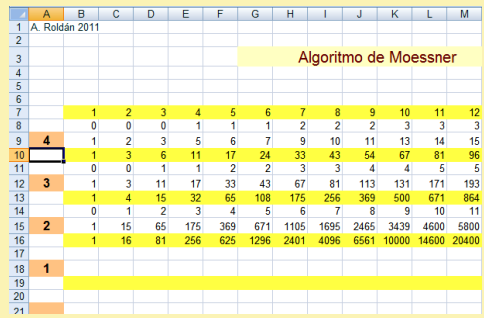
La hoja de cálculo



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	A. Roldán 2011												
2													
3							Algoritmo de Moessner						
4													
5													
6													
7		1	2	3	4	5	6	7	8	9	10	11	12
8		0	0	0	1	1	1	2	2	2	3	3	3
9	4	1	2	3	5	6	7	9	10	11	13	14	15
10		1	3	6	11	17	24	33	43	54	67	81	96
11		0	0	1	1	2	2	3	3	4	4	5	5
12	3	1	3	11	17	33	43	67	81	113	131	171	193
13		1	4	15	32	65	108	175	256	369	500	671	864
14		0	1	2	3	4	5	6	7	8	9	10	11
15	2	1	15	65	175	369	671	1105	1695	2465	3439	4600	5800
16		1	16	81	256	625	1296	2401	4096	6561	10000	14600	20400
17													
18	1												
19													
20													
21													

Edición Otoño 2015

Colección Hojamat.es

http://www.hojamat.es

PRESENTACIÓN

A muchos lectores les extrañará que existiendo hoy en día herramientas muy sofisticadas, usemos la hoja de cálculo para temas que exigen más potencia. Lo tomamos como un reto, es como quien prefiere ir en bicicleta aunque tenga el coche en el garaje. Por otra parte, en ellas aparecen los cálculos de forma muy clara y ordenada y permiten tres niveles: el simple cálculo, los esquemas con cálculos enlazados y la programación en Basic. Con ellos es más que suficiente para transmitir ideas de una forma entretenida y clara.

Otro motivo para su uso es la cantidad de funciones que el autor ha implementado en ella, que facilitan búsquedas y hallazgos de curiosidades. Si bien la mayoría de estas funciones no se están publicando, permiten el estudio de temas que de otra forma resultarían algo pesados de gestionar.

Como advertiremos en todos los documentos de esta colección, el material presentado no contiene desarrollos sistemáticos, ni pretende ser un manual teórico. En cada tema se incluirán cuestiones curiosas o

relacionadas con las hojas de cálculo, con la única pretensión de explicar algunos conceptos de forma amena.

TABLA DE CONTENIDO

FUNCIONES DEFINIDAS

En ocasiones desearás definir otras funciones además de las que una hoja de cálculo te ofrece. Por ejemplo, sería útil una función tal que si se aplica a un número entero positivo devuelva su mayor divisor. Para ello disponemos del lenguaje Basic (de macros) que llevan incorporado Calc y Excel. Coincide en gran parte de estas hojas, y sólo se diferencia en el tratamiento propio de hojas y celdas, pero en el caso de las funciones apenas existe diferencia en la forma en la ue debemos proceder.

A partir de esta entrada iremos explicando en otras sucesivas la forma de implementar algunas funciones. Antes de nada hay que aprender a definirlas en el Editor de Basic.

Editor de Basic

Crea una hoja o abre alguna existente. Para abrir el editor sigue la secuencia:

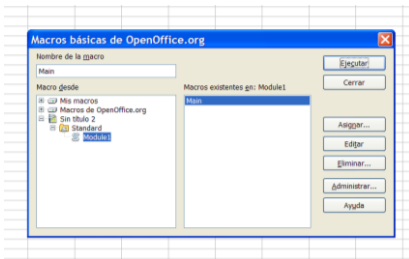
En Calc (OpenOffice o LibreOffice): **Menú Herramientas > Macros > Organizar macros > OpenOffice.org Basic.**

En Excel: **Ficha Programador > VisualBasic** (En algunos equipos no verás la ficha *Programador* y deberás activarla desde **Opciones de Excel > Más frecuentes**)

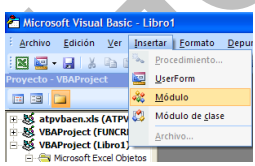
Uso de un Módulo contenedor

Si es la primera función que defines, busca la carpeta correspondiente al nombre de tu hoja de cálculo (si lo acabas de crear, se llamará Sin Nombre o Sin Título). Si ya has definido algunas funciones, habrás definido un módulo contenedor. Abre ese módulo.

Si no habías creado ningún módulo, una vez elegida la carpeta, pulsa el botón **Nuevo** para abrir un nuevo módulo contenedor. Se te ofrecerá el nombre de *module1, module2* u otro similar. Acepta el nombre o cámbialo según tu criterio. Al aceptar el nombre se abrirá el editor de macros.



Esta es la imagen que te aparece en OpenOffice y LibreOffice una vez creado el módulo. Pulsando Editar podrás comenzar a escribir código.



En Excel también tendrás que crear un módulo nuevo cuando entres en el Editor de Basic. Usa el comando **Insertar** y elige Módulo

Escritura del código

Terminada la secuencia anterior, borra lo que esté escrito de la macro Main (en Calc) y escribe el código de una función:

Debes comenzar con

Public function nombre de la función (argumento)

y terminar con

End function

y entre ambas, el código de la función. El argumento es la variable sobre la que actuará la función. En ese código debemos usar el nombre de la función seguida del signo igual y de su definición

Es mejor verlo con un ejemplo:

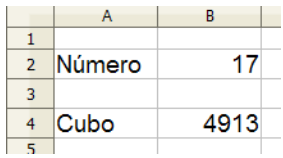
```
Public function cubo ( numero )
cubo=numero*numero*numero
End function
```

En el ejemplo, el nombre de la función es cubo, y su argumento numero (lo traduciríamos como "Cubo de un número")

Después volvemos a escribir cubo, el signo igual, y su definición.

Uso de la función

Una vez escrito el código, cierra el Editor de Basic y usa tu función en cualquier celda. En la imagen puedes ver que en B2 se ha escrito un número y en B4 la fórmula =CUBO(B2). En excel la función de autocompletar también funcionará para tus funciones.



	A	B
1		
2	Número	17
3		
4	Cubo	4913
5		

Con esto ya tienes definida la función.

Con la técnica explicada, esa función sólo estará activa en la hoja de cálculo en la que la has creado, no en otras. Al cerrar la hoja ya no podrás usarla.

SOLVER NOS AYUDA EN UN PROBLEMA

Hemos leído el siguiente problema en el blog **"Problemas matemáticos"**

(http://problemate.blogspot.com)

La igualdad 2008 = 1111 + 444 + 222 + 99 + 77 + 55 es un ejemplo de descomposición del número 2008 como suma de números distintos de más de una cifra, cuya representación (en el sistema decimal) utiliza un sólo dígito.

i) Encontrar una descomposición de este tipo para el número 2009.

ii) Determinar para el número 2009 todas las posibles descomposiciones de este tipo que utilizan el menor número posible de sumandos (el orden de los sumandos no se tiene en cuenta).

Nos ha parecido de interés para comentar las posibilidades de la herramienta Solver, tanto de Excel como de Calc (tan sólo se diferencian en pequeños detalles) para ayudarnos a encontrar una de las descomposiciones pedidas.

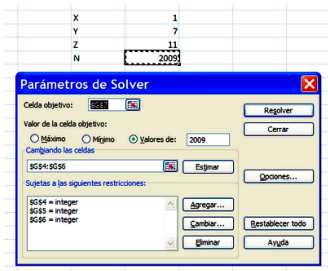
Este problema se puede plantear a partir de esta ecuación diofántica: N=1111x+111y+11z (si N es un número de cuatro cifras), para una vez hallados x, y, z, descomponerlos en números de una cifra del 1 al 9, para cumplir lo pedido por el problema.

Para ello podemos escribir en la hoja de cálculo y en columna, tres conjeturas para esas variables, supongamos que en las celdas G4, G5 y G6. Después, en la G7 escribimos la fórmula =1111*G4+111*G5+11*G6. Esta celda es la que deseamos que tenga el valor 2009, según pide el problema.

Elegimos la herramienta Solver, en la que concretaremos estos datos:

- Los contenidos de las celdas han de ser enteros positivos
- Como tipo de resolución no elegimos máximo o mínimo, sino **un valor concreto**
- La celda G7 ha de tener el valor 2009, por ejemplo

  En la imagen puedes estudiar el proceso de resolución



La solución obtenida, X=1, Y=7, Z=11 se puede descomponer, por ejemplo, en

2009=1111+777+66+55

Puede darte un valor de z mayor que 101, con lo que podemos pasar parte del mismo a X, ya que 1111=101*11

Como ves, el método te demanda unos ajustillos posteriores, pero es curioso que Solver ayude en este tipo de cuestiones.

FUNCIÓN CIFRAS_DISTINTAS

(A propósito de una entrada de Claudio Meller)

¿Cuál es el mayor número que entre él y su doble, no tienen dígitos repetidos?

Cuando quise emprender la búsqueda me di cuenta de que tenía que comenzar con el 49000 y después ir descendiendo en los ensayos hasta dar con el número pedido. Era una tarea para encargársela al ordenador y que él trabajara por mí.

Cuando planifiqué esta búsqueda vi que sería muy útil disponer de una función que me devolviera un 1 si su argumento no tuviese cifras repetidas o un 0 en caso contrario (preferí 1 y 0 porque es más cómodo que VERDADERO o FALSO). Pues bien, el estudio de esta función es el objetivo de esta entrada.

Función cifras_distintas

Para construir esta función sobre un argumento N son necesarios tres cálculos al menos:

- Encontrar el número de cifras de N
- Saber encontrar la cifra de N que ocupe el lugar K
- Comprobar si las cifras son distintas o existe alguna repetición.

Número de cifras

Un número entero tiene K cifras si está comprendido entre 10^{K-1} y 10^{K} por lo que tomando logaritmos decimales, K-1 <= log (N) < K

```
Public function numero_cifras(n)
numero_cifras=int(log(n)/log(10))+1
end function
```

Se declara Public para que se pueda usar como una función más de Excel o Calc y el dividir entre log(10) se justifica porque LOG se corresponde con el logaritmo natural.

