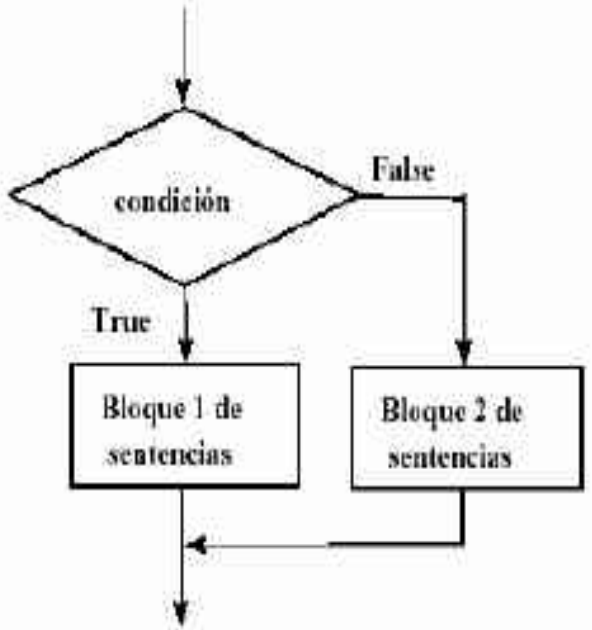
Pseudocódigo	Diagrama de flujo del programa

SI <condicion> ENTONCES
 	
//Bloque para verdadero
Sentencias 1
SINO
//Bloque para falso
Sentencias 2
 	
FIN
	

1º BACH TICO – IES Julio Verne

Tema 5 – Metodología de programación

¿Qué es programar?

Las máquinas internamente trabajan con tensiones (5V, 0V) que se representan por códigos binarios (0 y 1), pero como te puedes imaginar, es imposible hacer programas así, ya que nuestro cerebro "piensa" usando 10 números, 28 letras y todas sus combinaciones

Por eso, desde que existe la informática, el hombre ha creado diferentes lenguajes de programación (cada uno de ellos pensado para algo específico).Por ejemplo, Visual Basic, C, HTML, JAVA, Robolab, etc. Entonces, ¿qué es programar?

Programar es hacer un programa (ya veremos lo qué es) que traduzca lo que yo pienso a código máquina ("0" y "1"), que es lo que el PC entiende y puede ejecutar

Para ello, los programas utilizan instrucciones o comandos que representan las diferentes acciones lógicas de nuestro cerebro y que luego traduciremos (compilaremos) a código máquina

