiquement. En outre, **un contrôle des performances globales et locales est également entrepris.** Cette phase consomme aux alentours de 5 pourcents du temps total du cycle de vie et se finalise par la rédaction des résultats des tests.

La phase d'intégration permet de vérifier l'assemblage des différentes parties du logiciel. Les différents modules du logiciel sont successivement intégrés jusqu'à aboutir à la construction complète, en respectant rigoureusement les spécifications des tests d'intégration. Chaque module doit parfaitement être assimilé sans que le fonctionnement des modules précédemment intégrés n'en soit aucunement affecté. **Les résultats de cette phase sont consignés dans un document des tests d'intégration.** En général, une présentation du logiciel est également réalisée. Les tests d'intégration représentent en moyenne 20 pourcents du temps total du cycle de développement.

La dernière phase a pour vocation de valider le logiciel dans son environnement extérieur. Le produit applicatif est mis en situation d'utilisation finale afin de vérifier s'il répond parfaitement aux besoins énoncés dans les spécifications textuelles (première phase). Un document appelé résultat de la recette est produit au terme de la phase de validation qui dure 10 pourcents du temps total du cycle de vie du développement du logiciel.

La finalité du cycle de vie en V consiste à parvenir sans incident à livrer un logiciel totalement conforme au cahier des charges. Lors de la phase de spécification textuelle, une négociation avec le client permet d'affiner ou d'enrichir les besoins à propos de certains points techniques omis ou obscurs dans le cahier des charges. **Lorsque la spécification est validée, la suite du processus de développement doit être parfaitement encadré, contrôlé et approuvé de telle sorte qu'à aucun moment, il ne soit possible de diverger des règles énoncées lors de la première phase.**

21 / L'algèbre relationnelle appliquée au SGBDR

L'algèbre relationnelle a été introduite par M. CODD en 1970, pour formaliser les opérations sur les ensembles. C'est grâce à ce système formel que l'on peut démontrer l'équivalence des requêtes sur des bases de données relationnelles, et que l'on peut construire des optimiseurs de requêtes.

Les opérations relationnelles permettent de combiner des ensembles de valeurs, appelées relations, entre eux dans le but d'obtenir en fonction du type de l'opération, un certain résultat.

Plusieurs opérations relationnelles peuvent être appliquées à des tables de base de données dans des SGBDR. De telle opérations, appelées requêtes, sont lancées par l'intermédiaire du langage SQL afin d'extraire d'une ou plusieurs tables, une certaine combinaisons de colonnes (ou champs) et de lignes (ou uplets, tuples, enregistrements).

21.1 / Le modèle relationnel

Avant d'aborder les opérations relationnelles à proprement parler, il est nécessaire de définir le vocabulaire adapté au modèle relationnel.

Une base de données comporte une à plusieurs tables contenant diverses informations organisées par colonnes (ou champs) et lignes (ou enregistrements).

Dans le modèle relationnel, **une table de données correspond à une relation**, désignant un ensemble d'informations au sens mathématique. La relation est-elle même **décomposée en attributs et en n-uplets**, correspondant respectivement aux colonnes et aux lignes des tables.

Les attributs d'une relation forment un schéma, représentant la structure de cette relation. Le schéma caractérise une relation ainsi que tous les n-uplets qui y sont contenus.

Un n-uplet correspond à une combinaison de valeurs des n-attributs que possèdent la relation. Par exemple, si une relation possède six attributs, alors cette relation sera composée d'un certain nombre de "six-uplets".

Les n-uplets doivent tous être identifiables d'une manière unique. Pour assurer l'unicité d'un n-uplet, il est nécessaire de définir dans une relation, une clé primaire. Cette dernière, peut être **un attribut ou une combinaison d'attributs** (clé primaire composite), produisant respectivement une certaine valeur ou un ensemble de valeurs qui doit impérativement être unique, afin d'**identifier précisément un et un seul n-uplet** dans la relation.

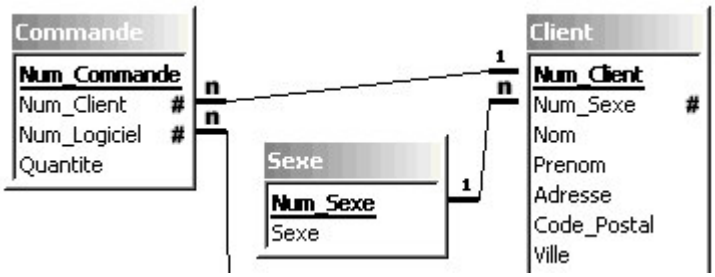
Les relations peuvent être reliées à d'autres relations par l'intermédiaire d'une liaison entre des clés identificatrices. **Une relation peut contenir une clé étrangère faisant référence à la clé primaire d'une seconde relation.** De cette manière, il devient possible de créer, pour les attributs d'une relation, des domaines de valeurs contenus dans d'autres relations liées. Par exemple un attribut dénommé *couleur* pourrait ne contenir qu'un seul code couleur défini dans une liste externe prédéterminée.

