Tabla de Contenido

SIMBOLOS UTILIZADOS EN ESTE MANUAL

Simbolo	Significado
	Código más elaborado. Muestra ejemplos de código más complejo en la sección.
	Nota histórica. Nos mostrará un poco de historia.
	Dato incierto. Cuando se refiera a algún dato incierto o elemento incierto.
	La llave del conocimiento a un nuevo truco para muchos desconocido.

Introducción

Este documento está orientado hacia aquellas personas que desean entrar al fascinante mundo de la programación con el lenguaje Visual Basic For Application (en algunas ocasiones lo he llamado erróneamente seudo-lenguaje).

Veremos como dentro de poco tiempo tendremos que cambiar nuestra mentalidad con respecto a la programación con este Lenguaje ya que desaparecerá. ¿Cómo que va a desaparecer?, bueno así es, y para adelantarles algo va a ser sustituido por el Visual Studio Tools For Microsoft Office (algo así como Herramientas de Visual Studio para Microsoft Office). Esto nos obliga a aprender algo más, teniendo que actualizarnos en todo momento.

Este cambio tan repentino afectará a muchos que no han comprendido que para poder desarrollar una pequeña rutina se necesita aprender, tener ganas de estudiar, impregnarse de un lenguaje, casi que hasta soñar con él. Para otros que hemos aprendido diferentes Lenguajes de Programación no será tan rudo.

Empezaremos por aprender qué es Programación para luego ver qué es el Lenguaje Visual Basic For Applications. Pasaremos a ver varias definiciones como Variables y Constantes (junto con los ámbitos de estas), Tipos de Datos, Operadores (lógicos, de asignación, de comparación), Matrices (de una y varias dimensiones), Estructuras de Decisión (If…Then…Else…End If, Select Case…End Select), Estructuras de Bucle (Do While…Loop, Do Until, For…Next, For Each…Next). Luego veremos Procedimientos y Funciones Definidas por Usuarios para luego programar tareas sencillas en las Hojas de Trabajo con varios ejemplos, trabajar con Rangos (copiar, mover y otros). Veremos cómo dar una mejor presentación visual a las Hojas.

Por último entraremos en el mundo de los Formularios (UserForm) y sus diferentes controles. Trataré de ser lo más detallado posible pero sin llegar a dar una solución completa (con esto me refiero a que no voy a desarrollar un sistema de contabilidad por ejemplo). También veremos cómo colocar de manera ordenada, lógica y clara a la vista cada control.

Se intentará poner más ejemplos que teoría, pero sin embargo habrá casos que será al contrario ya que la teoría es muy necesaria para el aprendizaje. Recordemos “El que no lee, no aprende”.

Me basaré en varios libros, algunos tendré que traducirlos y otros no, pero trataré de que sea un aprendizaje bueno, eficiente[1] y eficaz[2].

[1] Eficiente: adj. Que consigue un propósito empleando los medios idóneos.

[2] Eficaz: adj. Que logra hacer efectivo un intento o propósito

Capítulo 1 PROGRAMACIÓN

En informática la programación es un proceso por el cual se escribe (en un lenguaje de programación), se prueba, se depura y se mantiene el código fuente de un programa informático. Dentro de la informática, los programas son los elementos que forman el software, que es el conjunto de las instrucciones que ejecuta el hardware de una computadora para realizar una tarea determinada. Por lo tanto, la programación es una de las principales áreas dentro de la informática.

Para el desarrollo de programas de cierta envergadura o complejos, con ciertas garantías de calidad, es conveniente seguir alguno de los modelos de desarrollo de software existentes, en donde la programación es sólo una de las etapas del proceso de desarrollo de software. Los modelos de desarrollo de software son tratados específicamente en la disciplina ingeniería del software dentro del campo de la informática.

PROGRAMAS Y ALGORITMOS

Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce a un lenguaje de programación concreto) un algoritmo. Nótese que es la secuencia de instrucciones en sí la que debe ser finita, no el número de pasos realizados como la ejecución de ellas.

Los programas suelen subdividirse en partes menores (módulos), de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa.

Según Niklaus Wirth un programa está formado por algoritmos y estructura de datos.

Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las siguientes:

- Programación Lineal.
- Programación Estructurada.
- Programación Modular.
- Programación Orientada a Objetos (POO).

COMPILACIÓN

El programa escrito en un lenguaje de programación (comprensible por el ser humano, aunque se suelen corresponder con lenguajes formales descritos por gramáticas independientes del contexto) no es directamente ejecutable por una computadora. La opción más común es compilar el programa obteniendo un módulo objeto, aunque también puede ser ejecutado a travéz de un intérprete informático. El código fuente del programa se debe someter a un proceso de traducción para convertirse en lenguaje máquina, interpretable por el procesador. A este proceso se le llama compilación.

Normalmente la creación de un programa ejecutable (un típico.exe para Microsoft Windows o DOS) conlleva dos pasos. El primer paso se llama compilación (propiamente dicho) y traduce el código fuente escrito en un lenguaje de programación almacenado en un archivo a código en bajo nivel, (normalmente en código objeto, no directamente a lenguaje máquina). El segundo paso se llama enlazado (del inglés *link* o *linker*) se enlaza el código de bajo nivel generado de todos los ficheros y subprogramas que se han mandado compilar y se añade el código de las funciones que hay en las bibliotecas del compilador para que el ejecutable pueda comunicarse directamente con el sistemas operativo, traduciendo así finalmente el código objeto a código máquina, y generando un módulo ejecutable.

Estos dos pasos se pueden mandar hacer por separado, almacenando el resultado de la fase de compilación en archivos objetos (un típico.obj para Microsoft Windows,DOS, o para Unix) para enlazarlos en fases posteriores, o crear directamente el ejecutable con lo que la fase de compilación se almacena sólo temporalmente. Un programa podría tener partes escritas en varios lenguajes (generalmente C, C++ y Asm), que se podrían compilar de forma independiente y enlazar juntas para formar un único ejecutable.

PROGRAMACIÓN E INGENIERÍA DEL SOFTWARE

Existe una tendencia a identificar el proceso de creación de un programa informático con la programación, que es cierta cuando se trata de programas pequeños para uso personal, y que dista de la realidad cuando se trata de grandes proyectos.

El proceso de creación de software desde el punto de vista de la Ingeniería tiene los siguientes pasos:

1. Reconocer la necesidad de un programa para solucionar un problema o identificar la posibilidad de automatización de una tarea.
2. Recoger los requisitos del programa. Debe quedar claro qué es lo que debe hacer el programa y para qué se necesita.
3. Realizar el análisis de los requisitos del programa. Debe quedar claro cómo debe realizar el programa las cosas que debe hacer. Las pruebas que comprueben la validez del programa se pueden especificar en esta fase.
4. Diseñar la arquitectura del programa. Se debe descomponer el programa en partes de complejidad abordable.
5. Implementar el programa. Consiste en realizar un diseño detallado, especificando completamente todo el funcionamiento del programa, tras lo cual la codificación debería resultar inmediata.
6. Implantar (instalar) el programa. Consiste en poner el programa en funcionamiento junto con los componentes que pueda necesitar (bases de datos, redes de comunicaciones, etc.)

La Ingeniería del Software se centra en los pasos de planificación y diseño del programa, mientras que antiguamente (programación artesanal) la realización de un programa consistía únicamente en escribir el código.

REFERENCIAS HISTÓRICAS

La primera programadora de computadora conocida fue Ada Lovelace, hija de Anabella Milbanke Byron y Lord Byron. Anabella introdujo en las matemáticas a Ada, quien después de conocer a Charles Babbage, tradujo y amplió una

descripción de su máquina analítica. Incluso aunque Babbage nunca completó la construcción de cualquiera de sus máquinas, el trabajo que Ada realizó con éstas le hizo ganarse el título de primera programadora de computadoras del mundo. El nombre del lenguaje de programación Ada fue escogido como homenaje a esta programadora. No olvidemos que este proceso está aplicado a todos los metodos científicos que actualmente se practican.

OBJETIVOS DE LA PROGRAMACIÓN

La programación debe perseguir la obtención de programas de calidad. Para ello se establece una serie de factores que determinan la calidad de un programa. Algunos de los factores de calidad más importantes son los siguientes:

- *Corrección.* Un programa es correcto si hace lo que debe hacer tal y como se estableció en las fases previas a su desarrollo. Para determinar si un programa hace lo que debe es muy importante especificar claramente qué debe hacer el programa antes de desarrollarlo y una vez acabado compararlo con lo que realmente hace.

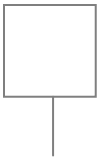
- *Claridad.* Es muy importante que el programa sea lo más claro y legible posible para facilitar así su desarrollo y posterior mantenimiento. Al elaborar un programa se debe intentar que su estructura sea sencilla y coherente, así como cuidar el estilo en la edición; de esta forma se ve facilitado el trabajo del programador, tanto en la fase de creación como en las fases posteriores de corrección de errores, ampliaciones, modificaciones, etc. Fases que pueden ser realizadas incluso por otro programador, con lo cual la claridad es aún más necesaria para que otros programadores puedan continuar el trabajo fácilmente.

- *Eficiencia.* Se trata de que el programa además de realizar aquello para lo que fue creado, es decir que sea correcto, lo haga gestionando de la mejor forma posible los recursos que utiliza. Normalmente al hablar de eficiencia de un programa se suele hacer referencia al tiempo que tarda en realizar la tarea para la que ha sido creado y en la cantidad de memoria que necesita, pero hay otros recursos que también pueden ser de consideración al obtener la eficiencia de un programa dependiendo de su naturaleza (espacio en disco que utiliza, tráfico de red que genera, etc.).

- *Portabilidad.* Un programa es portable cuando tiene la capacidad de poder ejecutarse en una plataforma, ya sea hardware o software, diferente a en la que ha sido elaborado. La portabilidad es una característica muy deseable para un programa ya que permite, por ejemplo, a un programa que se ha desarrollado para sistemas GNU/Linux ejecutarse también en la familia de sistemas operativos Windows. Esto permite que el programa pueda llegar a más usuarios más fácilmente.

VISUAL BASIC FOR APPLICATIONS

Microsoft VBA *(Visual Basic for Applications)* es el lenguaje de macros de Microsoft Visual Basic que se utiliza para programar aplicaciones Windows y que se incluye en varias aplicaciones Microsoft. VBA permite a usuarios y programadores ampliar la funcionalidad de programas de Microsoft Office. Visual Basic para Aplicaciones es un subconjunto casi completo de Visual Basic 5.0 y 6.0.



Microsoft VBA viene integrado en aplicaciones de Microsoft Office, como Word, Excel y Access, Powerpoint y Visio. Prácticamente **casi** cualquier cosa que se pueda programar en Visual Basic 5.0 o 6.0 se puede hacer también dentro de un documento de Office, con la sola limitación que el producto final no se puede compilar separadamente del documento, hoja o base de datos en que fue creado; es decir, se convierte en una macro (o más bien súper macro). Esta macro puede instalarse o distribuirse con sólo copiar el documento, presentación o base de datos.

Su utilidad principal es automatizar tareas cotidianas, así como crear aplicaciones y servicios de bases de datos para el escritorio. Permite acceder a las funcionalidades de un lenguaje orientado a eventos con acceso a la API de Windows. Este lenguaje ha sido implementado en otros productos como StarBasic en StarOffice que después heredaría OpenOffice.

EJEMPLOS

VBA es útil para automatizar tareas en una base de datos, como por ejemplo, recorrer una tabla:

```
Sub LoopTableExample

    Dim db As DAO.Database
    Dim rcs As DAO.Recordset
    Set db = CurrentDb
    Set rcs = db.OpenRecordset("SELECT * FROM tblMain")
    Do Until rcs.EOF
        MsgBox rcs!FieldName
        rcs.MoveNext
    Loop
    rcs.Close
    db.Close
    Set rcs = Nothing
    Set db = Nothing
End Sub
```

VBA puede ser usado para crear una función definida por el usuario para usar en una hoja de Microsoft Excel:

```
Public Function BUSINESSDAYPRIOR(dt As Date) As Date
    Select Case Weekday(dt, vbMonday)
    Case 1
        BUSINESSDAYPRIOR = dt -3
    Case 7
        BUSINESSDAYPRIOR = dt -2
    Case Else
        BUSINESSDAYPRIOR = dt -1
    End Select
End Function
```

VBA también tiene acceso a funciones internas de Windows en diversos grados, y puede accesar recursos desde horarios hasta archivos y control:

```
Sub ObtenerFecha()
    MsgBox "La fecha es " & Format(Now(), "dd-mm-yyyy")
End Sub
```

Se puede accesar al lenguaje al ingresar al menú **herramientas**. Y una vez allí **MACRO** y **EDITOR DE VISUAL BASIC**. Otra forma más rápida sería presionando las teclas ALT-F11

FUTURO DE VBA

El siguiente paso natural en la evolución de VBA es dejar de ser un subconjunto de Visual Basic y serlo de la plataforma .NET. Microsoft no planea hacer mejoras significativas a VBA en el futuro. Aunque continuará dando soporte a las licencias de VBA que se han ido ofreciendo, VBA está siendo sustituido por las Herramientas para Aplicaciones de Microsoft Visual Studio (VSTA: Visual Studio Tools for Applications) y las Herramientas para Office de Microsoft Visual Studio (VSTO: Visual Studio Tools for Office). Estas herramientas funcionan bajo la plataforma .NET. Desde el 1 de Julio de 2007, Microsoft ya no ofrece nuevas licencias de VBA a nuevos clientes. Los que poseían una licencia de VBA podrán conseguir una licencia de las nuevas soluciones por parte de Microsoft.

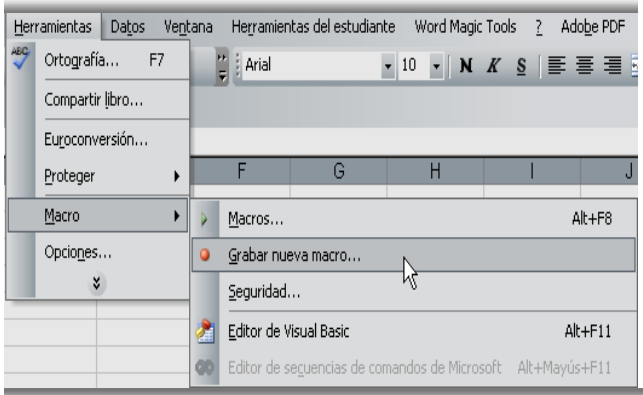
Capítulo 2 VARIABLES, CONSTANTES, TIPOS DE DATOS Y OPERADORES

En este apartado veremos los diferentes conceptos sobre Variables, Constantes, Tipos de Datos y Operadores. Trataré de poner ejemplos de cada uno de los conceptos para que se entienda major la forma en que trabajan cada uno y de qué manera se pueden utilizar.

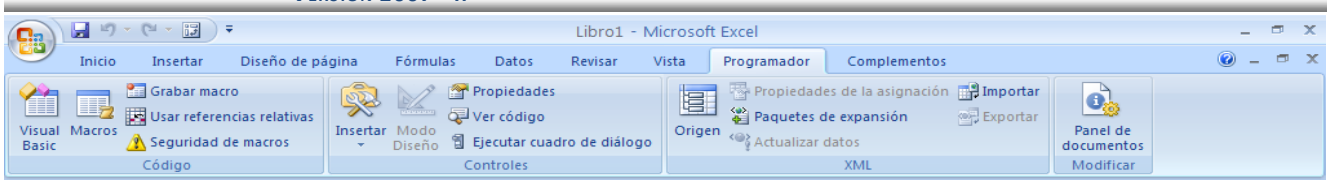
MACROS

Una macro es la automatización de una tarea que se repite constantemente. Por ejemplo, si se desea poner en mayúsculas cierta cantidad de celdas y esto se tiene que hacer todos o casi todos los días, lo mejor es hacer una macro para no tener que estar repitiendo el proceso.

VERSIÓN 2003- I



VERSIÓN 2007 - II



CREACIÓN (GRABACIÓN) DE UNA MACRO

Para crear una macro con la versión de Excel 2003, en el menú Herramientas-Macros-Grabar Nueva Macro. En la versión 2007 se escoge la plantilla Programador y en la sección de Código empezamos una nueva macro.

Si no ves la carpeta de Programador la puedes activar de la siguiente manera: Presionas en el Botón de Oficina, seleccionas Opciones de Excel. En la pantalla que sale, en la parte derecha podrás activarla.

Para detener la macro tienes que seguir los mismos pasos de la creación, con la diferencia que en la opción donde decía “Grabar nueva macro…”, se cambiará por “Detener Grabación”.

Para corer la macro puedes presionar las teclas Alt-F8, en la pantalla escoges la macro y presionas el botón de Ejecutar. Esto es igual en las versiones 2003 y 2007.

Como ejemplo vamos a suponer que queremos poner en negrita el rango de A2:B15. Iniciamos la creación de la macro, marcamos el rango ántes mencionado, presionamos Ctrol-N y luego detenemos la grabación.

Como se puede ver, crear una macro es de lo más sencillo del mundo. Aún recuerdo cuando usaba Lotus 123, para crear una macro se tenía que seguir ciertos pasos y para activarla se presionaba la teclas Alt para desplegar el menu y correrla (qué tiempos aquellos). También se puede deducir que muchas tareas repetitivas se pueden simplificar con la simple creación de una macro.

PROGRAMACIÓN VBA VRS. MACROS

Bueno amigos, entramos a la parte fuerte del tema. Espero que lo disfruten al igual que lo haré yo.

Ya vimos anteriormente que una macro es para automatizar una serie de pasos repetitivos. Ahora daré mi opinión sobre lo que es la programación en VBA.

A mi concepto, programar en VBA es tratar de explotar nuestra imaginación al máximo. Note que dije imaginación, ya que en muchos lugares lo que menos enseñan es a explotar esta parte de nuestro cerebrito. La programación en VBA no solo requiere que sepamos un conjunto de instrucciones, también es necesario imaginar el producto final. Al igual que la escultura, pintura y muchas otras profesiones, en donde el creador primeramente ve su obra en la mente y luego la plasma por medio del material que está utilizando para que muchos la admiren.

Muchas personas piensan que porque tienen seis meses aprendiendo VBA ya saben mucho, pero lejos están de saber todo lo que se puede explotar este mini lenguaje. Muchos solo programan ciertas cosas en Excel y se olvidan de las demás aplicaciones en las que se puede ejecutar como en Word, PowerPoint, Access, Visio, AutoCad y algunos otros.

Por medio de la programación damos más que una solución a nuestros clientes, damos una obra de arte que es utilizada para un fin específico.

Por medio de VBA manipulamos los objetos de las diferentes aplicaciones antes mencionadas. Controlamos los eventos de los mismos para que realicen lo que queremos en el momento y lugar deseado. Los objetos de hojas, libros, celdas, formularios y otros, los controlamos casi a nuestro antojo junto con sus eventos.

Sería bueno que aparte de aprender VBA, tengamos conocimientos sobre Bases de Datos (no me refiero al simple uso de Access), otros lenguajes de programación, Redes, Ingeniería Forense y cualquier otra cosa sobre informática. Esto nos permite cotizarnos más alto en el mercado laboral ya que estaremos en capacidad de dar un incentivo adicional a nuestra clientela.