¿Qué es programar?: Lenguaje/Entorno de programación y compilador

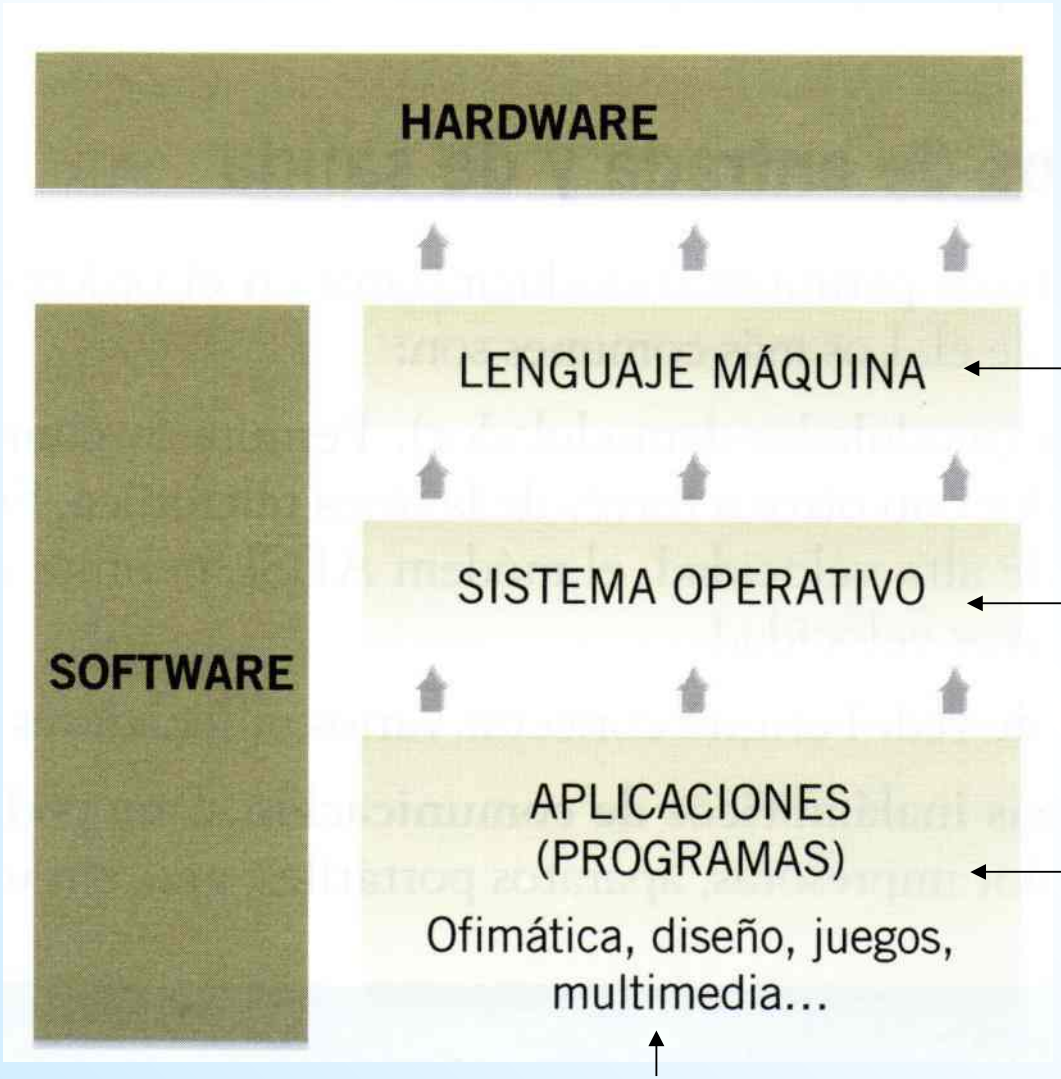
Lenguaje de programación: Es como un idioma inventado con una lógica y una serie de palabras específicas que combinadas entre sí forman un programa y representan lo que la persona ha pensado y quiere resolver. Estas palabras específicas se llaman palabras reservadas y tienen un significado claro y conciso (por ejemplo, en Visual Basic "MsgBox", que saca una ventana con un texto en pantalla).

Los programas se escriben dentro de editores de programación, dentro de los llamados entornos de programación.