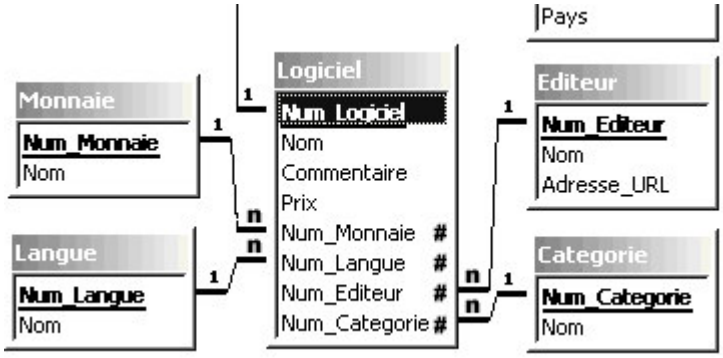
Relation A	ClePrim	Couleur	...
	1	c1	...
	2	c3	...
	3	c3	...
	4	c4	...
	5	c1	...

Relation B	ClePrim	CodeCoul
	c1	#FFFFFF
	c2	#FF0000
	c3	#FFFF00
	c4	#00FF00
	c5	#00FF00
	c6	#00FFFF
	c7	#000000

Une clé primaire peut être également une clé étrangère référençant une autre relation.

Les liaisons entre relations possèdent des cardinalités représentant le nombre d'occurrences de valeurs uniques autorisées entre des attributs de même domaine ou ayant le même sens. En fait, il existe trois combinaisons génériques de cardinalités : **Un à Un, Un à Plusieurs ou Plusieurs à Plusieurs**. La première cardinalité s'applique à une clé primaire et la seconde à une clé étrangère. De cette manière, il est possible de définir pour une relation son type de dépendance avec d'autres relations. Par exemple, un relation dénommée *famille* contient plusieurs personnes et à l'inverse une personne ne peut appartenir qu'à une et une seule *famille* (parentale).





Les conventions de présentation définissent qu'une clé primaire doit être soulignée et qu'une clé secondaire doit être suivie d'un signe dièse '*#*' ou d'une étoile '*'. Dans les modèles d'analyse Merise ou UML (Unified Modeling Language), les cardinalités *Plusieurs* peuvent être symbolisées par une lettre '*n*' ou parfois un signe infini, notamment dans le cas de Microsoft Access.

21.2 / L'union

L'union de deux relations possédant un schéma identique produit une relation résultante. Cette dernière de même schéma que celles dont elle est issue contient l'ensemble des n-uplets appartenant à chacune des deux relations ainsi que les uplets communs aux deux.

Formalisme : **R = UNION(Ra, Rb)** ou **R = Ra U Rb**

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i
	m	n	o
	p	q	r
	s	t	u

Rb	A	B	C
	g	h	i
	j	k	l
	v	w	x
	y	z	a

Ra U Rb	A	B	C	
	a	b	c	
	d	e	f	
	g	h	i	<< Intersection
	j	k	l	
	m	n	o	
	p	q	r	
	s	t	u	
	v	w	x	
	y	z	a	

Langage SQL :

```
SELECT * FROM Ra
UNION
SELECT * FROM Rb;
```

21.3 / L'intersection

L'intersection de deux relations possédant un schéma identique produit une relation résultante. Cette dernière de même schéma que celles dont elle est issue contient l'ensemble des n-uplets communs aux deux relations.

Formalisme : **R = INTERSECTION(Ra, Rb)**

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i
	m	n	o
	p	q	r
	s	t	u

Rb	A	B	C
	a	b	c
	g	h	i
	j	k	l
	p	q	r
	v	w	x
	y	z	a

INTERSECT(Ra, Rb)	A	B	C
	a	b	c
	g	h	i
	p	q	r

Langage SQL :

```
SELECT * FROM Ra
INTERSECT
SELECT * FROM Rb;
'ou
SELECT A, B, C FROM Ra
WHERE A IN (SELECT A FROM Rb)
    AND B IN (SELECT B FROM Rb)
    AND C IN (SELECT C FROM Rb) ;
```

21.4 / La différence

La différence entre deux relations possédant un schéma identique produit une relation résultante. Cette dernière de même schéma que celles dont elle est issue contient l'ensemble des n-uplets appartenant à la relation soustraite et n'appartenant pas à l'autre relation.

