There are no translations available.
Cuando conozcas estas herramientas te resultarán insustituibles para desarrollar tus hojas Excel...

Programación en Excel

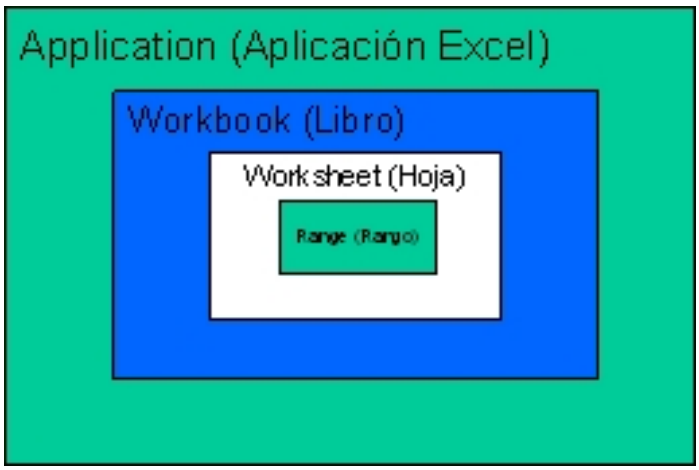
1 Programación

Con este curso queremos iniciar una serie que te acerque a las herramientas de que dispone Excel, para mejorar el rendimiento de todos tus trabajos.

Puede que programar hayas pensado que resulta una tarea muy complicada. Mi objetivo es demostrarte que, una vez que hayas alcanzado el grado suficiente de confianza, estas herramientas te resultarán insustituibles para desarrollar tus hojas Excel.

¡Adelante con ello!

1.1 Objetos Application, Workbook, WorkSheets, Range.



Visual Basic es un entorno de programación orientado a objetos. Si te sirve de algo, la primera vez que oí esto, también me quedé igual de perplejo. Al final he descubierto que es todo muy sencillo: En Visual Basic, tu programas todo lo que está relacionado con lo que pasa con un objeto; es decir, que si tienes un botón, puedes programar lo que pasa cuando haces "clic" o "doble clic" o sólo pasas el cursor por encima o pulsas el botón derecho del ratón. Es decir, que a cada objeto le corresponden unos eventos (cosas que le pasan al objeto).

Excel te facilita una jerarquía de **objetos** que te sirven para automatizar o personalizar el trabajo diario.

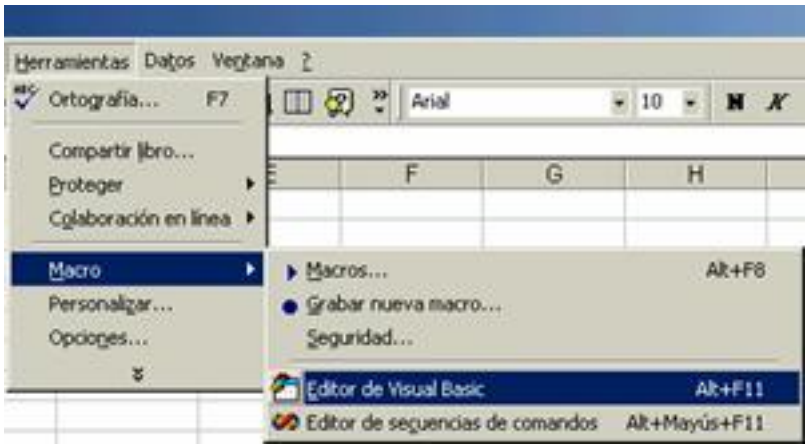
Así, por ejemplo tenemos:

· El objeto **Application** es el objeto superior y representa a la aplicación Excel.

· El objeto **WorkBook** se refiere a los distintos libros abiertos dentro de la aplicación Excel. Depende del objeto Application.

· El objeto **WorkSheet** es el conjunto de hojas de un libro. Depende de un objeto WorkBook.

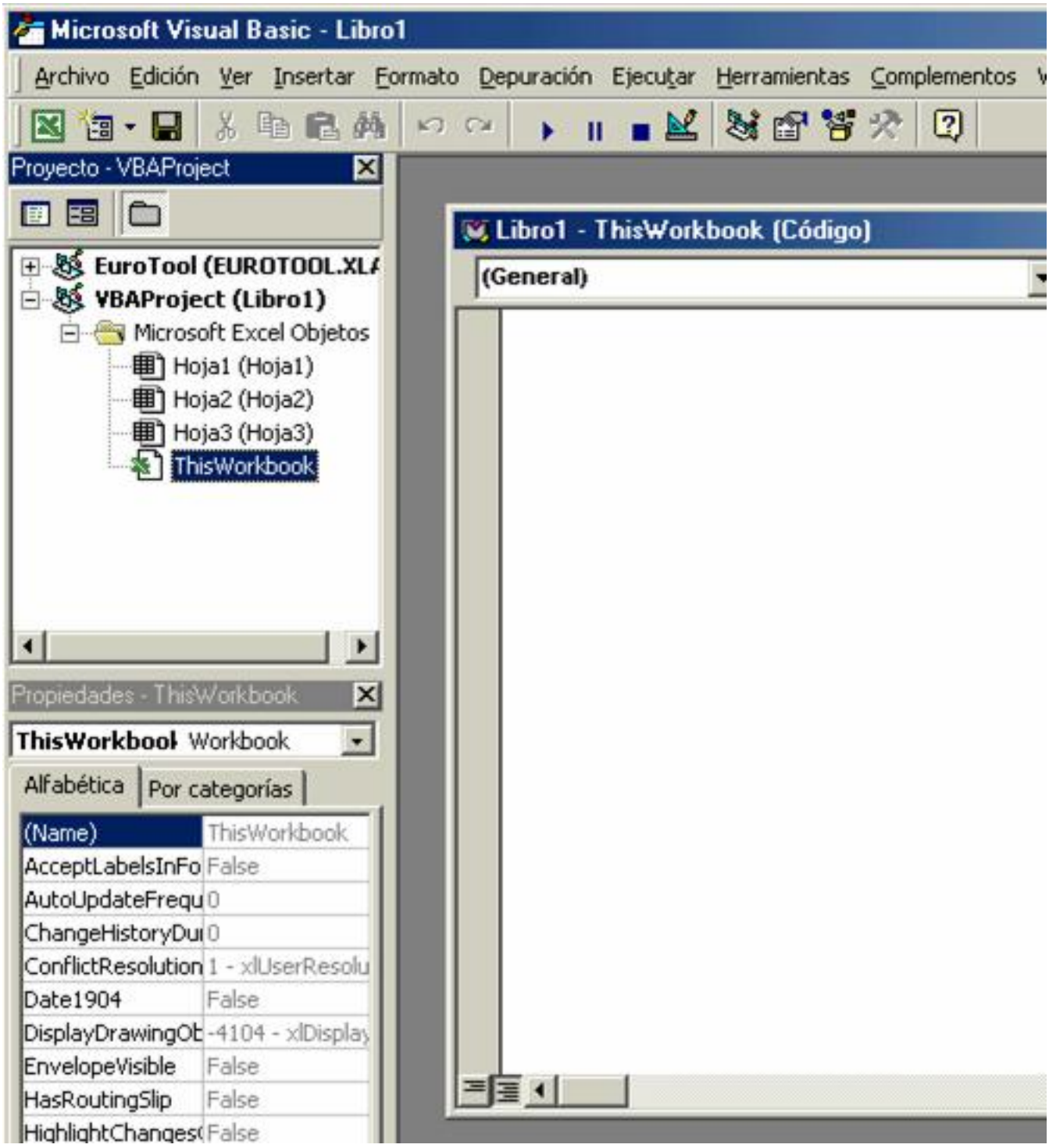
· El objeto **Range** se refier a una celda o a un rango de celdas. Normalmente depende de un objeto WorkSheet.