Selección de cifras

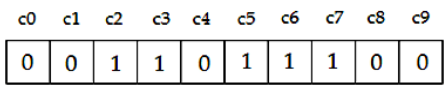
Para extraer una cifra de un número entero podemos acudir a calcular su cociente entero respecto a 10^{K-1}, y nos resultará el mismo número pero escrito sólo hasta la cifra K. Así INT(23543/100) se nos convierte en 235. Si ahora lo dividimos entre 10, desaparecerá la cifra K: INT(235/10)=23. Con algún pequeño detalle más llegamos a la función deseada. Si el número no posee tantas cifras nos devolverá un cero.

```
Public function cifra(m,n)
dim v
v=int(m/10^(n-1))
v=v-10*int(v/10)
cifra=v
end function
```

Función cifras_distintas

Ya sabemos contar las cifras de un número y separarlas unas de otras. Ahora hay que recorrerlas y detectar si alguna se repite.

Para ello crearemos diez memorias C, que se rellenarán con un 1 si la cifra existe en el número, y con un 0 si no existe. Así, cuando se recorre el número, se va marcando con un 1 cada cifra encontrada, pero si posteriormente nos encontramos con un 1 al marcar, es que existe una repetición. En la imagen vemos cómo se rellenarían las memorias con el número 27653:



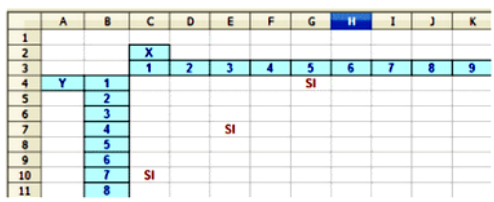
Con esto ya tenemos preparado todo para la función. Se transcribe a continuación un posible listado en Basic con comentarios en cursiva:

Public function cifras_distintas(n)
dim nc,i,d
dim c(10)
dim vale
nc=numero_cifras(n) *Se cuentan las cifras y se almacenan en la memoria* ***nc***
for i=0 to 9:c(i)=0:next i *Ponemos todas a cero*
vale=1 *Provisionalmente damos el número como con cifras distintas*
i=1 *Iniciamos el recorrido*
while vale=1 and i<=nc *Mantenemos el recorrido mientras no haya repetición*

```
d=cifra(n,i)                    Leemos la cifra
if c(d)=0 then                  Si no está ya marcada, la marcamos
c(d)=1
else
vale=0                          Si está marcada hay repetición y hacemos vale=0
end if
i=i+1
wend                            Fin del recorrido
cifras_distintas=vale
end function
```

EXPLORACIÓN DE SOLUCIONES

Esta propuesta, más que de Matemáticas, es de uso de una hoja de cálculo. Consiste en crear un instrumento que nos permita explorar las soluciones de una ecuación diofántica de dos variables (eventualmente tres). Por ejemplo, la solución más pequeña, con enteros positivos, de la ecuación $x^3+y^4 = z^5$ está formada por potencias de 2. ¿Cómo comprobarlo con una hoja de cálculo?



Proponemos una estructura sencilla, como la de la

imagen, en la que las posibles soluciones de X están situadas en la primera fila y las de Y en la primera columna, con los valores que deseemos darles, tanto en cantidad como en tipo de números:

El secreto de este instrumento es la fórmula que escribimos en la celda C4, y que después extenderemos a toda la tabla. En la imagen se ha programado la búsqueda de soluciones enteras positivas para la ecuación 3X+2Y=17. Se ha escrito en C4 la fórmula =SI(2*$B4+3*C$3=17;"SI";""). Obsérvese el uso del signo “$” para que al extender la fórmula a toda la tabla se respete la columna B y la fila 3.

Con esta fórmula, cuando 3X+2Y sea igual a 17, aparecerá la palabra “SI” y en caso contrario quedará en blanco.

En este ejemplo, como en todas las ecuaciones lineales, se observa que las soluciones forman sucesiones aritméticas (se sitúan en línea recta): X=1, X=1+3=4; X=4+3=7 Y=5; Y=5-2=3; Y=3-2=1 Esta observación es muy útil si se usa en el aula.

Se incluyen a continuación algunas propuestas por si deseáis construiros vuestro propio explorador:

(1) El ejemplo anterior de $x^3+y^4=z^5$ necesitaría una fórmula tan complicada como esta:

=SI(($B4^3+C$3^4)^(1/5)=ENTERO(($B4^3+C$3^4)^(1/5));"SI";"")

Y se obtendría la solución X=64 Y=256 y, mediante un cálculo, Z=32. Inténtalo.

(2) Prueba a encontrar las soluciones de la ecuación de Pell

$X^2-3Y^2=1$. Las primeras son:

X=2, Y=1; X=7,Y=4; X=26, Y=15 ¿Puedes aportar alguna más?

(3) La ecuación X*Y*(X+Y)=4860 posee las soluciones X=12 Y=15 (y su simétrica) Para tener la seguridad de que no existen otras podríamos rellenar la fila de X y la columna de Y con todos los divisores de 4860. ¿Hay más?

Como indiqué al principio, esta es una idea sencilla para comprobar resultados y realizar algunas investigaciones.

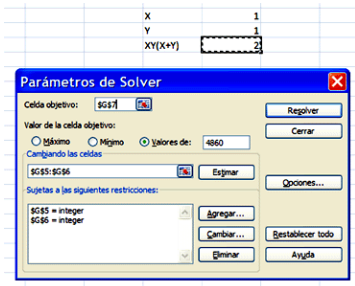
Solver

También puedes usar la herramienta Solver de Excel (la de OpenOffice.org Calc sólo resuelve el caso lineal). Sitúas el valor X en una celda, el de Y en otra, y el primer miembro de la ecuación (como fórmula) en una tercera.

Marcas en Solver el valor de la celda objetivo, y declaras X e Y como enteras. Si nos apoyamos en valores cercanos a una posible solución, es fácil que la obtengas. Al ser un problema generalmente

indeterminado, este método necesitará nuestra ayuda mediante valores cercanos apropiados.

En la imagen se ha planteado la ecuación XY(X+Y)=4860 con valores de apoyo X=1, Y=1. Al pedir Resolver nos devuelve las soluciones 12 y 15.



OBTENCIÓN DE LA LISTA DE DIVISORES

Algoritmo con hoja de cálculo

Para encontrar los divisores de un número N la búsqueda más simple consiste (salvo alguna pequeña modificación que la acelere) en recorrer todos los números menores o iguales a N y tomar nota de los que son divisores suyos. Es muy sencilla de programar y sólo ocupa unas pocas líneas de código. Tiene el inconveniente de que para números de más de cuatro o cinco cifras decimales resulta muy lenta en hoja de cálculo, que es la herramienta que usamos en este blog.

Un algoritmo más rápido consiste en reproducir en la hoja el esquema que siempre se ha usado en las clases de Matemáticas, el que desarrolla las distintas potencias del primer divisor primo y después las combina con las de los demás en una tabla bidimensional como la siguiente, que se corresponde con los divisores del número 33075=33*52*72

3	1	3	9	27
5	5	15	45	135
	25	75	225	675
7	7	21	63	189
	35	105	315	945
	175	525	1575	4725
	49	147	441	1323
	245	735	2205	6615
	1225	3675	11025	33075

En ella se han situado en la primera columna los factores primos, en la primera fila las potencias del 3 y en las demás filas los distintos productos que se pueden formar entre las potencias, sumandos formados en la fórmula

$$\sigma(N) = \prod \frac{{p_i}^{e_i+1} - 1}{p_i - 1} = \prod (1 + p_i + {p_i}^2 + \cdots {p_i}^{e_i})$$

de la obtención de sigma(N).

Para poder construir este esquema en la hoja necesitamos saber qué factores primos posee el

número y con qué exponentes. La siguiente subrutina en Basic nos los proporciona:

Global p(50) as long Se definen las variables p para los factores, e los exponentes y np el número total
Global e(50) as integer
Global np%

Sub sacaprimos(a)
Dim f,i
dim vale as boolean

f=2:i=1:e(i)=0:vale=false:np=0
while f<=a El bucle while_wend divide el número entre el factor todas las veces posibles
if esprimo(f) then
while a/f=int(a/f) es divisor primo
vale=true
p(i)=f
e(i)=e(i)+1 aumenta el exponente
a=a/f
wend
if vale then
np=np+1 aumenta el número de factores
i=i+1:e(i)=0 se inicia un nuevo exponente
end if
vale=false
end if
f=f+1 buscamos otro factor primo
wend
End sub

Con este código obtenemos la lista de factores primos p(i), la de exponentes e(i) y el número total de factores primos np.

Al usar estos datos la búsqueda de divisores se simplifica mucho, pues todo el trabajo consistirá ahora en construir la tabla que estudiamos en nuestros años escolares:

(a) Formamos una fila con las potencias posibles del primer factor primo. Tomamos nota de que la altura de la tabla es de 1. También recogeremos el dato de la variable fila en la que se han escrito.

(b) Vamos multiplicando esas potencias de la primera fila por todas las de los otros primos, pero teniendo cuidado de:

- En cada nuevo primo la altura queda multiplicada por e(i)+1, que es el número de sus potencias posibles.

- En cada nuevo producto deberemos incrementar la variable fila en una unidad, para que se vaya formando la tabla hacia abajo.

- Deberemos usar muchos bucles anidados, porque intervienen as variables del número de potencias de la primera fila, el de las del resto de primos y las del número de primos diferentes.

Un posible código en OpenOffice sería:

sub todosdivisores

dim i,j,k,l, altura,fila
dim divi as long

Rellena la primera fila con las potencias del primer primo
if np>=1 then
StarDesktop.CurrentComponent.sheets(1).GetCellByPosition(2,11).value=p(1)
for k=0 to e(1)
StarDesktop.CurrentComponent.sheets(1).GetCellByPosition(3+k,11).value=p(1)^k
next k
end if

Va multiplicando las potencias de los demás primos
if np>=2 then
altura=1:fila=12
for k=2 to np Este bucle recorre los primos
StarDesktop.CurrentComponent.sheets(1).GetCellByPosition(2,fila).value=p(k)
for j=1 to e(k) Recorre las potencias del primo actual
for i=1 to altura Recorre todos los divisores anteriores
for l=0 to e(1) Ídem los elementos de cada fila
divi=StarDesktop.CurrentComponent.sheets(1).GetCellByPosition(3+l,10+i).value
divi=divi*p(k)^j Efectúa el producto y lo escribe
StarDesktop.CurrentComponent.sheets(1).GetCellByPosition(3+l,fila).value=divi
next l
fila=fila+1 Una vez escritos, aumenta la fila
next i
next j

altura=altura*(e(k)+1) La altura se amplía
next k
end if
end sub

Pues ya tienes una idea. El resto es cosa tuya. Seguro que la mejoras.