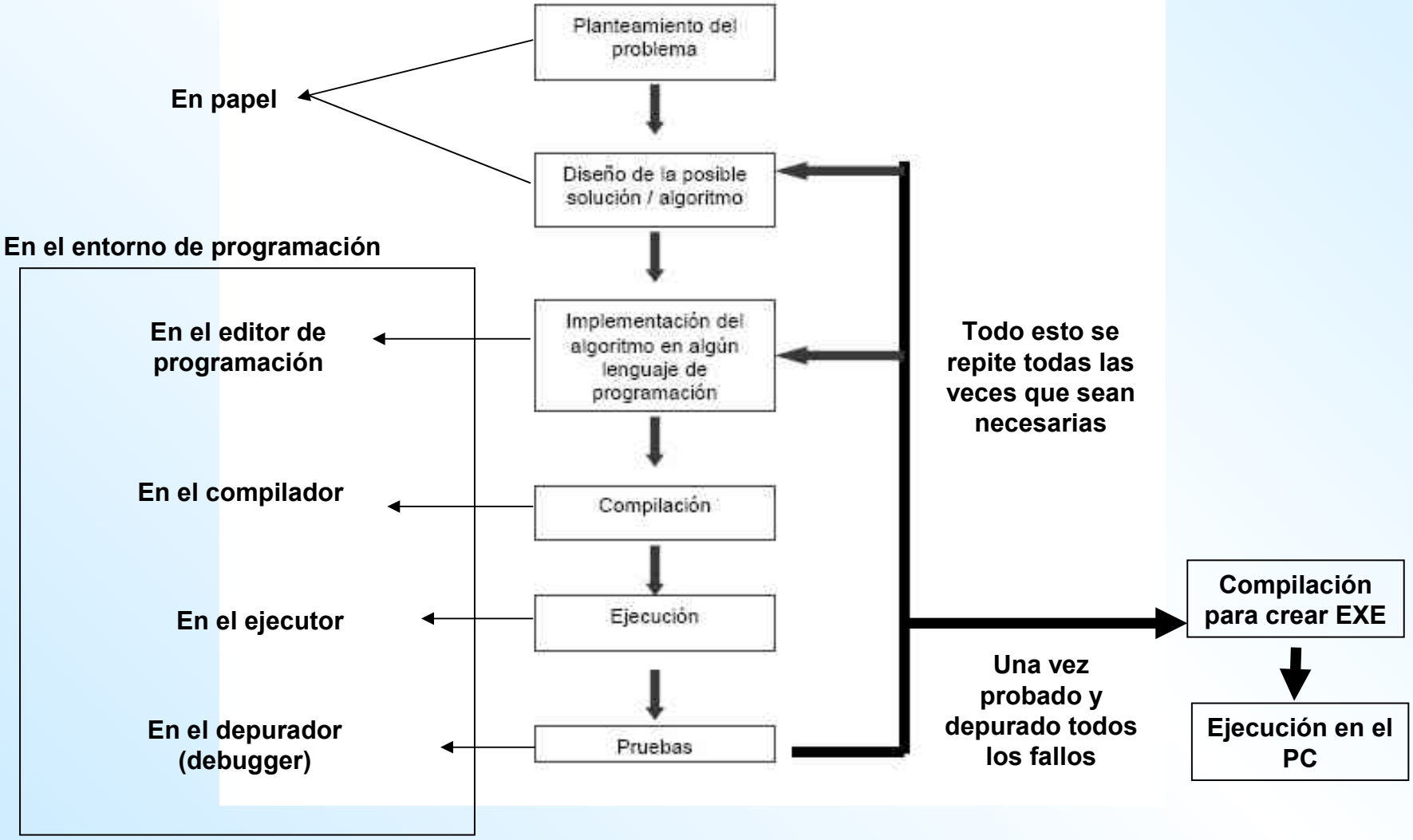
Compilador: Es la parte interna del entorno de programación que se encarga de compilar (traducir) los comandos del lenguaje a código máquina. Del compilador, yo no me preocupo, es interno y simplemente lo uso. Ya viene integrado en los entornos de programación del lenguaje en cuestión.

Importante es saber que un mismo lenguaje puede tener varios entornos de programación con diferentes compiladores, cada uno de ellos para aplicarse a un HW y/o SW determinado, por ejemplo, en Visual Basic, utilizamos el entorno de programación Visual Studio que tiene compiladores para Microsoft Windows.

¿Qué es programar?. Resumen gráfico



Fases de la programación



Programas: Tipos

EL HW del PC lo que ejecuta es un programa compilado.