En resumen un objeto Application puede contener varios Libros (WorkBooks), que contienen hojas (WorkSheets), que a su vez contienen otros objetos (por ejemplo Range).

Para acceder al editor de Visual Basic selecciona en el menú Herramientas -> Macros -> Editor

de Visual Basic.



Se te abrirá el editor de Visual Basic, en el que puedes observar diferentes ventanas: A la

izquierda arriba la ventana con los distintos objetos del proyecto. Debajo la ventana de propiedades del objeto seleccionado. A su derecha la ventana donde escribirás el código de programación del objeto seleccionado.

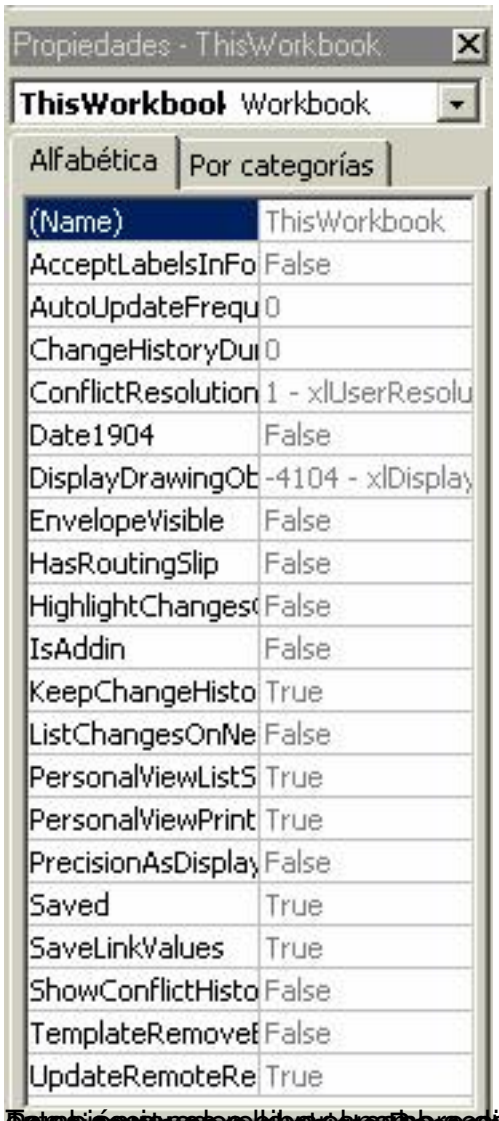
Puedes modificar o conocer las características de cada uno de estos objetos, accediendo a las **propiedades**
de los mismos.

Por ejemplo para el objeto Range tienes las siguientes propiedades:

· **Value,** contiene el valor de la celda (su contenido).

· **Column** y **Row,** que contienen respectivamente los datos de la fila y la columna que se corresponden con la celda.

· **Font,** que contiene la fuente de los caracteres que muestra la celda (Arial, Courier, etc).



En los objetos que te facilita Excel. Cada una de

1.1.1 Práctica 1. Trabajar con los objetos de Excel

Escribir un texto en la celda A1

1. Abre Excel y guarda el libro actual como Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Inserta un nuevo módulo: Insertar → Módulo en la barra de menús del Editor de Visual

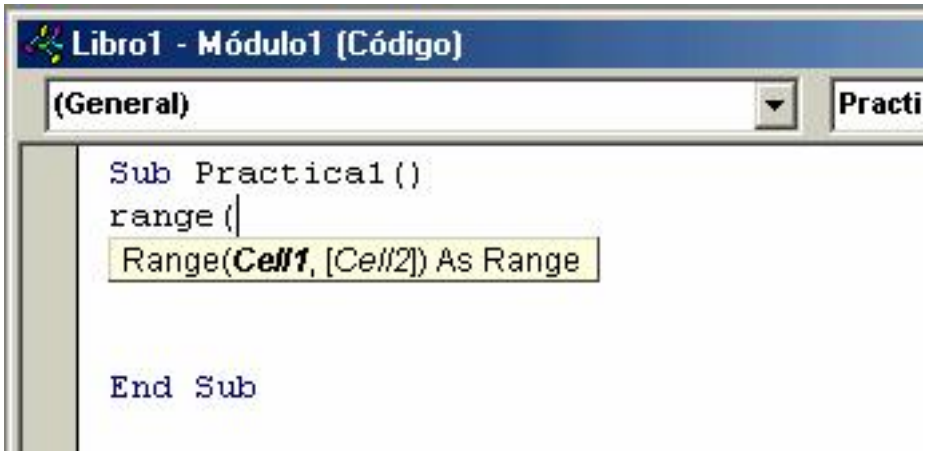
Basic.