Que no le pase lo de muchos, que se ponen a crear ciertas rutinas bajando código de Internet o modificando un poquito las macros que crearon, y después se enojan con los expertos porque no les dan soluciones completas. Quieren hacer conexiones entre Access y Excel y ni siquiera saben las principales funciones del Excel.

Ahora entrando en materia vamos a la siguiente sección.

LO QUE NO ES EXCEL

A diferencia de lo que muchos piensan, Excel no es un buen manejador de Bases de Datos. Muchos se refieren a un archivo como “Mi Base de Datos”. En realidad son hojas de cálculo. A un archivo de Excel se le llama Libro ya que puede contener varias hojas de cálculo en él.

Estas hojas las podemos integrar por medio de funciones, fórmulas o con VBA para que funcione parecido a una Base de Datos, con el inconveniente que no graba en tiempo real la información, lo cual nos puede traer pérdida de datos en caso de que falte el fluido eléctrico. Esto es muy crítico para una persona que digite muy rápido ya sea con el teclado numérico o alfanumérico.

Excel no lo recomiendo para llevar una contabilidad; tampoco para el manejo de inventarios, pero en este último es un poquito más pasable.

A mi concepto, Excel es un buen asistente financiero, estadístico y matemático. Nos permite crear muy buenos gráficos para la buena interpretación de los datos.

VARIABLES

En programación, las variables son estructuras de datos que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal pudiendo ser de longitud:

- Fija.- Cuando el tamaño de la misma no variará a lo largo de la ejecución del programa. Todas las variables, sean del tipo que sean tienen longitud fija, salvo algunas excepciones — como las colecciones de otras variables (arrays) o las cadenas.
- Variable.- Cuando el tamaño de la misma puede variar a lo largo de la ejecución. Típicamente colecciones de datos.

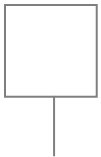
Las variables pueden albergar una gran variedad de tipos de datos, para simples valores Boolean (Falso, Verdadero) o valores de doble precisión. Los nombres de variables tienen que ser lo más descriptivos posibles.

REGLAS PARA LOS NOMBRES DE VARIABLES

i. Se pueden usar caracteres del alfabeto, números y algunos caracteres de puntuación, pero el primer carácter debe ser del alfabeto.

ii. El VBA no distingue entre mayúsculas y minúsculas. Esto nos obliga a darle nombres legibles, por ejemplo: InvInicial (para definir el inventario inicial). Caso contrario ocurre en C, C++, Visual C++, C#, Delphi y otros, los cuales si distinguen entre mayúsculas o minúsculas.

iii. No se pueden utilizar espacios, periodos (:) ni puntos. Para que sea más legible se utiliza el guion bajo (Inv_Inicial).



iv. No se deben utilizar estos caracteres: #, $, %, &, ¡ o !.

v. Los nombres pueden contener hasta 254 caracteres (caso que nunca se da por lo ilógico de usar un nombre de este largo).

vi. El VBA tiene algunas palabras reservadas que no se pueden utilizar ya que dará un error a la hora de ejecución del programa. Por ejemplo, si le ponemos de nombre Next a una variable, nos dará un error en tiempo de sintaxis. Si recibe un mensaje de error puede consultar la ayuda Online. Se tiene que verificar bien los nombres de variable para evitar esto.

MI PRIMERA MACRO - ¡YA ERA HORA!

Bueno amigos y amigas, me imagino que para esta hoja ya están ansiosos de saber cómo se crea una macro. Bueno, vamos a crear una hoja, pero no la vamos a crear a lo que salga. Vamos a hacerla de manera más presentable. Utilizaremos dos celdas como variables para poder desplegar nuestra información en el rango que deseamos. Los datos de la hoja deberán ser como sigue:

	A	B	C	D	E	F	G
1	Fecha	Fecha	Monto $	T.C.$	Monto ¢	Inquilino	Ced_Juridica
2	>=39356	<=39721					
3							
4							
5							
6	Fecha	Monto $	T.C.$	Monto ¢	Inquilino	Ced_Juridica	
7	07/10/2007	1,942.50	520.74	1,011,537.45	Grupo de Ctros. de Serv. S.A.	3-101-375733	
8	23/11/2007	1,942.50	521.23	1,012,489.28	Grupo de Ctros. de Serv. S.A.	3-101-375733	
9	17/12/2007	1,942.50	501.84	974,824.20	Grupo de Ctros. de Serv. S.A.	3-101-375733	
10	03/01/2008	1,942.50	500.97	973,134.23	Grupo de Ctros. de Serv. S.A.	3-101-375733	
11	03/03/2008	900.00	499.02	449,118.00	Daniel Mena Soto	11-128-092936	
12	11/04/2008	900.00	497.14	447,426.00	Daniel Mena Soto	11-128-092936	
13	14/05/2008	900.00	516.16	464,544.00	Daniel Mena Soto	11-128-092936	
14	19/06/2008	900.00	522.90	470,610.00	Daniel Mena Soto	11-128-092936	
15	21/07/2008	900.00	556.32	500,688.00	Daniel Mena Soto	11-128-092936	
16	21/08/2008	900.00	556.30	500,670.00	Daniel Mena Soto	11-128-092936	
17	22/09/2008	900.00	558.89	503,001.00	Daniel Mena Soto	11-128-092936	
18							
19							
20							
21							
22							
23							
24							
25							

TblResumen | TblAlquileres | Hoja2 | Hoja3

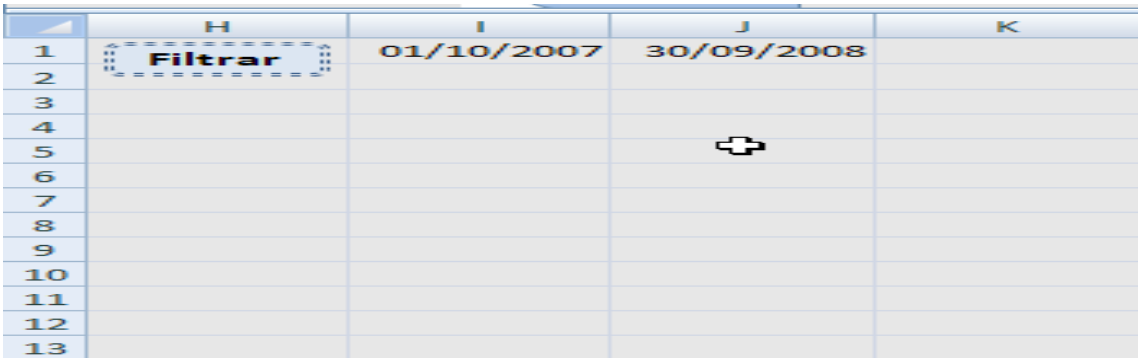
Listo

TABLA DE DATOS 1

Observen que no estoy empezando la tabla a partir de la celda A1, en su lugar estoy poniendo los títulos principales a partir de la celda A6:F6. Estos mismos títulos los copio en A1:G1. Sin embargo también observen que el título Fecha lo pongo dos veces. Esto es para cuando voy a crear mi Filtro Avanzado programado

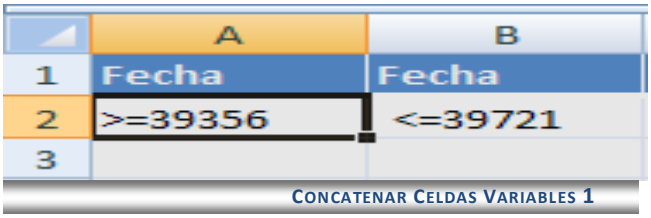
me tome el primer título Fecha como inicial y el otro como final, sería algo como decirle: desde la fecha X hasta la fecha Y.

Aquí estamos aprendiendo a hacer un Filtro Avanzado pero con la diferencia que va a ser programado. Ahora vamos a insertar una Autoforma y en las celdas I1 y J1 pondremos nuestras fechas (inicial y final) quedando de la siguiente forma:



CELDAS VARIABLES Y AUTOFORMA 1

En el siguiente paso vamos a crear una fórmula en las celdas A1 y B1 las cuales serán una concatenación con los operadores =, >, < y los datos de las celdas I1 y J1, las cuales son: en la celda A1 ponemos =">="&I1, en B1 ="<="&J1. Esto hará que nos quede como lo muestra la imagen.



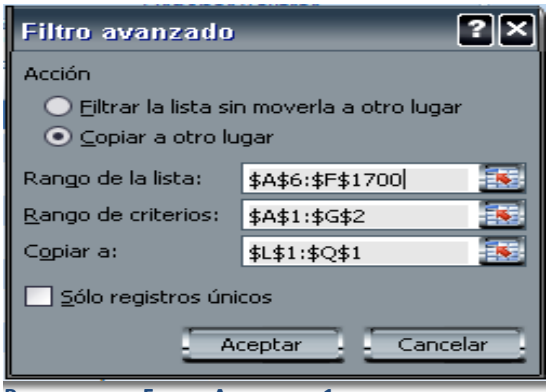
CONCATENAR CELDAS VARIABLES 1

Notemos que no muestra las fechas que pusimos en las celdas variables ya que Excel nos lo convierte para poder trabajar con cálculos. Le podríamos dar formato para que aparezca la fecha pero mejor lo dejamos como está (a menos que lo quieras hacer de ejercicio). No explico cómo insertar Autoformas ya que considero que la persona que está aprendiendo macros ya debería saberlo. Los pasos dados son para la versión 2007 y 2003.

Ahora que tenemos nuestra Tabla formada empezaremos a crear nuestra primer macro. Nos posicionamos en la celda A6. Activamos la grabadora de macros (los pasos ya fueron explicados para las dos versiones de Excel) y empezaremos a crear un Filtro Avanzado.

I. Versión 2003 – menú Datos-Filtro-Filtro Avanzado.

II. Versión 2007 – ficha Datos-Avanzadas



PANTALLA DEL FILTRO AVANZADO 1

Esto nos desplegará una pantalla en la cual escogemos la segunda opción. Aquí le daremos los rangos de Lista, Criterios y dónde copiar, así como lo muestra la imagen. Observemos que en el rango de Lista he modificado el alcance para que no solo tome los datos hasta la celda F17, sino hasta la F1700 por si decido introducir más datos no tener que estar modificando la macro. Los datos los copiaremos a partir de la celda L1.

Presionamos el botón Aceptar. Detenemos la Grabadora de Macros y listo, ya hemos creado nuestra primer macro.

Para no tener que repetir los pasos anteriores cada vez que cambiemos las fechas, lo que hacemos es asignar la macro a la Autoforma. Posicionamos el puntero del Mouse sobre la Autoforma y presionamos el botón derecho, en el menú que se despliega seleccionamos Asignar Macro. En la pantalla que sigue escogemos nuestra macro y presionamos el botón de Aceptar.

Como ejercicio podemos cambiar las fechas de inicio y fin, luego presionamos sobre la Autoforma y veremos nuestra maravillosa macro. También podemos agregar más datos y verás que los toma en cuenta ya que aumentamos el rango en la Lista.

Muchos se preguntaran "Y si quiero pegar los datos en otra hoja ¿cómo hago?". Bueno ese paso lo veremos cuando empecemos a ver Programación VBA, aún es muy temprano. Recordemos que primero empezamos a gatear como los bebes y después podremos correr.

OTRO EJEMPLO – CAMBIANDO Y RESTAURANDO UN RANGO

	A	B	C	D	E
1	Fecha	Concepto	Cargos	Abonos	Saldos
2					
3	30/05/2005	Cierre del Balance, Mayo			29,344.00
4	01/06/2005	Compra de la Póliza del Seguro Médico		6,864.00	22,480.00
5	01/06/2005	Compra de Materiales para Oficina		3,194.00	19,286.00
6	04/06/2005	Recibo de Bancos	4,690.00		23,976.00
7	04/06/2005	Cheque para pagar Materiales de Oficina	91.00		24,067.00
8	07/06/2005	Recibo de Bancos	1,006.00		25,073.00
9	11/06/2005	Recibo de Bancos	8,207.00		33,280.00
10	14/06/2005	Recibo de Bancos	9,592.00		42,872.00
11	14/06/2005	Compra de libros, Editorial Neal		6,023.00	36,849.00
12	14/06/2005	Compra de libros, Distribuidora Lenay		8,474.00	28,375.00
13	18/06/2005	Recibo de Bancos	4,663.00		33,038.00
14	18/06/2005	Cuentas por Cobrar pago de Mayo	17,951.00		50,989.00
15	21/06/2005	Recibo de Bancos	5,514.00		56,503.00
16	23/06/2005	Recibo de Bancos	3,791.00		60,294.00
17	27/06/2005	Factura del Teléfono, Mayo		1,835.00	58,459.00
18	27/06/2005	Recibo de Bancos	9,050.00		67,509.00
19	27/06/2005	Compra de libros, Editorial Neal		6,440.00	61,069.00
20	29/06/2005	Cheque de Sueldo, Rodgers		2,950.00	58,119.00
21	29/06/2005	Cheque de Sueldo, Rouse		2,761.00	55,358.00
22	29/06/2005	Cheque de Sueldo, Tofoya		4,377.00	50,981.00
23	29/06/2005	Factura de Publicidad, Mayo		3,116.00	47,865.00
24	30/06/2005	Recibo de Bancos	6,841.00		54,706.00
25					
26					

TABLA PARA FORMATOS 1

En este ejemplo trabajaremos con una tabla que no tiene ningún tipo de formato. La Tabla que utilizaremos debe quedar como la de la imagen.

Por medio de macros y le pondremos Negrita a los títulos y las fechas, crearemos un Formato Condicional que nos cambie el fondo a gris en las filas impares. Por último lo restauramos. También insertaremos dos Autoformas para asignarlas a las respetivas macros de dar formato y quitarlo.

Activamos la grabadora de macros, en la pantalla que aparece le ponemos como nombre FormatoTabla. Ahora los pasos a seguir son los siguientes:

1. Seleccionamos el rango de A1:E1
2. Presionamos sobre el botón de relleno y escogemos el color negro. 
3. Presionamos el botón de color de fuente y escogemos el blanco. 
4. Presionamos el botón de Negrita para que las letras queden más resaltadas.
5. Ahora seleccionamos el rango A3:E24.
6. Vamos al menú Formato y escogemos Formato Condicional. En la pantalla que nos aparece, en la primera opción, cambiamos a Fórmula.

7. Esto nos cambiará el aspecto de la pantalla. A la par de Fórmula, en el espacio que hay digitamos la siguiente fórmula: =Residuo(Fila(),2)
8. Presionamos el botón que dice Formato y en la pestaña que dice Tramas escogemos el color gris. Presionamos en Aceptar, al igual volvemos a presionar en aceptar.
9. Detenemos la grabadora de macros.

De esta forma ya hemos creado la macro para dar formato a nuestra Tabla, la cual debe quedar como la imagen.

	A	B	C	D	E
1	Fecha	Concepto	Cargos	Abonos	Saldos
2					
3	30/05/2005	erre del Balance, Mayo			29,344.00
4	01/06/2005	Compra de la Póliza del Seguro Médico		6,864.00	22,480.00
5	01/06/2005	Compra de Materiales para Oficina		3,194.00	19,286.00
6	04/06/2005	Recibo de Bancos	4,690.00		23,976.00
7	04/06/2005	Cheque para pagar Materiales de Oficina	91.00		24,067.00
8	07/06/2005	Recibo de Bancos	1,006.00		25,073.00
9	11/06/2005	Recibo de Bancos	8,207.00		33,280.00
10	14/06/2005	Recibo de Bancos	9,592.00		42,872.00
11	14/06/2005	Compra de libros, Editorial Neal		6,023.00	36,849.00
12	14/06/2005	Compra de libros, Distribuidora Lenay		8,474.00	28,375.00
13	18/06/2005	Recibo de Bancos	4,663.00		33,038.00
14	18/06/2005	Cuentas por Cobrar pago de Mayo	17,951.00		50,989.00
15	21/06/2005	Recibo de Bancos	5,514.00		56,503.00
16	23/06/2005	Recibo de Bancos	3,791.00		60,294.00
17	27/06/2005	Factura del Teléfono, Mayo		1,835.00	58,459.00
18	27/06/2005	Recibo de Bancos	9,050.00		67,509.00
19	27/06/2005	Compra de libros, Editorial Neal		6,440.00	61,069.00
20	29/06/2005	Cheque de Sueldo, Rodgers		2,950.00	58,119.00
21	29/06/2005	Cheque de Sueldo, Rouse		2,761.00	55,358.00
22	29/06/2005	Cheque de Sueldo, Tofoya		4,377.00	50,981.00
23	29/06/2005	Factura de Publicidad, Mayo		3,116.00	47,865.00
24	30/06/2005	Recibo de Bancos	6,841.00		54,706.00

A continuación haremos una macro que nos quite todos los formatos creados invirtiendo los pasos anteriores. Los pasos a seguir son los siguientes:

I. Activamos la grabadora de macros.

II. Seleccionamos el rango A1:E1. Presionamos sobre el botón de relleno y escogemos la opción Sin Relleno. 

III. Ahora presionamos sobre el botón de color de fuente y escogemos la opción de Automático. 

IV. Seleccionamos el rango A3:E24. En el menú Formato seleccionamos Formato Condicional.

V. En la pantalla que aparece presionamos el botón de Eliminar.

VI. Nos posicionamos en la celda A1 y detenemos la grabadora de macros.

Solo nos queda asignar cada macro a una Autoforma. Este paso ya se explicó anteriormente. A una Autoforma le ponemos el nombre de Dar Formato y a la otra Quitar Formato.

Listo, con esto ya tenemos una forma de trabajar más profesionalmente dándole una presentación más agradable a nuestra Tabla.

CONSTANTES

En programación, las constantes son tipos de datos (con valores numéricos o de cadena) que permanecen invariables, sin posibilidad de cambiar el valor que tienen durante el curso del programa. Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

Por ejemplo:

- El valor de pi = 3.141592

Por conveniencia, el nombre de las constantes suele escribirse en mayúsculas en la mayoría de lenguajes.

TIPOS DE DATOS

Según mi maestro John Walkenbach el tipo de datos se refiere a cómo se guardan los datos en la memoria, como números enteros, reales, series, etc.

VBA tiene en cuenta los tipos de datos pero eso nos puede inducir a ciertos errores, por lo que es conveniente declarar el tipo de dato que vamos a utilizar ya que nuestro código se puede hacer lento en su ejecución a la hora de estar convirtiendo un tipo de dato a otro. Esto no parece estorbar mucho, pero si tenemos aplicaciones grandes realmente lo vamos a lamentar. Asignar el tipo de dato adecuado nos ayuda a manejar mejor la memoria de nuestro ordenador (ya extrañaba esa palabra). Otra ventaja es que VBA se puede dar cuenta de algún error a la hora de la compilación del código. Cuando hablo de compilación no crean que se va a crear un archivo ejecutable como ocurre con C#, Visual Basic.NET u otros lenguajes, lastimosamente Excel no cuenta con esta ventaja.

En la siguiente tabla muestro los diferentes tipos de datos que podemos utilizar.

