Aide-mémoire LibreOffice

LibreOffice Basic
Bibliothèque d'exécution

Débutant

v. 2.0 – 06/04/2021

Rédigé avec LibreOffice v. 7.0.5 – Plateforme : Toutes

Options d'exécution

À spécifier, **par module**, avant tout code exécutable.

`Option Explicit`	Impose la déclaration explicite des variables.
`Option Compatible`	LibO Basic se comporte comme VBA.
`Option VBASupport 1`	Active le support de VBA.
`Option Base 1`	Les tableaux sont indexés à partir de `1` au lieu de `0`.
`Option ClassModule`	À utiliser pour créer des classes (+ `Option Compatible`).

Constantes Basic

`True, False`	Vrai, Faux (Boolean)
`Pi`	`3.14159265358979` (Double)
`Empty`	Var. non initialisée.
`Null`	Var. ne contient pas de donnée utile.
`Nothing`	(objets) supprime l'assignation antérieure.

Fonctions

☞ Syntaxe générale des fonctions : `Resultat = NomDeLaFonction(arguments)`

Fonctions Chaînes de caractères (type String)

`Asc()` Renvoie la valeur Ascii (du premier caractère) d'une chaîne.
☛ *Chr(), Table Ascii.* `Asc("Azerty")` → `65`

`Chr()` Retourne le caractère dont le code Ascii est passé.
☛ *Asc(), Table Ascii.* `Chr(65)` → `"A"`

`ConvertFromURL()` Convertit un nom de fichier du format URL vers le format de l'OS.
Format URL : `protocole:///hote/chemin/vers/fichier.ext`
Ex. Windows : `file:///c:/rep/fichier.ods`
Ex. Linux : `file:///home/user/rep/fichier.ods`

`ConvertToURL()` Convertit un nom de fichier du format de l'OS vers le format URL.
☛ *ConvertFromURL()*

`Format()` Convertit un nombre en chaîne, en la formatant selon un masque.
☛ Fonction Format – Masques de format.

`InStr()` Renvoie la position d'une chaîne de caractères dans une autre.
Si pas trouvé, retourne `0`.
`InStr("LibreOffice", "Office")` → `6`

`Join()` Renvoie une chaîne de car. à partir d'un tableau de chaînes.
☛ *Split()* `Tablo = Array("C:", "Rep", "SsRep", "MonFichier.ods")`
`Join(Tablo, "\")` → `"C:\Rep\SsRep\MonFichier.ods"`

`LCase()` Retourne une chaîne mise en minuscules.
☛ *UCase()* `LCase("LibreOffice")` → `"libreoffice"`

`Left()` `Left(chaine, N)` Extrait `N` car. d'une chaîne à partir de la gauche.
☛ *Mid(), Right()* `Left("LibreOffice", 5)` → `"Libre"`

`Len()` renvoie le nombre de caractères de la chaîne.
`Len("LibreOffice")` → `11`

`Ltrim()` Supprime les espaces à gauche (au début) d'une chaîne de car.
☛ *RTrim(), Trim()*

`Mid()` `Mid(chaine, P, N)`. Extrait `N` car. à l'intérieur d'une chaîne à partir de la position `P` : `Mid("14/7/2017", 4, 1)` → `"7"`
☛ *Left(), Right()*

`Right()` `Right(chaine, N)`. Extrait `N` car. d'une chaîne à partir de la droite.
☛ *Left(), Mid()*

`Rtrim()` Supprime les espaces à droite (à la fin) d'une chaîne de car.
☛ *LTrim(), Trim()*

`Space()` Renvoie une chaîne de car. constituée d'une suite d'espaces.
☛ *String()* `Space(3)` → `"   "`

`Split()` Renvoie un tableau de chaînes à partir d'une chaîne de car. en coupant sur un car. spécifié.
☛ *Join()*
`MaChaine = "C:\Rep\SsRep\MonFichier.ods"`
`Split(MaChaine, "\")` → un tableau de 4 éléments : `"C:"`, `"Rep"`, `"SsRep"`, `"MonFichier.ods"`

`Str()` Convertit une expression numérique en chaîne.
☛ *CStr(), Val()* `Str(-65)` → `" -65"`
☞ Un espace précède le texte. Le séparateur décimal est le point.