4. En este módulo guardarás los procedimientos que desarrolles en los ejemplos.

5. Inserta un procedimiento: escribe Sub Practica1 dentro del módulo que has insertado y pulsa Enter.

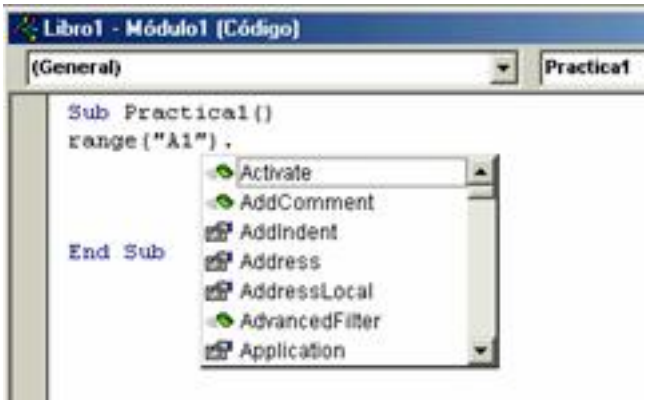
6. Automáticamente aparecerá debajo End Sub. En el espacio existente entre estas dos instrucciones escribirás el código del procedimiento.

7. Escribe Range("A1").Value = "Esta es la Celda A1"



8. Comprueba que si escribes solamente Range("A1"). Visual Basic te indicará las propiedades y métodos que puedes aplicar al objeto Range
.

9. Vas a cambiar el contenido de la celda, cambiando la propiedad Value (asignándole un nuevo valor) . Observa que para separar el objeto de su propiedad se utiliza la notación punto (.).



10. Ejecuta el procedimiento

11. Sitúa el cursor dentro del procedimiento. Selecciona en la barra de menús Ejecutar → Ejecutar Sub/Userform. También puedes hacer clic sobre el botón o pulsar la tecla F5.

12. Comprueba en la hoja Excel el resultado: Verás que en la celda A1 ahora está el texto que has escrito: "Esta es la Celda A1"

¡Felicidades, ya has programado tu primer procedimiento!

1.1.2 Práctica 2. Trabajar con los objetos de Excel

Escribir un texto en la celda A1 utilizando la jerarquía completa de objetos

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de

Excel.

3. Abre el módulo Modulo 1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica2 dentro del módulo que has abierto y pulsa Enter.

5. Automáticamente aparecerá debajo End Sub.

6. Escribe WorkSheets(2).Range("A1").Value = "Esta es la Celda A1 de la hoja 2" en el espacio que hay entre
Sub
y
End Sub
.

7. Al escribir WorkSheets(2) delante te estás refiriendo a la celda A1 de la hoja 2.

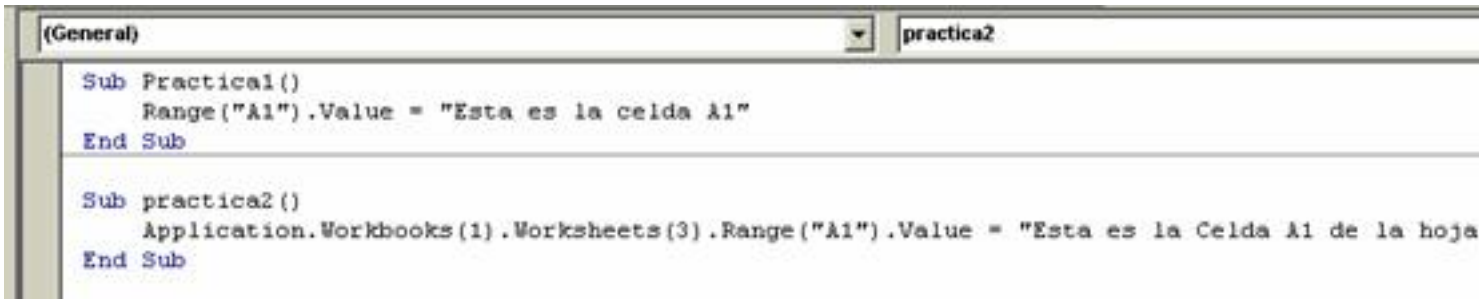
8. Ejecuta el procedimiento y comprueba los resultados

9. Escribe WorkBooks(1).WorkSheets(3).Range("A1").Value = "Esta es la Celda A1 de la hoja 3 del libro1" en el espacio que hay entre Sub y End Sub.

10. Al escribir WorkBooks(1) delante te estás refiriendo a la celda A1 de la hoja 3 del libro 1.

11. Ejecuta el procedimiento y comprueba los resultados

12. Escribe Application.WorkBooks(1).WorkSheets(3).Range("A1").Value = "Esta es la Celda A1 de la hoja 3 del libro1 (jerarquía completa)" en el espacio entre Sub y End Sub.



13. Al escribir Application delante utilizas la jerarquía completa de objetos Excel. Observa que en algunos casos no es necesario especificar todas las dependencias entre objetos de Excel. Lo que ocurre es que Excel utiliza objetos por defecto. Así, si no especificas hoja, Excel supone que te refieres a la hoja activa; si no especificas libro, se supone que te refieres al libro activo, etc.