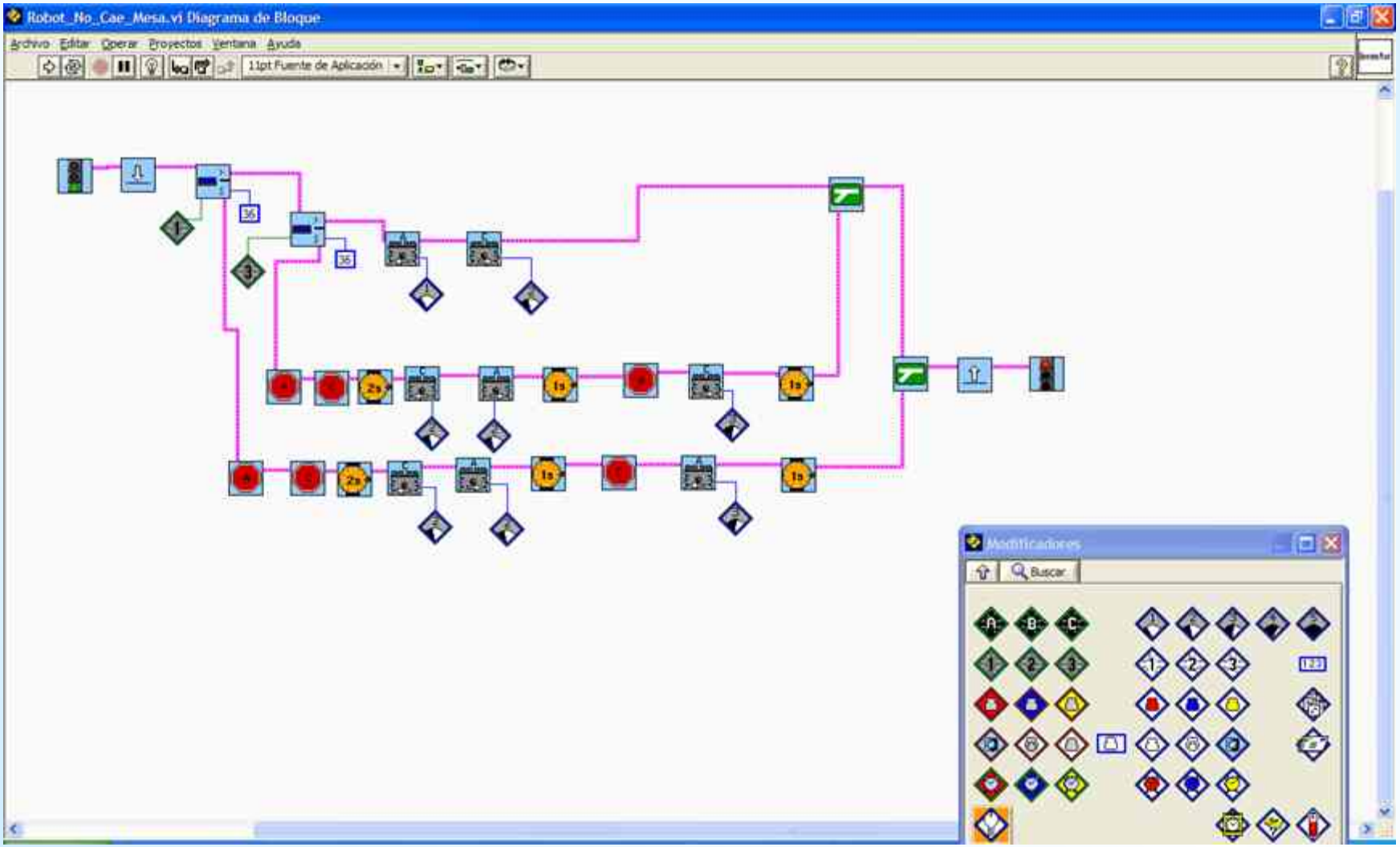
Un programa es un conjunto de instrucciones o sentencias (formadas por operandos (variables y constantes) y operadores) que básicamente lo que hace es consultar valores externos (teclado, ratón, etc.) o internos (temporizadores, contadores, estados de variables, etc.) y, en función de su valor, dar órdenes para hacer algo (calcular un valor, sacar valores por pantalla, etc.).

Los programas se realizan hoy en día dentro de los llamados entornos de programación. Esto proporcionan una serie de herramientas para que podamos escribirlos, comprobarlos, compilarlos y transferirlos, es decir, hacer todo lo que necesito.

Hay muchos entornos de programación, pero básicamente hay 3 tipos:

- **Gráficos: Todo el programa se hace con el ratón, conectando las instrucciones que están disponibles en forma de cajas.**
- **No gráficos: Todo el programa se hace escribiendo comandos o instrucciones una detrás del otro, en forma de lista.**
- **Mixtos: Todo el programa se hace escribiendo comandos o instrucciones una detrás del otro, en forma de lista.**