SIMULACIÓN PARA VAGOS

El otro día, buscando temas por ahí, me topé con la secuencia http://oeis.org/A051293

1, 2, 5, 8, 15, 26, 45, 76, 135, 238, 425, 768, 1399, 2570, 4761, 8856, 16567, 31138… que representa el número de subconjuntos de {1,2,3,...,n} cuya media aritmética es entera.

Por ejemplo, en el caso de 4, los subconjuntos con media entera son {1}, {2},{3},{4},{1,3},{2,4},{1,2,3} y {2,3,4}, en total 8.

Me pareció un tema interesante e intenté abordarlo usando congruencias y después particiones, pero la cosa se complicó y me sentí *vago*. Quien tenga más disposición que yo puede seguir con ello. Además, el autor de esta secuencia en OEIS se vio obligado a conjeturar cosas. Malo.

¿Y si lo simulara? Yo nunca lo había intentado con conjuntos de este tipo y quizás podría reproducir la secuencia, al menos con algún pequeño error. Me puse a ello:

(a) Simulación de subconjuntos

Se puede usar la técnica de tirar una moneda: recorremos los elementos de {1,2,3,...,n} y para cada uno de ellos tiramos una moneda. Si sale *cara*, lo incluimos en el subconjunto, y si sale *cruz*, no lo incluimos. En los lenguajes de programación disponemos de la función ALEATORIO(), que en Basic es RND. Así que el algoritmo de formación de un subconjunto de {1,2,3,...,n} podría ser similar a este esquema:

Randomize
For i=1 to n
If rnd(1)>1/2 then … se incluye en el subconjunto
Next i

Para quienes no lo sepan, ***randomize*** hace que cada vez que usemos el algoritmo se forme una secuencia distinta de números aleatorios (en realidad, seudoaleatorios)

(b) Identificación de las medias aritméticas enteras

Como en nuestro caso nos interesa la media en cada subconjunto, las sentencias presentadas en Basic se podrían concretar en el sentido de que para cada

subconjunto tomemos nota del número de elementos y de su suma, para luego dividir y hallar la media.

Podemos definir la variable n para recoger el número de elementos y la variable s para formar la suma. En ese caso las sentencias anteriores se podían modificar así:

Randomize
n=0:s=0
For i=1 to n
If rnd(1)>1/2 then n=n+1:s=s+i
Next i
media=s/n

De esta forma obtendríamos la media de los elementos de cada subconjunto.

Ahora sólo nos quedaría comprobar si la media es entera. Esta prueba consistirá en ver si media=INT(media). Podemos generar muchos conjuntos aleatorios, por ejemplo 10000 y contar aquellos en los que la media es entera. En caso afirmativo incrementamos un contador.

Por último. Lo que marque el contador lo convertimos en proporción dividiendo entre 10000 y multiplicamos después por 2^n para que cuente subconjuntos. Después presentamos resultados. Aquí tienes un listado aproximado en Basic de Excel:

Randomize
a = 0 Contador de exitos

Input n o cualquier otra forma de capturar n

For i = 1 To 10000

ActiveWorkbook.Sheets(1).Cells(3, 3).Value = i esto sólo sirve para saber por dónde va la simulación
c = 0 cuenta elementos del conjunto aleatorio
s = 0 suma elementos del conjunto aleatorio
b = 1 / 2 es la media, que la declaramos al principio no entera.

For k = 1 To n se genera el conjunto aleatorio
m = Rnd(1)
If m < 1 / 2 Then c = c + 1: s = s + k: b = s / c
Next k

If b = Int(b) Then si es entero se incrementa el contador
a = a + 1
ActiveWorkbook.Sheets(1).Cells(fila, 6).Value = a / i * 2 ^ n Valor adaptado a 2^n
End If

Next i

Los resultados obtenidos han sido bastante acertados. Transcribimos los correspondientes a una sola pasada del algoritmo:

N	S(N)	Simulación	Errores relativos
3	5	5,0576	0,01152
4	8	8,031209363	0,00390117
5	15	14,8704	-0,00864

6	26	25,5872	-0,015876923
7	45	44,6464	-0,007857778
8	76	76,34683468	0,004563614
9	135	133,9707708	-0,00762392
10	238	236,032	-0,008268908

Como era de esperar, al aumentar N se presentan desviaciones mayores. Los errores relativos son más estables.

Pues hemos hecho el vago, pero con diversión.

FUNCIONES RECURSIVAS EN LAS HOJAS DE CÁLCULO

Cuando yo programaba hace años en Pascal se nos vendía su posibilidad de usar la recursión, es decir, que una función se llamara a sí misma, en declaraciones del tipo

Factorial(n)=n*factorial(n-1)

Esta y otras características nos hizo abandonar el Basic como un lenguaje más primitivo y que no admitía funciones recursivas ni por asomo. Pasados bastantes años dejé la confección de programas ejecutables y consiguientemente el Pascal. Ahora que mis trabajos, por voluntad propia, los restrinjo a las hojas de cálculo y a su Basic, no uso la recursión…hasta hoy.

Preparando una próxima entrada se me ocurrió usar funciones recursivas en Excel, OpenOffice y LibreOffice (en Google Docs no funcionan las macros en Basic) con la sorpresa de que sí funcionaban bastante bien.

Toda función recursiva contiene una llamada a sí misma, directa o indirectamente a través de otra función. Como esto nos puede llevar a un proceso sin fin, debe contener también un código de parada, que suele ser una definición en un caso concreto, como veremos en los ejemplos.

La recursividad no se resuelve hasta que no desemboca en ese caso de parada. Mientras tanto hay que guardar los datos pendientes situándolos en una pila. Por tanto, ahí está el único problema de usar la recursividad en las hojas, y es que se puede agotar la pila si se alargan mucho los cálculos, con el consiguiente mensaje de error. Un fallo de principiante es programar una función recursiva sin facilitar su salida. En ese caso el error será más grave aún: un cálculo sin fin.

Explicamos a continuación algunas funciones recursivas, comenzando con el factorial, que es la más popular y que nos servirá para explicar algunos detalles:

```
Public Function factorial(n)
Dim f
If n = 0 Then f = 1 Else f = factorial(n - 1) * n
```

factorial = f
End Function

Es fácil entender el código: Para evitar confusiones, comenzamos almacenando el factorial en la variable f, para al final recoger su valor en factorial. El cálculo de f es el clásico de la función n!: si n es cero, definimos el factorial como 1 y en caso contrario multiplicamos por n el factorial de n-1.

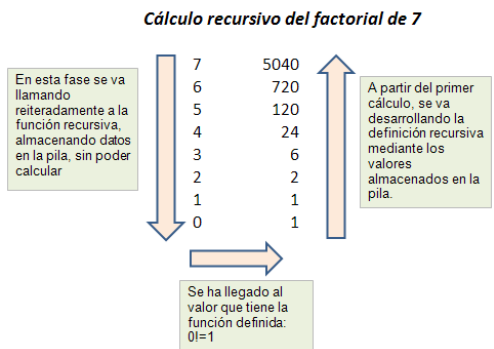
Prueba esta función en cualquiera de las tres hojas propuestas más arriba.

Este primer ejemplo contiene las dos partes imprescindibles en una función de este tipo:

- Alguna asignación de un valor concreto, que servirá para detener la primera fase de lectura de datos y comenzar los cálculos hacia atrás. Aquí es la asignación 0!=1

- La definición recursiva propiamente dicha, que, como es conocido, consiste en exigir que n!=n*(n-1)!

Lo explicamos con un esquema:



Al intentar calcular el factorial de 7, el programa se encuentra con una referencia al factorial de 6, guarda el 7 en la pila y se dedica a calcular el nuevo factorial. Como no puede, almacena el 6 encima del 7 (es una pila) y lo intenta con el 5, y así va de fracaso en fracaso (flecha descendente) hasta llegar al 0.

El valor 0 admite el cálculo, porque está definido como 1. Resuelto esto, es como si el programa se preguntara: ¿por dónde iba? Acude a la pila y ve un 1, con lo que ya puede calcular 1!=1*0!=1 y así sigue (flecha ascendente) buscando datos en la pila y resolviendo los cálculos según la definición recursiva.

Es evidente que para números grandes la pila se puede agotar por falta de memoria asignada.

Con esta función recursiva (la más inútil que me he inventado nunca) puedes tener una idea de la amplitud de la pila de tu hoja de cálculo.

```
Public Function identidad(n)
Dim i
If n = 1 Then i = 1 Else i = identidad(n - 1) + 1
identidad = i
End Function
```

La he elegido para que no influya la magnitud de los números, sino la cantidad de ellos que permanecen en la pila. Con ella he llegado la valor n=3270 como el último que no me da error. En los siguientes no consigo realizar el cálculo.

¿Qué margen tendrá tu hoja de cálculo? Prueba a ver.

Ejemplos varios

Si deseas el enésimo número triangular, sólo tienes que usar este código:

```
Public Function triang(a)
Dim p
If a = 1 Then p = 1 Else p = triang(a - 1) + a
triang = p
End Function
```

También se entiende bien: en los números triangulares vamos añadiendo en cada paso una base del triángulo nueva con a elementos. Prueba esta función y si quieres compara los resultados con la clásica fórmula Tn=n(n+1)/2.

¿Puedes analizar esta función?

```
Public Function cuad(a)
Dim p
If a = 1 Then p = 1 Else p = cuad(a - 1) + 2 * a - 1
cuad = p
End Function
```

¿Por qué produce como resultado el cuadrado de a? Este es un bonito ejemplo de elevar un número al cuadrado sin multiplicar en ningún momento.

Y ya que estamos con números poligonales, podríamos generarlos todos con una función recursiva única que dependiera de a y también del número de lados del polígono. ¿Te atreves con ella?