1.1.3 Ejercicios

1. Escribe un procedimiento que escriba "hola" en negrita en la celda B4.

2. Igual que el anterior pero que escribe el texto en un nuevo libro, en la Hoja3.

1.2 Tipos de datos. Declaración de variables

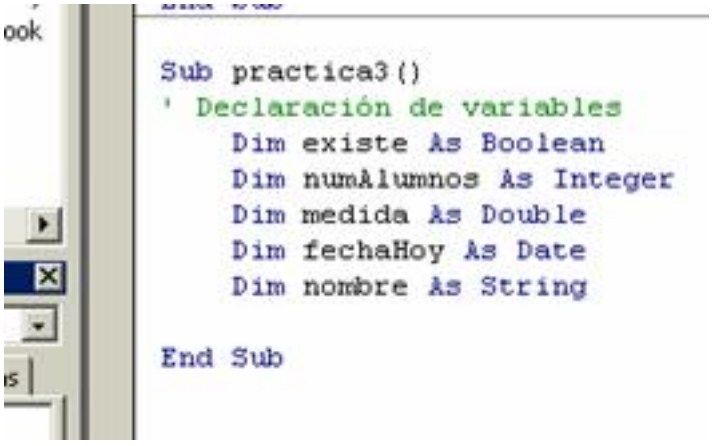
Cuando se programa, puede ser necesario almacenar información para su uso posterior en otra parte del programa. Para ello dispones de **variables** en las que puedes guardar distintos **tipos de datos**
.

Los tipos de datos de que dispones son **Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Single**, **Double**
,
Date

,
String
,
Object
,
Variant
(Variant es el tipo de datos predeterminado).

Los que más se suelen utilizar son:

- Boolean: Valor lógico. Sólo tiene dos valores posibles, **True** (verdadero) o **False** (falso).
- Integer: Numérico entero
- Double: Numérico de doble precisión
- Date: Fecha
- String: Cadena de caracteres



Para declarar una variable se utiliza la instrucción Dim. Su sintaxis es:

DIM variable **AS** ***tipo*** (Declara variable la que sea- del tipo el que corresponda-)

Observa unos ejemplos en la imagen de la izquierda.

1.2.1 Práctica 3. Declaración de variables

Solicita por pantalla variables y establécelas en distintas casillas

1. Abri Excel y abre el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Mólulo 1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica3 dentro del módulo que has abierto y pulsa Enter.

Automáticamente aparecerá debajo **End Sub**. Escribe el siguiente código para el procedimiento:

```
Sub practica3()
' Declaración de variables
Dim nombre As String
Dim edad As Integer

' Asignar valores a las variables
nombre = InputBox("Nombre:", "Introducir texto")
Range("B2").Value = nombre
edad = InputBox("Edad:", "Introducir número")
Range("B3").Value = edad


End Sub
```

… utilizar aquí una función muy útil de Visual Basic, **InputBox** (Abre un

… el procedimiento y comprueba los resultados.

1.2.2 Ejercicios

1. Pide por pantalla dos números y asigna la suma a la casilla C4, declarando las variables que utilices.

2. Igual que el anterior pero, al solicitar los números, el primero será 100 por defecto y el segundo será 1000 por defecto. Recuadra el resultado.

1.3 Option Explicit

Es conveniente para evitar escribir incorrectamente el nombre de una variable existente o para evitar confusiones en el código, utilizar la instrucción **Option Explicit.**

```
Sub practica3()
' Declaración de variables
Option Explicit
Dim nombre As String
Dim edad As Integer
```

Esta instrucción se usa en el módulo para forzar las declaraciones explícitas de todas las variables en dicho módulo, es decir, que no las des por supuestas , sino que se tengan que declarar antes de comenzar.

Si usas, la instrucción **Option Explicit,** ésta debe aparecer en un módulo antes de cualquier otro procedimiento.

Si intentas usar un nombre de variable que no hayas declarado antes, te dará un error de compilación.

Si no usas la instrucción **Option Explicit,** todas las variables que no hayas declarado previamente, tendrán el tipo **Variant**, a menos que el tipo de variable predeterminado esté especificado de otra manera.

1.4 Sentencias condicionales

Con las sentencias condicionales, puedes controlar la ejecución de un fragmento de código en función de si se cumple o no una condición, que hayas fijado previamente.

La sintaxis es la siguiente:

If *condición* **Then**
[*instrucciones*]

[**ElseIf** *condición-n* **Then**
[*instrucciones_elseif*] ...

[**Else**
[*instrucciones_else*]]

End If

1.4.1 Práctica 4. Sentencias Condicionales

Establece distintos colores según el valor de una celda

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica4 dentro del módulo que has abierto y pulsa Enter.

Automáticamente aparecerá debajo **End Sub**. Escribe el siguiente código para el procedimiento:

```
Sub practica4()

ActiveSheet.Range("A3").Value = ActiveSheet.Range("A1").Value - ActiveSheet.Range("A2").
If ActiveSheet.Range("A3").Value < 0 Then
ActiveSheet.Range("A3").Font.Color = RGB(255, 0, 0)
Else
ActiveSheet.Range("A3").Font.Color = RGB(0, 0, 255)
End If

End Sub
```

Un comentario: Vas a utilizar el objeto **ActiveSheet** para referirte explícitamente a la hoja activa y la función **RGB** para establecer el color de la fuente de la celda sobre la que trabajas.

5. Prueba con distintos números en las casillas A1 y A2, ejecuta el procedimiento y comprueba los resultados.

1.4.2 Ejercicios

1. Escribe un procedimiento que escriba "Mayor de 10" o "Menor de 10" en la casilla A2, en función de como sea el número existente en la celda A1.

2. Igual que el anterior pero si el número de la casilla A1 es menor de 0 se pone el texto en rojo.

1.5 Bucles

Seguro que has pensado alguna vez, en cómo repetir varias veces una misma acción, hasta que se cumpla una condición o hasta que hayas llegado a repetirla un número determinado de veces. Para eso sirven los bucles.