Ejemplo de programa gráfico: ROBOLAB para LEGO



Ejemplo de programa no gráfico: WinLOGO para ENCONOR

TEMA_2.LOG - Bloc de notas

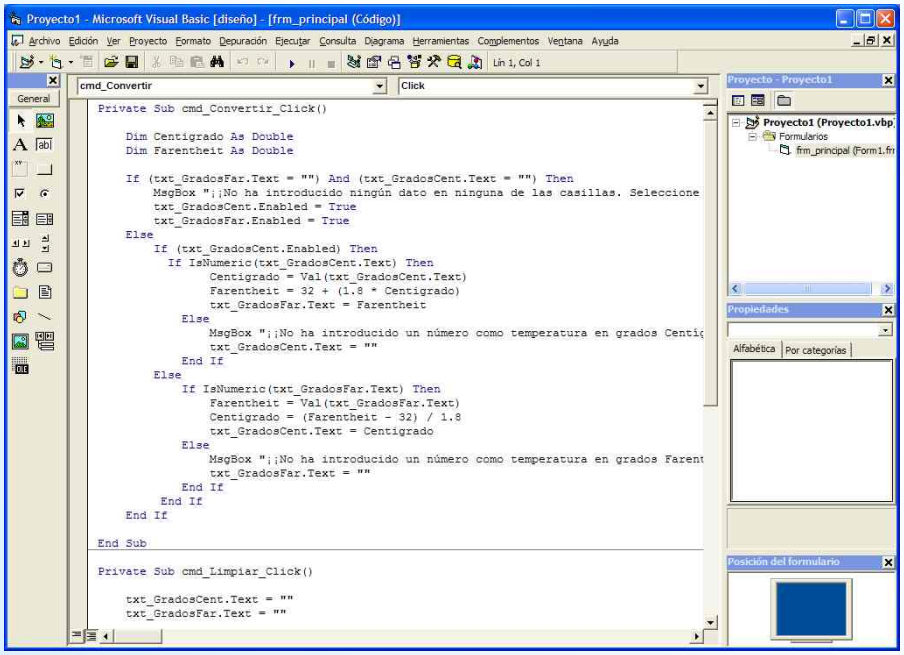
Archivo Edición Formato Ver Ayuda

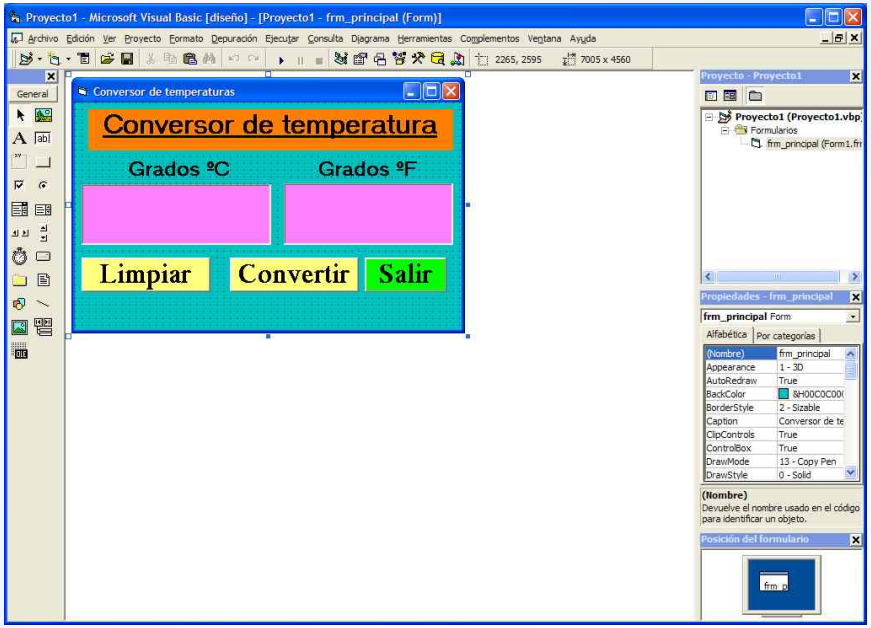
```
PARA ALEATORIO
BT
HAZ "X AZAR 50
REPITE 10 [ ESCRIBE :X
      HAZ "X :X+1]
FIN

PARA CONDICIONAL
BT
ESCRIBE [NUMERO GENERADO:]
HAZ "X AZAR 100
ESCRIBE :X
PRUEBA :X<50
SICIERTO [ESCRIBE [EL NUMERO GENERADO ES MENOR DE 50]]
SIFALSO [ESCRIBE [EL NUMERO GENERADO ES IGUAL O MAYOR DE 50]]
FIN

PARA CONDICIONAL2
BT
ESCRIBE [NUMERO GENERADO:]
HAZ "Y AZAR 100
ESCRIBE :Y
ESCRIBE [LA TABLA DEL 3 DEL NUMERO GENERADO ES:]
HAZ "CONTADOR 0
MIENTRAS [:CONTADOR <:Y+1] [ESCRIBE :CONTADOR
                                    ESCRIBE [RESULTADO:]
                                    ESCRIBE :CONTADOR * 2
                                    ESCRIBE [] ; Esto es un comentario. Deja una línea en blanco
                                    HAZ "CONTADOR :CONTADOR+1]
FIN
```