`StrComp()` Compare deux chaînes de caractères et renvoie une valeur entière représentant le résultat de la comparaison.

`String()` Crée une chaîne constitué de `N` fois un caractère.
☛ *Space()* `String(4, "Y")` → `"YYYY"`

`Trim()` Supprime les espaces à gauche et à droite d'une chaîne de car.
☛ *LTrim(), RTrim()*

`Ucase()` Retourne une chaîne mise en majuscules.
☛ *LCase()* `UCase("LibreOffice")` → `"LIBREOFFICE"`

`Val()` Convertit une chaîne en valeur numérique (`0` si non convertible).
☛ *Str(), Val()* `Val("12,34")` → `12,34`

Fonctions Numériques

`Abs()` Retourne la valeur absolue d'un nombre.
`Exp()` Exponentielle. Renvoie e élevé à une puissance.
`Fix()` Renvoie la partie entière d'un nombre (sans arrondir).
`Hex()` Retourne la valeur hexadécimale d'un nombre décimal.
`Int()` Renvoie la partie entière d'un nombre (arrondie à la valeur inférieure).
`Log()` Retourne le logarithme d'un nombre.
`Oct()` Retourne la valeur octale d'un nombre décimal.
`Randomize()` Initialise le générateur de nombres aléatoires (pour la fonction `Rnd()`).
`Rnd()` Renvoie un nombre aléatoire entre `0` et `1`. ☛ `Randomize()`
`Sgn()` Retourne le signe d'un nombre.
`Sqr()` Calcule la racine carrée d'un nombre.

Fonctions Trigonométriques

Angles en radians. `radians = (degrés * Pi)/180`
`Atn()` Arc tangente. `Cos()` Cosinus. `Tan()` Tangente.

Fonctions Date/heure

Format de date « UNO »

L'API LibreOffice utilise souvent les dates « Uno », c-à-d de type `com.sun.star.util.DateTime` (ou `.Date` ou `.Time`), structuré ainsi :

`IsUTC`	`True` si fuseau hor. est UTC.	`Hours`	Les heures (0-23).
`Year`	N° de l'année	`Minutes`	Les minutes (0-59).
`Month`	N° du mois (`0` si date vide).	`Seconds`	Les secondes (0-59).
`Day`	N° du jour (`0` si date vide).	`NanoSeconds`	Les nanosecondes.

☞ `Date` ↔ Uno Date : utilisez les fonctions de conversion `CDateXxx` ci-dessous.

Fonctions Date/heure

`CDateFromISO()` Renvoie la valeur de type Date correspondant à chaîne de caractères de date au format ISO (`AAAAMMJJ`).
`CDateFromISO("20170714")` → cette date au format Date.

`CDateFromUnoDate()` Convertit une structure UNO `com.sun.star.util.Date` en une valeur de type Date.

`CDateFromUnoDateTime()` Convertit une structure UNO `com.sun.star.util.DateTime` en une valeur de type Date.

`CDateFromUnoTime()` Convertit une structure UNO `com.sun.star.util.Time` en une valeur de type Date.

`CDateToISO()` Renvoie la date au format ISO (`AAAAMMJJ`) à partir d'une valeur type Date : le 14/7/2017, `CDateToISO(Now())` → `"20170714"`

`CDateToUnoDate()` Renvoie la date sous forme d'une structure UNO `com.sun.star.util.Date`.

`CDateToUnoDateTime()` Renvoie la date et l'heure sous forme d'une structure UNO `com.sun.star.util.DateTime`.