Tipo de Dato	Bytes que usa	Serie de Valores
Byte	1 Byte	0 a 255
Boolean	2 Bytes	Verdadero o Falso (True – False)
Entero (Integer)	2 Bytes	-32,768 a 32,767

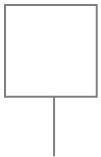
Tipo de Dato	Bytes que usa	Serie de Valores
Largo (Long)	4 Bytes	-2,147,483,648 a 2,147,483,647
Único (Unique)	4 Bytes	-3.402823E38 a -1.401298E-45 para valores negativos 1.401298E-45 a 3.402823E38 para valores positivos
Doble (Double)	8 Bytes	-1.79769313486232E308 hasta -4.94065645841247E-324 para valores negativos 4.94065645811247E-324 hasta 1.79769313486232E308
Moneda (Currency)	8 Bytes	-922,337,203,685,477.5808 hasta 922,337,203,685,477.5807
Decimal	14 Bytes	+/-79,228,162,514,264,337,593,543,950,335 sin punto +/-7.9228162514264337593543950335 con 28 elementos a la derecha del punto decimal
Fecha (Date)	8 Bytes	Enero 1, 0100 a Diciembre 31, 9999
Objeto (Object)	4 Bytes	Ninguna referencia de Objeto
Cadena (Largo de la cadena variable)	10 Bytes + largo de la cadena	De 0 a aproximadamente 2 billones
Cadena (Con largo mixto)	El largo de la cadena	1 a aproximadamente 65,400
Variante (Variant) con números	16 Bytes	Cualquier valor de número mayor que un tipo de datos doble
Variante (Variant) con letras	22 Bytes + el largo de la serie	0 a aproximadamente 2 billones
Definido por usuario (UD)	Varía	Varía según el elemento

Cabe destacar que el tipo Decimal funciona de la versión 2000 en adelante.

FUNCIONES PARA CONVERTIR TIPOS DE DATOS

En Excel tenemos varias funciones con las cuales nos podemos ayudar a convertir datos de un tipo a otro. Esto nos da la seguridad de utilizar el tipo correcto a la hora de hacer conversiones o trabajar con las Variables y Constantes adecuadas.

A continuación la tabla para la conversión de tipo de datos:



Sintaxis

CBool(*expresión***)**	**CByte(***expresión***)**	**CCur(***expresión***)**	**CDate(***expresión***)**
CDbl(*expresión***)**	**CDec(***expresión***)**	**CInt(***expresión***)**	**CLng(***expresión***)**
CSng(*expresión***)**	**CStr(***expresión***)**	**CVar(***expresión***)**	

El argumento obligatorio *expresión* es cualquier expresión de cadena o expresión numérica.

Tipos devueltos

El nombre de la función determina el tipo devuelto, como se muestra a continuación:

Función	Tipo devuelto	Intervalo del argumento *expresión*
CBool	Boolean	Cualquier expresión de cadena o numérica válida.
CByte	Byte	0 a 255.
CCur	Currency	-922.337.203.685.477,5808 a 922.337.203.685.477,5807.
CDate	Date	Cualquier expresión de fecha.
CDbl	Double	-1.79769313486231E308 a -4,94065645841247E-324 para valores negativos; 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.
CDec	Decimal	+/-79.228.162.514.264.337.593.543.950.335 para números basados en cero, es decir, números sin decimales. Para números con 28 decimales, el intervalo es +/-7,9228162514264337593543950335. La menor posición para un número que no sea cero es 0,0000000000000000000000000001.
CInt	Integer	-32.768 a 32.767; las fracciones se redondean.
CLng	Long	-2.147.483.648 a 2.147.483.647; las fracciones se redondean.
CSng	Single	-3,402823E38 a -1,401298E-45 para valores negativos; 1,401298E-45 a 3,402823E38 para valores positivos.
CStr	String	El mismo intervalo que **Double** para valores numéricos. El mismo intervalo que **String** para valores no numéricos.
CVar		

EJEMPLOS DE CONVERSIÓN DE TIPOS DE DATOS

Aquí les dejo unos cuantos ejemplos que vienen en el mismo Excel

Ejemplo de la función CBool

En este ejemplo se utiliza la función **CBool** para convertir una expresión a un tipo de datos **Boolean**. Si la expresión se evalúa como un valor diferente de cero, **CBool** devolverá **True**; de lo contrario, devolverá **False**.

```
Dim A, B, Prueba
A = 5: B = 5                ' Inicializa variables.
Prueba = CBool(A = B)     ' Prueba contiene True.
A = 0     ' Define variable.
Prueba = CBool(A)     ' Prueba contiene False.
```

Ejemplo de la función CByte

En este ejemplo se utiliza la función **CByte** para convertir una expresión a un tipo de datos **Byte**.

```
Dim MiDoble, MiByte
MiDoble = 125.5678    ' MiDoble es un Double.
MiByte = CByte(MyDouble)     ' MiByte contiene 126.
```

Ejemplo de la función CCur

En este ejemplo se utiliza la función **CCur** para convertir una expresión a un tipo de datos **Currency**.

```
Dim MiDoble, MiMoneda
MiDoble = 543.214588    ' MiDoble es del tipo Double.
MiMoneda = CCur(MiDoble * 2) ' Convierte el resultado
de MiDoble * 2 (1086.429176) al tipo
Currency(1086.4292).
```

Ejemplo de la función CDate

En este ejemplo se utiliza la función **CDate** para convertir una cadena en un tipo de datos **Date**. En general, no se recomienda utilizar fechas y horas como cadenas de caracteres en el código definitivo de una aplicación (como se muestra en el ejemplo). Use en su lugar literales de fecha y literales de hora (como #2/12/1969# y #4:45:23 PM#).

```
Dim MiFecha, MiHoraCorta, MiHora, MiHoraCorta
MiFecha = "12 febrero 1969"        ' Define la fecha.
MiFechaCorta = CDate(MiFecha)' Convierte al tipo Date.
MiHora = "4:35:47 PM"    ' Define la hora.
MiHoraCorta = CDate(MiHora) 'Convierte al tipo Date.
```

Ejemplo de la función CDbl

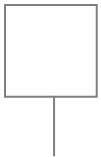
En este ejemplo se utiliza la función **CDbl** para convertir una expresión a un tipo de datos **Double**.

```
Dim MiMoneda, MiDoble
MiMoneda = CCur(234.456784)     ' MiMoneda es Currency.
MiDoble = CDbl(MiMoneda * 8.2 * 0.01)
' Convierte el resultado a un tipo de datos Double.
```

Ejemplo de la función CDec

En este ejemplo se utiliza la función **CDec** para convertir un valor numérico a **Decimal**.

```
Dim MiDecimal, MiMoneda
MiMoneda = 10000000.0587    ' MiMoneda es una moneda.
MiDecimal = CDec(MyCurr)     ' MiDecimal es un decimal.
```



Ejemplo de la función CInt

En este ejemplo se utiliza la función **CInt** para convertir un valor a un tipo de datos **Integer**.

```
Dim MiDoble, MiEntero
MiDoble = 2345.5678         ' MiDoble es un Double.
MiEntero = CInt(MiDoble)     ' MiEntero contiene 2346.
```

Ejemplo de la función CLng

En este ejemplo se utiliza la función **CLng** para convertir un valor a un tipo de datos **Long**.

```
Dim MiValor1, MiValor2, MiLargo1, MiLargo2
MiValor1 = 25427.45: MiValor2 = 25427.55    ' MiValor1,
MiValor2 son Doubles.
MiLargo1 = CLng(MiValor1)     ' MiLargo1 contiene 25427.
MiLargo2 = CLng(MiValor2)     ' MiLargo2 contiene 25428.
```

Como podemos observar, los ejemplos son sencillos pero entendibles para cualquiera. Muchos a estas alturas se estarán preguntando para qué sirve el Dim. Bueno el Dim quiere decir Dimension, este nos permite declarar Variables y asignarle un espacio de almacenamiento.

DECLARACIÓN DE VARIABLES

Ya vimos anteriormente que para declarar una variable únicamente ponemos la palabra Dim, luego el nombre de la Variable y por último el tipo de Variable que utilizaremos. Con esta información solo nos queda hacer lo siguiente:

Dim MiVariable As Integer

Dim MiFechas As Date

Estos son solo dos ejemplos, con el transcurso del tiempo ya veremos en la práctica la forma de aplicarlo a nuestros códigos.

OPTION EXPLICIT

Esta instrucción nos permite forzar la declaración de las variables. La ventaja de esto es no tener que poner nombres a lo loco y luego no saber si el despliegue de nuestros resultados es el correcto.

Hay personas que no utilizan esta opción y se aventuran a declarar variables con un nombre, en el transcurso del código necesitan esa variable pero resulta que se le olvida cómo fue que la escribió obteniendo resultados inesperados. Por ejemplo:

Dim MiSaldo As Double

… (Resto del código)

MisSaldos = Debitos – Créditos

Como puedes ver primeramente se declaró MiSaldo y después utilizamos MisSaldos. En esta parte VBA hará un cálculo con la nueva variable MisSaldos pudiendo producir un resultado no deseado.

Con Option Explicit activado y compilamos, inmediatamente nos dará un error, que no hemos declarado la variable y se detendrá sobre esta poniendo el texto en amarillo. Como vemos esto nos ahorra dolores de cabeza.

La misma explicación anterior es aplicable para las Constantes.

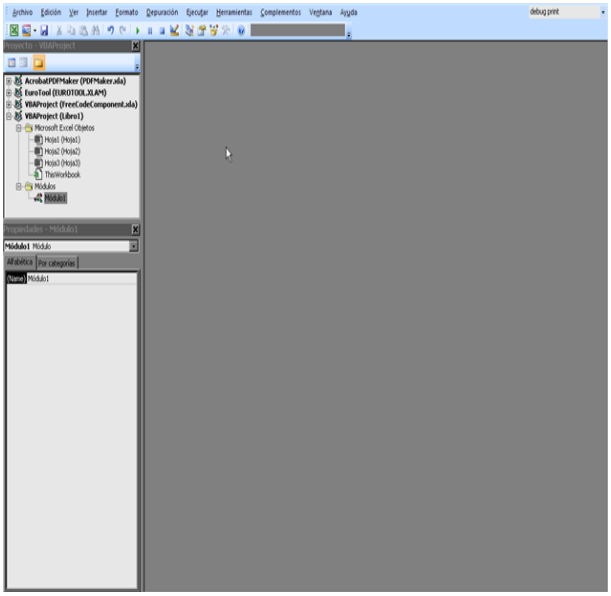
Ahora pasaremos a ver los distintos tipos de operadores en Visual Basic For Applications. Estos se dividen en los siguientes grupos: Lógicos, de Asignación, de Comparación y Aritméticos.

Operadores Aritméticos

Operador ^ (Exponenciación)

Este carácter se utiliza para elevar a la potencia deseada un determinado número. La sintaxis que se utiliza es la siguiente: Numero ^ Exponente.

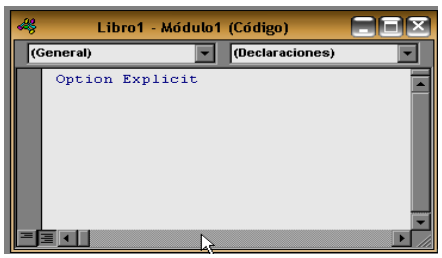
```
Sub expone()
   Dim MiNumero As Integer
   Dim Exponente As Integer
   Dim Resultado As Integer
   MiNumero = 3
   Exponente = 2
   Resultado = MiNumero ^ Exponente
   Debug.Print Resultado
End Sub
```



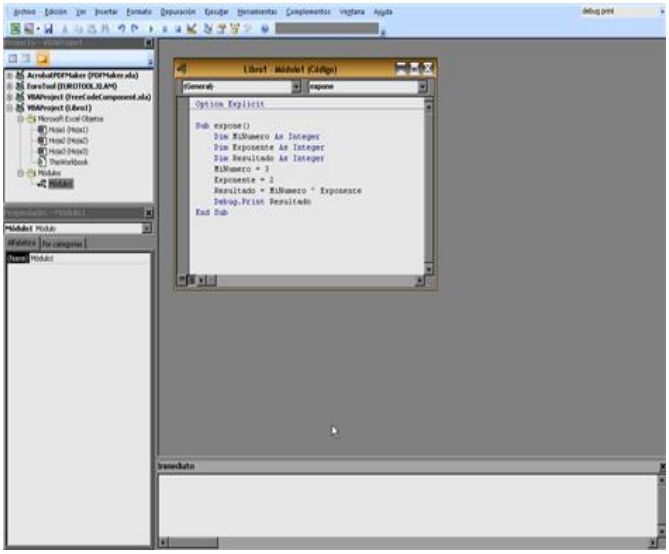
Editor de Visual Basic For Applications

Como vemos, aquí tenemos un nuevo concepto que es Debug.Print el cual nos permite ver el resultado de nuestra macro.

Para que comprendan mejor el uso de esa instrucción tenemos que entrar al Editor de Visual Basic For Applications. Esto lo hacemos presionando las teclas Ctrol-F11. Una vez dentro veremos una pantalla como la imagen.



Pantalla de Módulo


EDITOR VISUAL BASIC FOR APPLICATIONS CON VENTANA INMEDIATO

Ahora vamos a configurarlo para que active el Option Explicit cada vez que creamos una rutina. Para ello posicionamos el mouse sobre el menú Herramientas-Opciones. En la primera pestaña seleccionamos la casilla de "Requerir Declaración de Variables".

Para poder crear la macro anterior tenemos que posicionar el mouse sobre el menú Insertar, luego sobre Módulo, esto nos abrirá una pantalla similar a la imagen anterior. Ahora puedes digitar el código anterior. Para poder ver los resultados utilizaremos la ventana Inmediato. Esta opción la activamos por medio del menú Ver-Ventana Inmediato, otra forma es presionando las teclas Ctrol-G, esto nos dejará la pantalla como la imagen. Presionamos la tecla F5 y en la ventana Inmediato verás el resultado de tu macro el cual tiene que ser 9. Puedes cambiar los valores para que veas diferentes resultados.

De ahora en adelante solo pondré el código, los demás pasos los haces tú. Con esto doy por terminado el Operador de Exponenciación.

OPERADOR * (MULTIPLICACIÓN)

Este operador nos permite multiplicar una cantidad por otra. La sintaxis utilizada es la siguiente: MiNumero * Numero2. Como ejemplo tenemos:

```
Sub Multiplica()
   Dim MiNumero As Integer
   Dim Numero2  As Integer
   Dim Resultado As Integer
   MiNumero = 3
   Numero2  = 2
   Resultado = MiNumero * Numero2
   Debug.Print Resultado
End Sub
```

Esto nos desplegará el resultado de 6. Recuerda que es por medio de la ventana Inmediato que verás el resultado.

Recuerda que si quieres desplegar resultados con decimales tienes que declarar las variables como Double, Single o Currency.

OPERADOR / (DIVISIÓN)

Este nos permite dividir dos números. La sintaxis utilizada es la siguiente: Numero1 / Numero2. Como ejemplo tenemos:

```
Sub Divide()
   Dim Numero1  As Double
   Dim Numero2  As Double
   Dim Resultado As Double
   Numero1  = 15
   Numero2  = 2
   Resultado = Numero1 / Numero2
   Debug.Print Resultado
End Sub
```

El resultado debe ser de 7.5 como puedes ver.

Para todos los impacientes, dentro de poco veremos resultados en las Hojas de Trabajo de Excel. Primero se debe aprender la base para después construir la edificación. Como te dije casi al principio, esto es un arte y todo artista primero pone una base y realiza su obra sobre ese fundamento.

Operador \ (Cociente)

Este operador nos devuelve el cociente sin decimales al dividir dos números. La sintaxis es la siguiente: Numero1 \ Numero2. Por ejemplo:

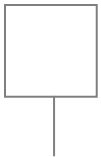
```
Sub Divide2()
   Dim Numero1  As Double
   Dim Numero2  As Double
   Dim Resultado As Double
   Numero1  = 15
   Numero2  = 2
   Resultado = Numero1 \ Numero2
   Debug.Print Resultado
End Sub
```

El resultado debe ser de 7 como puedes ver.

Operador Mod (Residuo)

Este operador nos devuelve el residuo en una división. La sintaxis sería: Numero1 Mod Numero2. Por ejemplo:

```
Sub Divide()
   Dim Numero1  As Double
   Dim Numero2  As Double
   Dim Resultado As Double
   Numero1  = 15
   Numero2  = 2
   Resultado = Numero1 Mod Numero2
   Debug.Print Resultado
End Sub
```



El resultado debe ser de 1 como puedes ver.

Operador + (Suma)

Este nos sirve para sumar dos o más números. También puede servir para unir cadenas. La sintaxis sería: Numero1 + Numero2, por ejemplo:

```
Sub Suma()
   Dim Numero1  As Double
   Dim Numero2  As Double
   Dim Resultado As Double
   Numero1  = 15
   Numero2  = 2
   Resultado = Numero1 +  Numero2
   Debug.Print Resultado   'El resultado sería 17
End Sub
```

Operador – (Resta)

Este nos sirve para restar dos o más números. También puede servir para unir cadenas. La sintaxis sería: Numero1 - Numero2, por ejemplo:

```
Sub Resta()
   Dim Numero1  As Double
   Dim Numero2  As Double
   Dim Resultado As Double
   Numero1  = 15
   Numero2  = 2
   Resultado = Numero1 +  Numero2
   Debug.Print Resultado   'El resultado sería 13
End Sub
```

Operadores de Comparación

Permiten comparar expresiones. En Visual Basic for Application podemos utilizar:

Operador	True si	False si	Null si
< (Menor que)	*expresión1* < *expresión2*	*expresión1* >= *expresión2*	*expresión1* o *expresión2* = **Null**
<= (Menor o igual que)	*expresión1* <= *expresión2*	*expresión1* > *expresión2*	*expresión1* o *expresión2* = **Null**
> (Mayor que)	*expresión1* > *expresión2*	*expresión1* <= *expresión2*	*expresión1* o *expresión2* = **Null**
>= (Mayor o igual que)	*expresión1* >= *expresión2*	*expresión1* < *expresión2*	*expresión1* o *expresión2* = **Null**
= (Igual a)	*expresión1* = *expresión2*	*expresión1* <> *expresión2*	*expresión1* o *expresión2* = **Null**
<> (Distinto de)	*expresión1* <> *expresión2*	*expresión1* = *expresión2*	*expresión1* o *expresión2* = **Null**

Estos operadores no los voy a explicar a detalle ya que son muy obvios en su función.

Operadores Lógicos

Operador And (Y lógico)

Para realizar operaciones lógicas en expresiones Booleanas. También se puede utilizar en operaciones de bits en expresiones numéricas, por ejemplo

```
Dim X As Integer = 8
Dim Y As Integer = 7
Dim Z As Integer = 5
Dim Resultado As Boolean   'Recuerde que Boolean se refiera e Falso o Verdadero
Resultado = X > Y And Y > Z   'Resultado Verdadero
Resultado = Y > X And Y > Z  'Resultado Falso
```

Operador Eqv (Equivalencia)

Para equivalencia lógicas entre dos expresiones, por ejemplo:

```
Dim A, B, C, D, MiPrueba
A = 10: B = 8: C = 6: D = Null    ' Inicializa variable.
MiPrueba = A > B Eqv B > C        ' Devuelve True.
MiPrueba = B > A Eqv B > C        ' Devuelve False.
MiPrueba = A > B Eqv B > D        ' Devuelve Null.
MiPrueba = A Eqv B                ' Devuelve -3 (comparación bit por bit).
```

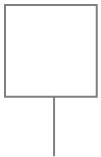
Operador Imp (Implicación)

Para una implicación lógica entre dos expresiones, por ejemplo:

```
Dim A, B, C, D, MiPrueba
A = 10: B = 8: C = 6: D = Null   ' Inicializa variables.
MiPrueba = A > B Imp B > C       ' Devuelve True.
MiPrueba = A > B Imp C > B       ' Devuelve False.
MiPrueba = B > A Imp C > B       ' Devuelve True.
MiPrueba = B > A Imp C > D       ' Devuelve True.
MiPrueba = C > D Imp B > A       ' Devuelve Null.
MiPrueba = B Imp A               ' Devuelve -1 (comparación bit por bit).
```

Bueno amigos y amigas, por ahora creo que es suficiente. En este capítulo he incluido los operadores importantes que se utilizan en la programación. En el siguiente entraremos un poco más de lleno en el arte de programar.