Ejemplo de programa mixto: VISUAL BASIC para Windows





Características de un buen programa

Correcto: Que no tenga errores de sintaxis (errores de escritura)

Legible y estructurado: Que esté bien ordenado y cualquier otro programador sea capaz de entenderlo

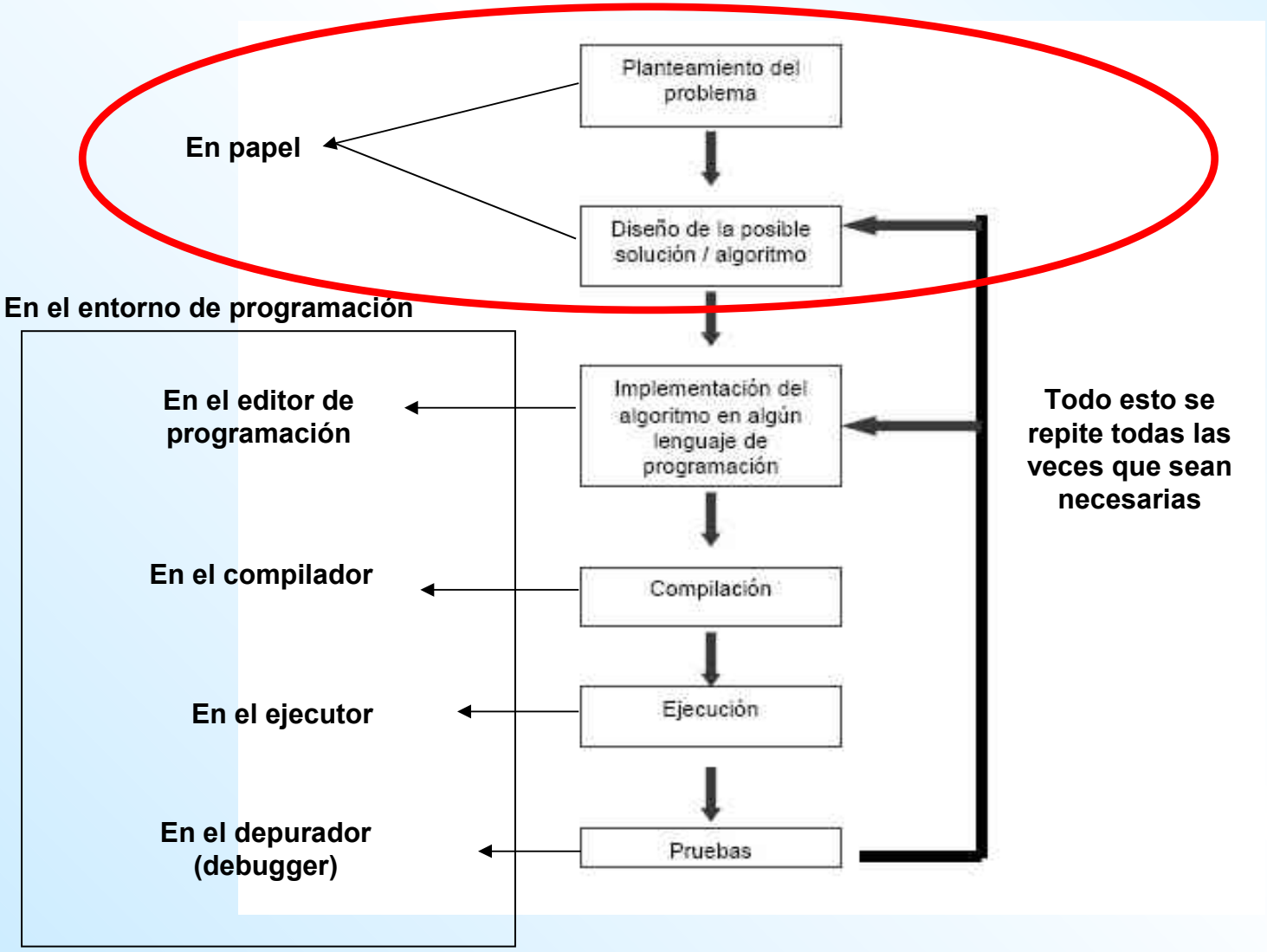
Modificable: Que sea fácilmente modificable o ampliable