`CDateToUnoTime()` Renvoie l'heure sous forme d'une structure UNO `com.sun.star.util.Time`.

`Date()` Retourne la date actuelle (type Date).
☛ *Now(), Time()*

`DateAdd()` Renvoie une nouvelle date calculée à partir d'une date de départ et un critère d'ajout (±).
Le 14/7/2017, `DateAdd("m", 1, Now())` → `14/8/2017`
Masques de type d'ajout :

`yyyy`	Année	`ww`	Semaine de l'année
`q`	Trimestre	`d`	Jour
`m`	Mois	`h`	Heure
`y`	Jour de l'année	`n`	Minute
`w`	Jour de la semaine	`s`	Seconde

`DateDiff()` Calcule la différence entre deux dates, exprimée dans une unité à choisir (☛ tableau dans `DateAdd()`).
`DateDiff("m", "14/8/2017", "14/7/2017")` → `1`

`DatePart()` Renvoie la partie spécifiée de la date (☛ tableau dans `DateAdd()`) : `DatePart("q", "14/7/2017")` → `3`

`DateSerial()` Retourne une valeur numérique pour une date, calculée à partir des éléments année, mois, jour : `DateSerial(2017,7,14)` →

`DateValue()` Renvoie une valeur de date à partir d'une chaîne de caractères la représentant.
`DateValue("14/7/2017")` → `14/07/2017` (type Date)

`Day()` Renvoie le numéro du jour dans le mois.
`Day("14/7/2017")` → `14`

`Hour()` Renvoie l'heure : à midi, `Hour(Now())` → `12`

`Minute()` Renvoie les minutes d'une valeur de type Date.
Il est midi. `Minute(Now())` → `0`

`Month()` Renvoie le numéro du mois : `Month("14/7/2017")` → `7`

`Now()` Retourne l'horodatage actuel (type Date).
☛ *Date(), Time()*

`Second()` Renvoie la seconde d'une valeur de type Date.
Il est midi. `Second(Now())` → `0`

`Time()` Retourne l'heure courante au type Date.
☛ *Date(), Now()*

`Timer()` Retourne un Double qui indique le nombre de secondes écoulées depuis minuit.
☞ Affectez `Timer()` à une variable avant de l'utiliser !

`TimeSerial()` Retourne une valeur date pour une heure, calculée à partir des éléments heure, minute, seconde.
`TimeSerial(12,25,14)` → `12:25:14` (typeDate)

`TimeValue()` Renvoie une valeur horaire (type Date) à partir d'une chaîne de car. la représentant : `TimeValue("12:25:14")` → `12:25:14`

`Wait` (instruction) Attend un nombre spécifié de millisecondes.
`Wait 1000` → pause de 1 sec.

`WeekDay()` Renvoie le numéro du jour de la semaine (`1` = dim).
`Weekday("14/7/2017")` → `6` (vendredi)

`Year()` Renvoie le numéro de l'année : `Year("14/7/2017")` → `2017`

Fonctions Couleurs

Les couleurs sont stockées dans des `Long`.

`Red(), Green(), Blue()` Extraient les composantes d'une couleur.

`RGB()` Renvoie une couleur à partir des trois composantes rouge, verte et bleue : `RGB(128,0,0)` → `8388608` (rouge)

Fonctions Tableaux

`Array()` Crée un tableau à partir de valeurs discrètes.
`MonTablo = Array("Un", 2, Now())`

`DimArray()` Comme `Array()` : `MonTablo = DimArray("Un", 2, Now())`
☞ Utilisez si déclaration implicite des variables. Sinon, utilisez `Array()`.

`Erase` (Instruction) Efface le contenu du tableau. Si tableau dynamique, libère la mémoire. `Erase MonTablo`

`LBound()` Borne inférieure. `UBound()` Borne supérieure.

Fonctions d'interrogation de type

Ces fonctions permettent d'obtenir des informations à propos des variables.