¿Y qué opinas de esta, con dos variables?¿Qué resultado produce?

```
Public Function combi(m, n)
Dim c
If n = 0 Then c = 1 Else c = combi(m, n - 1) * (m - n + 1) / n
combi = c
End Function
```

Ahora un ejemplo más serio:

En una entrada anterior (http://hojaynumeros.blogspot.com/2012/02/suma-de-los-elementos-de-todos-los.html) descubrimos que la suma de todos los elementos de los subconjuntos de un conjunto de n elementos venía dada por la fórmula de recurrencia

$$S_n=2S_{n-1}+ n*2n-1$$

y que da lugar a esta sucesión de valores en función de n

0, 1, 6, 24, 80, 240, 672, 1792, 4608, 11520, 28160, 67584, 159744, 372736, 860160, 1966080,… (http://oeis.org/A001788)

Si definimos una función según esta recurrencia podremos reproducir esta lista en nuestra hoja de cálculo. Podría ser esta:

```
Public Function sumaelem(n)
Dim s
If n = 1 Then s = 1 Else s = 2 * sumaelem(n - 1) + n * 2 ^ (n - 1)
sumaelem = s
End Function
```

Con ella hemos construido esta tabla que coincide con la de OEIS

1	2	3	4	5	6	7	8
1	6	24	80	240	672	1792	4608

Un ejemplo elegante

Define esta función de texto

```
Public Function simetrico$(a$)
Dim s$
If a$ = "" Then s$ = "" Else s$ = simetrico(Right$(a$, Len(a$) - 1)) + Mid$(a$, 1, 1)
simetrico = s$
End Function
```

Escribe una palabra en una celda y aplícale esta función desde otra celda ¿Cuál es el resultado?

Como ves, todo esto es bastante divertido, pero no muy útil a causa del agotamiento del espacio de memoria asignado a la pila de datos.

Y ahora tú. ¿Cómo hallarías, mediante una función recursiva, el término general en estas sucesiones?

Progresiones aritméticas y geométricas.
La sucesión de Fibonacci (¡cómo no!)
La enésima potencia de un número dado.

A PROPÓSITO DE ORMISTON

Un algoritmo de comparación de cifras

En la entrada anterior señalábamos, un poco de pasada, que los pares de Ormiston están formados por dos números primos consecutivos que presentan las mismas cifras, aunque en distinto orden. El primer par está formado por 1913 y 1931. Todo esto está estudiado y puedes consultar estas secuencias en OEIS:

Pares de Ormiston: https://oeis.org/A072274

Tripletes: https://oeis.org/A075093

Conjuntos de cuatro primos consecutivos: https://oeis.org/A161160

Pares que sólo se diferencian en las dos últimas cifras: https://oeis.org/A162765

No vamos a seguir la teoría de estos números, no muy interesante, sino las posibles búsquedas de los mismos

con hoja de cálculo. Como de hecho ya están encontrados, nuestro interés se dirigirá **al procedimiento de búsqueda**.

Si se piensa un poco en la misma se pueden distinguir tres fases:

Identificar los números primos

Para cada uno de ellos encontrar el siguiente primo

Comparar las cifras de ambos para ver si son las mismas.

Como los dos primeros presentan menos novedad, los abordaremos al final. Comenzaremos con la detección de igualdad en el conjunto de cifras.

1) Las mismas cifras en distinto orden

Aquí tenemos el problema que deseábamos resolver hoy:

¿Qué algoritmo podemos usar para saber si dos números enteros tienen las mismas cifras, con el mismo número de repeticiones, y posiblemente en distinto orden?

El problema está en las repeticiones, porque saber si un número contiene a las cifras del otro es fácil, pero el que el número de repeticiones coincida, ya es más difícil de averiguar (para la máquina). Por ejemplo,

51613 y 51631 forman un par de Ormiston y el algoritmo ha de detectar que el 1 aparece dos veces en ambos números. Si no, no serían de Ormiston. ¿Cómo hacerlo?

Se nos ha ocurrido ir tachando una cifra cada vez que comprobemos que también está en el otro par. Leemos la primera cifra del primer número y la buscamos en el otro, y si la encontramos se tacha. Así seguimos hasta que exista una discrepancia o el agotamiento de las cifras. En el caso del ejemplo:

51613 **1**613 **6**13 **1**3 **3**

51631 **1**631 **6**31 3**1** **3**

Expresado en pseudocódigo:

Se leen los números m y n
Se convierten en texto (para que el manejo de las cifras sea más rápido)
**** Mientras no se agoten las cifras de m se hace lo siguiente:***
Buscamos una coincidencia de la primera cifra de m con cualquiera de n
Si se da la coincidencia, se tacha esa cifra tanto en m como en n
Si no hay coincidencia se para el bucle y se avisa
**** Fin del mientras***
Se lee si hay coincidencia plena o hubo un fallo

En atención a quienes no tienen interés por el código en Basic lo situamos al final.

2) Identificar un número como primo

Aprovechamos este tema para introducir una mejora en el algoritmo de averiguar si un número es primo o no, sugerida por nuestro amigo Goyo Lekuona. En este blog, por simplicidad, buscábamos los divisores de un número entre los pares y los impares, pero una vez descartado el 2, se puede seguir con los impares. Con ello el tiempo se reduce casi a la mitad. Gracias, Goyo.

El nuevo código puede ser (hay alguna otra variante posible):

```
Public Function esprimo(a) As Boolean
Dim n, r
Dim es As Boolean

'Devuelve true si es primo. No analiza el que sea entero, por lo que en un decimal puede dar respuesta ilógica

If a = 1 Then es = False 'El 1 no es primo
If a = 2 Then es = True 'El 2 sí lo es
If a > 2 Then
If a / 2 = Int(a / 2) Then
es = False  'Si el número es par lo descartamos
Else
   n = 3: es = True: r = Sqr(a)
   While n <= r And es = True
   If a / n = Int(a / n) Then es = False 'probamos con todo los impares hasta la raíz cuadrada
   n = n + 2
   Wend
End If
```

```
End If
esprimo = es
End Function
```

Una pequeña cuestión: ¿funciona para N=3? ¿Por qué?

3) Buscar el próximo primo

Es muy simple y no necesita explicación:

```
Function primprox(a) As Long
Dim p, prim As Long
Dim sale As Boolean

'Encuentra el menor número primo mayor o igual al dado

p = a + 1: sale = False: prim = 0
While Not sale
If esprimo(p) Then prim = p: sale = True
p = p + 1
Wend
primprox = prim
End Function
```

Con estas herramientas hemos reproducido fácilmente los primeros pares de Ormiston, y te invitamos a intentarlo

1913	1931
18379	18397
19013	19031
25013	25031
34613	34631
35617	35671
35879	35897
36979	36997
37379	37397
37813	37831

Para probar las funciones y aunque bastaba con una inspección visual, hemos pedido a la hoja un listado de pares de Ormiston en los que no haya cifras repetidas. Aquí tienes los primeros:

18379	18397
25013	25031
35617	35671
35879	35897
40213	40231
40639	40693
45613	45631
48091	48109
56179	56197
56713	56731
58613	58631

Anexo

Código de la función cifras_identicas

```
Public Function cifras_identicas(m, n) As Boolean
Dim i
Dim vale, esta As Boolean
Dim nn$, mm$, c$



nn$ = haztexto(n) 'Convierte los números en textos
mm$ = haztexto(m)
```

```
vale = True
While Len(mm$) > 0 And vale
c$ = Mid$(mm$, 1, 1) 'toma la primera cifra
esta = False
i = 1
  While i <= Len(nn$) And Not esta
    If c$ = Mid$(nn$, i, 1) Then
    esta = True
    mm$ = borracar(mm$, 1) 'si ha coincidencia, borra la cifra
    nn$ = borracar(nn$, i)
    End If
    i = i + 1
Wend
If Not esta Then vale = False 'si no hay coincidencia, sale del bucle con el valor "false"
Wend
cifras_identicas = vale
End Function
```

LA HOJA DE CÁLCULO GANA CIFRAS

Calculadora STCALCU

Los cálculos exactos con operadores muy grandes no son posibles en las hojas de cálculo. Ya es sabido que en ellas cuando las cifras significativas de un número llegan a unas 15, se tratan automáticamente en coma flotante y notación científica. La única forma de

mantener la expresión con todas las cifras es aumentando las prestaciones mediante un complemento o, como presentaremos aquí, mediante una colección de nuevas funciones.

Como mero divertimento emprendimos hace tiempo la tarea de dotar a las hojas de cálculo de la posibilidad de manejar números enteros con todas sus cifras, sin las limitaciones a las que nos hemos referido.

Después de intentarlo con registros múltiples desembocamos en la decisión de usar variables de texto (string), como hemos visto en un trabajo similar al nuestro. Del hecho de manejar strings viene el prefijo ST que incorpora tanto la calculadora (STCALCU) como las funciones: stsuma, stresta,…La llamamos calculadora, aunque en realidad es una colección de funciones, pero para quien no se anime a manejarlas hemos incluido un esquema de cálculo con botones en la última hoja.

Representar un número mediante un string tiene una limitación, y es que las hojas de cálculo manejan en general cadenas de 255 caracteres. En la herramienta que presentamos se ha puesto un tope de 250, útil para la mayoría de los trabajos con números enteros.

Operaciones como funciones

Lo que presentamos ahora ha tenido un desarrollo totalmente personal (y por tanto sin la garantía de un

producto profesional) y realiza los cálculos en forma de funciones. Así se pueden crear tablas o enlazar unos cálculos con otros con toda libertad. También así se podrán mezclar, con cuidado, nuestras funciones con las propias de Excel, OpenOffice o LibreOffice.

Con la implementación de las operaciones como funciones se consiguen varias ventajas:

- Puedes escribir la función en cualquier celda, con lo que es fácil construir tablas y esquemas de cálculo.
- Son independientes del resto de funciones de las hojas y se pueden mezclar con ellas (con cuidado)
- No hay que preocuparse en exceso por la sintaxis de las expresiones algebraicas, que aquí se reducen a la aplicación reiterada de funciones. Así, A*B+C*D se escribiría como Stsuma(stmulti(A;B),stmulti(C;D)). Parece más complejo, pero todo es cuestión de costumbre.

La hoja que contiene la calculadora la tienes alojada en http://hojamat.es/sindecimales/aritmetica/herramientas/herrarit.htm#calcula

En STCALCU dispones de las funciones más usuales. Si sabes programar podrás enriquecerlas con otras nuevas. Son estas:

Operaciones básicas

Son las clásicas (la raíz cuadrada resultaba costosa y poco útil), más el residuo

Operación	Formato
SUMA	=stsuma(A;B)
RESTA	=sresta(A;B)
PRODUCTO	=stmulti(A;B)
COCIENTE	=stdivi(A;B)
POTENCIA	=stpotencia(A;B)
RESTO	=stresto(A;B)

Damos algunos ejemplos por si deseas reproducir alguno:

=stsuma(771662374885756;12636645869121) = 784299020754877

=stmulti(777654556988;299818) = 233154833967028184

=stpotencia(7;51) = 125892552985318850263419623839875454447587 43

=stresto("27677841276031200";301)= 163

A y B son, en general, referencias a celdas, pero como ves en los anteriores ejemplos, se pueden usar números enteros directamente.