Portable: Que se pueda utilizar en el mayor número de HW y sistmas operativos posibles

Modular: Preferiblemente que esté hecho en varios módulos independientes, lo que facilitará que sea más legible, estructurado, modificable y portable

Eficiente: Que aproveche al máximo los recursos del sistema y sea lo más rápido posible

Fase de diseño



Pseudocódigo

Quedamos en que un programa es un conjunto de instrucciones que básicamente lo que hace es consultar valores externos (teclado, ratón, etc.) o internos (temporizadores, contadores, estados de las variables, etc.) y, en función de su valor, dar órdenes para hacer algo (calcular un valor, sacar valores por pantalla, etc.).

Para facilitar las tareas de programación, se suelen utilizar los psudocódigos y/o los diagramas de flujo

Un pseudo código es un código inventado por el programador que expresa con palabras fácilmente reconocibles lo que tengo que hacer, sin llegar a usar las instrucciones reales del lenguaje de programación.

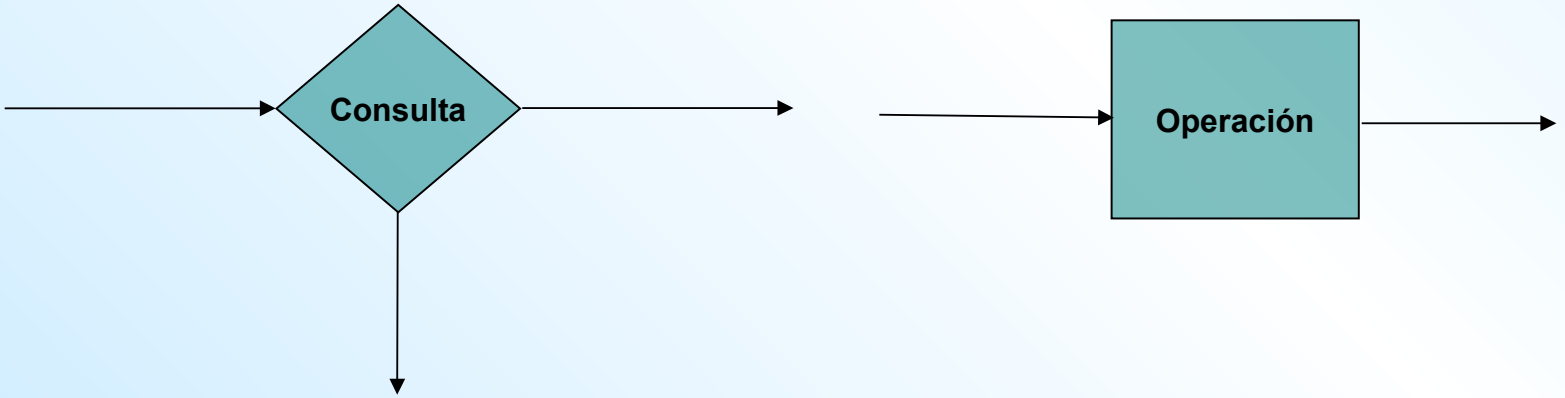
Su utiliza simplemente para entenderme y saber lo que debo hacer. Normalmente se suele usar junto con los diagramas de flujo. Luego vemos un ejemplo.

Pseudocódigo y diagramas de flujo

Los diagramas de flujo son representaciones pictóricas de las instrucciones más comunes de los programas, expresadas en pseudocódigo, que luego nos facilitarán la codificación en el lenguaje usado.