Formalisme : **R = DIFFERENCE(Ra, Rb)** ou **R = Ra - Rb**

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i
	m	n	o
	p	q	r
	s	t	u

Rb	A	B	C
	a	b	c
	g	h	i
	j	k	l
	p	q	r
	v	w	x
	y	z	a

Ra - Rb	A	B	C
	d	e	f
	m	n	o
	s	t	u

Langage SQL :

```
SELECT * FROM Ra
EXCEPT
SELECT * FROM Rb;
'ou
SELECT A, B, C FROM Ra
WHERE A NOT IN (SELECT A FROM Rb)
    AND B NOT IN (SELECT B FROM Rb)
    AND C NOT IN (SELECT C FROM Rb);
```

21.5 / Le produit cartésien

Le produit de deux relations possédant un schéma quelconque produit une relation résultante. Cette dernière possède les attributs des deux relations dont elle est issue, ainsi que toutes les combinaisons entre les n-uplets de chacune des relations précitées.

Formalisme : **R = PRODUIT(Ra, Rb)** ou **R = Ra x Rb**

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i

Rb	D	E	F
	u	v	w
	x	y	z

Ra x Rb	A	B	C	D	E	F
	a	b	c	u	v	w
	a	b	c	x	y	z
	d	e	f	u	v	w
	d	e	f	x	y	z
	g	h	i	u	v	w
	g	h	i	x	y	z

Langage SQL :

```
SELECT * FROM Ra, Rb;
```

21.6 / La projection

La projection ne s'applique qu'à une seule relation en produisant une relation résultante. Cette dernière ne possède que certains attributs déterminés de la relation dont elle est issue, et contient tous les n-uplets du ou des attributs projetés, de la relation précitée à l'exception des doublons. Cette opération de projection a pour but d'**éliminer des attributs d'une relation**.

Formalisme : **R = PROJECTION(Ra, Atribut1, ..., AttributN)**
ou
R = PROJECTION$_{Atribut1, ..., AttributN}$(Ra)

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i

PROJ$_B$(Ra)	B
	b
	e
	h

Langage SQL :

```
'Avec doublons
SELECT B FROM Ra;
'Sans doublons
SELECT DISTINCT B FROM Ra;
'ou
SELECT B FROM Ra GROUP BY B;
```

21.7 / La sélection

La sélection (ou restriction) ne s'applique qu'à une seule relation en produisant une relation résultante. Cette dernière de même schéma de la relation dont elle est issue, ne contient que certains n-uplets correspondant à une condition exprimées par l'intermédiaire d'opérateurs logiques (OU, ET, NON) et/ou arithmétiques (=, !=, <, >, <=, >=).

Formalisme : **R = SELECTION(Ra, Attribut opérateur "Valeur")**
ou
R = SELECTION$_{\text{Attribut opérateur "Valeur"}}$(Ra)

Ra	A	B	C
	a	b	c
	z	b	a
	m	u	b
	b	k	i
	x	b	f

SELECT$_{B="b"}$(Ra)	A	B	C
	a	b	c
	z	b	a
	x	b	f

Langage SQL :
SELECT * FROM Ra **WHERE B = "b"**;

21.8 / La jointure

La jointure entre deux relations possédant un schéma quelconque mais avec au moins un attribut défini dans le même domaine de valeurs, produit une relation résultante. Cette dernière comporte la totalité des attributs de chacune des relations dont elle est issue, et contient la combinaison de tous les n-uplets des deux relations précitées, correspondant à la condition de jointure.

Cette condition peut tester l'égalité entre un ou plusieurs attributs toujours définis dans le même domaine, c'est-à-dire, dans un ensemble de valeurs autorisées pour chacun des attributs communs aux deux relations. Par ailleurs, les attributs comparés sur les deux relations peuvent ne pas avoir le même nom.

Ainsi dans les théta-jointures, on distingue l'équijointure, vérifiant l'égalité des valeurs d'attributs, et d'autres vérifiant une comparaison de différence, d'infériorité ou de supériorité (!=, <, >, <=, >=).