Aunque sé que muchos estaban deseosos de programar controles en una hoja o en un UserForm, tenemos que aprender las bases como lo dije anteriormente. No podemos correr sin saber que tenemos pies y para qué nos sirven. Y aunque nos falte mucho camino, verán que vale la pena el esfuerzo. No es promesa pero trataré de incluir algo en Visual Studio Tools For Office (VSTO).



Los que tengan curiosidad de cómo programar herramientas VSTO deberían empezar por conseguir un manual sobre Visual Studio .NET.

En la próxima entrega veremos estructuras de decisión (If…Then…Else, Select… Case), estructuras de Bucle (While…Loop, Do…While, For…Next, For Each…Next). Veremos controles Activex y su aplicación en las hojas, trabajaremos con rangos y hojas de trabajo, en fin, trataré de hacerlo más entretenido.

Capítulo 3 -Diseño de Pantallas – Ciclos – Decisiones – Programación con Rangos – Trabajo con Hojas y Más

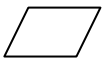
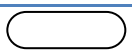
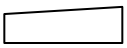
Diseño de Pantallas y Algo Más

Hola amigos y amigas, aquí estoy de nuevo con una nueva entrega del maravilloso mundo de VBA. Quiero antes que nada que aprendamos sobre Diagramas de Flujo.

Muchos se preguntarán, ¿para qué aprender eso si ya ni se usa?, puede que tengan razón, pero siempre es bueno aprender como un profesional que primero diseña su lógica y luego la plasma en el código.

Diagramas de Flujo

El uso de dichos diagramas me imagino que se remontan desde antes de 1985. En 1990 conseguí un libro llamado Programación Basic para Microcomputadoras de Luis Joyanes Aguilar, muy bueno por cierto y lastimosamente no he vuelto a ver una obra de él. La siguiente tabla muestra los símbolos y sus significados como lo mostraba el autor para esas fechas.

SÍMBOLO	FUNCIÓN O SIGNIFICADO
	**Proceso**: Indica cualquier proceso interno de la computadora, esto es, cualquier serie de transferencia de datos u operaciones aritméticas.
	**Entrada/Salida (E/S)**: Representa cualquier operación de entrada o salida, como Lectura, Escritura, etc. La operación específica se realiza indicándola por medio de un comentario dentro del símbolo.
	**Decisión**: Se utiliza para verificar una comparación lógica. Básicamente se utiliza cuando la computadora ha de realizar una pregunta.
	**Conector**: Significa un punto de referencia que indica donde debe continuar el diagrama de flujo. Se utiliza para indicar un cambio en el flujo normal de datos (transferencia o bifurcación).
	**Terminal**: Significa el comienzo o punto final de un programa.
	**Entrada desde Teclado**: Introducción de datos desde el teclado.
	**Salida hacia una Impresora**: Presentación de resultados por impresora (imprimir).
	Sentido Flujo de Datos: Conexión lógica entre unos símbolos y otros.

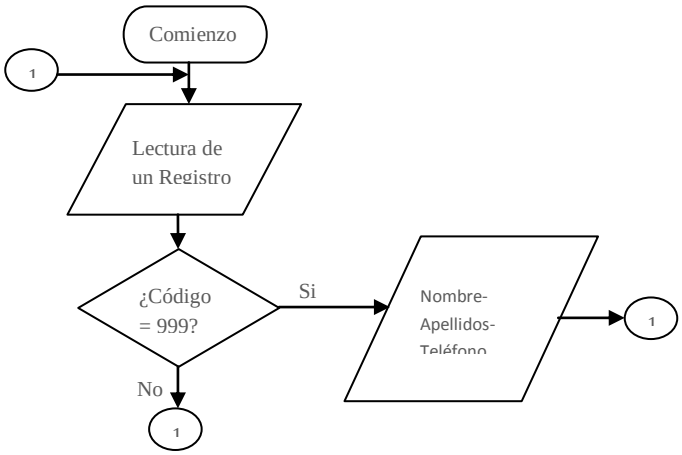
Reglas para la Realización de Diagramas de Flujo

1. Cada símbolo significa un tipo de operación: Entrada/Salida, Proceso, Decisión, Transferencia o Bifurcación.

2. Se escribe un comentario dentro de cada símbolo para indicar la función específica que se ha de ejecutar.
3. Los Diagramas de Flujo se leen de arriba hacia abajo.
4. Una secuencia de operaciones se ejecuta hasta que un símbolo terminal designa el final de la ejecución o un conector de Bifurcación transfiere el control.

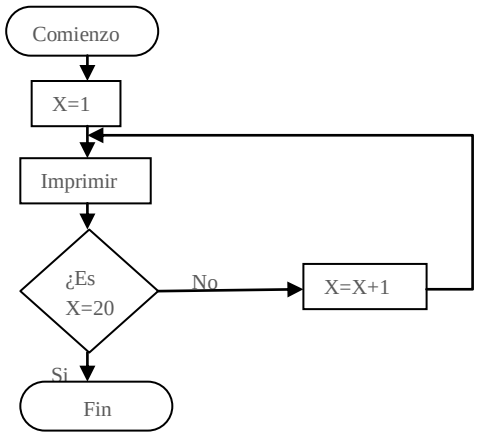
Ejemplo de uso de Diagramas de Flujo

Diagrama para imprimir el Nombre, Apellidos y número de Teléfono desde una Tabla:



Imprimir los números enteros del 1 al 20, ambos inclusive:

```
Sub ContarNum()
Dim i As Integer
For i = 1 To 20
   i = i
   Debug.Print i
Next i
End Sub
```

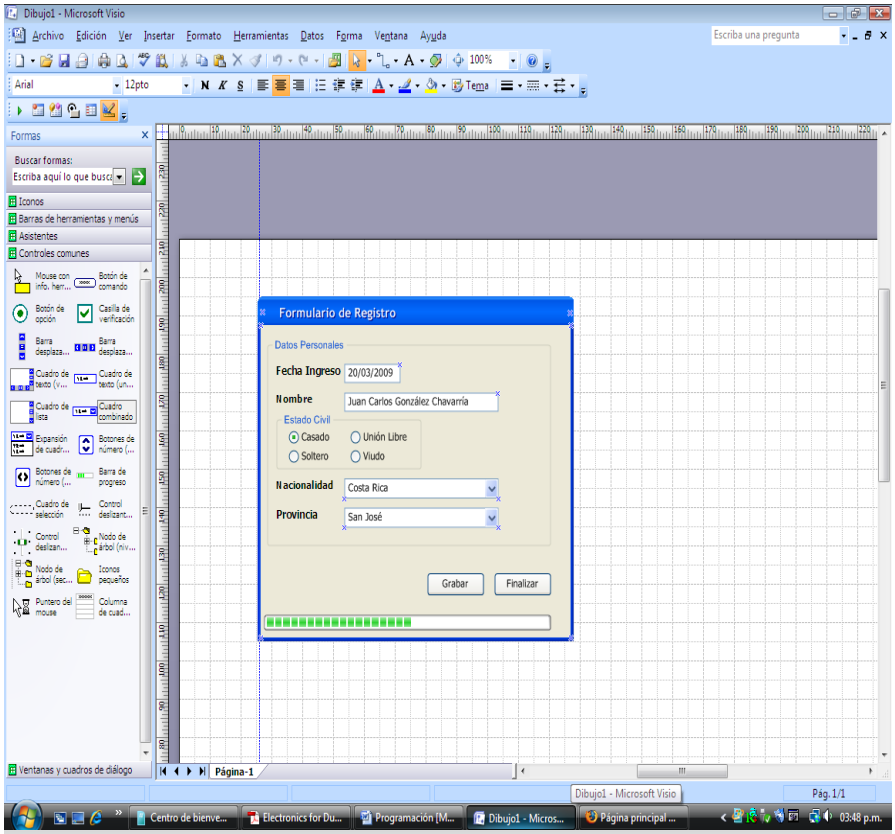


Como podemos ver se puede trabajar de una manera más ordenada y con la posibilidad de cometer menos errores.

No continuaré más con este tema ya que tendría que escribir un libro aparte.

Diseñando con el VISIO

Bueno me imagino que para muchos el nombre les parecerá un poco raro, pero sí, es un programa del conjunto de Microsoft Office (según la versión). Este nos ayuda a poder diseñar desde Diagramas de Flujo, Diseño de Pantallas, Diseños de Bases de Datos y otras cositas más muy interesantes. Sería bueno que lo consiguieran si quieren entrar en el mundo de la programación en serio.



Diseño de Pantallas con VISIO

De ahora en adelante no volveré a mencionar el nombre de ninguna compañía porque no me están patrocinando para crear este libro, solo voy a mencionar el nombre del programa y si tienen curiosidad a quien pertenece, lo buscan en Internet u otro lado.

Este es un ejemplo de lo que se puede diseñar con este programa, que para mi criterio, es muy bueno.

En caso de no contar con dicha herramienta, les aconsejo primero diseñar sus pantallas a mano, en una simple hoja. De esta manera se mostrará al usuario un trabajo más profesional.

No enseño cómo utilizarlo ya que se sale del objetivo del presente libro.

Diseñando las salidas en Hojas

Siempre he dicho que una buena presentación en la hoja, la hará más agradable para trabajar. Me he encontrado con usuarios que por no tomarse su tiempo para darle un buen formato de salida a sus Hojas la presentación es tan desagradable que ni siquiera dan ganas de trabajar con ellas. Algunos por falta de experiencia y otros por dejados y perezosos.

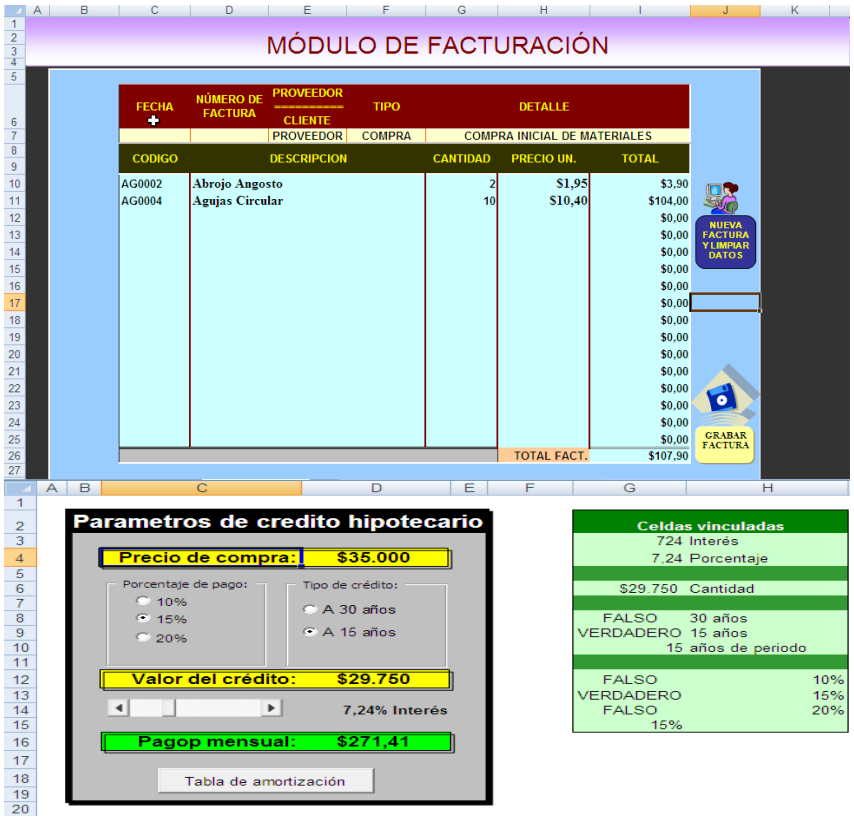
Creo que si trabajamos para una empresa y mostramos un trabajo bien detallado, siempre tendremos trabajo. Si somos independientes, de igual modo. Si

queremos que nuestros clientes nos sigan contratando, debemos tomarnos el tiempo adecuado para diseñar una salida agradable y entendible.

Las presentaciones en las hojas no deben tener demasiados elementos ni muy pocos. Deben tener los necesarios para trabajar con ella. Deben estar en un orden que no permita la confusión y haga perder tiempo buscando un control x.

Código	Descripción	Crédito	No_Factura	Fecha	Monto	Abono1	Abono2	Saldo	Fecha_Vence	Días_Vencidos	No_Vencida	De 1-30	De 31-60	De 61-90	Más de 90
2502	Materiales La Niña	30	6525	01-oct	45.000,00	5.000,00	3.000,00	37.000,00	31-oct	140	-	-	-	-	37.000,00
2505	Confites El Dulce	15	8525	25-abr	85.000,00	-	-	85.000,00	10-may	314	-	-	-	-	85.000,00
2501	Distribuidora Bele	15	8525	15-sep	13.000,00	3.500,00	-	9.500,00	30-sep	171	-	-	-	-	9.500,00
2503	Maderas La Tabla	0	4585	10-sep	14.000,00	8.000,00	3.000,00	3.000,00	10-sep	191	-	-	-	-	3.000,00
2504	Canastas La Única	30	1685	10-jul	95.000,00	12.000,00	8.000,00	75.000,00	09-ago	223	-	-	-	-	75.000,00
2506	Magali Ltda.	30	8525	10-oct	55.000,00	18.000,00	-	37.000,00	09-nov	131	-	-	-	-	37.000,00
2507	Medallas Mías	30	8684	10-sep	125.000,00	50.000,00	-	75.000,00	10-oct	161	-	-	-	-	75.000,00
2501	Distribuidora Bele	15	9525	02-ago	18.000,00	3.000,00	-	15.000,00	17-ago	215	-	-	-	-	15.000,00
2502	Materiales La Niña	30	1068	03-abr	185.000,00	12.500,00	14.000,00	158.500,00	03-may	321	-	-	-	-	158.500,00

Este es un ejemplo de una hoja bien diseñada y entendible para cualquier persona. Se puede apreciar perfectamente que es un historial de facturas en la cual se muestra sus vencimientos, abonos, proveedores y otros detalles.



Si queremos insertar objetos, estos deben ir colocados de manera que no estorben con el entorno de donde se van a digitar datos.

Como se puede apreciar en la imagen, los objetos para una nueva factura y para grabarlas se encuentran a un lado del ambiente de trabajo. En lo personal prefiero utilizar otros controles pero todo es cuestión de gustos.

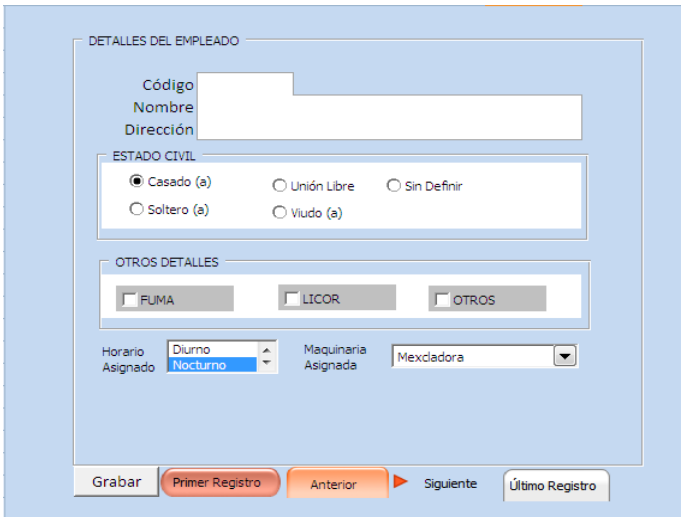
En el libro que me compré viene un ejemplo que demuestra un buen formato de salida y distribución de controles en una hoja.

Como habrán observado, es muy agradable trabajar con algo que está en lo máximo posible bien estructurado. Nos dará la sensación de estar trabajando con un sistema sofisticado.

Controles de Formulario

Con este tipo de controles podemos dar un aspecto a nuestra hoja como lo tiene la imagen anterior. Podemos observar que contiene Botones de Opción, Cuadro de Grupo y Barra de Desplazamiento. Excel tiene los siguientes Controles de Formulario:

Nombre	Característica
Botón	Lo puedo asignar a una macro. Este control no se puede asignar a una celda específica.
Cuadro Combinado	Me permite asignarle un rango de celdas o un nombre de rango y mostrar el contenido del mismo.
Casilla de Verificación	Me permite asignarlo a una celda. Al estar activado en la celda se mostrará Verdadero, caso contrario mostrará Falso.
Control de Número	Al asignarlo a una celda me incrementa o disminuye una numeración en la cantidad que se le asigne.
Cuadro de Lista	Me permite asignarle un rango de celdas o un nombre de rango y mostrar el contenido del mismo.
Botón de Opción	Nos permite asignarlo a una celda. Al incluir varios el valor de la celda cambiará según la opción deseada.
Cuadro de Grupo	Nos sirve para agrupar objetos o para dar una mejor presentación.
Etiqueta	Para mostrar una leyenda. También se puede asignar a una celda.
Barra de Desplazamiento	Al igual que el Control de Número nos sirve para aumentar o disminuir en un número deseado.

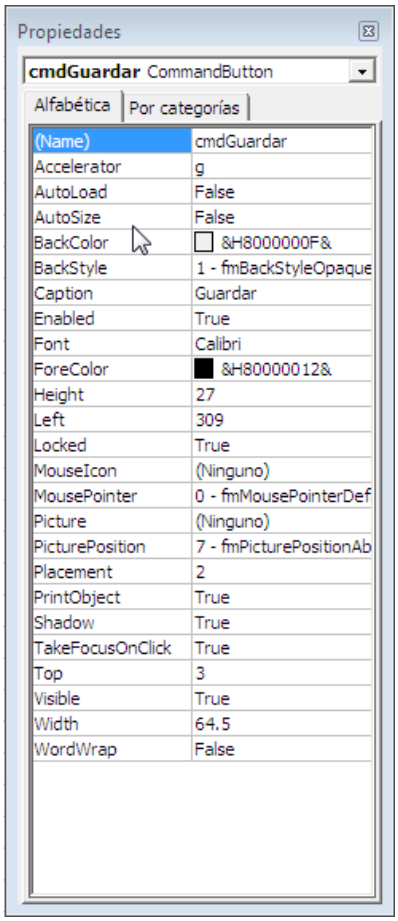


Como se deduce, estos controles son muy útiles en ciertas ocasiones según sea el trabajo a realizar. Se pueden utilizar en conjunto con una macro. Por ejemplo si queremos grabar una factura con varias opciones estos controles nos pueden ayudar.