Hay que tener en cuenta estas consideraciones:

(a) Las funciones actúan sobre datos de tipo texto, pero frecuentemente la hoja los interpreta bien aunque no se escriban comillas. Observa el ejemplo anterior del resto, en el que sin comillas nos hubiera dado error. Para evitar esto es preferible usar las funciones sobre celdas, como en **=stpotencia(Z12;W12**), procurando que tengan formato de texto, también para ver mejor los datos, que, de otra forma aparecerían en formato de coma flotante.

(b) Las operaciones se pueden combinar como si fueran funciones. Esta fórmula te daría el séptimo número de Fermat

=stsuma(stpotencia(2;stpotencia(2;7));1) = 340282366920938463463374607431768211457

(c) Sobre ellas puedes definir otras funciones. Por ejemplo, el STCUBO

Public function stcubo$(x$)
Stcubo$=stpotencia(x$,"3")
End function

Observa que prudentemente definimos todo como string

Otras funciones

Están orientadas a la divisibilidad, por lo que pueden ralentizarse en exceso si hay que factorizar

Función	Formato
FACTORIZAR	=stfactores(A)
ES MÚLTIPLO	=stmultiplo(A;B)
ES PRIMO	=stprimo(A)

STFACTORES

Te devuelve el conjunto de factores primos de un número con el formato usual de [primo1,exponente1] [primo2,exponente2] [primo3,exponente3]…

Ya se ha advertido que puede resultar muy lenta. Si tu paciencia se agota, pulsa ESC en Excel o CTRL+Mayúscula+Q en las otras.

Ejemplo:

stfactores(277311825)= [3,2],[5,2],[7,2],[25153,1]

Es decir, que 277311825=3^2*5^2*7^2*25153

STMULTIPLO

Devuelve VERDADERO si un número es múltiplo de otro

Ejemplo

Stmultiplo(366727538765709894016572635240878771
131528192937591261004187463844 84;
182273662712) = VERDADERO

STPRIMO

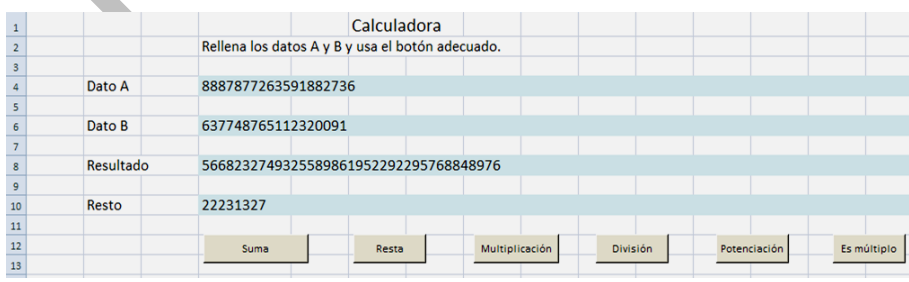
Devuelve VERDADERO si su argumento es primo. **También puede resultar lenta**.

Ejemplo

stprimo(7726631) = FALSO porque 7726631 = [11,1],[239,1],[2939,1]

Calculadora

Para quienes no se sientan muy a gusto manejando funciones se ha implementado en la tercera hoja de ***Stcalcu*** una calculadora simple en la que realizar los cálculos. Consultando el código de los botones implementados se pueden insertar otros nuevos. No es difícil.



Se ha habilitado una línea para el resto de la división, que aquí siempre es euclídea.

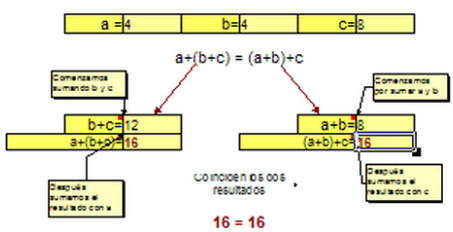
En la parte inferior figuran los botones de divisibilidad y aún más abajo unas memorias para almacenar datos intermedios. Todo el esquema es fácilmente revisable.

Con estas funciones y las estructuras del Basic de las hojas de cálculo puedes enriquecer el catálogo. Debes tener cuidado en declarar como ***string*** las variables que uses. Suerte.

Y de nuevo la advertencia: estás ante un trabajo no profesional. Si luego falla algo, ten paciencia.

CONVERTIR ESQUEMAS DE CÁLCULO EN TABLAS

Desde este blog y nuestra página hojamat.es hemos promovido siempre el uso de esquemas de cálculo y apuntes interactivos http://hojamat.es/contenidos/apuntes.htm



La limitación que presentan es que al provenir de cálculos de cierta complejidad es difícil recogerlos en forma de función o tabla. En esta entrada presentaremos una herramienta sencilla para recoger resultados de esquemas en forma de tabla.

La herramienta

Al principio intentamos recogerlos como funciones, pero ni Excel ni OpenOffice ni LibreOffice lo permiten. Hay algo en la programación de funciones que hace que si se altera el valor de una celda cualquiera dentro del proceso de cálculo de la función, esta no recoja el valor que ha de devolver. Continuamente da mensajes de error.

Lo que sí podemos es construir una tabla que altere los parámetros del esquema y recoja el valor final del cálculo. Como estamos abusando de generalidades, lo explicaremos mejor con un ejemplo.

Partiremos del cálculo del fósil de un número

(http://hojaynumeros.blogspot.com.es/2008/10/dndole-vueltas-2.html)

Este valor se halla multiplicando las cifras de un número, volviendo a realizar esta operación en el resultado y en los siguientes hasta llegar a un número de una cifra al que llamamos fósil.

Por ejemplo, partimos de 876, multiplicamos 8*7*6=336. Volvemos a multiplicar y reiteramos. 3*3*6=54, 5*4=20, 2*0=0, luego el fósil de 876 es 0.

Este cálculo lo podemos tener implementado en una hoja mediante un esquema (en este momento no nos va a interesar qué fórmulas se han usado)

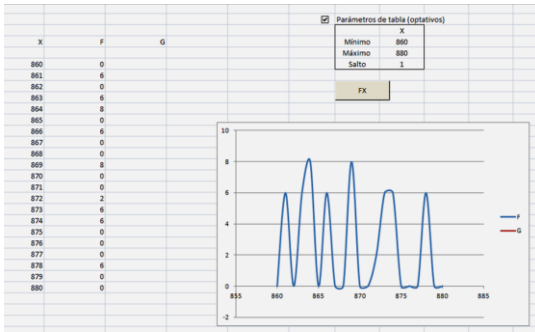
		Esquema de cálculo						
F	0							
X	876	87	8	0	0	0	0	0
		6	7	8	1	1	1	1
	336	33	3	0	0	0	0	0
		6	3	3	1	1	1	1
	54	5	0	0	0	0	0	0
		4	5	1	1	1	1	1
	20	2	0	0	0	0	0	0
		0	2	1	1	1	1	1
	0	0	0	0	0	0	0	0
		0	1	1	1	1	1	1
	0	0	0	0	0	0	0	0

Lo que deseamos es poder extraer de este esquema una tabla de valores y un gráfico si vamos cambiando el 876 por el intervalo 860…880. Esta tarea la realiza la hoja de cálculo ***esquefun***, que está alojada en (aquí dirección)

Tabla simple

Si la abres verás que la primera hoja estará en blanco o contendrá un esquema de cálculo cualquiera. En el segundo caso puedes borrarlo todo y construir tu propio proceso. No sobrepases el tamaño de una pantalla o algo más (unas treinta filas por treinta columnas).

En la segunda hoja se te ofrece la posibilidad de construir la tabla



¿Cómo se consigue esto? Lo explicaremos paso a paso para una tabla simple X,FX:

(1) En la primera hoja, en la celda que está a la izquierda del valor de la variable independiente escribes una X. Debes procurar tener esa celda siempre libre. También es conveniente que uses las primeras filas y columnas.

(2) También a la izquierda del resultado que te interese como valor de la función (en nuestro caso el fósil) escribes una F.

Si tu resultado no está en ellas siempre puedes copiarlas dinámicamente. Por ejemplo, si tienes el resultado en la celda AH44, basta que en una celda más arriba y a la izquierda escribas la fórmula **=AH44** y así todos los resultados se copiarán a esa celda.

F	**0**
X	**876**

Con eso ya has terminado la preparación de la hoja 1: Escribir “X” a la izquierda de la variable independiente y una “F” al lado de la variable dependiente.

(3) Define el mínimo, máximo y salto de la tabla que deseas para concretar los valores de X en la misma (puedes hacerlo de forma manual, pero sería cosa tuya borrar en la columna de la X todos los datos sobrantes)

(4) Pulsa el botón FX y obtendrás tabla y gráfico de los resultados. En el volcado de pantalla de más arriba puedes observar que la tabla va de 860 a 880 y que el comportamiento del fósil es totalmente irregular.

Ya está. No hay que trabajar más. Después tabla y gráfico los puedes exportar a cualquier otro documento.

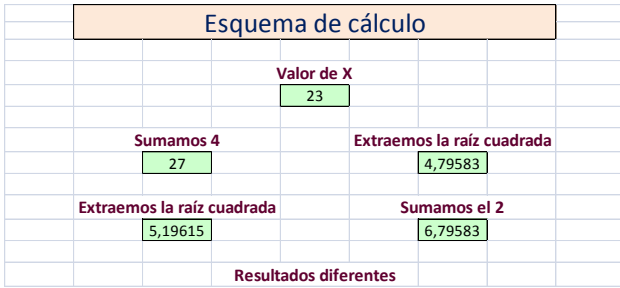
Tabla simple con dos funciones

Lo explicamos también con un ejemplo:

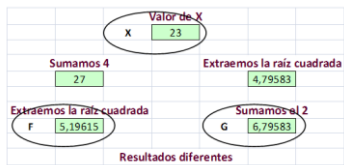
Deseamos hacer entender a nuestro alumnado que la raíz de una suma no da el mismo resultado que la suma de raíces. Para ello hemos pensado en usar

$$\sqrt{X+4} \neq \sqrt{X}+2$$

Preparamos un esquema de cálculo en el que se manifiesten las diferencias. Podía ser este:

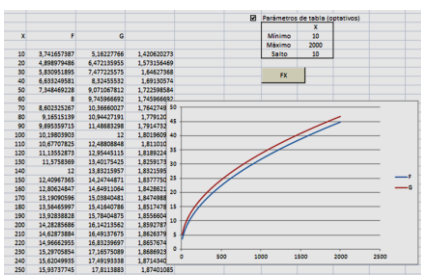


Si ahora deseamos ver en un gráfico cómo evolucionan las diferencias necesitaremos definir dos funciones. Así que rotulamos con X el número, con F el primer cálculo y con G el segundo (estos nombres son obligatorios)



A partir de este esquema ya rotulado podemos crear tabla y gráfico

Hemos construido una tabla del 10 al 2000 con saltos de 10, con la ¿sorpresa? de que las diferencias tienden a estabilizarse a valores cercanos al 2



En la imagen aparece una tercera columna de diferencias que se han creado manualmente. El objetivo de construir la tabla a partir del esquema se ha conseguido.