Sur toute variable

`TypeName()` Renvoie une chaîne détaillant une variable donnée.
`VarType()` Renvoie une valeur numérique relative à une variable donnée.
`IsUnoStruct()` Renvoie `True` si l'argument est une structure UNO.

Les deux premières fonctions rendent les valeurs listées ci-dessous :

VarType	TypeName	VarType	TypeName	VarType	TypeName
0	Empty	5	Double	11	Boolean
1	Null	6	Currency	12	Variant
2	Integer	7	Date	17	Byte
3	Long	8	String	37	Decimal
4	Single	9	Object		

☞ Tableaux: `8192 + vartype`

Sur des variants

Renvoient `True` selon le type réel détecté.

Fonction	Vérifie le type	Fonction	Vérifie le type
`IsArray()`	Tableau.	`IsNull()`	`Null` (pas de données).
`IsDate()`	Date.	`IsNumeric()`	Valeur numérique.
`IsEmpty()`	Variable non initialisée.	`IsObject()`	Objet OLE.
`IsError()`	Valeur d'erreur.	`IsUNOStruct()`	`True` si structure UNO.

Sur des objets

☛ Structures et objets UNO

Fonctions de transtypage (*typecast*)

Ces fonctions convertissent une valeur d'un type compatible vers un autre. Le nom de la fonction reflète le type cible.

☞ Lisibilité du code : préférez toujours un typecast explicite à un transtypage implicite !

`CBool()`	Vers `Boolean`	`CDbl()`	Vers `Double`	`CSng()`	Vers `Single`
`CByte()`	Vers `Byte`	`CDec()`	Vers `Decimal`	`CStr()`	Vers `String`
`CCur()`	Vers `Currency`	`CInt()`	Vers `Integer`	`CVar()`	Vers `Variant`
`CDate()`	Vers `Date`	`CLng()`	Vers `Long`	`CVErr()`	Vers `Variant` (Erreur)

☞ Compatibilité des types entre eux : voir A-M n°2A.

Structures et objets UNO

`CreateUnoService(Nom)` Crée un service UNO. Nom est sensible à la casse !
`IsUNOStruct()` `True` si *structure* UNO.
`(struct.)Dbg_Properties` Renvoie le nom de la *structure* UNO (`String`).
`HasUnoInterfaces()` `True` si objet UNO supporte des interfaces.
`(obj.)SupportsService()` `True` si `obj` supporte le service en argument (`String`).
`EqualUnoObjects(o1, o2)` `True` si deux var. se réfèrent à la même instance d'objet.

Fonctions d'information sur les erreurs

`Erl` N° de ligne de l'erreur
`Err` Code de l'erreur
`Error` Message associé à l'erreur.

Fonctions diverses

`GetGUIType()` Renvoie une valeur qui caractérise le système, parmi :
1 Windows
3 MacOS
4 OSX ou Linux

`GetSolarVersion()` Renvoie la version de LibreOffice.
`IsMissing()` Teste si un paramètre facultatif de sous-programme est omis.

Appeler des commandes système

Syntaxe de la commande : `Shell(Commande, Style, Param, Synchro)`, avec :

`Commande` La commande à exécuter (`String`).
`Style` La fenêtre dans laquelle l'exécution a lieu, parmi (`Integer`) :
- `0` Le programme reçoit le focus et sa fenêtre est masquée.
- `1` Le programme reçoit le focus et est lancé dans une fenêtre standard.
- `2` Le programme reçoit le focus et est lancé dans une fenêtre minimisée.
- `3` Le programme reçoit le focus et est lancé dans une fenêtre maximisée.
- `4` Le programme est démarré dans une fenêtre standard non focalisée.
- `6` Le programme est lancé dans une fenêtre minimisée ; la focalisation reste sur la fenêtre actuelle.
- `10` Le programme est lancé en mode plein écran.