Todos los bucles necesitan una condición, que se debe cumplir, para que se produzca la siguiente repetición del bucle.

Hay distintos tipos de bucles:

· **Do...Loop**: Seguir en el bucle mientras o hasta una condición sea cierta (True).

· **For...Next**: Utilizar un contador para ejecutar las instrucciones un número determinado de veces.

· **For Each...Next**: Repetición del grupo de instrucciones para cada uno de los objetos de una colección.

1.5.1 Práctica 5. Trabajar con bucles 1

Mostrar en pantalla los números del 1 al 5

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica5 dentro del módulo que se ha abierto. Automáticamente aparecerá debajo **End Sub**.

5. Escribir el siguiente código para el procedimiento:

```
Sub practica5()

Dim i As Integer
For i = 1 To 5
MsgBox i
Next

End Sub
```

Un comentario: Vas a utilizar una función muy útil de Visual Basic, **MsgBox,** que muestra un mensaje en un cuadro de diálogo. Si quieres obtener una descripción completa de esta función

consulta la ayuda.

6. Ejecuta el procedimiento y comprueba los resultados.

1.5.2 Ejercicios

1. Crea un procedimiento que pida por pantalla el nombre y la edad de 3 alumnos.

2. Igual que el anterior pero insertando una fila de un color entre cada alumno.

1.6 Comprender la ayuda de Visual Basic (Microsoft Excel 2002)

Si quieres acceder a la ayuda desde el editor de Visual Basic, sólo necesitas pulsar la tecla F1 para mostrar la ayuda relacionada con la palabra en la que se encuentra el cursor.

Si esta palabra es del lenguaje de Visual Basic aparece la ayuda de lo que es el lenguaje de programación únicamente.

Si la palabra sobre la que pulsamos F1 está relacionada con los objetos de Excel, verás una ayuda diferente, en la que el primer nodo de la solapa contenido es **Referencia Visual Basic de Microsoft Excel** .

Es interesante echar un vistazo a esta ayuda con especial atención a los temas:

· **Objetos de Microsoft Excel**. Contiene toda la ayuda relacionada con los objetos, propiedades y métodos.

· **Conceptos de programación**. La ayuda te ofrece ejemplos generales interesantes que

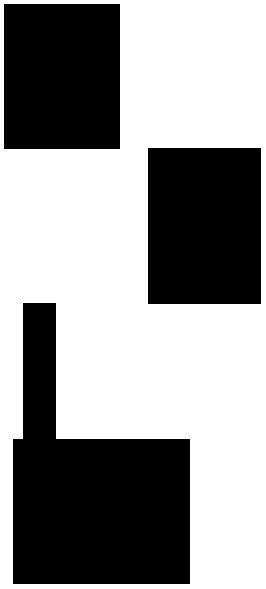
se pueden aplicar después a un caso particular.

Hay que tener en cuenta las siguientes consideraciones:

· En la sintaxis de los ejemplos que verás en la ayuda, los argumentos de una función o procedimiento que no van entre corchetes son obligatorios. Los que van entre corchetes son opcionales. Si no deseas especificar algún argumento es necesario poner la coma que delimita ese argumento, es decir, dejar el espacio que le corresponde en blanco.

Ejemplo:

MsgBox(*prompt*[, *buttons*][, *title*][, *helpfile*, *context*])



Obligatorio Opcionales

1.7 Recorrer celdas de una hoja de cálculo

Una operación bastante habitual cuando se trabaja con Excel es el recorrido de grupos de celdas para llenarlas con valores, mirar su contenido, etc. Los bucles son imprescindibles para recorrer grupos de celdas o rangos.

Para realizar esta tarea son también de gran ayuda las siguientes propiedades:

· **ActiveCell:** Nos proporciona la celda activa.

Ejemplo: Escribe hola en la celda activa: ActiveCell.Value = hola

· **Cells:** Sirve, como el objeto **Range**, para referenciar una casilla o rango de casillas, pero en lugar de utilizar la referencia de la forma A1, B1,... utiliza la fila y la columna que ocupa la casilla dentro de la hoja.

Ejemplo: Poner hola en la casilla A1 de la hoja activa: ActiveSheet.Cells(1,1).Value="Hola"

· **Offset:** significa desplazamiento, es una propiedad del objeto **Range** y se utiliza para referenciar una casilla situada a n Filas y n Columnas de una casilla dada.

Ejemplos:

ActiveSheet.Range("A1").Offset(2, 2).Value = "Hola" Resultado: Casilla C3 = Hola, 2 filas hacia abajo y 2 columnas hacia la derecha desde A1.

ActiveCell.Offset(3,1).Value = "Hola" Resultado: 3 Filas por debajo de la casilla Activa = Hola

ActiveCell.Offset(-2,4).Activate Resultado: Activar la casilla que está 2 filas por encima y 3 columnas a la derecha de la activa

1.7.1 Práctica 6. Recorrer celdas de una hoja 1

Llena el rango de las casillas A1..A5 con valores consecutivos empezando por el 7.