En cursos algo más avanzados puedes intentar demostrar que efectivamente el límite de la diferencia entre ambos resultados es 2.

Tabla doble

Sería también útil estudiar una función que dependiera de dos variables. Para eso dispones de la tercera hoja de esta herramienta. No se ha incluido el gráfico para no tener que insertar otra cuarta hoja, pero nuestros lectores sabrán cómo construirlo.

En este caso deberemos rotular **con X e Y las dos variable independientes y con F la función**. Lo explicaremos con un ejemplo que no tiene más interés que la mera curiosidad:

Sabemos que los pasos necesarios en el algoritmo de Euclides para obtener el MCD de dos números varía mucho según los datos usados. Intentemos formar una tabla de doble entrada con ellos.

Imagina que hemos trasladado a la primera hoja el algoritmo de nuestra herramienta ***Euclides***

(http://hojamat.es/sindecimales/congruencias/herramientas/herrcong.htm)

			Primer número	6554			Segundo número	8720				
					Técnica manual, con el uso de las prestaciones de una hoja de cálculo							
	0	1	3	38	1	2	9	0	0	0	0	0
6554	8720	6554	2166	56	38	18	2	0	0	0	0	0
6554	2166	56	38	18	2	0	0	0	0	0	0	0
			Número de cocientes		7		Máximo común divisor				2	

Debemos ahora, después de comprobar que funciona bien, borrar los rótulos “Primer número” y “Segundo número” y sustituirlos por X e Y respectivamente. Abajo también sustituiremos “Número de cocientes” por F, para recoger su valor como una función. En este ejemplo tenemos un problema, y es que esas celdas están combinadas. Debes primero anular la combinación y después escribir X,Y,F de forma contigua a su valor.

Pasamos a la tercera hoja y definimos intervalos y saltos para X e Y, por ejemplo, de 20 a 30 con saltos de 1 (el carácter optativo se incluye porque se puede efectuar un relleno manual, aunque no es muy aconsejable).

☑ Parámetros de tabla (optativos)

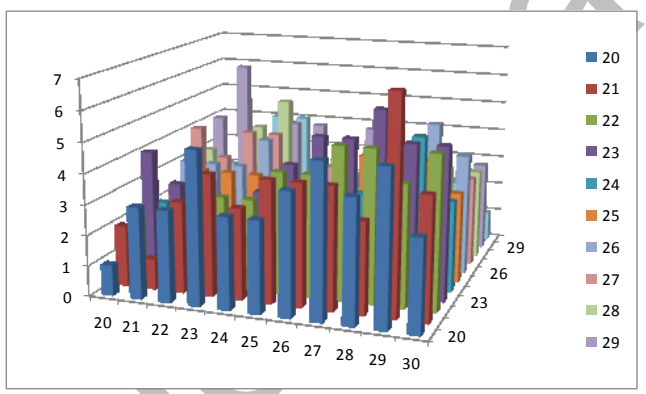
	X	Y
Mínimo	20	20
Máximo	30	30
Salto	1	1

Pulsamos el botón **Fxy** y se formará la tabla de doble entrada:

	Y										
X	20	21	22	23	24	25	26	27	28	29	30
20	1	3	3	5	3	3	4	5	4	5	3
21	2	1	3	4	3	4	4	4	3	7	4
22	2	2	1	3	3	4	4	5	5	4	5
23	4	3	2	1	3	4	5	5	6	5	5
24	2	2	2	2	1	3	3	3	3	5	3
25	2	3	3	3	2	1	3	4	4	4	3
26	3	3	3	4	2	2	1	3	3	5	4
27	4	3	4	4	2	3	2	1	3	4	3
28	3	2	4	5	2	3	2	2	1	3	3
29	4	6	3	4	4	3	4	3	2	1	3
30	2	3	4	4	2	2	3	2	2	2	1

Llama la atención que no es simétrica para intercambios entre X e Y, pero es que si el primer número es menor que el segundo nos cuesta un paso más en el algoritmo.

Con los procedimientos habituales podemos traducirla a un gráfico 3D:



Estas son las tres modalidades de creación de tablas que hemos incluido en ***esquefun***. Con ellas basta para encontrar usos en la enseñanza y como herramienta de búsqueda. Que os sea útil.

RECOGIDA DE DATOS EN TABLAS DE MARCADO DE CASILLAS.

Ya hacía tiempo que no escribíamos sobre el manejo de las hojas de cálculo sin relacionarlas con el estudio de los números. Lo hacemos hoy con un problema que se presenta al recoger valoraciones cumplimentadas mediante el marcado de casillas.

Cuando se plantea una encuesta de valoración es fácil adivinar la orientación cultural de quien la ha confeccionado. Si es alguien con mentalidad numérica, la crea pensando ya en la recogida de datos y aplicación de medidas estadísticas. Utiliza escalas numéricas o ceros y unos. Por el contrario, personas más cercanas a una cultura de tipo humanístico prefieren esquemas sencillos, visuales y que transmitan bien la idea que se desea valorar.

Una estructura muy usada es la de una tabla de doble entrada en la que se marcan algunas celdas según la valoración deseada. En la imagen presentamos una muy popular, y es la de elegir del 1 al 5, como escala ordinal y subjetiva en las columnas, para la valoración de los aspectos que figuran en las filas.

	Escala del 1 al 5, 1 muy mal, 5 muy bien				
	1	2	3	4	5
Seguridad					*
Prestaciones		*			
Acabado		*			
Precio			55	*	
Servicio técnico			*		

En una tabla pequeña como esta, los totales por filas y columnas son muy sencillos de obtener, pero imaginemos que se manejan cientos de filas o que se han agrupado muchas encuestas en una, ¿cómo automatizar la traducción del símbolo "*" a datos numéricos? Deseamos dos cosas:

(a) Crear unas frecuencias en la parte baja de la tabla con los distintos resultados
(b) Traducir cada asterisco a la valoración numérica entre 1 y 5 correspondiente.

	Escala del 1 al 5, 1 muy mal, 5 muy bien					
	1	2	3	4	5	
Seguridad					*	5
Prestaciones		*				2
Acabado		*				2
Precio				*		4
Servicio técnico			*			3
	0	2	1	1	1	

En la imagen recogemos nuestras pretensiones:

Las celdas coloreadas son las que deseamos obtener de forma automática.

Frecuencias por columnas

Con estas no hay problema, pues la función CONTAR.SI nos lo resuelve. En cada columna escribimos algo así como =CONTAR.SI(E5:E921;"*"), donde el primer argumento recorre toda la columna de la tabla y el segundo contiene el símbolo usado.

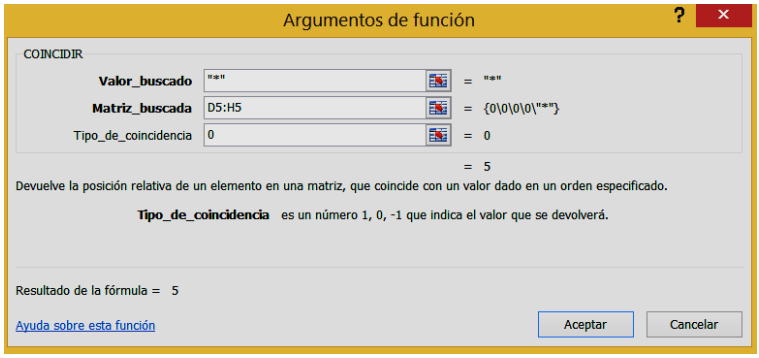
Como vemos, el problema se resuelve sin dificultad y no le prestamos más atención. Pasamos al otro.

Conversión de un símbolo en una valoración numérica

Este otro problema es más difícil de resolver y con el objeto de repasar técnicas de las hojas de cálculo lo abordaremos de varias formas.

Función COINCIDIR

Es la solución más sencilla, pues esta función nos devuelve la posición del asterisco dentro de la fila, si la organizamos de esta forma:



- Valor buscado: el símbolo “*”
- Matriz: La fila en la que estamos trabajando
- Tipo de coincidencia: Usamos el 0 para que sea de tipo exacto: o es un asterisco o no lo es.

  En la celda se escribiría una fórmula similar a esta: =COINCIDIR("*";D5:H5;0)

  Este procedimiento tiene la ventaja de poder arrastrar la fórmula hacia abajo, porque sólo maneja referencias relativas.

  El inconveniente es que si las puntuaciones no son del 1 al 5, sino otras, como 2,4,5,20, o A,B,C,D…esta técnica nos devolvería el número de orden y no el valor. Esto se puede arreglar con la función INDICE. Buscamos el asterisco con COINCIDIR y después lo volcamos en la primera fila con INDICE para localizar la puntuación.

	A	B	C	D	E	
Seguridad					*	E
Prestaciones		*				B
Acabado		*				B
Precio				*		D
Servicio técnico			*			C
	1	2	3	4	5	

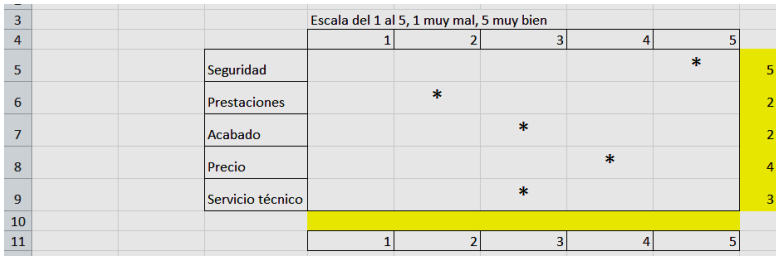
Para obtener el resultado de la imagen hemos usado este tipo de fórmula:

=INDICE(D$4:H$4;COINCIDIR("*";D6:H6))

Función BUSCARH

Esta función es muy útil en estos casos, pero aquí tiene dos problemas, como veremos. BUSCARH actúa sobre una matriz recorriendo la primera fila para buscar el valor deseado, y nos devuelve el valor correspondiente en la misma columna pero situado unas filas más abajo.