`Param` Les paramètres d'exécution à passer à la commande (`String`).
`Synchro` Drapeau de suivi de l'exécution :
- `True` Attendre la fin de l'exécution de la commande.
- `False` Ne pas attendre la fin de l'exécution de la commande.

Fonction Format – Masques de format

La fonction `Format()` convertit un nombre en chaîne, en la formatant selon un masque.
Un **masque de format** est une chaîne qui peut être divisée en trois sections séparées par des points-virgules : `val>0;val<0;val=0`. Une seule section = tous les nombres.

☞ Formatage des nombres selon la langue, configurez **Outils > Options > Paramètres linguistiques > Langue**.

Nombres

`0`	Chiffre obligatoire à la position (0 si manquant)	`%`	Le résultat est affiché au format pourcentage.
`#`	Chiffre facultatif	`E- E+`	Format scientifique.
`.`	Séparateur décimal	`e- e+`	
`+- espace ( )`	Caractères littéraux, tels quels dans le résultat.	`\`	Car. d'échappement : le car. qui suit est affiché dans le résultat.

Dates

`D` ou `DD`	N° du jour (1 ou 2 car.)	`Q` ou `QQ`	N° du trimestre (1 ou 2 car.)
`M` ou `MM`	N° du mois (1 ou 2 car.)	`W` ou `WW`	N° semaine (1 ou 2 car.).
`MMMM`	Nom du mois.	`h` ou `hh`	Heure (1 ou 2 car.)
`YY` ou `YYYY`	N° de l'année (2 ou 4 car.)	`m` ou `mm`	Minutes (1 ou 2 car.)
`NNN`	Nom du jour.	`s` ou `ss`	Secondes (1 ou 2 car.)

Exemple

Le 14/7/2021 : `Format(Now(), "yyyy")` → `"2021"`

Support VBA

☞ La prise en charge de VBA n'est pas complète.

Options d'environnement

Outils > Options > LibreOffice > Chargement/enregistrement > Général

Charger le code Basic Charge et enregistre le code VBA d'un document MSOffice dans un module spécial LibreOffice Basic.
Code exécutable Le code VBA sera chargé prêt à être exécuté.
Enregistrer le code Basic d'origine Le code VBA dans le document est enregistré à part pendant le chargement du document dans LibO.

Options d'exécution

Le support VBA requiert les options : `Option VBASupport 1` et `Option Compatible`.

Fonctions VBA

`AscW`	`FV()`	`IRR()`	`Round()`
`ChrW`	`Input()`	`Me()`	`RTL()`
`DDB()`	`InStrRev()`	`MIRR()`	`StrReverse()`
`FormatDateTime()`	`IPmt()`	`NPer()`	`WeekDayName()`

Plus de détails dans l'aide en ligne.

Instructions VBA

`Enum`

```
Enum MonEnumeration
  W1ND0WS   = 1 ' Windows
  OS2PM     = 2 ' OS/2 Presentation Manager
  MACINTOSH = 3 ' Macintosh
  MOTIF     = 4 ' Motif Window Manager / Unix-like
  OPENLOOK  = 5 ' Open Look / Unix-like
End Enum
```

Les valeurs énumérées sont rendues comme `Long`.

Les noms des énumérations et de leurs valeurs doivent être **uniques** à l'intérieur d'une bibliothèque.

Table Ascii

Déc	Hex	Val	Déc	Hex	Val	Déc	Hex	Val	Déc	Hex	Val
0	0	NUL	32	20	SPC	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Crédits

Auteur : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

Nous sommes comme des nains assis sur des épaules de géants. Si nous voyons plus de choses et plus lointaines qu'eux, ce n'est pas à cause de la perspicacité de notre vue, ni de notre grandeur, c'est parce que nous sommes élevés par eux. (Bernard de Chartres [attr.])

Historique

Version	Date	Commentaires
2.0	06/04/2021	Mise en forme.