Overview

Development time: Coding 20% – Maintaining 80%

Entities Naming

Variables, constants, subs and functions must be identified. Allowed chars: unaccented chars, numbers, underscore (_).

An identifier can’t start with a number nor contain a space.

Do not use any Basic keyword to name an entity!

Easy to read names CamelCase, Name_with_separators

Explicit names IsCell(), SaveSpreadsheet()

Comments

¢ (apostrophe) or REM. What follows is a comment.
¢ Comments as are important as code! They apply to the current line only.

Code Indent

Indented code is easier to read. Indent each code level with Space / Tabulation

Continuing An Instruction On The Next Line

Last two chars or the first line: (space + underscore).

Variables

¢ By default, variable declares are not mandatory but this is dangerous (typos lead to double declares).
¢ Adding Option Explicit on top of a module forces variable declaration.

Declaring Variables

Variable : a memory place. A variable contents may be modified at run-time.

Simple Variables

Dim MyVar As AType
Ex : Dim MyText As String

Arrays

¢ Array indices are zero-based!

Dim MyArray(NumDim) As AType
Number of dimensions : any.
Dim MyArray(2,4) : 2 dimensions. 3 items for the 1st, 5 for the 2nd (base 0).
Dim MyArray(9) As AType
Declares a 1-dim array with 10 items (base 0).
Dim MyArray() As AType
Declares an array of unknown dimension. Calling Dim MyArray(1) will be required.
Dim MyArray As Variant
Declaring Arrays Items

MyArray(1, 3) = Value
Sets value to item 1, 3
Inf = LBound(MyArray()) [n]
Lower bound for dimension n.
Sup = UBound(MyArray()) [n]
Upper bound for dimension n.

Redimensioning

Redim MyArray(NewDim) With data loss.
Redim Preserve MyArray(NewDim) Without data loss.

Emptying

Erase MyArray() or use ReDim with data loss.

Test if Empty
IsEmpty = LBound(MyArray()) = 0 And UBound(MyArray()) = -1

Test if Exists
Exists = Not IsNull(MyArray) And Not IsEmpty(MyArray)

Setting Non-Object Variables

MyVar = SomeValue
¢ Basic often automatically typecasts when SomeValue is not of the same type as MyVar. Prefer typecasting explicitly using dedicated functions (Cxx(), ReCard #5).

Creating/Setting Object Variables

Dim MyObject As New AClass
MyObject = New AClass
Set MyObject = AClass

Initialization is immediate

Variables Visibility

Declaring...
gives visibility...

Dim MyVar As AType
In the current subprogram or module.
Static MyVar As AType
In the current subprogram.
Private MyVar As AType
In the current module.
Public MyVar As AType
In the current library.
Global MyVar As AType
In all libraries.

Type

Specifies the value set a variable can carry or a function return.

Predefined Types

Type name | Description | Initialized to
----------|-------------|----------------
Boolean   | Logical values True / False. | False
Byte      | Integer numbers (8 bits), from 0 to 255. | 0
Currency  | Currency numbers (4 decimals). | 0.0000
Date      | Dates and hours. In fact, doubles. | 0.0
Double    | Floating numbers (64 bits). | 0.0
Integer   | Integer numbers (16 bits), from -32 768 to +32 767. | 0
Long      | 32bit integers, -2 147 483 648 to 2 147 483 647. | 0

LibreOffice Basic
LibreOffice v. 5.3.3 – Platform: All
LibOffice RefCard
LibreOffice Basic Overview
v. 1.10 – February 11, 2018

Written using LibreOffice v. 5.3.3 – Platform: All

Type name | Description | Initialized to
----------|-------------|----------------
Object    | Objects. Allow to manipulate LibreOffice API objects. | Null
Single    | Floating numbers (32 bits). | 0.0
String    | Text (0 to 65 545 characters). | "" (null length)
Variant   | Any type, incl. object. | Empty

Every time a type is unspecified, Variant is implicit.
Always set initial values rather than rely upon implicit settings.

Custom Types

Type MyType | member1 As AType | member2 As AType | ’etc.
End Type

A custom type may only be referenced in the module where it is declared. This code is not possible elsewhere: Dim MyVar As New MyType

To create a var of this type in any other module, create a function that realizes the creation, within the same module as the type declaration. You then call that creation function in order to create an instance of that type:

Function CreateMyType() As MyType
Dim Result As New MyType
CreateMyType() = Result
End Function

Usage elsewhere (other modules):
Dim MyVar As Object
MyVar = CreateMyType()

Objects

LibreOffice offers many classes (aka services) to manipulate documents and their components. Service = Properties + Methods. An object is an instance of a service.

Property State (= variable) | Method Action (= subprogram)
Synced.object, SomeProperty or object, SomeMethod.

Empty, Null And Nothing

Empty
Uninitialized variable yet. Empty assignment possible.
Null
Invalid contents. Null assignment possible.
Nothing
(objects only) No (more) reference to the object. Assignation possible.

Functions

IsEmpty(SomeVar) Variable is empty.
IsNull(SomeObject) Unusable data.

Operators

Booleans
Not Not And And
Or Or (inclusive) Xor Exclusive or

Comparisons (return True Or False)
= Strictly equal < Strictly lower <= Lower or equal
< Strictly upper >= Upper or equal

Mind to floating numbers comparisons!