También se pueden utilizar las famosas AutoFormas, Imágenes Prediseñadas o Imágines de Archivo. Algo muy curioso es que me he dado cuenta que muchos no saben que el Office contiene una cantidad de botones, punto u otras imágenes que nos ayudarían a darle mejor presentación a nuestro trabajo. D:\Archivos de programa\Microsoft Office1\Templates\BUTTONS. En mi caso las imágenes se encuentran en la dirección anterior.

En esta imagen podemos observar todos estos controles juntos. Por supuesto que no deberían quedar botones mezclados pero los puse así solo para ilustración.

Controles Activex



Este tipo de controles es un poco más poderoso. La ventaja principal es que los podemos manipular por medio de programación, a diferencia de los anteriores que los asignamos a una celda. Este tipo de controles tiene sus ventajas y desventajas.

Posee los mismos controles que Controles de Formularios listados en la tabla anterior pero con la diferencia que a estos les podemos asignar un código o macro. Posee propiedades que podemos manipular casi a nuestro antojo.

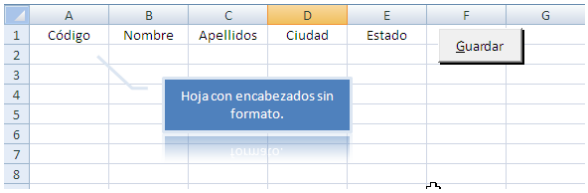
En este punto quiero aclarar que no estamos trabajando con Visual Basic 6.0, Visual Basic Editor o Visual Basic.NET, por lo que no se podrán explotar ciertas características. No es que del todo no se pueda, es que habría que programar mucho más código.

Para accesar estos controles vamos al menú Ver-Controles Activex. En la versión 2007 presionamos en la ficha Programador.

With … End With

Si son observadores me imagino que notaron esta construcción la cual nos permite realizar múltiples operaciones con un objeto. Esta construcción nos permite ahorrar mucho tiempo en programación ya que si hiciéramos todos estos cambios por medio de una macro el código sería muy extenso y menos entendible. En cambio de esta manera puedes ver que se empieza a trabajar con el rango A1:E1, y decimos Con el rangox haga x cosa, fin de Con (With Range("A1:E1"), demás instrucciones, End With).

Command Button



Como su nombre lo dice es un botón de comando. En las propiedades del mismo podemos ver la que dice "Aceelarator" y "Caption". Con "Caption" ponemos el texto que deseamos aparezca en nuestro botón. Con "Accelerator", escogemos la letra que combinada con la tecla Alt nos activará el control, algo similar a presionar click sobre él. En la versión 2007 cuando presionamos la ficha Programador se ve claramente.

Para acceder a las propiedades de cada control vamos al menú Ver-Controles Activex. Esto hará que aparezca una barra con todos los controles Activex disponibles. Dentro de la misma verán un ícono que parece una escuadra con un lápiz. Al presionar sobre este, nos mostrará la ventana de propiedades similar a la imagen.

Para tener una mejor idea de su funcionamiento empezaremos por trasladar datos de una hoja a otra.

En A1 pondremos "Código", B1 "Nombre", C1 "Apellidos", D1 "Ciudad" y en E1 "Estado". Todo esto sin ningún tipo de formato quedando como la imagen.

Ahora insertamos un CommandButton y lo posicionamos a la par de la celda E1. En sus propiedades, en “Accelerator” digitamos una “G” y en “Caption” digitamos “Guardar”. En la parte de “Name” ponemos “cmdGuardar”.

Para hacerlo más interesante digitamos la información de A2 hasta E2. Presionamos sobre el ícono de propiedades y damos doble click sobre nuestro CommandButton. Esto nos abrirá el editor de VBA, o sea, el VBE que nos desplegará una pantalla que dice Private Sub cmdGuardar_Click() y que finaliza con End Sub. Lo cual nos indica que nuestra rutina empezará con Private Sub y finalizará en End Sub.

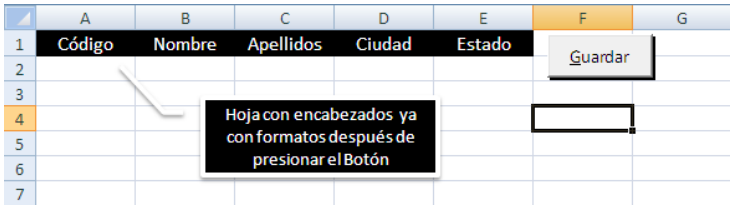
Antes de continuar permítanme contarles que los procedimientos y variables pueden alcanzar dos tipos:

a. Private, el cual se ejecutará dentro del mismo entorno. Aunque puede ser llamado desde otro lugar, sus variables solo son reconocidas en su contorno.

b. Public, el cual se ejecutará desde cualquier entorno.

Ahora digitamos el siguiente código:

```
Private Sub cmdGuardar_Click()
Range("A1:E1").Interior.ColorIndex = 1
With Range("A1:E1")        ‘Con el Rango
    .Font.Size = 12            ‘Cambio el tamaño del texto
    .Interior.ColorIndex = 1   ‘Color de fondo negro
    .Font.ColorIndex = 2       ‘Texto de color blanco
  .HorizontalAlignment = xlCenter
  ‘Centrar horizontalmente
    .VerticalAlignment = xlBottom
      ‘Centrar verticalmente
End With
End Sub
```



Si nos pasamos a nuestra hoja y presionamos el botón verán su código actuar. Por ahora no pasa la información de una hoja a otra pero pueden ver que el formato del rango A1:E1 ha cambiado. Verán que el fondo del rango es de color negro y el texto es blanco. El código se encarga de centrar los textos tanto de forma horizontal como vertical. Además, es de destacar que la presentación actual es más agradable que la que tenía antes de presionar el botón. Si queremos ocultar las líneas de división solo agregamos después del End With ActiveWindow.DisplayGridlines = False y así tendremos una mejor presentación.

Muchos empezarán a decir, pero por qué enseña esta construcción sin antes explicar cómo se pueden manipular los rangos, hojas, libros u otros. Bueno porque a mí me interesa que aprendan a programar sin entrar en muchos detalles por el momento, los cuales se pueden ir aprendiendo durante el camino.

Ahora borremos el código anterior y pongamos el siguiente código:

```
Private Sub cmdGuardar_Click()
        ActiveWindow.DisplayGridlines = False
        Application.ScreenUpdating = False
        Dim SigFila As Long
        Range("A2:E2").Select
        Selection.Copy
        Hoja2.Select
        SigFila = Application.WorksheetFunction.CountA(Range("A:A")) + 1
        ActiveSheet.Paste
        ActiveCell.Offset(1, 0).Select
        Hoja1.Select
        Application.CutCopyMode = False
        Range(“A2:E2”) = “”
        Range("A2").Select
        Application.ScreenUpdating = True
End Sub
```

Con este código sí se pasa la información de la Hoja1 a la Hoja2 con solo presionar el botón que insertamos en la hoja.

EXPLICACIÓN DEL CÓDIGO

I. ActiveWindow.DisplayGridLines = False: Con esta instrucción le decimos a Excel que quite las líneas de división que se muestran en la hoja, esas que nos muestran las celdas y columnas.

II. Application.ScreenUpdating = False: Con esta instrucción desactivamos los parpadeos de pantallas al hacer alguna operación.

III. Dim SigFila As Long: Aquí declaramos una variable de tipo Long. Esto porque si se acuerdan de la Tabla de Tipos de Variables se darán cuenta que la tipo Integer solo soporta hasta 32,767 y la cantidad de filas de Excel 2003 son de 65,532 y en la versión 2007 es de 1, 000,000, bueno un poquito más.

IV. Range(“A2:E2”).Select: Aquí seleccionamos el rango que deseamos copiar hacia la otra hoja.

V. Selection.Copy: Con esta damos la instrucción de Copiar el rango.

VI. Hoja2.Select: Seleccionamos la Hoja2. También lo podemos hacer con Sheets("Hoja2").Select pero se recomienda con la otra instrucción porque si el usuario cambia el nombre de la hoja entonces nos generaría un error al tratar de seleccionar la hoja.

VII. SigFila = Application.WorksheetFunction.CountA(Range("A:A")) + 1: Esto le dice al programa que la siguiente fila vacía será igual que el resultado de aplicar la función =ContarA() de la columna A y le sume 1. Esto porque la función =ContarA () cuenta la cantidad de filas no vacías.

VIII. ActiveSheet.Paste: Con esta le indicamos al programa que copie el rango que tiene en memoria, el cual se seleccionó anteriormente.

IX. ActiveCell.Offset(1, 0).Select: Aquí le decimos que baje una celda.

X. Hoja1.Select: Seleccionamos nuevamente la hoja1 con la que estabamos trabajando y en la cual introducimos la información que queremos pasar.

XI. Application.CutCopyMode = False: Cada vez que hacemos una copia de una o varias celdas podemos observar que quedan unas rayitas alrededor del rango seleccionado. Pues con esta instrucción cancelamos esas rayitas.

XII. Range("A2:E2") = "": Limpiamos el rango para que quede listo nuevamente y volver a introducir datos en él.

XIII. Range("A2").Select: Nos posicionamos en la celda A2 para volver a introducir datos

XIV. Application.ScreenUpdating = True: Volvemos a activar el parpadeo de pantalla.

Como podemos notar, es un código bastante útil para copiar de una hoja a otra con solo presionar un botoncito de la hoja de trabajo, lo cual nos ahorra bastante tiempo y no tener que estar copiando, buscando la última fila, pegando la información y volver a donde estábamos.

Sé que muchos quieren ya saber cómo hacer si tienen una factura con varias líneas y pasarlas a una hoja de registro, pero aún estamos empezando a gatear, vamos con calma porque aún no hemos aprendido sobre Bases de Datos.

Para concluir con esta parte podrás darte cuenta de la importancia de mostrar una pantalla agradable a la vista, con un uso sencillo y entendible para cualquier usuario y que no se le canse la vista con colores muy llamativos. Si sigues esta secuencia diseñar tus archivos de una forma más profesional.

CICLOS (ESTRUCTURAS DE BUCLE)

DO WHILE … LOOP

Este tipo de construcciones puede tener dos formas: Do While … Loop o Do … Loop While

Para que nos entendamos mejor veamos ejemplos de las dos construcciones anteiores.

1) Do While … Loop: Vamos a crear una macro e introducimos este código.

```
Sub ejemploDoWhileLoop()
Dim Contador As Integer
Contador = 0
Do While Contador <= 5
   Contador = Contador + 1
   Range("H1").Value = Contador
   MsgBox "El valor actual es: " & Contador
Loop
End Sub
```

Aquí creamos un ciclo que vaya de 1 hasta 5, el resultado se pondrá en la celda H1 y nos desplegará un mensaje con el valor actual de la variable.

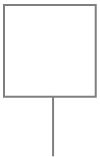
Debemos prestar atención ya que el código anterior tiene un pequeño error. No llegará solo hasta 5, sino que nos desplegará hasta el 6. Esto se debe a que el ciclo Do While se ejecuta una vez más después de haber llegado a 5 para hacer otra evaluación y ver si termina.

2) Do … Loop While: Creamos otra macro con el siguiente código:

```
Sub DoLoopWhile()
Dim Contador As Integer
Contador = 0
Do
   Contador = Contador + 1
   Range("H1").Value = Contador
   MsgBox "El valor actual es: " & Contador
Loop While Contador <= 5

End Sub
```

Ese código hace exactamente lo mismo que el anterior solo que la evaluación se hace al final y no al principio. Tiene el mismo problema que el anterior.

Veamos otro ejemplo, vamos a suponer que queremos verificar si el usuario ha introducido la clave correcta. La clave la vamos a digitar en la celda M1 y en base a lo que esté escrito dejará al usuario pasar al sistema. El código es el siguiente:



```
Sub VerificaClave()
Dim Contador As Integer
Dim Pase As String
Do
   Pase = InputBox("Introduzca su contraseña: ")
   If Pase = Range("M1").Value Then
      MsgBox "Bienvenido al sistema."
      Exit Sub
   Else
      MsgBox "Lo siento, contraseña incorrecta. Intente nuevamente."
      Contador = Contador + 1
   End If

Loop While Contador <= 5

End Sub
```

Pondremos cualquier palabra en la celda M1, ejecutamos el código y daremos la opción de poder intentar hasta 5 veces.

Por supuesto que esto se puede depurar mejor, podemos poner un UserForm con los nombres de usuarios y evaluar sus contraseñas para dejarlos o no accesar nuestro archivo.

CICLO FOR … NEXT

Nos permite repetir una serie de instrucciones un determinado número de veces. Veamos un ejemplo sencillo de cómo utilizar este ciclo:

```
Sub ForNext()
Dim i As Integer
For i = 1 To 5
   Range("H10").Value = i
   MsgBox "El valor con For...Next es: " & i
Next i
End Sub
```

En el ejemplo anterior nos posicionaremos en la celda H10 y se nos mostrará un mensaje con el valor que contenga i al momento de la ejecución.

Veamos ahora otra instrucción con el For, el Step. Esta nos permite saltar la cantidad de veces que le indicamos. Para que me entiendan veamos un ejemplo:

```
Sub ForNextStep()
Dim i As Integer
For i = 0 To 20 Step 2
```

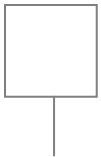
```
   Range("M10").Value = i
   MsgBox "El valor con For...Next es: " & i
Next i
End Sub
```

En el ejemplo anterior le decimos al programa que nos despliegue el resultado cada 2 números, o sea, pondrá el 0 2 4 6… y así sucesivamente.

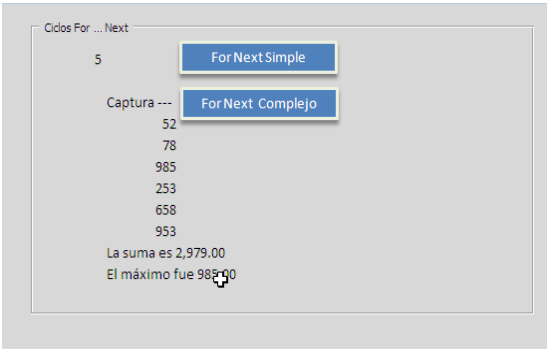
Como vemos el ejemplo es muy sencillo. Ahora vamos a complicar un poquito más las cosas y aprenderemos nuevas instrucciones. He adaptado un ejemplo de mi maestro virtual en Visual Basic 6.0 José Felipe Ramírez R., y aunque el código es de Visual Basic 6.0, podemos ver que es casi exactamente lo mismo, porque aunque no lo he mencionado, **casi todo** lo que se puede hacer en Visual Basic, se puede hacer en VBA.

Con el siguiente código aprenderemos cosas nuevas y luego comentaré el mismo:

```
Sub ForNext2()
Dim intVentas(1 To 6) As Integer
Dim strMensaje As String * 25
Dim intContador As Integer
Dim intSuma As Integer
Dim intMaximo As Integer

'Preguntar Datos a procesar.
For intContador = 1 To 6
   strMensaje = "Captura del número " & Format(intContador, "##")
   intVentas(intContador) = InputBox("Dame un número ENTERO ", strMensaje)
Next intContador

'Segundo proceso: desplegar lo capturado
Range("I12").Value = "Captura ---"
Range("I12").Select
For intContador = 1 To 6
   ActiveCell.Offset(1, 0).Select
   ActiveCell.Value = Format(intVentas(intContador), "###,###.00")
Next intContador

'Tercer proceso: Sumar las cantidades y determinar el máximo.
For intContador = 1 To 6
   intSuma = intSuma + CInt(intVentas(intContador))
   If intVentas(intContador) > intMaximo Then
      intMaximo = intVentas(intContador)
   End If
Next intContador
```



```
Range("I19").Value = "La suma es " & Format(intSuma, "###,###.00")
Range("I20").Value = "El máximo fue " & Format(intMaximo, "###,###.00")
```




```
MsgBox "La suma es: " & Format(intSuma, "###,###.00") & Chr(13) & _
   Chr(13) & "El máximo fue: " & Format(intMaximo, "###,###.00")
End Sub
```

Bueno que creyeron que iba a ser fácil esto de programar, pues están muy equivocados. Entre más cosas se quieran hacer, más código hay que digitar. El código anterior nos pide 6 cantidades y las guarda en una variable tipo Array o Matriz (luego explicaré qué es). Luego las suma y por último nos las coloca en el rango respectivo, nos da la sumatoria de todos y para rematar nos dice cual es el número máximo que se digitó quedando como la imagen. Claro está que yo le di un formato de salida más presentable para que sea más agradable a la vista y entendible para cualquier persona.

Explicación del Código

Para que no se me pierdan en la programación voy a explicar el código anterior y las nuevas instrucciones ingresadas en él.

Antes quisiera decirles que si queremos poner un comentario en cualquier parte de nuestro código solo tenemos que digitar una comilla (‘). También podemos utilizar el REM pero ya casi nadie lo usa y por lo general lo he visto en códigos de las versiones Office 95 y 97.

Explicación del Código

Instrucción	Explicación
Dim intVentas(1 To 6) As Integer	Declaración de una variable tipo Array o Matriz y que va a ser Integra (Integer). Consta de 6 elementos (1 to 6).
Dim strMensaje As String * 25	Declaración de una variable tipo cadena pero con la diferencia de que le damos un límite de 25. Solo 25 caracteres podrán introducirse.
Dim intContador As Integer	Declaración de variable que nos servirá para el contador.
Dim intSuma As Integer	Declaración de una variable que nos permitirá acumular la suma de intVentas.
Dim intMaximo As Integer	Variable que nos permitirá guardar el valor máximo entre los digitados.
'Preguntar Datos a procesar.	Como vemos tiene una comilla al inicio, esto nos indica que es un comentario. Todo comentario inicia de esa forma o con REM.
For intContador = 1 To 6	Le decimos que el contador irá de 1 hasta 6 (Para intContador = desde 1 hasta 6)

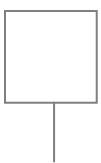
strMensaje = "Captura del número " & Format(intContador, "##")	Le asignamos una leyenda a la variable strMensaje y a la vez con el carácter "&" unimos la variable intContador. A esta variable que unimos le damos un formato de número con el Format (más adelante lo veremos con más detalle).
intVentas(intContador)= InputBox("Dame un número ENTERO ", strMensaje)	A la varible intVentas que contiene el índice de intContador se le asigna una función InputBox que nos pedirá un número entero. El InputBox() lo veremos más adelante detalladamente.
Next intContador	Le indicamos al ciclo que continúe con el siguiente intContador.
Range("I12").Value="Captura---"	A la celda I12 le asignamos la leyenda "Captura ---"
Range("I12").Select	Nos posicionamos en la celda I12.
For intContador = 1 To 6	Volvemos a asignar el contador.
ActiveCell.Offset(1, 0).Select	Bajamos una celda y nos posicionamos en I13.
ActiveCell.Value=Format(intVentas(intContador), "###,###.00")	Asignamos a la celda actual el valor que contenga intVentas con su índice respectivo (intContador). A la vez le damos un formato numérico.
Next intContador	Indicar que continúe con el siguiente intContador o índice.
For intContador = 1 To 6	Volver a inicializar el intContador.
intSuma=intSuma+ CInt(intVentas(intContador))	Asignamos a la variable intSuma la sumatoria de intVentas y su respectivo índice con intContador. Aquí notemos que la convierto en Entera con Cint().
If intVentas(intContador) > intMaximo Then	Le decimos que si intVentas es mayor que intMaximo entonces ejecute el resto.
intMaximo = intVentas(intContador)	Asignamos a intMaximo el valor de intVentas con el índice respectivo.
End If	Finalizamos el If.
Next intContador	Indicamos que continúe con el siguiente intContador
Range("I19").Value = "La suma es " & Format(intSuma, "###,###.00")	Asignamos a la celda I19 una leyenda y a la vez la unimos con intSuma con su respectivo formato. Esto nos desplegará la suma total.
Range("I20").Value = "El máximo fue " & Format(intMaximo, "###,###.00")	Asignamos a la celda I20 una leyenda y a la vez la unimos con intMaximo y su respectivo formato. Esto nos desplegará el valor máximo digitado.
MsgBox "La suma es: " & Format(intSuma, "###,###.00") & Chr(13) & _	Esta instrucción no es necesaria pero la puse para que vean el resultado de otra forma. Por medio de un Cuadro de Mensaje mostramos una leyenda, la unimos con intSuma con su respectivo formato, luego mandamos un Enter con Chr(13) y por último con el "_" le indicamos que la instrucción continúa en la fila siguiente.
Chr(13) & "El máximo fue: " & Format(intMaximo, "###,###.00")	Ingresamos otro Enter con Chr(13) y ponemos otra leyenda que la unimos con intMaximo y su respectivo formato.