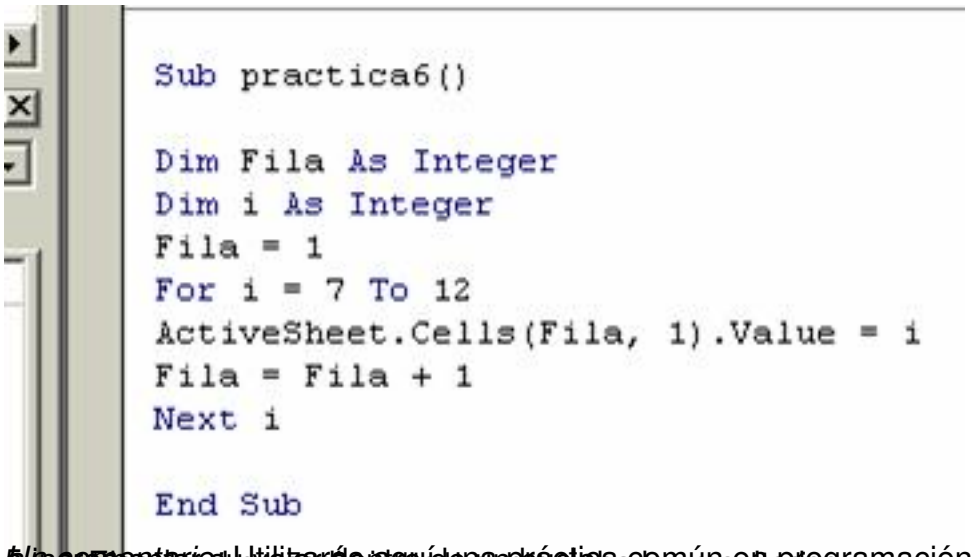
1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica6 dentro del módulo que se ha abierto y pulsa Enter.

Automáticamente aparecerá debajo **End Sub**. Escribe el siguiente código para el procedimiento:



```
Sub practica6()

Dim Fila As Integer
Dim i As Integer
Fila = 1
For i = 7 To 12
ActiveSheet.Cells(Fila, 1).Value = i
Fila = Fila + 1
Next i

End Sub
```

6. Ejecuta el procedimiento y comprueba los resultados.

1.7.2 Práctica 7. Recorrer celdas de una hoja 2

Llenar el rango de las casillas A1..A5 con valores consecutivos empezando por el 0.

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica7 dentro del módulo que se ha abierto y pulsa Enter.

```
Sub practica7()

Dim Fila As Integer
ActiveSheet.Range("A1").Activate
For Fila = 0 To 4
ActiveCell.Offset(Fila, 0).Value = Fila
Next Fila

End Sub
```

Automáticamente aparecerá debajo **End Sub**. Escribe el siguiente código para el procedimiento:

1.7.3 Práctica 8. Recorrer celdas de una hoja 2

Busca la primera celda vacía en la columna A de la Hoja1

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica8 dentro del módulo que se ha abierto y pulsa Enter.

Automáticamente aparecerá debajo **End Sub**. Escribe el siguiente código para el procedimiento:

```
Sub practica8()

Worksheets("Hoja1").Activate
ActiveSheet.Range("A1").Activate
Do While Not IsEmpty(ActiveCell.Value)
ActiveCell.Offset(1, 0).Activate
Loop

End Sub
```

Ejecutar el procedimiento y comprobar los resultados. Comprobar si una celda esta vacía.

1.7.4 Ejercicios

1. Localizar la primera celda vacía. Solicitar nombre y edad, y rellenar filas hasta que el nombre introducido este vacío

2. Igual que el anterior pero si la edad es mayor que 18, solicitar el DNI e introducirlo en color azul en la celda contigua a la edad.

1.8 Creación y utilización de procedimientos

Seguramente no te has dado cuenta, pero llevas practicando con procedimientos todo el tiempo.

Un procedimiento es un bloque de instrucciones de código que sirven para llevar a cabo alguna tarea específica.

Cada tarea la realizará un procedimiento y, si esta tarea implica la ejecución de otras tareas, cada una se implementará y solucionará en su correspondiente procedimiento de manera que cada uno haga una cosa concreta. Así, los diferentes pasos que se deben ejecutar para que un programa haga algo, quedaran bien definidos cada uno en su correspondiente procedimiento. Si el programa falla, fallará a partir de un procedimiento y de esta forma podremos localizar el error más rápidamente.

Los procedimientos son también un eficaz mecanismo para evitar la repetición de código en un mismo programa e incluso en diferentes programas.

Sintaxis:

Sub Nombre_Procedimento

Sentencias.

End Sub

1.8.1 Llamada a un procedimiento

Para llamar un procedimiento desde otro se utiliza la instrucción Call Nombre_Procedimiento.

Sub Programa1

Sentencias.

.

Call Programa2

.

Sentencias

.

End Sub

Las sentencias del procedimiento Programa1 se ejecutan hasta llegar a la línea *Call Programa 2* , entonces se salta al procedimiento Programa2, se ejecutan todas las sentencias de este procedimiento y el programa continúa ejecutándose en el procedimiento Programa1 a partir de la sentencia que sigue a *Call Programa2* .

No es necesario poner **call** para llamar a un procedimiento. Si se pone los parámetros tienen que ir entre paréntesis. Si no se pone **call** los parámetros van sin paréntesis (excepto si solo se pone uno).

1.8.2 Práctica 9. Utilización de procedimientos

Localizar la primera celda vacía utilizando un procedimiento. Solicitar nombre y edad, y rellenar filas hasta que el nombre introducido este vacío

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el módulo Módulo1 si no estuviese abierto.

4. Inserta un nuevo procedimiento: escribe Sub Practica9 dentro del módulo que se ha abierto y pulsa Enter.

Automáticamente aparecerá debajo **End Sub**. Escribir el siguiente código para el procedimiento:

```
Sub practica9()

Dim nombre As String
Dim edad As Integer

Call buscaCasillaVacia

nombre = InputBox("Nombre:", "Introducir texto")
Do While nombre <> ""
edad = InputBox("Edad:", "Introducir número")
ActiveCell.Value = nombre
ActiveCell.Offset(0, 1).Value = edad
nombre = InputBox("Nombre:", "Introducir texto")
ActiveCell.Offset(1, 0).Activate
Loop

Sub buscaCasillaVacia()
Worksheets("Hoja1").Activate
ActiveSheet.Range("A1").Activate
Do While Not IsEmpty(ActiveCell.Value)
ActiveCell.Offset(1, 0).Activate
Loop
End Sub

End Sub
```

Un comentario: La llamada al procedimiento buscaCasillaVacia rompe la secuencia del programa, que en ese punto ejecuta las instrucciones del procedimiento, y devuelve el control a la siguiente instrucción (nombre □).