Primer problema: La fila de búsqueda es siempre la primera de la matriz y la de devolución de valores es otra, pero aquí lo que deseamos devolver, 1, 2, 3, 4 o 5 está precisamente situado en la primera fila. Una solución es copiar esa fila al final de la tabla y tomar nota de donde está situada. En la imagen la hemos copiado en la fila 11



		Escala del 1 al 5, 1 muy mal, 5 muy bien					
		1	2	3	4	5	
Seguridad						*	5
Prestaciones			*				2
Acabado				*			2
Precio					*		4
Servicio técnico				*			3
		1	2	3	4	5	

Después el truco consiste en que al dar la fila de búsqueda damos la actual (por ejemplo, para el concepto “Acabado” sería la fila 7 y para la fila de devolución escribimos **12-FILA()** y así la hoja cuenta las filas que van desde la nuestra hasta la final situada en el 11 incluida.

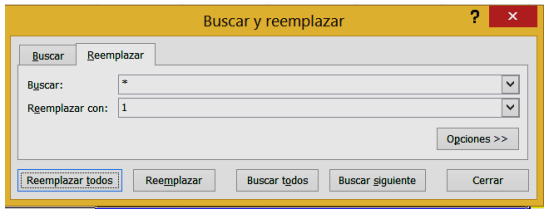
Segundo problema: La dimensión de la matriz cambia al rellenar hacia abajo, pero eso no nos va a afectar porque no importa si sobran filas, ya que sabemos que la que nos interesa está siempre en el 11 y el cálculo **12-FILA()** nos garantiza que llegamos a ella. En resumen, usaríamos una fórmula como esta: =BUSCARH("*";D8:H14;12-FILA();0), que es la correspondiente a la fila 8.

Resulta algo artificioso el procedimiento. Se ve que es mejor el que usa la función COINCIDIR. No importa, porque nuestro objetivo es descubrir posibilidades.

¿Qué haría alguien de Matemáticas?

Esto va un poco en broma.

Cambiaría los asteriscos por un 1. Esto se puede conseguir con la orden Reemplazar.



Los huecos se pueden reemplazar por ceros, pero no es necesario. Una vez que nuestra matriz es numérica, para traducir la posición del asterisco a un número basta usar SUMAPRODUCTO, que multiplique la fila actual por la primera, y así sólo aparecerá la puntuación situada en la misma fila que el 1.

	1	2	3	4	5
Seguridad	0	0	0	0	1
Prestaciones	0	1	0	0	0
Acabado	0	1	0	0	0
Precio	0	0	0	1	0
Servicio técnico	0	0	1	0	0

Sería una fórmula similar a esta:

=SUMAPRODUCTO(D6:H6;D$4:H$4)

Observa que la primera fila se usa con referencia absoluta, para que al rellenar hacia abajo se conserven siempre las puntuaciones 1, 2,…5.

El problema está en que la persona que haya diseñado la encuesta nos proteste por manipularla. Por eso decíamos que esto se trataba un poco como broma.

PROCEDIMIENTO PARA EXCEL

El autor necesita frecuentemente descomponer un número en factores primos. Como esta función no viene implementada en la hoja de cálculo, ha tenido que programarla en el Basic de Excel. El problema que surge es que sólo está disponible en la hoja que contiene el código y no en cualquier otra que se cree. Esto tiene un remedio, y es la construcción de un complemento de Excel que nos permita acceder a esa factorización cuando se abra cualquier hoja.

Complementos de Excel

Para saber de qué estamos hablando, entra en las **Opciones de Excel** y busca **Complementos**. En la ventana que se abre podrás comprobar qué complementos tienes instalados en tu equipo

(el volcado de pantalla corresponde al Excel 2007 sobre Windows XP, una querida antigüedad, pero igual te funciona en Excel 2010)

En la imagen vemos que el autor tiene instaladas dos herramientas de análisis, el Solver y un complemento suyo titulado Micomplemento. Como habrás comprendido, los cuatro contienen funciones y rutinas que no vienen implementadas en Excel originariamente.

Crea tu propio complemento

Al final de este apartado se ha incluido el código mínimo necesario para implementar la descomposición factorial de un número entero (dentro de los límites de Excel y del propio código, no le pidas milagros) como un regalo del autor a sus lectores.

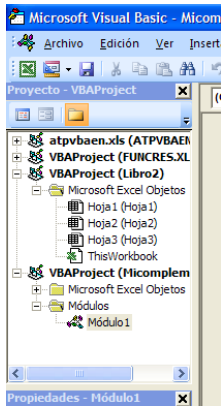
Pasos a seguir

En primer lugar tienes que escribir tus funciones. En el caso que estamos desarrollando basta con que las copies desde el final de este apartado. Abre un archivo nuevo y pega en él las definiciones que desees según te explicamos a continuación:

Una vez decidido el código deberás pasarlo a Excel. Para ello acude a la pestaña **Programador** de la cinta de opciones. Si no la tienes visible deberás activarla en **Opciones de Excel – Más frecuentes.**

Entras en el ámbito de programación mediante el primer botón de la ficha **Programador**:

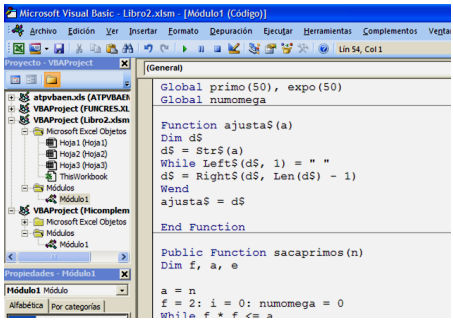
Te aparecerá el acceso a las macros que utiliza tu hoja de cálculo en este momento:



A ti no te aparecerá la referencia a Micomplemento. También, si usas la versión 2010 los colores podrán cambiar, pero el contenido será el mismo.

Ahora debes crear un módulo que aloje tu código. Pide **Insertar – Módulo** y Excel lo hará con el nombre de **Modulo 1** (salvo que tengas otro anterior).

En la hoja en blanco que aparece pega el código que habrás copiado desde aquí o que haya sido creado por ti:

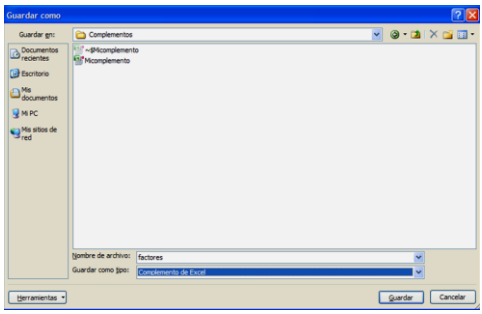


Ahora puede ser un buen momento para comprobar si todo va bien. Guarda el archivo nuevo como **Libro habilitado para macros**. Vuelve a la hoja. Escribe cualquier número entero, por ejemplo 366220 en la celda B4. En otra celda escribe **=factores(B4**). Si ves escrito [2,2][5,1][18311,1] es que tu función se comporta bien. La interpretación de lo que ves es que el primer número de cada corchete es el factor primo y el segundo el exponente al que está elevado. En este caso $366220=2^2*5*18311$. No intentes cálculos con esta expresión, que tiene formato de texto.

Lo que has construido hasta ahora sólo te vale para el archivo que contiene el código. Para que se active en cualquier hoja hay que convertirlo en complemento.

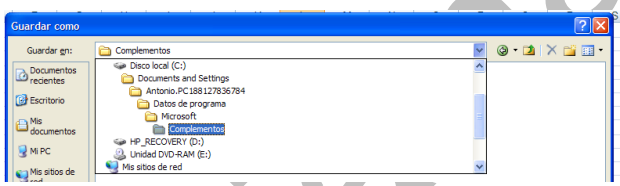
Instalación del complemento.

Borra si acaso los cálculos efectuados y vuelve a guardar el libro como ***complemento de Excel***. Puedes cambiarle el nombre a **factores.** Guíate por la imagen

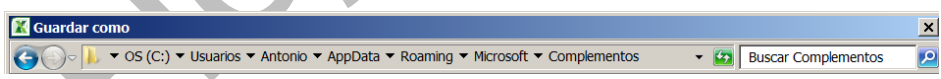


Observa que Excel te guía ya a la carpeta **Complementos**, que es donde debe estar alojado el tuyo. **No cambies esa carpeta, que si no, no podrás instalar el complemento.**

Puedes acceder a la ruta en la que está situada la carpeta



En Office 2010 se te muestra también toda la ruta, que es distinta a la anterior

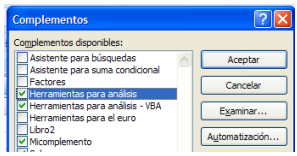


Es interesante conocer esa ruta, por si deseas borrar el archivo.

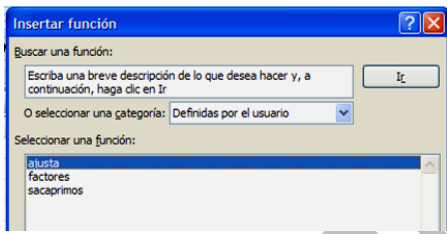
Instalación

Ya sólo te falta instalar tu complemento. Vuelve a las opciones de Excel y busca **Complementos**. En la parte inferior de la ventana tendrás el botón **Ir…** Úsalo y

descubrirás que tu trabajo está preparado ya para ser usado:



Activa la casilla de verificación que está junto al nombre **Factores** y pulsa **Aceptar**. Si todo ha ido bien, cuando abras Excel de nuevo, en el catálogo de funciones definidas por el usuario dispondrás de la función **factores**:



Las otras dos funciones ***ajusta*** y ***sacaprimos*** son auxiliares y no tienes por qué usarlas, ya que quizás no interpretarías bien su resultado.

Ahora define tú un complemento propio ¡Suerte!