Bueno espero haberme explicado bien, de no ser así, no se preocupen que con el pasar del tiempo y los demás ejemplos aprenderemos a utilizar todas estas instrucciones.

ESTRUCTURAS DE DECISIÓN

IF… THEN

Nos permite construir un árbol de toma de decisiones, ya que es la única manera de especificar la secuencia lógica de un proceso. Se ejecutarán las instrucciones



dependiendo de un valor True o False (Verdadero o Falso), correspondiente a una expresión lógica. Su sintaxis es:

```
If condición then
        MiCodigo
End If
```

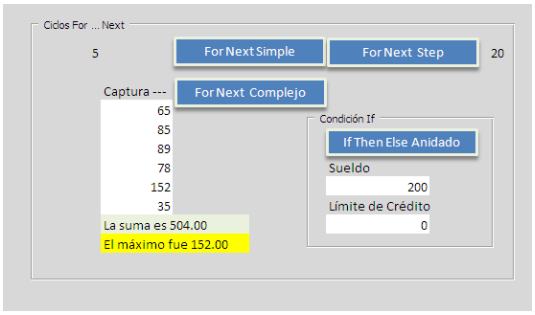
Este se puede utilizar también con Else, que nos indica que sino se cumple una condiciones, entonces ejecute otra. La sintaxis sería:

```
If Condicion Then
        MiCodigo
Else
        MiOtroCodigo
End If
```

Otra de sus variantes es utilizar el ElseIf que como Else pero habilitando otro bloque If. Obliga a la definición de una condición nueva a evaluar; el Else solo implica solamente que la última condición evaluada fue Falsa.

Veamos unos ejemplos para comprender mejor este tipo de decisiones.

```
If Sueldo > 4000 Then
        Limite_Credito = 3000
Else
        Limite_Credito = 1000
End If
```



Como vemos si el Sueldo es mayor de 4,000, el límite de crédito será 4,000, sino será de 1,000.

Podemos anidar todos los If Then Else End If que necesitemos, siempre y cuando sea de una forma lógica y sin abusar de ellos para no perdernos en la programación, veamos un ejemplo de mi maestro virtual en Visual Basic 6.0 José Felipe Ramírez R.:

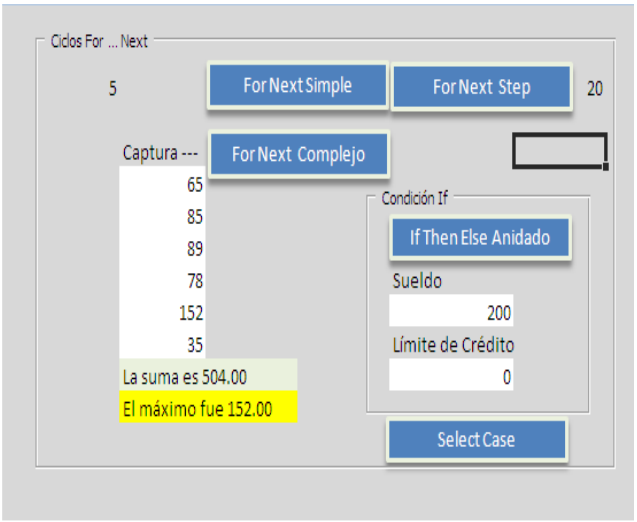
```
Sub IfThen()
Dim Sueldo As Integer
Dim Limite_Credito As Integer
Range("L17").Select
Sueldo = InputBox("Por favor ingrese el monto del sueldo.", "Sueldos")
Range("L17").Value = Sueldo
If Sueldo > 4500 Then
   Limite_Credito = 3000
Else
   If Sueldo > 3500 Then
```

```
        Limite_Credito = 1500
    Else
        If Sueldo > 1500 Then
            Limite_Credito = 500
        Else
            Limite_Credito = 0
        End If
    End If
End If
Range("L19").Value = Limite_Credito
End Sub
```

Notemos la importancia de tabular nuestro código ya que la diferencia de distancia con respecto a la primera posición de la izquierda permite visualizar de manera fácil que todos los bloques If se han cerrado de manera adecuada con End If.

SELECT CASE… END SELECT

Esta construcción es muy útil para elegir entre tres o más opciones. Al igual que el If Then, cuenta con un Else; y su sintaxis sería: Select Case… Case Else… End Select

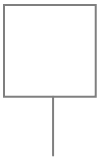


Cuando se requiere realizar una cantidad de comparaciones sobre una misma expresión, se recomienda utilizar la instrucción Select Case.

Como el mismo caso del If, se pueden ir anidando tantos Select Case como necesitemos y agregar tantos Case a la medida que se necesiten. Veamos un pequeño ejemplo:

```
Sub SelectCase()
Dim i As Integer
i = InputBox("Digite un número entre 1 y 10.", "Validar números")
Select Case i
    Case 2, 4, 6, 8, 10
        MsgBox "El número digitado es par."
    Case 1, 3, 5, 7, 9
        MsgBox "El número es impar."
    Case Else
        MsgBox "El número digitado no está en el rango de intervalo."
End Select
End Sub
```

Noten que hasta ahora he insertado AutoFormas y les he asignado las macros que he ido creando tal como se ve en la imagen. Si mal no recuerdo, anteriormente mencioné que una macro se ejecuta presionando las teclas Alt-F8 y se escoge la macro a ejecutar.



Ustedes no tendrán el mismo formato que yo le he dado, pero pueden utilizar su imaginación para crear uno nuevo o modificar el presente.

Hasta ahora podrán notar que programar no es tan fácil ni tan difícil, es solo crear una estructura lógica para obtener el resultado deseado. Tampoco hay que caer en el error de que aprenderemos a programar en 6 meses o menos, como lo mencioné al principio. Hay que conocer no solo del uso de Visual Basic For Applications, también es bueno aprender sobre Bases de Datos, Redes y todo lo que sea posible.

TRABAJO CON HOJAS Y MÁS

Aquí iniciaremos una nueva etapa en la programación ya que hasta ahora hemos trabajado en base a una solicitud por medio del InputBox. Como vimos en el ejemplo de pasar datos a otra hoja, se puede seleccionar un rango, copiarlo, pegarlo en otra hoja y luego limpiarlo para volver a digitar nuevos datos.

Antes de entrar de lleno al tema quisiera explicar los diferentes tipos de eventos.

EVENTOS

Existen diferentes tipos de eventos dependiendo de lo que se esté utilizando, ya sea una hoja, un rango, una celda o el libro de trabajo, una Etiqueta, un ComboBox, un TextBox, etc.

Algunos pensarán que mi secuencia está errada por no mostrar otras funciones de programación como el InputBox, MsgBox, MessageBox y otros, pero lo hago así ya que cuando yo aprendía me era muy difícil entender muchas cosas que se me aclararon cuando llegué a estas secciones.

Dichos eventos tienen la particularidad de poder ser controlados por medio de programación (VBA). Por ejemplo, si presionamos sobre una AutoForma o un CommandButton se ejecuta un evento Click de los mismos, los cuales ejecutarán las instrucciones contenidas en ellos.

Las clases de eventos que Excel puede reconocer según lo lista el señor John Walkenbach en su libro “Excel 2002 – Programación con VBA” son:

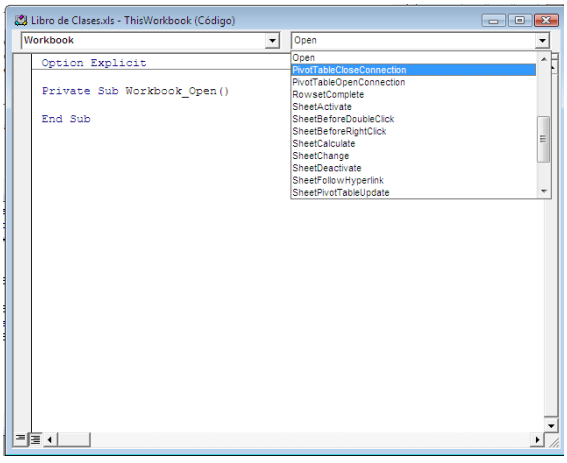
- Un libro de trabajo es abierto o cerrado.
- Se activa una ventana.
- Se activa o desactiva una hoja de trabajo.
- Se introducen datos en una celda o se edita una celda.
- Se guarda un libro de trabajo.

- Se calcula una hoja de trabajo.
- Se hace click sobre un objeto.
- Se actualizan los datos de un gráfico.
- Se presiona una tecla o una combinación de teclas en particular.
- Se hace doble click sobre una celda.
- Tiene lugar una hora en particular del día.

TIPO DE EVENTOS QUE EXCEL PUEDE DIRIGIR

Eventos	*Descripción*
Libro de Trabajo	***Tienen lugar para un libro en particular. Open, BeforeSave, NewSheet.***
Hoja de Trabajo	***Para una hoja en particular. Change, SelectionChange, Calculate.***
Gráfico	***Para un gráfico en particular. Select, SeriesChange.***
Aplicación	***Para una aplicación. NewWorkBook, WorkBeforeClose, SheetChange.***
UserForm	***Para un Formulario en particular. Initialize, Click***
No asociados con objetos	***Consta de dos útiles eventos de nivel de aplicación. On y OnKey.***

Esto es solo un resumen de varios eventos, conforme sigamos la secuencia aprenderemos algunos otros. Estos deben ser colocados en el lugar adecuado para que se pueda ejecutar el código. Por ejemplo, muchas veces se quiere que cuando se seleccione una hoja x, aparezca un formulario, sin embargo los usuarios nuevos colocan este código en un Módulo en lugar del evento de hoja Activate. Por detalles como estos es que explico primero los eventos para continuar con lo demás.



EVENTOS DE LIBRO DE TRABAJO

Para conocer los eventos que se pueden manipular a nivel de libro, hoja u otros solo tenemos que ingresar al VBE (Visual Basic Editor) con Alt-F11, presionamos doble click sobre el libro u hoja y en la parte derecha nos mostrará todos los eventos contenidos en el objeto tal como se aprecia en la imagen.

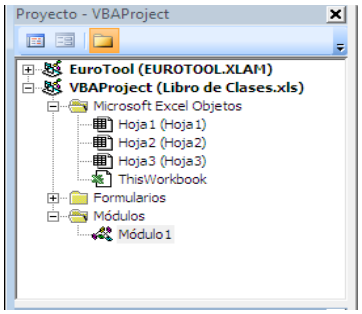
Pueden apreciar que son bastante los eventos que se pueden llevar a cabo a nivel de Libro de Trabajo.

Para que entendamos un poco más, la siguiente tabla del señor John Walkenback nos muestra en detalle cada uno de ellos con su respectiva explicación

Evento de Libro de Trabajo	Descripción
Activate	Se activa un Libro de Trabajo.
AddinInstall	Instalar complementos.
AddinUninstall	Desinstalar complementos.
BeforeClose	Está a punto de cerrar un libro de Trabajo.
BeforePrint	Está a punto de imprimir un libro de Trabajo.
BeforeSave	Está a punto de salvar un libro de Trabajo.
Deactivate	Se desactiva un libro de Trabajo.
NewSheet	Se inserta una nueva hoja de Trabajo.
Open	Se abre un libro de Trabajo
PivotTableCloseConnection *	Una fuente de datos externa conectada para una tabla dinámica se cierra.
PivotTableOpenConnection *	Una fuente de datos externa conectada para una tabla dinámica se abre.
SheetActivate	Se activa cualquier hoja de trabajo.
SheetBeforeDoubleClick	Se hace click sobre cualquier hoja de trabajo. Tiene lugar antes de la acción por defecto del doble click.
SheetBeforeRightClick	Se hace click con el botón derecho del mouse sobre la hoja. Ocurre antes de la acción por defecto del doble click.
SheetCalculate	Se calcula cualquier hoja (o se vuelve a calcular).
SheetChange	Se cambia a cualquier hoja de trabajo por el usuario o por un enlace.
SheetDeactivate	Se desactiva cualquier hoja de trabajo.
SheetFollowHyperlink	Se hace click en un hipervínculo de una hoja de trabajo.
SheetPivotTableUpdate *	Se actualiza una tabla dinámica con nuevos datos.
SheetSelectionChange	Se cambia la selección de cualquier hoja de trabajo.
WindowActivate	Se activa cualquier ventana perteneciente al libro de trabajo.
WindowDeactivate	Se desactiva cualquier ventana perteneciente al libro de trabajo.
WindowResize	Se cambia el tamaño de cualquier ventana del libro de trabajo.

- * Eventos que solo tienen lugar en Excel 2002, no se soportan en versiones anteriores.

Ahora veamos algunos ejemplos para comprender mejor algunos de estos eventos:




```
Private Sub Workbook_Open()
 Dim Msg As String
 If Weekday(Now) = vbFriday Then
    Msg = "Hoy es viernes, por fin a descanzar."
    Msg = Msg & Chr(13) & "Por favor haga su respaldo semanal."
    MsgBox Msg, vbInformation
 End If
 End Sub
```

En este ejemplo cada vez que abrimos el libro de trabajo verifica si el día es viernes, si es así despliega un mensaje que nos recuerda hacer nuestro respaldo semanal. Esto se ha hecho dando doble clic sobre ThisWorkBook en la ventana de proyectos del VBA tal como lo muestra la imagen.

Más adelante veremos otros ejemplos con otros tipos de evento a nivel de libro.

Eventos a nivel de Hoja de Trabajo

Al igual que un libro de trabajo, una hoja puede contar con sus propios eventos. Los mismos son listados a continuación:

Evento de Hoja de Trabajo	Descripción
Activate	Se activa la hoja de trabajo.
BeforeDoubleClick	Se hace doble clic sobre la hoja de trabajo.
BeforeRightClick	Se hace doble clic sobre la hoja de trabajo.
Calculate	Se calcula la hoja de trabajo o se vuelve a calcular.
Change	Las celdas de la hoja de trabajo son cambiadas por el usuario o por un enlace externo
Deactivate	Se desactiva la hoja de trabajo.
FollowHyperlink	Se hace clic sobre un hipervínculo de la hoja de trabajo.
PivotTableUpdate *	Se actualiza una tabla dinámica de la hoja.
SelectionChange	La selección de la hoja de trabajo se cambia.

Al igual que los eventos de un libro de trabajo, para accesarlos se tiene que dar doble clic sobre la hoja deseada e introducir el código deseado para cada evento.

Como dice mi profesor virtual John, dirigir estos eventos puede hacer que sus aplicaciones realicen proezas que serían imposibles de otra forma.

Aquí mostraré ejemplos más detallados ya que son eventos que se usan más en la realidad. No quiero decir que los de libro no se usen, solo insinúo que se utilizan solo ciertos eventos. También se pueden hacer maravillas en los eventos de libro.

Ahora veremos algunos ejemplos del maestro John que lista en su libro.

EVENTO ACTIVATE

Este evento ocurre cada vez que se activa una hoja de trabajo en particular. Podemos crear código en cada hoja para que una vez que se active se ejecute el mismo. El código de una hoja no interfiere con el de otra. Si queremos afectar hojas con el mismo código, recordemos que se debe crear en el libro de manera Public. De esta forma cualquier hoja lo puede utilizar, un ejemplo de esto sería el famoso convertidor de números a letras que tal vez ponga el código más adelante.

De aquí en adelante voy a tratar de adaptar algunos ejemplos del maestro José Felipe Ramírez R.

```
Private Sub Worksheet_Activate()
Dim Msg As String
        Msg = "Bienvenido " & Application.UserName
        Msg = Msg & Chr(13) & "Se encuentra en la hoja " & ActiveSheet.Name
        Msg = Msg & Chr(13) & "Que tenga un excelente día de trabajo."
        MsgBox Msg, vbInformation
End Sub
```

En este ejemplo, cada vez que se active la hoja de trabajo nos desplegará una leyenda de bienvenida, el nombre de la hoja y deseándonos un buen día.

Ahora lo vamos a mejorar un poco, daremos acceso a la hoja solo si se digita la contraseña correcta y solo tendrá 5 oportunidades para hacerlo.

```
Private Sub Worksheet_Activate()
Dim Msg As String
Msg = "Bienvenido " & Application.UserName
Msg = Msg & Chr(13) & "Se encuentra en la hoja " & ActiveSheet.Name
Msg = Msg & Chr(13) & "Que tenga un excelente día de trabajo."
MsgBox Msg, vbInformation
Dim Contador As Integer
Dim Pase As String
Do
   Pase = InputBox("Introduzca su contraseña: ", "Contraseñas")
```

```
  If Pase = "Clave" Or Pase = "clave" Then
    MsgBox "Bienvenido al sistema."
    Range("A2").Select
    Exit Sub
  Else
    MsgBox "Lo siento, contraseña incorrecta. Intente nuevamente."
    Contador = Contador + 1
  End If
Loop While Contador <= 4
If Contador = 5 Then Hoja1.Select
End Sub
```

Aquí tenemos un código más elaborado, si digitamos 5 veces la clave incorrecta, nos posicionará en la hoja1.

También podemos ver dos nuevas instrucciones: Application.UserName, que nos dará el nombre del usuario del libro y ActiveSheet.Name que nos dará el nombre de la hoja de trabajo. Vemos también el Or trabajando ya que nos dice que si Pase es igual a “Clave” O a “clave” entonces ejecute el resto del código.

No explico el código en detalle porque ya es hora de empezar a tratar de memorizar cada instrucción. Una ventaja que tiene Excel y que pueden aprovechar aquellos que son un poco olvidadizos, es que con solo digitar una instrucción y poner un punto (.), nos desplegará las propiedades (si las tiene) de esa instrucción.

EVENTO CHANGE

Ocurre cuando cualquier celda de la hoja activa es cambiada por el usuario o por un enlace externo. No se ejecuta cuando un cálculo genera un valor diferente o se añade un objeto a la hoja.