Numerical
+ Addition / Subtraction
\ Division \ Mod \ Integer division
^ Exponentiation & Raising to the power

& Strings concatenation (fusion) (’*’ is possible; better not use because of its ambiguity).

Constants

Constant: a memory place; fixed value (immutable during execution).
Declaring Constants
Const MYCONST = SomeValue
SomeValue must be a simple type: no array, no object.

Naming Constants
It is frequent to name constants in all UPPERCASE.

Constants Visibility

Declaring...
gives visibility...
Const MYCONST = SomeValue
In the current subprogram or module.
Public MYCONST = SomeValue
In the current library.
Global MYCONST = SomeValue
In all libraries.

File Paths

To ensure multi-platform compliance, file paths are often expressed using the URL format: file:///support/path/to/afile.txt instead of the native OS format.

Two functions allow to switch from one to the other representation:
From OS native to URL
URLname = ConvertToURL(NativeName)
From URL to OS native
NativeName = ConvertFromURL(URLname)

Subprograms

Ensure arguments ↔ parameters correspondence, in number and type.
Premature subprogram exit Exit Sub, Exit Function
Sub
Executes an action.

Naming hint: verb at the infinitive: DoXXX(), etc.

Declaration
Sub SubName(parameters)

Structure
Sub SubName(parameters)

Instructions
End Sub
Use
SubName(arguments). If no argument: SubName()
Calling A Command Associated With A LibreOffice Menu

101

Use the Dispatcher, and pass it the wanted UNO menu command.

Knowing UNO Menus Commands

UNO menu commands: see the menubar.xml files in the LibreOffice installation directory (OS specific), under share/config/sofficecfg/modules. Subdir menubar of the wanted module (eg: sglobal/menubar.xml etc.).

All commands start with .uno:

Ex: ".uno:open" (File > Open), ".uno:OptionsTreeDialog" (Tools > Options), etc.

Program Skeleton

Dim Frame As Variant
Dim Dispatch As Object
Dim Args() As Variant  'contents depends from context
Dim unoCed As String
Frame = ThisComponent.CurrentController.Frame
unoCed = "UNO command to run (above)
Dispatch = createUnoService("com.sun.star.frame.DispatchHelper")
Dispatch.executeDispatch(Frame, unoCed, ",", 0, Args())

where unoCed is the command found in the files above.

Examples

(only modified parts are shown)

Ex1. Calling Print Preview

Dispatch.executeDispatch(Frame, ",.uno:PrintPreview", ",", 0, Args())

Ex2. Showing/Hiding The Sidebar

Dim Args() As New com.sun.star.beans.PropertyValue
Args(0).Name = "Sidebar"
Args(0).Value = True  'or False depending on aim
Dispatch.executeDispatch(Frame, ",.uno:Sidebar", ",", 0, Args())

Error Management

In Basic, error management is available using:

• On Error Xxx: instructions for error interception;
• Err, Erl and Error: functions to get information about the last error met.

Error Information Functions

Err  The error code
• An error code of 0 (zero) means "no error".
Use If Err Then ... to check error presence.

Error  The message that describes the error.
Erl  The line number where the error occurred.

You may create custom errors by setting a value to Err:

• Er = 1234 generates error 1234.

On Error – Interception Errors

◊ Error interception is active as long as it has not been canceled
On Error Goto MyLabel  Activates error interception. If an error occurs, the execution continues to MyLabel.

◊ In the program body, you must define the label
MyLabel: (beware to the semicolon character).

On Error Resume Next  Cancels error interception.

◊ In a Sub or Function, you might prefer the On Local Error Xxx syntax. This doesn't require calling On Error Goto 0 to cancel error interception: canceling is automatically performed when leaving the Sub or Function.

◊ On Local Error Goto Xxx has precedence on any preexisting On Error Goto Xxx.

Different Ways Of Running A Macro

◊ Method  LibreOffice  Document Type  Current Document

| Using a toolbar button | ⬤ | ⬤ |
| Using a menu | ⬤ |
| Using a shortcut | ⬤ |
| Through an event | ⬤ |

Loading A Code Library

For readability and maintainability, organize your code in several libraries (RefCard #1).

◊ The standard code library is the only loaded library at document opening. Others must be explicitly loaded to gain access to their code.

◊ Library names are case sensitive!

Loading From The Local Container (document)

Checking existence 
LibExists = BasicLibraries.hasByName("MyLib")
Loading 
BasicLibraries.loadLibrary("MyLib")

Loading From A Global Container

Same as above but BasicLibraries is replaced with GlobalScope.BasicLibraries.

◊ Mind to identifiers collisions between libraries! You may qualify names using:
container.library.module.name (all or part).
Ex GlobalScope.Tools.Strings.ClearMultiDimArray(MyArray, 3)

Credits

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

We are like dwarves perched on the shoulders of giants, and that we are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because we are carried aloft and elevated by the magnitude of the giants (Bernard de Chartres [attr.])

History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.10</td>
<td>02/11/2018</td>
<td>First EN version</td>
</tr>
</tbody>
</table>

License

This refcard is placed under the CreativeCommons BY-SA v4 (intl) license.

More information: https://creativecommons.org/licenses/by-sa/4.0/