4 formas de acelerar y optimizar tus macros excel

Esta información es muy útil para quienes manejen el tema de **programación de macros excel.** ¿Tus macros van lentas? ¿Problemas a la hora de ejecutarlas? ¿Cuáles son las técnicas recomendadas?

Cuando de **programación de macros excel** se trata, el tema de la eficiencia y la velocidad es clave. Hay 2 leyes fundamentales que hay que recordar:

a. Cuanto menos código tiene una macro mejor…¿por qué?
Ayuda a que la macro se ejecute mucho más rápido
Simplifica la tarea a la hora de modificar/ampliar/reparar la macro

b. Cuanto más rápido se ejecuta una macro mejor!…¿por qué?
Mejora la experiencia del usuario
No mantiene la PC ocupada tanto tiempo

Respecto de usar menos código dependerá de las habilidades del **programador excel** en cuestión. Hemos visto infinidad de casos donde 30 o 40 líneas de código VBA se pueden resumir en 5 o 6 líneas (algo similar pasa con las fórmulas excel). Siempre hay macros o fórmulas que hacen la tarea de forma más directa y sin dar tantas vueltas!

Otra recomendación clave es **invertir mucho tiempo inicial en planificar y analizar la lógica del trabajo.** Esto nos va a ahorrar muchos problemas y dolores de cabeza posteriores!

Hay algunas instrucciones puntuales que siempre conviene usar y que **van a acelerar y optimizar nuestras macros en todos los casos**. Vamos a ver repasar algunas técnicas puntuales que podemos usar al comienzo, durante y al final de nuestras macros.

AL COMIENZO DE LAS MACROS

1. Apagar el parpadeo de pantalla
Lo hacemos con la instrucción: *Application.screenupdating=False*
*E*vita los movimientos de pantalla que se producen al seleccionar celdas, hojas y libros

2. Apagar los cálculos automáticos
Lo hacemos con la instrucción: *Application.calculation=xlCalculationManual*
Evita que se recalcule todo cada vez que se pegan o modifican datos

3. Apagar los eventos automáticos
Lo hacemos con la instrucción: *Application.EnableEvents=False*
Evita que se disparen macros de evento si las hubiere

4. Apagar visualización de saltos de página
Lo hacemos con la instrucción: *ActiveSheet.DisplayPageBreaks = False*
Sirve para evitar algunos problemas de compatibilidad entre macros Excel 2003 vs. 2007/2010

En resumen, siempre debemos comenzar las macros así:

Application.screenupdating=False
Application.calculation=xlCalculationManual
Application.EnableEvents=False
ActiveSheet.DisplayPageBreaks = False

AL FINAL DE LAS MACROS

5. Borrar contenido de portapapeles
Lo hacemos con la instrucción: *Application.CutCopyMode = False*
Permite limpiar el portapapeles en caso de haber copiado datos
Además debemos volver a su estado original las instrucciones con las que comenzamos la macro.

En resumen, siempre debemos finalizar las macros así:

Application.screenupdating=True
Application.calculation=xlCalculationAutomatic
Application.EnableEvents=True
ActiveSheet.DisplayPageBreaks = True
Application.CutCopyMode = False

OTRAS TECNICAS UTILES

6. Usar la instrucción WITH
Se usa para evitar tener que referenciar un mismo objeto muchas veces

Ejecución leeenta…
```
Sheets(1).Range("A1:Z1").Font.Italic = True
Sheets(1).Range("A1:Z1").Font.Interior.Color = vbRed
Sheets(1).Range("A1:Z1").MergeCells = True
```

Ejecución rápida!
```
With Sheets(1).Range("A1:Z1")
.Font.Italic = True
.Font.Interior.Color = vbRed
.MergeCells = True
End With
```

7. Evitar la instrucción SELECT

Se genera sobre todo en las macros grabadas
La mayoría de las veces no es necesario seleccionar para cumplir el objetivo

Ejecución leeenta…
```
Range("E1").Select
Selection.Copy
Range("D10").Select
ActiveSheet.Paste
```

Ejecución rápida!
```
Range("E1").Copy Range("D10")
```

8. Evitar loops FOR EACH

Tener que ir celda por celda consume mucho tiempo
Se puede resolver el problema de forma más directa!

Ejecución leeenta…
```
For Each cell In Range("A1:A10000")
If cell = Empty Then cell = 0
Next cell
```

* Los loops siempre son leeentos
* En este caso recorre 10.000 celdas!

Ejecución rápida!
Existen diversas formas de evitar los loops. La solución dependerá del caso concreto en cuestión. Generalmente se usan algunas de estas técnicas: agrupar, ir a especial, filtros, filtros avanzados. La idea es poder realizar la acción sobre todos los elementos al mismo tiempo, en lugar de tener que ir uno a uno!

9. Usar las funciones nativas de Excel

No quieras reinventar la rueda. Quizás ya exista una función Excel que lo haga!

Las macros siempre ejecutan más rápido las funciones nativas de Excel

```
Ejecución leeenta…
mProducto = 1
For i = 1 to 100
mProducto = mProducto * Cells(3,i)
Next
Ejecución rápida!
mProducto = Application.WorkSheetFunction.Product(Range("C1:C100"))
```

10. Forzar la declaración de variables

En el editor VBA, menú Herramientas > Opciones > pestaña Editor > marcar "Requerir declaración de variables"
Luego usar la variable correcta: si es fecha usar Date, si es texto usar String, si es valor usar Long…
Evitar el uso de la variable Variant ya que insume más recursos…
Usar nombres de variables que nos digan algo (por ej. "UltimaFila" o "FilaZ" en lugar de "f" o "uf")

11. Escribir las macros en módulos y no en hojas

Las hojas pueden ser borradas o copiadas y esto generaría problemas inesperados

12. Separar el proceso en varias macros (divide y conquistarás)

Si tu macro hace muchas cosas conviene separarla en muchas macros pequeñas y luego unirlas
Es más fácil para controlar, auditar, etc…
Además te permite luego poder rehusar alguna parte del proceso en otras macros

```
Macro muy laaarga…
Sub MegaMacro()
'Codigo limpia datos
'Codigo carga datos
'Código arregla datos
'Código arma reporte
End Sub()
```

```
Mejor dividir en diferentes macros para cada proceso
Sub LimpiaDatos()
'Codigo…
End Sub Sub
```

```
CargaDatos()
'Codigo…
End Sub

Sub ArreglaDatos()
'Codigo…
End Sub

Sub ArmaReporte()
'Codigo…
End Sub
```

Finalmente podemos unir todos los procesos

```
Sub ProcesoCompleto()
Call LimpiaDatos
Call CargaDatos
Call ArreglaDatos
Call ArmaReporte
End Sub()
```

13. Ser cuidadoso con la instrucción ON ERROR RESUME NEXT

Esta instrucción hace que la macro siga avanzando aunque encuentre un error
En algunos casos esto hará que se ignoren errores que no deberían ser ignorados
Podrías tener errores (bugs) y no enterarte!

14. Comentar bien las macros

¿Qué pasaría si tuvieras que volver a revisar/arreglar/ampliar tu código 8 meses después?
Añadir comentarios te ayudará a describir y recordar la lógica y te ahorrará mucho tiempo!