Formalisme : **R = JOINTURE(Ra, Rb, Ra.Attribut opérateur Rb.Attribut)**
ou
R = JOINTURE$_{\text{Ra.Attribut opérateur Rb.Attribut}}$(Ra, Rb)

Ra	A	B	C
	a	b	2
	d	e	3
	g	h	4
	m	n	8
	p	q	9
	s	t	10

Rb	C	D	E
	1	b	c
	4	h	i
	3	k	l
	8	q	r
	10	w	x
	3	z	a

JOIN$_{\text{Ra.C = Rb.C}}$(Ra, Rb)	A	B	C	D	E
	d	e	3	k	l
	d	e	3	z	a
	g	h	4	h	i
	m	n	8	q	r
	s	t	10	w	x

Langage SQL :
SELECT a.A, a.B, a.C, b.D, b.E FROM Ra AS a, Rb AS b
WHERE a.C = b.C;

21.8.1 / La jointure naturelle

La jointure naturelle est une équijointure dont la condition porte sur l'égalité de valeurs entre tous les attributs de même nom, des relations concernées. Le schéma de la relation résultante correspond à une concaténation de l'ensemble des attributs des deux relations dont elle est issue, autour du ou des attributs communs.

Formalisme : **R = JOINTURE(Ra, Rb)**

Ra	A	B	C
	a	b	2
	d	e	3
	g	h	4
	m	n	8
	p	q	9
	s	c	1

Rb	B	C	D
	c	1	b
	b	2	h
	l	3	k
	n	8	q
	x	10	w
	b	2	z

$JOIN_{Ra.C = Rb.C}(Ra, Rb)$	A	B	C	D
	a	b	2	h
	a	b	2	z
	m	n	8	q
	m	n	8	q
	s	c	1	b

Langage SQL :
SELECT a.A, a.B, a.C, b.D FROM **Ra AS a, Rb AS b**
WHERE a.B = b.B AND a.C = b.C;

21.8.2 / L'auto jointure

L'auto jointure permet d'effectuer une opération de jointure réflexive sur une relation unique dédoublée. Le schéma de la relation résultante sera composé par une concaténation de tous les attributs des relations dont elle est issue.

Le langage SQL :
SELECT * FROM **Ra AS a, Ra AS b**
WHERE a.B = b.B;

Ra (1)	A	B	C
	a	b	2
	d	e	3
	g	h	4
	m	b	8
	p	q	9
	s	h	10

Ra (2)	A	B	C
	a	b	2
	d	e	3
	g	h	4
	m	b	8
	p	q	9
	s	h	10

Résultat	Ra.A	Ra.B	Ra.C	Rb.A	Rb.B	Rb.C
	a	b	2	a	b	2
	d	e	3	d	e	3
	m	b	8	a	b	2
	m	b	8	m	b	8
	p	q	9	p	q	9
	s	h	10	g	h	4
	s	h	10	s	h	10

21.8.3 / La jointure externe

Les jointures externes sont utilisées pour afficher tous les n-uplets, y compris ceux n'ayant pas de correspondance dans l'une ou l'autre des relations concernées. Le schéma de la nouvelle relation est composée de l'ensemble des attributs des relations dont elle est issue.

Les jointures externes peuvent être appliquées en complet, à droite ou à gauche.

- **Une jointure externe complète** provoque la création d'une relation résultante contenant tous les n-uplets, y compris ceux ne pouvant être joints des relations impliquées.
- **Une jointure externe à gauche** provoque la création d'une relation résultante contenant tous les n-uplets, y compris ceux ne pouvant être joints de la relation placée à gauche.
- **Une jointure externe à droite** provoque la création d'une relation résultante contenant tous les n-uplets, y compris ceux ne pouvant être joints de la relation placée à droite.

Langage SQL :
SELECT * FROM Ra
{FULL | LEFT | RIGHT} OUTER JOIN Rb
ON Ra.B = Rb.B

Dans ce cas, *Ra* est la relation de gauche et *Rb* la relation de droite.

Ra	A	B	C
	a	b	c
	d	e	f
	g	h	i

Rb	D	E	F
	r	b	m
	u	q	w
	x	h	k

FULL	A	B	C	D	E	F
	a	b	c	r	b	m
	d	e	f			
	g	h	i	x	h	k
				u	q	w

LEFT	A	B	C	D	E	F
	a	b	c	r	b	m
	d	e	f			
	g	h	i	x	h	k

RIGHT	A	B	C	D	E	F
	a	b	c	r	b	m
	g	h	i	x	h	k
				u	q	w

21.9 / Les fonctions d'agrégation

Les fonctions d'agrégation calculent une résultat à partir des valeurs numériques d'un attribut d'une relation.

Ces fonctions sont des mots-clés SQL prédéfinis et sont utilisées pour fournir des informations de synthèse, à l'image d'une somme, d'une moyenne, d'un minimum et d'un maximum des valeurs d'un attribut ou encore d'un nombre de n-uplets.