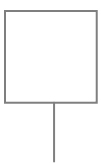
Este recibe un objeto Target que es un objeto Range que representa la celda o rango cambiado. Un ejemplo sencillo sería:

```
Private Sub WorkSheet_Change(ByVal Target As Excel.Range)
        MsgBox “Rango “ & Target.Address & “ fue cambiado.”
End Sub
```

El código anterior nos desplegará un mensaje cada vez que cambiemos una celda o rango de celdas. Lo pongo solo como ejemplo, no lo recomiendo que se use porque cada vez que hagamos algo en la hoja se desplegará el mensaje.

Bueno como a mí me gusta lo complicado, vamos a ver el siguiente código:

```
Private Sub Worksheet_Change(ByVal Target As Range)
  'MsgBox "Rango " & Target.Address & " fue cambiado."
Dim ccc As String
```



```
If Target.Column <> 2 Then Exit Sub
  ccc = Target.Value _
    & "     " & Format(Date + Time, "mm/dd/yy hh:mm")
  If Target.Comment Is Nothing Then
    Target.AddComment.Text ccc
   Else
    Target.Comment.Text (Target.Comment.Text & Chr(10) & ccc)
  End If
Target.Comment.Shape.TextFrame.AutoSize = True
Target.Comment.Visible = False
End Sub
```

Que belleza es Excel, me permite registrar los cambios realizados a una celda específica. Eso es lo que hace este código. Si hago algún cambio en la columna B, me insertará un comentario con el dato que tenía anteriormente y el actual junto con la fecha en que se realizaron.

Veamos otra modificación que nos hará casi lo mismo, pero con la diferencia que la hora será más detallada y nos mostrará el usuario que hizo el cambio al momento de estar usando la hoja (me refiero al usuario que activo la aplicación de Excel, no lo he probado en red). Para hacerlo más llamativo insertaremos un control CheckBox. Si este está activo registrará los comentarios, caso contrario no lo hará.

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim Cell As Range
Dim TextoAnt As String, NuevoTexto As String
   If Target.Column <> 3 Then Exit Sub
 If CheckBox1 Then
   For Each Cell In Target
      With Cell
         On Error Resume Next
         TextoAnt = .Comment.Text
         If Err <> 0 Then .AddComment
         NuevoTexto = TextoAnt & "Cambiado a " & Cell.Text & _
          " por " & Application.UserName & " el " & Now & vbLf
         .Comment.Text NuevoTexto
         .Comment.Visible = True
         .Comment.Shape.Select
          Selection.AutoSize = True
         .Comment.Visible = False
      End With
```

```
    Next Cell
  End If
End Sub
```

Ahora sí voy a comentar varias partes de código nuevas.

Explicación de nuevas Instrucciones

Instrucción	Explicación
Target	En español quiere decir “Objetivo”. Solo reconoce una celda como “Objetivo”.
Target.Column	Podemos referenciar que el objetivo se ejecute solo en cierta columna. En el ejemplo anterior se ejecuta solo en la columna C. Las columnas las representamos por medio de números: 1 es la A, 2 la B, 3 la C y así sucesivamente.
For Each…Next	Es un ciclo muy parecido al For … Next. Solo que se lee Para Cada Condicion … Siguiente Condicion.
On Error Resume Next	En caso de error continúe con el siguiente código. Se utiliza para ignorar errores.
Err <> 0	Nos indica que si el objeto Err es diferente de 0 continuemos con el código. Este objeto atrapa ciertos errores y tomaremos decisiones en base a ellos. En nuestro caso agregamos un comentario a la celda modificada.
With … End With	Aunque este ya lo expliqué anteriormente, vemos que aplicado a la variable Cell podemos agregar
.Comment.Text	Agregamos una leyenda al comentario, en este caso lo que contenga TextoNuevo.
Comment.Visible=True	Indicamos que el comentario se muestre con la opción True (Verdadero). Al final lo ponemos en False (Falso) para que se oculte y no quede en la hoja hasta que se seleccione la celda que lo contenga.
.Comment.Shape.Select	Lo utilizamos para seleccionar la figura del comentario.
Selección.AutoSize	Con esto indicamos que el comentario tome el tamaño según la cantidad de caracteres contenidos.

Ya vamos aprendiendo mucho más, así que paciencia que ya sabes más de cuando empezaste y estoy seguro que hay cosas que no sabías que se podían hacer con Excel.

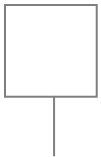
EVENTO SELECTIONCHANGE

Tiene lugar cada vez que el usuario hace una selección en la hoja de trabajo. Al igual que el anterior utilizamos un Target.

Un pequeño ejemplo nos ilustrará mejor su uso:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
Cells.Interior.ColorIndex = xlNone
With ActiveCell
   .EntireRow.Interior.ColorIndex = 36
   .EntireColumn.Interior.ColorIndex = 36
End With
End Sub
```

Con el segmento anterior aprendemos nuevas instrucciones. Por ejemplo:



- Cells.Interior.ColorIndex = xlNone: Le indicamos a Excel que no ponga ningún color de fondo a las Celdas.
- .EntireRow.Interior.ColorIndex = 36: Aquí escogemos un color amarillo para toda la fila. Podemos jugar poniendo otros números.
- .EntireColumn.Interior.ColorIndex = 36: Al igual que el anterior pero con la variación de que es la columna la que se pondrá en amarillo.

Este código hará que cada vez que nos posicionemos en una celda, toda la fila y la columna de esa celda se pongan de color amarillo. Es bueno para ciertos casos pero tiene el problema que si tenemos algunas celdas con ciertos formatos, por ejemplo, celdas con fondo negro y letras blancas, eliminará este formato.

TRABAJAR CON RANGOS

Bueno amigos, hemos estado trabajando con ciertos métodos sobre libros y hojas, ahora empezaremos a trabajar con rangos casi específicos. Recordemos que un rango es el conjunto de una o más celdas. Podemos manipular estos casi a nuestro antojo, ya sea insertando comentarios, cambiando los formatos, haciendo operaciones, comparándolos y muchas otras cosas.

Antes de continuar y ver ejemplos, es bueno saber sobre Objetos y Colecciones, las propiedades y métodos.

TECNOLOGÍA ORIENTADA A OBJETOS.

En Visual Basic (me refiero desde la versión 6.0, .Net y For Applications) todo es un objeto. Para entenderlo mejor usaré una práctica de mi profesor José Felipe Ramirez. Como dice él se requiere de mucha imaginación para la práctica:

I. Usted está sentado en una silla, frente a un escritorio.

II. Sobre ese escritorio se encuentran dos cosas: un sombrero de mago y un botón rojo.

III. Al presionar el botón, del sombrero sale un conejo. A dicho conejo usted lo llama “Fred”.

IV. Se presiona nuevamente el botón y sale otro conejo, al cual le llama “Angus”.

V. En este momento sobre el escritorio hay dos conejos, de igual color, tamaño y peso (mismas propiedades), y éstos corren alegremente sobre el escritorio.

VI. Sólo los puede diferenciar por el nombre, pero como saltan de un lugar a otro, es posible que en un momento determinado no sepa quién es Fred y quién es Angus.

VII. A usted le gustaría que Fred fuera más pequeño, y de otro color.

VIII. Usted toca a Fred y como por arte de magia (y como en nuestra imaginación todo es posible), Fred se transforma en un conejo más pequeño y de un color diferente que el de Angus.

Tenemos entonces que:

A. Cada conejo es de una naturaleza bien definida. Decimos que pertenece a la Clase Animal.

B. Como son dos conejos, se generaron dos Instancias de la Clase. A las instancias de una Clase se les llama ***Objetos***.

C. La ***Clase define todas las Propiedades*** que heredan los conejos, es por ello que Fred y Angus son iguales, y sólo se diferencian por el nombre, ya que no puede haber dos Objetos con el mismo nombre.

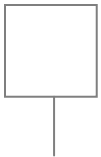
D. Fred y Angus no son inertes, por sí mismos hacen algo, a veces saltan. Lo que ellos pueden hacer pos sí mismos se llama ***Método***. Los Métodos que un objeto puede realizar, son sólo aquellos que la clase tiene definidos y permitidos. Ni Fred ni Angus pueden ladrar; no está en su clase.

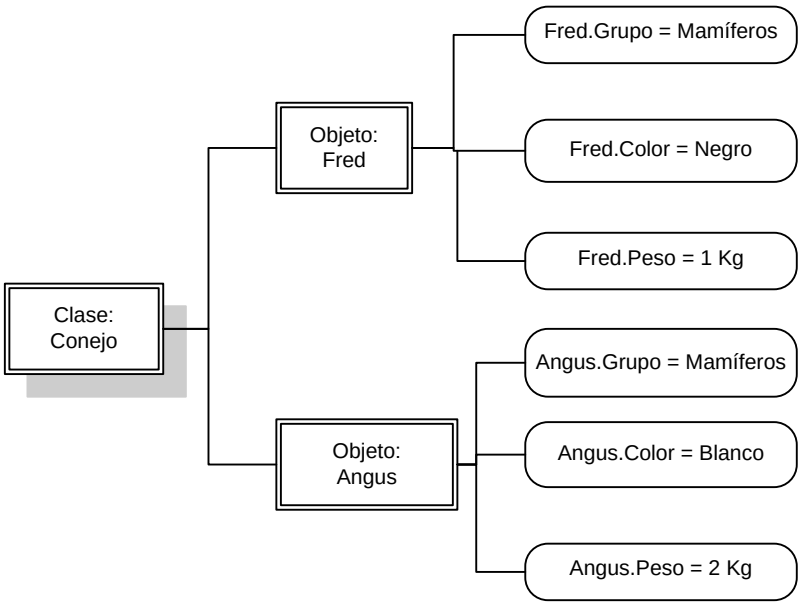
E. A Fred le sucedió una transformación, que no se llevó a cabo sino hasta que usted lo tocó. Lo que le sucede a un objeto por el hecho de interactuar con usted, se le llama Evento. Los Eventos que le pueden suceder a un Objeto, son sólo aquellos que la Clase tiene definidos y permitidos.

F. La Clase conejo existe, pero no se manifiesta sino a través de sus Instancias, mismas que son generadas solamente a través del uso del botón rojo. Si no se presiona el botón rojo, no hay conejos en el escritorio. Al Elemento que sirve para crear Instancia de una Clase se le denomina Control.

G. El escritorio sería el Objeto Contenedor de Objetos. Cumple con la misma función que cumplirá el formulario o controles en sus aplicaciones de Visual Basic for Applications.

Con el siguiente Organigrama podemos ilustrarlo un poco más detallado para que se entienda.





Como vemos, cada Clase puede tener sus Objetos y cada Objeto tiene definidos sus Eventos.

CONTROLES

Representaciones gráficas que permiten generar elementos de interfaz de Windows, como botones de comando, cuadros de texto, gráficos, listas desplegables, etiquetas, etc. Estos controles invocan una Clase determinada, con base a la cual se generan Objetos con una forma y comportamiento predefinido.

CLASE

Es la definición formal de un elemento de interfaz de Windows. Una Clase actúa como plantilla para cada una de las instancias generadas a partir de un determinado Control cuando es agregado a un formulario o a nuestra hoja; la Clase define las propiedades de un Objeto y los Métodos y Eventos utilizados para controlar su comportamiento. Visual Basic trata los términos Clase y Control como uno solo, ya que la Clase no se manifiesta sino a través de un Control.

OBJETOS

Los Objetos son ocurrencias de una Clase en un formulario o la hoja de trabajo. Son controles una vez que son insertados en una aplicación; el desarrollador les dará las propiedades que necesite y el comportamiento que requiera. Puede haber un solo control llamado CommandButton, pero muchos Objetos generados a partir de él, es decir, múltiples botones en el formulario u hoja.

Por cuestiones prácticas, en ocasiones los términos Control y Objeto se utilizan como sinónimos; es posible usarlos de esa manera, pero no se debe perder de

vista que ambos son diferentes. Un Control es un medio para generar un Objeto; un Objeto es una ocurrencia del Control (que a su vez es el ejercicio de una Clase), con nombre, tamaño, colores y otras especificaciones particulares.

Concuerdo con don Felipe que no deberíamos utilizarlos como sinónimos.

Colecciones

Grupo de Objetos de la misma Clase (una colección por sí misma es un Objeto). WorkBook es una Colección de todos los Objetos WorkBook. WorkSheet de todos los Objetos WorkSheet contenidos en WorkBook. Se puede trabajar con una Colección completa de Objetos o con Objetos individuales de una Colección.

Para referirnos a un Objeto en particular, lo haremos con solo poner su nonmbre: WorkSheets("Hoja1"). Utilizando la Colección Sheets que está hecha de todas las hojas de un libro de trabajo sería: Sheets(1).

Objetos Referidos

Para referirse a un Objeto utilizamos el punto (.). Si nos encontramos que tenemos dos libros abiertos y nos queremos referir a un objeto en particular de un libro, usamos: WorkBooks("Libro1").WorkSheets("Hoja1").

Con esto logramos indicarle al programa que queremos utilizar la Hoja1 del Libro1 y no se confundirá con los otros libros abiertos al momento.

Propiedades

Todos los Objetos tienen sus propiedades, como por ejemplo, si utilizamos un rango, algunas de sus propiedades serían: Value, Activate, Comment, Text, Column, Row.

Métodos

Es la acción que realiza un Objeto. Por ejemplo el método Clear: Range("A2:B2").Clear.

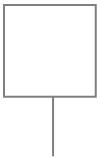
También existe el ClearContents que borra el contenido de un rango pero conservando el formato del mismo.

Trabajos con Rangos

Bueno amigos y amigas (y por qué no jovencitos y jovencitas), de aquí en adelante empezaremos a ver código más profundo. Empezaremos con lo básico y profundizaremos.

Trabajaremos a nivel de hojas y libros de trabajo. En el próximo capítulo entraremos a ver formularios (UserForm) y sus controles. Lo que nos dará una mejor presentación a nuestros libros de trabajo.

De aquí en adelante, ya es hora de saber leer un código desde su principio hasta el fin. No veo excusa para que alguien diga que no entiende que hace cierto



programa. Si bien es cierto que no saben algunas instrucciones, trataré de comentarlas al principio, pero conforme avancemos, iré quitando ciertos comentarios y dejaré solo los necesarios.

Los ejercicios hechos hasta ahora los pueden encontrar en el archivo Libro de Clases.xls. Están pensados para que corran en las versiones 2003 y 2007 de Excel. Aunque las pantallas que vean en los demás ejemplos sean hechas en la versión 2007, trataré de que todo corra en la versión 2003. A mi criterio, ya la mayoría debería tener la última versión, pero también soy consciente de que muchas empresas por razones de costos u otras, no han siquiera actualizado a la 2003.

Proyecto 1 – Filtro Avanzado en otra Hoja

Siempre he considerado que la persona que quiere aprender a programar es porque ya sabe todo, o casi todo, sobre las funciones, fórmulas y otras características del Excel. Pero me he encontrado con consultas en TodoExpertos.Com que han hecho cambiar mi forma de pensar.

He descubierto que muchos quieren hacer un programa y no tienen las nociones básicas para hacerlo. No saben que son variables, módulos, objetos. Peor aún, muchos y muchas no saben si quiera interpretar una fórmula. No tienen noción de lo que es una función y para qué sirve. Y para colmos, muchas veces quieren trabajar las hojas como si fuera una Base de Datos y no tienen las más mínima idea de cómo enlazar sus campos.

Todo este tipo de usuarios lo único que piensan es que la programación es un juego, en el cual solo presionas un botón y listo, ya todo está hecho. No saben la cantidad de código que tienen que introducir para poder lograr un objetivo. Creen que con dos comandos ya pueden hacer maravillas, pero que lejos están de la realidad. También piensan que en 6 meses van a saber cómo crear un súper programa, ja ja ja, pues se equivocan. Y no crean que me burlo de ellos, lo que pasa es que muchos y muchas quieren todo sin mover un solo dedo.

Aunque más adelante les enseñaré un poco sobre las Bases de Datos, esto no quiere decir que les explicaré a detalle las funciones y fórmulas de Excel.

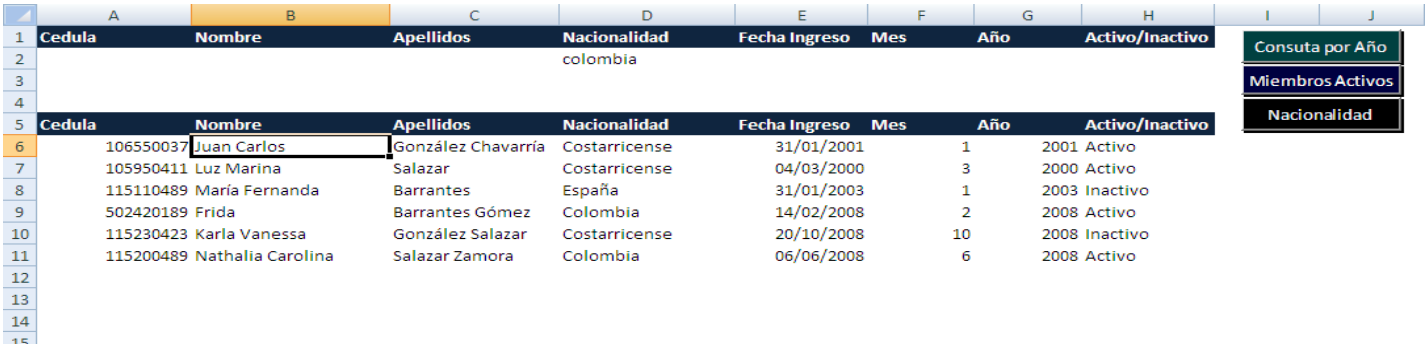
Y como los que están estudiando este libro quieren llegar a ser más profesionales, pues empecemos. Crearemos una tabla llamada HojaBase que nos quede de la siguiente forma:

	A	B	C	D	E	F	G	H
1	Cédula_Identidad	Nombre	Apellidos	Nacionalidad	Fecha Ingreso	Mes	Año	Activo/Inactivo
2								
3								
4								
5	Cédula_Identidad	Nombre	Apellidos	Nacionalidad	Fecha Ingreso	Mes	Año	Activo/Inactivo
6	106550037	Juan Carlos	González Chavarría	Costarricense	31/01/2001	1	2001	Activo
7	105950411	Luz Marina	Salazar	Costarricense	04/03/2000	3	2000	Activo
8	115110489	María Fernanda	Barrantes	España	12/12/2001	12	2001	Inactivo
9	502420189	Frida	Barrantes Gómez	Colombia	14/02/2008	2	2008	Activo
10	115230423	Karla Vanessa	González Salazar	Costarricense	20/10/2008	10	2008	Inactivo
11	115200489	Nathalia Carolina	Salazar Zamora	Colombia	06/06/2008	6	2008	Activo
12								

A partir de A6 la pueden llenar con los datos que gusten. Vamos a nombrar algunos rangos (a estas alturas ya deberían saber cómo nombrar rangos), los cuales serán los siguientes:

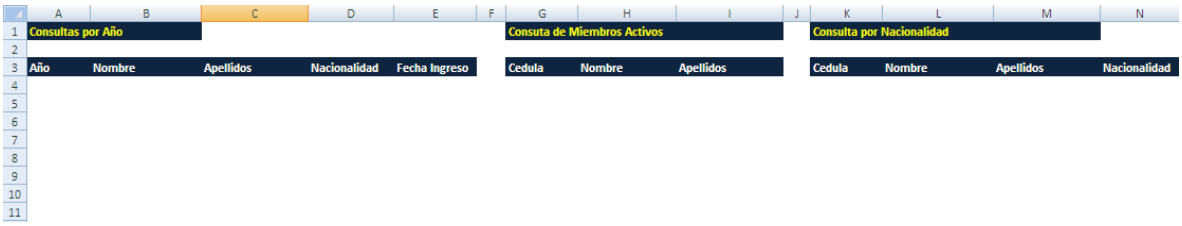
1. Marcamos de A1:H2 y lo nombramos “CritPrinc”.
2. Marcamos de A5:H30 y lo nombramos “BasePrinc”

Ahora insertamos tres botones de comando y nos quedará como esta imagen:



Cedula	Nombre	Apellidos	Nacionalidad	Fecha Ingreso	Mes	Año	Activo/Inactivo
			colombia				
Cedula	Nombre	Apellidos	Nacionalidad	Fecha Ingreso	Mes	Año	Activo/Inactivo
106550037	Juan Carlos	González Chavarría	Costarricense	31/01/2001	1	2001	Activo
105950411	Luz Marina	Salazar	Costarricense	04/03/2000	3	2000	Activo
115110489	María Fernanda	Barrantes	España	31/01/2003	1	2003	Inactivo
502420189	Frida	Barrantes Gómez	Colombia	14/02/2008	2	2008	Activo
115230423	Karla Vanessa	González Salazar	Costarricense	20/10/2008	10	2008	Inactivo
115200489	Nathalia Carolina	Salazar Zamora	Colombia	06/06/2008	6	2008	Activo

Observemos que he dado algunos formatos a los botones; eso se hace en las propiedades del control CommandButton. Las propiedades de los controles ya las expliqué anteriormente. Ahora crearemos otra hoja que la llamaremos “HojaConsultas”. En ella crearemos tres tipos de consultas que corresponderán a los botones que insertamos y nos quedará como la siguiente imagen:



Crearemos otros tres rangos que serán:

1. De A3:E3 y lo llamaremos “Salida01”.
2. De G3:I3 lo llamaremos “Salida02”.
3. De K3:N3 lo llamaremos “Salida03”

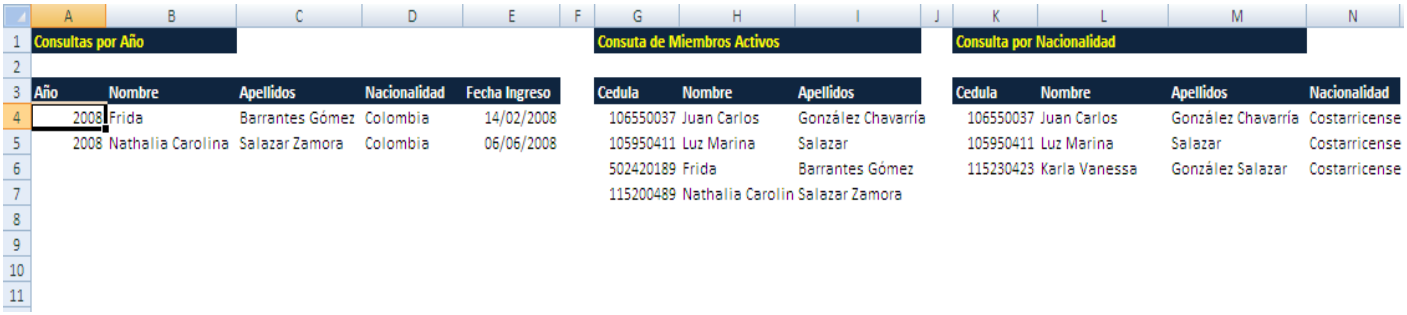
Nos pasamos a la hoja “HojaBase” y activamos el botón de Modo de Diseño en la barra de Controles. Una vez activa presionamos doble clic sobre el primer botón, lo cual nos abrirá el Editor de Visual Basic for Applications e insertamos el siguiente código:

```
Range("BasePrinc").AdvancedFilter Action:=xlFilterCopy, CriteriaRange:=Range( _
    "CritPrinc"), CopyToRange:=Hoja7.Range("Salida01"), Unique:=True
```

En la parte donde dice CopyToRange:=Hoja7, ustedes tienen que poner la hoja correspondiente, o pueden modificarlo así:
CopyToRange:=Sheets("HojaConsultas")

A los otros botones le insertan el mismo código pero con la diferencia que donde dice Hoja7.Range("Salida01"), lo cambian por "Salida02" y al tercero por "Salida03".

Digitemos en la celda G2, 2008, y presionamos el primer botón. Nos pasamos a la HojaConsultas y verán que despliega solo los que pertenecen al año 2008. Borramos el dato anterior y digitamos en H2, Activos, y presionamos el segundo botón, nos pasamos a la HojaConsultas y veremos que lista todos los activos. Borramos H2 y en D2 digitamos, Costarricense, y de nuevo vemos los maravillosos resultados que quedarían como la siguiente imagen:



Consultas por Año

Año	Nombre	Apellidos	Nacionalidad	Fecha Ingreso
2008	Frida	Barrantes Gómez	Colombia	14/02/2008
2008	Nathalia Carolina	Salazar Zamora	Colombia	06/06/2008

Consuta de Miembros Activos

Cedula	Nombre	Apellidos
106550037	Juan Carlos	González Chavarría
105950411	Luz Marina	Salazar
502420189	Frida	Barrantes Gómez
115200489	Nathalia Carolin	Salazar Zamora

Consulta por Nacionalidad

Cedula	Nombre	Apellidos	Nacionalidad
106550037	Juan Carlos	González Chavarría	Costarricense
105950411	Luz Marina	Salazar	Costarricense
115230423	Karla Vanessa	González Salazar	Costarricense

Podemos notar que trabajar con Filtros Avanzados es una herramienta muy poderosa. Imaginemos la gran utilidad que le podríamos dar a una hoja donde se almacenen las ventas diarias; podríamos consultar por facturas del año, las ventas a un proveedor específico, los proveedores o clientes activos, ventas en un rango de fechas, en fin, el límite es nuestra imaginación.

Se puede crear una tabla que contenga la información de nuestros empleados, proveedores, clientes u otros y hacer la consulta que queramos.

Y si han sido un poco curiosos, prueben insertando el año y Activo, o la nacionalidad y Activo, o los tres juntos. Inserten un número de cédula, y de nuevo, solo la imaginación es el límite.

Otra cosa muy importante, **no he tenido que digitar mucho código**, observen todo lo que se puede hacer con solo una pequeña modificación al código inicial.

PROYECTO 2 – COPIA DE RANGOS

Aquí empezamos una nueva fase de programación en VBA, la copia de rangos. Recordemos que un rango puede ser una celda o un conjunto de estas. Iniciaremos con algo sencillo y lo complicaremos durante el avance.

Para copiar una celda a otra se puede hacen de varias formas:

1. Copiar una celda a otra: Range("A3").Copy Destination:=Range("C1"). Esto me copia el contenido de la celda A3 en C1.

2. Copiar una celda en otra hoja: Range("A3").Copy Destination:= Hoja7.Range("C18").

3. Copiar una celda a otro libro:
   Workbooks("MiArchivo1.xls").Sheets("Hoja1").Range("A3").Copy Destination:=Workbooks("MiArchivo2.xls").Sheets("Hoja7").Range("C18")

Como pueden ver, es muy sencillo copiar una celda a otro lugar. Pero, ¿qué pasa si deseo copiar más de una celda?, bueno muy sencillo, lo hacemos de igual forma solo que se especifica el rango a copiar:
Range("A2:B19").Copy Destination:=Range("D2")

Podemos utilizar las mismas combinaciones anteriores para copiarlo a una hoja o libro en particular.

PROPIEDAD SET

Nos sirve para asignar una referencia de Objeto a una variable o propiedad. Podemos declarar una variable como: Dim MiRango as Range.

El Objeto tiene que ser válido. Vamos a asignar el Objeto declarado anteriormente a un rango específico: Set MiRango = Range("A2:B4").

Para descargar o desasignar se utiliza Nothing, así si queremos dejar vacía nuestra variable, entonces digitamos: Set MiRango = Nothing.

Ahora bien, si queremos pasar un rango que está variando constantemente ya que cada cierto tiempo se ingresan nuevos datos, entonces utilizaremos otra instrucción.

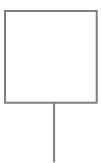
PROPIEDAD CURRENTREGION

Representa la región actual de trabajo del Objeto Range. Esta región está limitada por cualquier fila o columna vacía. Por ejemplo, si queremos pasar nuestra región de trabajo sin necesidad de especificar el rango, entonces se puede hacer con: Range("A1").CurrentRegion.Copy Sheets("Hoja2").Range("A1").

Esto nos copia todo lo que contengamos a partir de la celda A1 hasta Ax y hasta la columna n. Me copia exactamente todo con los formatos que contenga el rango.

MÉTODO END DEL OBJETO RANGE

Sirve para determinar la dirección a la que se extenderá una selección de rango. También es útil para desplazarse hacia alguna dirección específica, puede ser hacia arriba, abajo, izquierda o derecha (xlUp, xlDown, xlToLeft, xlToRight). Estas direcciones se colocan en inglés.



Para que sepamos cómo funciona, veamos un ejemplo para desplazarse a la última celda hacia arriba.

Range("A15").End(xlUp).Select. Aquí le decimos a Excel que se posicione en A15 y después que suba hasta la primera celda no vacía de la columna A.

Por ahorita no se preocupen si no entienden como aplicar estas instrucciones ya que pondré ejemplos que las incluyan.

Todo el código digitado hasta ahora, se encuentra en el archivo "Libro de Clases". El código funciona tanto en la versión 2002, 2003 y 2007 de Excel.

PROYECTO 3 – CREACIÓN DE FACTURA SENCILLA

Bueno, para no cansarlos con el cuento, empezamos con la creación de una factura sencilla. Primero crearemos una tabla que nombraremos "HojaFact". En ella tendremos el registro de los productos y los datos de los clientes a partir de la celda G1 como en la siguiente imagen:

	G	H	I	J	K	L	M	N
1	Código	Descripción	Precio		Código	Nombre	Dirección	Plazo
2	MB001	Tarjeta Madre P4	45,000.00		13165	Juan Carlos González	San José	30
3	IMP001	Impresora Inyección de Tinta	20,000.00		11511	Luz Marina Salazar	Cartago	15
4	IMP002	Impresora Laser	60,000.00		15223	Karla González Salazar	San José	0
5	CM001	Cámara Digital	70,000.00		15113	Nathalia Salazar Zamc	Cartago	45
6								
7								

	A	B	C	D	E
1	Fecha				
2					
3	Cliente				Plazo
4					
5	Código	Cant.	Descripción	Precio Unit.	Monto
6					
7					
8					
9					
10					
11					
12				Total	-
13					

Ustedes pueden poner los datos que quieran, yo trabajaré con los mostrados. Procedemos a diseñar la factura a partir de la celda A1 como lo muestra la imagen.

Nuevamente les menciono que son libres de utilizar el formato que deseen. En A2, estoy utilizando un formato de fecha (dd/mm/aaaa).

Ahora utilizaremos la función =BuscarV(); si esa función de la que muchos no tienen idea de cómo utilizarla, otros ni siquiera la comprenden.

Procedemos a marcar el rango G1:I20, y lo nombramos como "BaseProd"; luego marcamos de K1:N20 y lo nombramos como "BaseCli".

En B4 digitaremos la fórmula: =SI(A4="","",BUSCARV(A4,BaseCli,2,0)). Aquí le decimos al programa que si la celda A4 está vacía, entonces que deje B4 vacío, caso contrario, busque en la BaseCli el código que se digite en A4 y despliegue el nombre correspondiente.

En la celda E4: =SI(A4="","",BUSCARV(A4,BaseCli,4,0)). Esto buscará en la base de clientes, el plazo de crédito concedido.

Cabe aclarar que no es estrictamente necesario tener los productos y clientes en la misma hoja, yo lo pongo así para que me entiendan pero bien pueden tener estos datos en hojas diferentes.

Es hora de aprender una nueva técnica de insertar una fórmula en un rango de celdas digitándola una sola vez. Marcaremos el rango C6:C11 y digitaremos la fórmula =SI(A6="","",BUSCARV(A6,BaseProd,2,0)). Cuando terminemos de digitarla presionaremos las teclas Ctrol-Enter. Automáticamente la fórmula se introducirá en las demás celdas. De igual modo haremos con el rango D6:D11, pero la fórmula es =SI(A6="","",BUSCARV(A6,BaseProd,3,0)).

Bueno ahora tenemos dos opciones para poder trasladar los datos de la factura a la hoja de registro, una es creando un extenso código para trasladar los datos de manera adecuada, la otra es crear una serie de fórmulas de evaluación y después pasar las líneas correspondientes a la hoja de registro. A mi criterio vamos a utilizar la segunda opción que es más fácil ya que todos estamos familiarizados con las fórmulas que voy a poner (al menos eso creo).

A partir de T1 pondremos los títulos de todos los datos tal como la imagen:

	T	U	V	W	X	Y	Z	AA
1	Fecha	Cod_Cliente	Nombre	Plazo	Cod_Prod	Cantidad	Descripción	Precio
2	10/06/2009	13165	Juan Carlos C	30	mb001	1	Tarjeta Madr	45,000.00
3	10/06/2009	13165	Juan Carlos C	30	cm001	2	Cámara Digit	70,000.00
4								
5								
6								
7								

Aunque la imagen presenta datos, estos fueron creados con las siguientes fórmulas:

- Marcamos el rango T2:T7 y digitamos: **=SI($A6="","",$A$2)**, luego presionamos las teclas Ctrol-Enter. Esto le dirá al programa que si la celda A6 está vacía, entonces que T2 queda vacía también, sino que despliegue el contenido de A2, que en nuestro caso es la fecha.
- Seleccionamos el rago U2:U7 y digitamos: **=SI($A6="","",$A$4)**, de igual forma presionamos la combinación de teclas para desplegar el código del cliente. De aquí en adelante doy por entendido que se tienen que presionar las teclas Ctrol-Enter para introducir las fórmulas en las correspondientes celdas.
- Seleccionamos V2:V7 y digitamos: **=SI($A6="","",$B$4)**.
- Seleccionamos W2:W7 y digitamos: **=SI($A6="","",$E$4)**.
- De X2:X7: **=SI($A6="","",$A6)**.

- De Y2:Y7: **=SI($A6="","",$B6)**.
- De Z2:Z7: **=SI($A6="","",$C6)**.
- Por último de AA2:AA7: **=SI($A6="","",$D6)**.
- Para terminar con las fórmulas digitamos en A12: **=CONTAR(B6:B11)** y le damos al formato de texto un color blanco para que no se vea en pantalla.

Como pueden observar, son fórmulas muy, pero muy sencillas. No se les olvide darles el formato adecuado a cada columna para que se pase igual cuando se graben los datos.

Ahora insertaremos un botón de comando y le damos el formato que deseemos, en mi caso queda como la siguiente imagen:

	A	B	C	D	E
1	Fecha				Registrar
2	10/06/2009				
3	Cliente				Plazo
4	13165	Juan Carlos González			30
5	Código	Cant.	Descripción	Precio Unit.	Monto
6	mb001	1	Tarjeta Madre P4	45,000.00	45,000.00
7	cm001	2	Cámara Digital	70,000.00	140,000.00
8					
9					
10					
11					
12				Total	185,000.00
13					

ahora nos disponemos a digitar un código para poder trasladar los datos, pero antes crearemos una hoja que llamaremos RegFact con sus respectivos campos, que es donde guardaremos los datos de todas las facturas que se digiten de ahora en adelante.

	A	B	C	D	E	F	G	H
1	Fecha	Código	Nombre	Plazo	Cod_Prod	Cantidad	Descripción	Precio
2								
3								
4								

En la imagen anterior observen el formato que yo elegí para guardar los datos. Mi botón lo he nombrado cmdRegistra. Presiono el ícono “Modo de Diseño”, doy doble clic sobre el botón para ingresar al código del mismo y digito:

```
Private Sub cmdRegistra_Click()
'Declaración de Variables
Dim Count_Row As Long
Dim i As Integer
```

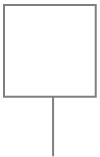
```
'Apagar los parpadeos
Application.ScreenUpdating = False
'Verificar que existan datos a copiar
If Range("A2, A4").Value = "" Or Range("A6, A7").Value = "" Then
   MsgBox "Lo siento faltan datos.", vbInformation, "Datos incompletos."
Else
   Range("T2").Select
   'Ciclo que me dice cuántos datos se van a trasladar
   For i = 1 To Range("A12").Value
      Range(Selection, Selection.End(xlToRight)).Select
      Selection.Copy
      Hoja10.Select
      Count_Row = WorksheetFunction.CountA(Range("A:A")) + 1
      ActiveCell.PasteSpecial xlPasteValues
      ActiveCell.Offset(1, 0).Select
      Hoja9.Select   'Volvemos a la hoja de Factura
   Next i
End If
Range("A2").Select  'Me posiciono en la celda A2

'Vuelvo a activar los parpadeos
Application.ScreenUpdating = True
End Sub
```

Si han seguido las instrucciones al paso de la letra, obtendrán un resultado como el siguiente:

	A	B	C	D	E	F	G	H
1	**Fecha**	**Código**	**Nombre**	**Plazo**	**Cod_Prod**	**Cantidad**	**Descripción**	**Precio**
2	10/06/2009	13165	Juan Carlos González	30	mb001	1	Tarjeta Madre P4	45,000.00
3	10/06/2009	13165	Juan Carlos González	30	mb001	1	Tarjeta Madre P4	45,000.00
4	04/03/2009	11511	Luz Marina Salazar	15	imp002	3	Impresora Laser	60,000.00
5	04/03/2009	11511	Luz Marina Salazar	15	imp002	3	Impresora Laser	60,000.00
6								

Ya tenemos nuestro primer registro de facturas automático. Y para hacerlo más efectivo pueden seleccionar los rangos de fecha, código, código de producto y cantidad, presionan el botón derecho del mouse y desmarcan la casilla de Bloqueada. Luego protegen la hoja. La ventaja de esto es que nos permitirá posicionarnos en la celda correspondiente cada vez que presionemos la tecla Enter.

Podrán observar que con lo que hemos aprendido hasta ahora, hemos podido automatizar un proceso y no hemos tenido que digitar mucho código. Otra de las ventajas es que este principio también se puede utilizar para registrar un Asiento Contable. También se darán cuenta que aunque puse que era una factura sencilla,



si lo vemos con cuidado, no es tan sencilla como parece y la podemos modificar y hacerla tan compleja como queramos agregando otros campos.

Hasta aquí dejamos todo lo correspondiente a Diseño de Pantallas, Ciclos, Decisiones, Programación con Rangos y Trabajos con Hojas.

A partir del próximo capítulo entraremos a ver UserForm (Formularios de Usuario) y sus controles. Aprenderemos cosas muy interesantes y útiles.

Continuaré con la misma forma de enseñar, me refiero a que aprenderemos nuevas cosas en la práctica y tratando de poner poca teoría. Quiero que sea lo más práctico posible.

Todo el código empleado lo pueden ver en el archivo “Libro de Clases”, el cual contiene las macros y las hojas con las que hemos trabajado hasta ahora.

Chao y hasta el próximo capítulo.