Código en Basic

```
Global primo(50), expo(50)
Global numomega

Function ajusta$(a)
Dim d$
d$ = Str$(a)
While Left$(d$, 1) = " "
```

```
d$ = Right$(d$, Len(d$) - 1)
Wend
ajusta$ = d$

End Function

Public Function sacaprimos(n)
Dim f, a, e

a = n
f = 2: i = 0: numomega = 0
While f * f <= a
e = 0
While a / f = Int(a / f)
e = e + 1
a = a / f
Wend
If e > 0 Then
numomega = numomega + 1
primo(numomega) = f
expo(numomega) = e
End If
If f = 2 Then f = 3 Else f = f + 2
Wend
If a > 1 Then
numomega = numomega + 1
primo(numomega) = a
expo(numomega) = 1
End If
sacaprimos = numomega

End Function

Public Function factores(n) As String
```

```
Dim a, nn
Dim s$

'saca factores en forma de string
a = n
nn = sacaprimos(a)
s$ = ""
For i = 1 To numomega
s$ = s$ + "[" + ajusta(primo(i)) + "," + ajusta(expo(i))
+ "]"
Next i
factores = s$
End Function
```

PROCEDIMIENTO PARA APACHE OPENOFFICE Y LIBREOFFICE

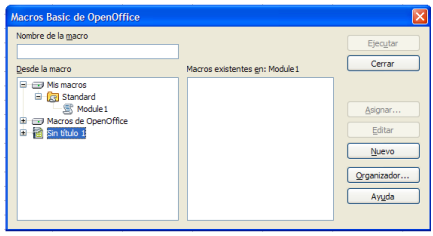
Si pasamos a Apache OpenOffice o LibreOffice, la creación de complementos (extensiones) se complica, porque está orientada al uso de terceros. Como aquí sólo nos interesa que tengas disponibles tus funciones en cualquier archivo nuevo que crees para tu propio uso, desarrollaremos un método mucho más sencillo.

Usaremos el código que se presentó en anteriormente para descomponer un número natural en sus factores primos. Cópialo y guárdalo, porque te servirá ahora.

Una vez decidido el código deberás pasarlo a **Apache OpenOffice** o a **LibreOffice**. El procedimiento es

similar en ambos programas, y sólo añadiremos los detalles específicos de LibreOffice si fuera necesario.

Abre una hoja nueva. Acude al menú **Herramientas** y en él elige **Macros**, después **Organizar macros** y finalmente **OpenOffice Basic** (o **LibreOffice Basic**)



Observa que tu archivo aparecerá en la parte baja (en la imagen aún no tiene título). Tú has de ir a la superior, "***Mis macros - Standard***". Pide **crear un módulo nuevo** con el botón "***Nuevo***" de la parte derecha. Si ya existe uno, como ocurre en la imagen, le asignará el nombre de Module 2 u otro similar.

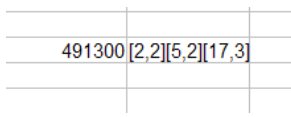
Abre el nuevo módulo que has creado (pinchando sobre su nombre) y pégale el código que desees. Acepta y cierra todo.

Ahora puede ser un buen momento para comprobar si todo va bien. Vuelve a la hoja. Escribe cualquier número entero, por ejemplo 366220 en la celda B4. En otra celda escribe **=factores(B4**). Si ves escrito [2,2][5,1][18311,1] es que tu función se comporta bien. La interpretación de lo que ves es que el primer número de cada corchete es el factor primo y el segundo el exponente al que está elevado. En este caso

366220=2^2*5*18311. No intentes cálculos con esta expresión, que tiene formato de texto.

Como has usado el contenedor "***Mis macros***", todo lo que has construido hasta ahora lo encontrarás implementado en cualquier libro que abras. Prueba a hacerlo. Cierra el archivo, abre uno nuevo, y en cualquier celda escribe un número entero y aplícale la función ***factores*** (no aparecerá en ningún catálogo. Te lo tienes que aprender)

En la imagen se ha descompuesto el número 491300 en factores dentro de un archivo recién creado:

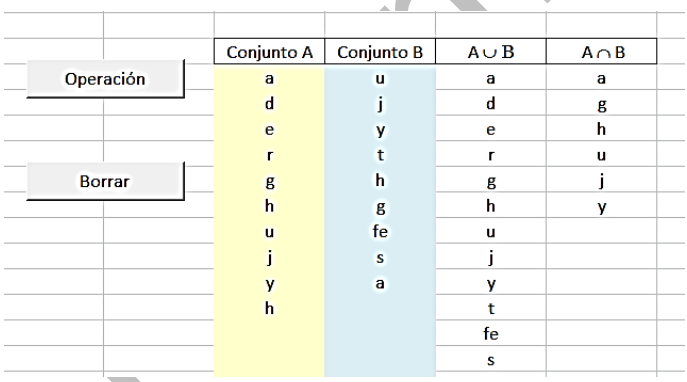


Ahora inténtalo tú.

UNIÓN E INTERSECCIÓN DE CONJUNTOS

Nos vamos a proponer la obtención de la unión y la intersección de dos conjuntos escritos en la hoja como dos columnas paralelas. Lo desarrollaremos en Excel sólo, para no duplicar las explicaciones, pero el contenido se puede adaptar a OpenOffice o LibreOffice. No se busca aquí la utilidad, sino la posibilidad de superar un reto. Lo que construyamos puede que aparentemente no sirva para nada.

Hemos preparado un esquema en el que a partir de la fila 7 se van escribiendo los conjuntos A y B con lo que deseamos operar. Después, con una simple pulsación de un botón, realizaremos las operaciones deseadas.



Intentamos en primer lugar resolver la cuestión sin el uso de macros, pero resultó un proceso tan complejo y artificioso que renunciamos a ello. Así, todo lo que sigue se basará en el lenguaje VBA de Excel. Como se observa en la imagen, se pueden usar números y también letras y palabras. Sólo hay que tener en cuenta

que un espacio en blanco cuenta como un elemento, por lo que el borrado se debe realizar con el botón correspondiente o con la tecla Supr, sin escribir nada.

Otro detalle interesante es que en las operaciones se eliminan los elementos repetidos, logrando con ello una gran limpieza en la presentación. En un segundo paso puedes ordenar los resultados sin son más extensos.

Conjunto A	Conjunto B	A ∪ B	A ∩ B
2	1	2	3
3	3	3	5
5	5	5	4
4	7	4	7
5	11	6	9
6	4	7	
7	9	8	
8		9	
8		1	
7		11	
9			

Procedimientos necesarios

Recorrido por un conjunto

Para obtener la unión e intersección de dos conjuntos se requiere, en primer lugar, el poder recorrer un conjunto del que no se sabe en principio cuántos elementos contiene. Para ello usaremos la idea de celda vacía. El recorrido se basará entonces en “avanzo mientras la celda no esté vacía”. Esta condición se puede verificar en VBA con la función IsEmpty, que nos devuelve True si la celda no contiene ningún dato. Con ella es fácil programar un recorrido:

fila = 7

While Not IsEmpty(Cells(fila, columna)) And … (cualquier otra condición)
‘ Aquí las operaciones que deseemos realizar con el elemento.
fila = fila + 1
Wend

Este sencillo esquema se repetirá cada vez que realicemos una operación elemento a elemento: ver si un dato pertenece o no al conjunto, buscar repetidos, incorporar elementos nuevos y otros. Comenzamos por la fila 7, que es donde comienzan nuestros conjuntos y después se va bajando de fila hasta que no queden elementos.

Por ejemplo, la siguiente función ESTA nos devuelve True si un valor **n** pertenece al conjunto situado en la cierta columna

Public Function esta(n, columna) As Boolean
Dim fila
Dim est As Boolean

est = False
fila = 7

While Not IsEmpty(Cells(fila, columna)) And Not est
If n = Cells(fila, columna).Value Then est = True

```
fila = fila + 1
Wend
esta = est
End Function
```

Es fácil identificar la estructura del recorrido por el conjunto. Esta función ESTA nos servirá para saber si podemos agregar un elemento nuevo a un conjunto mediante el procedimiento AGREGA, que servirá para ir incorporando términos a la unión y a la intersección.

```
Sub agrega(n, columna)
Dim i

i = 7
While Not IsEmpty(Cells(i, columna))
i = i + 1
Wend
If Not esta(n, columna) Then Cells(i, columna).Value
= n
End Sub
```

Recorre el conjunto, y si no encuentra el elemento dado, baja una fila y lo incorpora.

Con los procedimientos de recorrido y agregación y la función ESTA podemos ya planificar nuestra tarea:

- Se elige el primer conjunto
- Se recorre, y para cada elemento:
- Si no está en la unión, se agrega (así evitamos repetidos)
- Se compara con todos los elementos del segundo (mediante un recorrido por el mismo) y si está repetido, se incorpora a la intersección si todavía no está.
- Se repite la tarea con el segundo conjunto, pero esta vez no se busca la intersección, que ya estará resuelta.

Para entender el listado que sigue recuerda que los conjuntos están escritos en las columnas tercera y cuarta, que la unión se escribe en la quinta y la intersección en la sexta.

Así quedaría:

Option Explicit ‘Evita el uso de variables no dimensionadas

Public Function esta(n, columna) As Boolean ‘ Ya explicada. Determina si un elemento pertenece a un conjunto
Dim fila
Dim est As Boolean

est = False
fila = 7

While Not IsEmpty(Cells(fila, columna)) And Not est
If n = Cells(fila, columna).Value Then est = True
fila = fila + 1
Wend
esta = est
End Function

Sub agrega(n, columna) ‘También explicada: añade un elemento si aún no está
Dim i

i = 7
While Not IsEmpty(Cells(i, columna))
i = i + 1
Wend
If Not esta(n, columna) Then Cells(i, columna).Value = n
End Sub

Sub union() ‘Esquema general de trabajo. Se inicia al pulsar el botón “Operación”
Dim i, j, n1, n2

Dim esinter As Boolean

i = 7
Call borrar ‘Macro grabada aparte
While Not IsEmpty(Cells(i, 3)) ‘Recorre el primer conjunto

'Se recorre la primera columna
n1 = Cells(i, 3).Value
Call agrega(n1, 5) ‘Agrega el elemento a la unión

'se busca la intersección
j = 7
esinter = False
While Not IsEmpty(Cells(j, 4)) And Not esinter ‘Se recorre el segundo conjunto
n2 = Cells(j, 4).Value
If n1 = n2 Then esinter = True
j = j + 1
Wend
If esinter Then Call agrega(n1, 6) ‘Si está repetido se agrega a la intersección
i = i + 1
Wend
i = 7
While Not IsEmpty(Cells(i, 4))
'Se recorre la segunda columna y se repite el agregar a la unión.

```
n1 = Cells(i, 4).Value
Call agrega(n1, 5)
i = i + 1
Wend
End Sub
```

Si lo has entendido, intenta programar la diferencia entre dos conjuntos, los elementos que pertenecen a uno pero no al otro. Puedes repasar la hoja alojada en (http://www.hojamat.es/blog/union_de_conjuntos.xlsm) y modificarla libremente.

SOLUCIONES

Exploración de soluciones

La ecuación de Pell propuesta presenta las soluciones

X=2;Y=1 X=7;Y=4 X=26;Y=15 X=101;Y=56

El resto de soluciones requiere muchas filas y columnas y no merece la pena seguir con este método.

La segunda ecuación no tiene más soluciones.