21.9.1 / Le dénombrement

Un dénombrement des n-uplets de regroupements d'une relation ou de la totalité de cette dernière s'effectue par l'intermédiaire d'une fonction d'agrégation, dénommée *compte* (*count()*).

Cette fonction compte le nombre d'occurrences pour une relation et éventuellement un attribut donnés.

Syntaxe SQL :
SELECT **count(Champ | *)**
FROM Relation
[ORDER BY Champ]
[HAVING Champ = "valeur"];

Ra	A	B	C	D	E	F
	a	b	2	a	b	2
	d	e	3	d	e	3
	m	b	8	a	b	2
	m	b	8	m	b	8
	p	q	9	p	q	9
	s	h	10	g	h	4
	s	h	10	s	h	10
	m	h	12	s	d	1
	m	m	15	x	s	8

Compte(Ra)	Cpt
	9

Compte(Ra, A)	A	Cpt
	m	4
	s	2
	a	1
	d	1
	p	1

Langage SQL :
SELECT **count(*)** FROM Ra

SELECT A, **count(A)** FROM Ra GROUP BY A;

21.9.2 / La somme

Une somme d'un uplet de regroupements d'une relation ou de la totalité de cette dernière s'effectue par l'intermédiaire d'une fonction d'agrégation, dénommée *somme* (*sum()*).

Cette fonction effectue la somme des valeurs numériques pour un attribut d'une relation donnés.

Syntaxe SQL :
SELECT **sum(Champ)**
FROM Relation
[ORDER BY Champ]
[HAVING Champ = "valeur"];

Ra	A	B	C	D	E	F
	a	b	2	a	b	2
	d	e	3	d	e	3
	m	b	8	a	b	2
	m	b	8	m	b	8
	p	q	9	p	q	9
	s	h	10	g	h	4
	s	h	10	s	h	10
	m	h	12	s	d	1
	m	m	15	x	s	8

sum(Ra, C, F)	Sc	Sf
	77	47

Somme(Ra, A)	A	Sc
	m	43
	s	20
	a	2
	d	3
	p	9

Langage SQL :
SELECT **sum(C) AS Sc, sum(F) AS Sf** FROM Ra

SELECT A, **sum(C) AS Sc** FROM Ra GROUP BY A;

21.9.3 / La moyenne

La moyenne des valeurs d'un attribut d'une relation est calculable par l'intermédiaire de la fonction de moyenne (*avg()*).

Cette fonction calcule une moyenne à partir de la somme des valeurs d'un attribut spécifié en argument et du nombre total d'occurrences pour l'attribut d'une relation ou éventuellement de chaque groupe d'un autre attribut.

Syntaxe SQL :
SELECT [champG,]**avg(Champ)**
FROM Relation
[ORDER BY ChampG]
[HAVING ChampG = "valeur"];

Ra	A	B	C	D	E	F
	a	b	2	a	b	2
	d	e	3	d	e	3
	m	b	8	a	b	2
	m	b	8	m	b	8
	p	q	9	p	q	9
	s	h	10	g	h	4
	s	h	10	s	h	10
	m	h	12	s	d	1
	m	m	15	x	s	8

moy(Ra)	Moy
	8.55

moy(Ra, A)	A	Moy
	m	10.75
	s	10
	a	2
	d	3
	p	9

Langage SQL :
SELECT **avg(C)** FROM Ra

SELECT A, **avg(C)** FROM Ra GROUP BY A;

21.9.4 / Le minimum et le maximum

Le minimum et le maximum d'un attribut sont accessibles respectivement via des fonctions spécifiques d'extraction *min()* et *max()*.

Ces fonctions déterminent les valeurs numériques minimum et maximum à partir d'un attribut passé en argument.

Syntaxe SQL :
SELECT [ChampG,]**min(Champ), max(Champ)**
FROM Relation
[ORDER BY ChampG]
[HAVING ChampG = "valeur"];

Ra	A	B	C	D	E	F
	a	b	2	a	b	2
	d	e	3	d	e	3
	m	b	8	a	b	2
	m	b	8	m	b	8
	p	q	9	p	q	9
	s	h	10	g	h	4
	s	h	5	s	h	10
	m	h	12	s	d	1
	m	m	15	x	s	8

moy(Ra)	Mn	Mx
	2	5

moy(Ra, A)	A	Mn	Mx
	m	8	15
	s	5	10
	a	2	2
	d	3	3
	p	9	9

Langage SQL :
SELECT **min(C), max(C)** FROM Ra

SELECT A, **min(C), max(C)** FROM Ra GROUP BY A;