5. Ejecuta el procedimiento y comprueba los resultados.

1.8.3 Ejercicios

1. Localiza la primera celda vacía. Solicita nombre y edad en un procedimiento, y rellena filas preguntando en cada iteración si se desea seguir.

2. Igual que el anterior pero si la edad es mayor que 18, solicita el DNI y lo introduce en color azul en la celda contigua a la edad, realizando un procedimiento para poner en color azul la celda activa.

1.9 Creación y utilización de Funciones

Tu ya conoces lo que es una función. Las has estado utilizando de manera mas o menos consciente. En tus hojas de cálculo has usado funciones matemáticas (p.e. **SUMA()**) o funciones de fecha (p.e.
HOY()
), etc.

Excel te ofrece la posibilidad de crear tus propias funciones, a las que puedes llamar desde cualquier celda.

Una función es lo mismo que un procedimiento (tal y como ya los has visto hasta ahora), con la única diferencia, de que la función devuelve un valor al procedimiento o función desde la que lo has llamado. Este valor es el resultado de aplicar la función.

Para que la función realice su trabajo, le tienes que indicar con qué valores o elementos va a trabajar. Éstos son los argumentos o parámetros de la función.

Sintaxis:

***Function** nombre [(lista_argumentos)] [**As** tipo]*

End Function

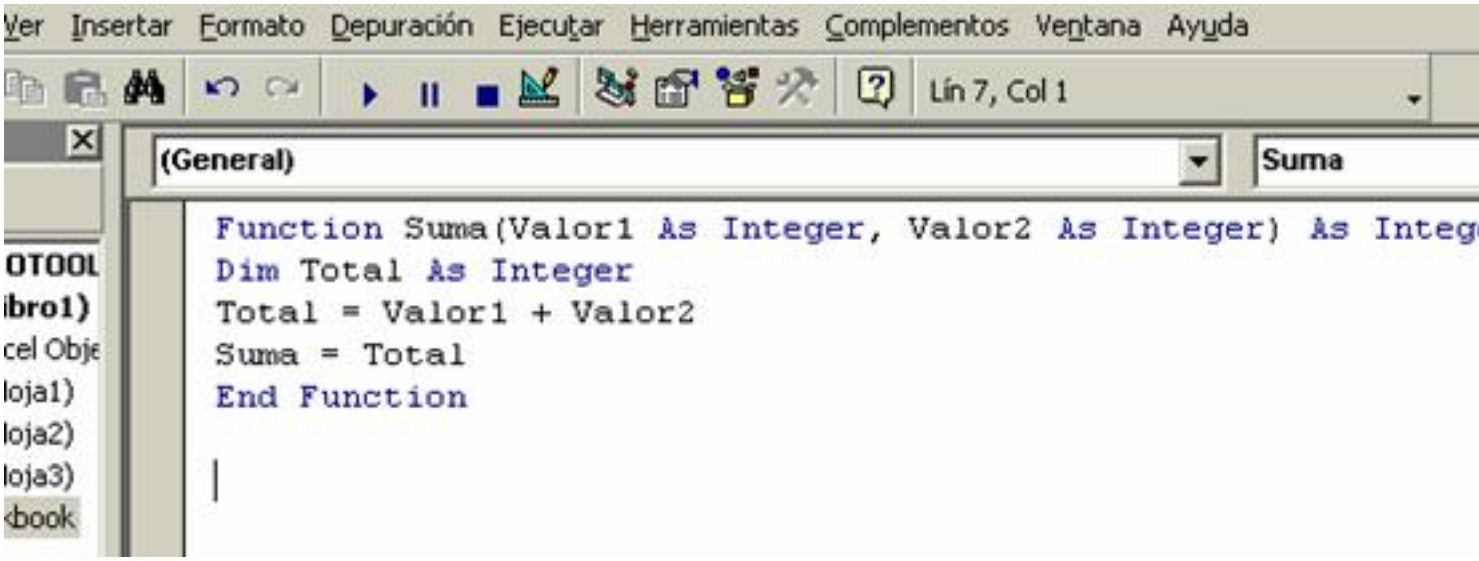
Mas fácil con un ejemplo:

Crea una función que devuelve la suma de dos valores que se le pasan como parámetros. Como las funciones devuelven un valor hay que definir de qué tipo de valor es.

En este caso numérico entero (**As integer**).

Dentro de las instrucciones de una función tiene que existir una que asigne un valor al nombre de la función. En este caso va a ser **Suma = Total.**

Observa:



1.9.1 Práctica 10. Utilización de funciones

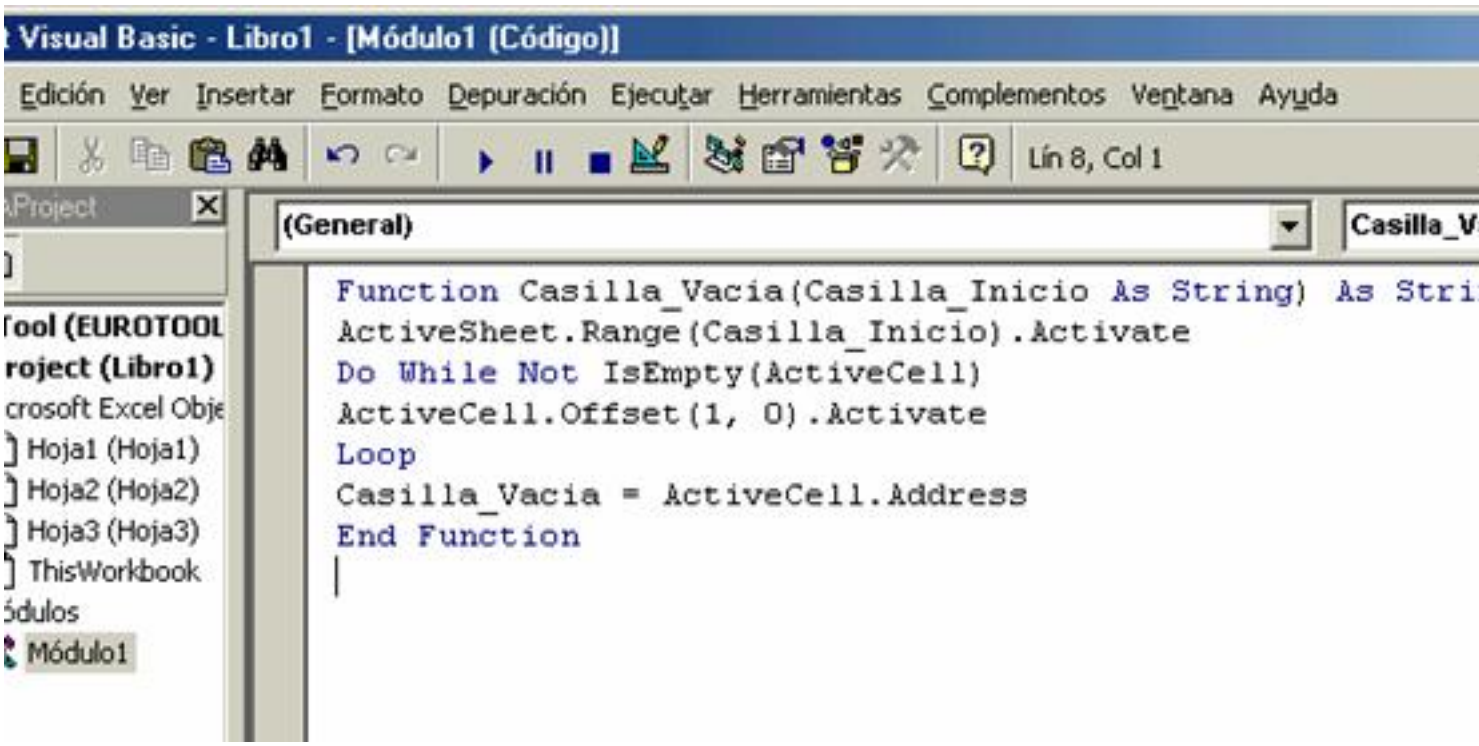
Devuelve la primera celda vacía a partir de una celda que se pasa como parámetro

1. Abre Excel y el libro Ejercicios de Programación.

2. Selecciona Herramientas → Macro → Editor de Visual Basic en la barra de menús de Excel.

3. Abre el Módulo1 si no estuviese abierto.

4. Inserta el código que verás en la página siguiente:



Un comentario: Has pasado como parámetro la casilla inicial. Utilizas la propiedad **Address** que te devuelve la referencia de la celda activa.

5. Llama a la función desde un procedimiento y comprueba los resultados.

1.9.2 Ejercicios

1. Crea una función que devuelva la celda que contenga un valor a partir de una casilla inicial.

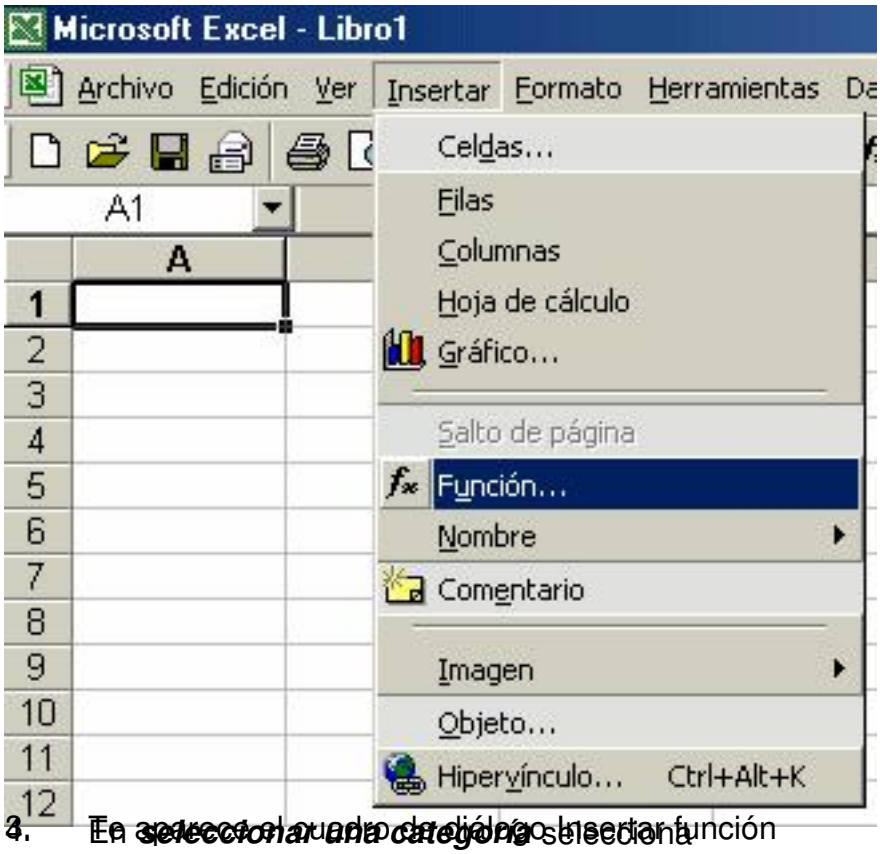
2. Haz un procedimiento que llame a la función anterior y ponga la celda encontrada en color rojo.

1.10 Funciones definidas por el usuario desde el asistente de funciones

Puedes crear funciones tuyas desde el asistente, para utilizarlas posteriormente. Para ello deberás:

1. Abrir Excel y el libro Ejercicios de Programación.

2. Seleccionar Insertar → Función en la barra de menús de Excel.



4. En ***seleccionar una categoría*** selecciona

Te aparece el cuadro de diálogo Insertar función