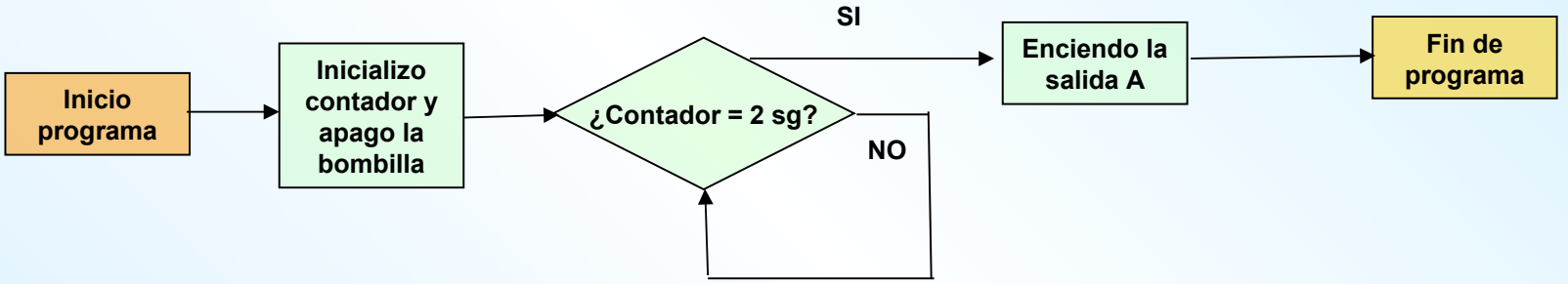
Las diferentes figuras de los diagramas de flujo se unen mediante líneas que se llaman conectores.

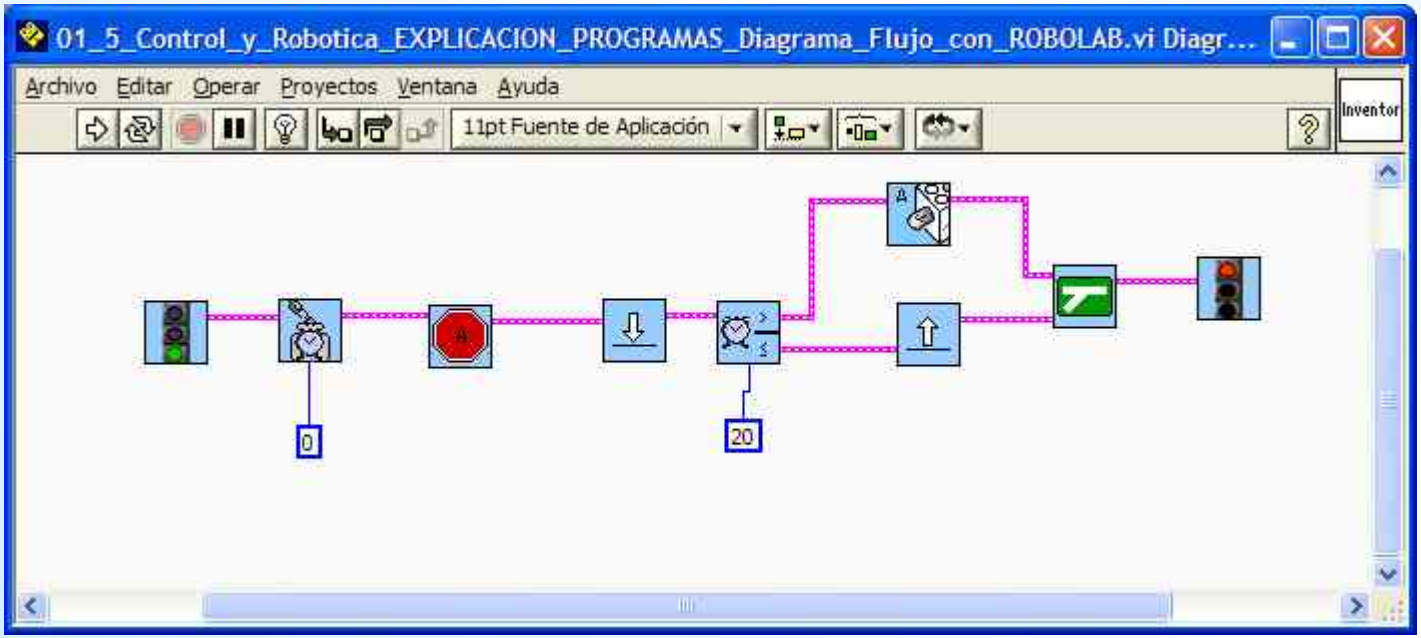
La dirección del proceso se muestra con una flecha



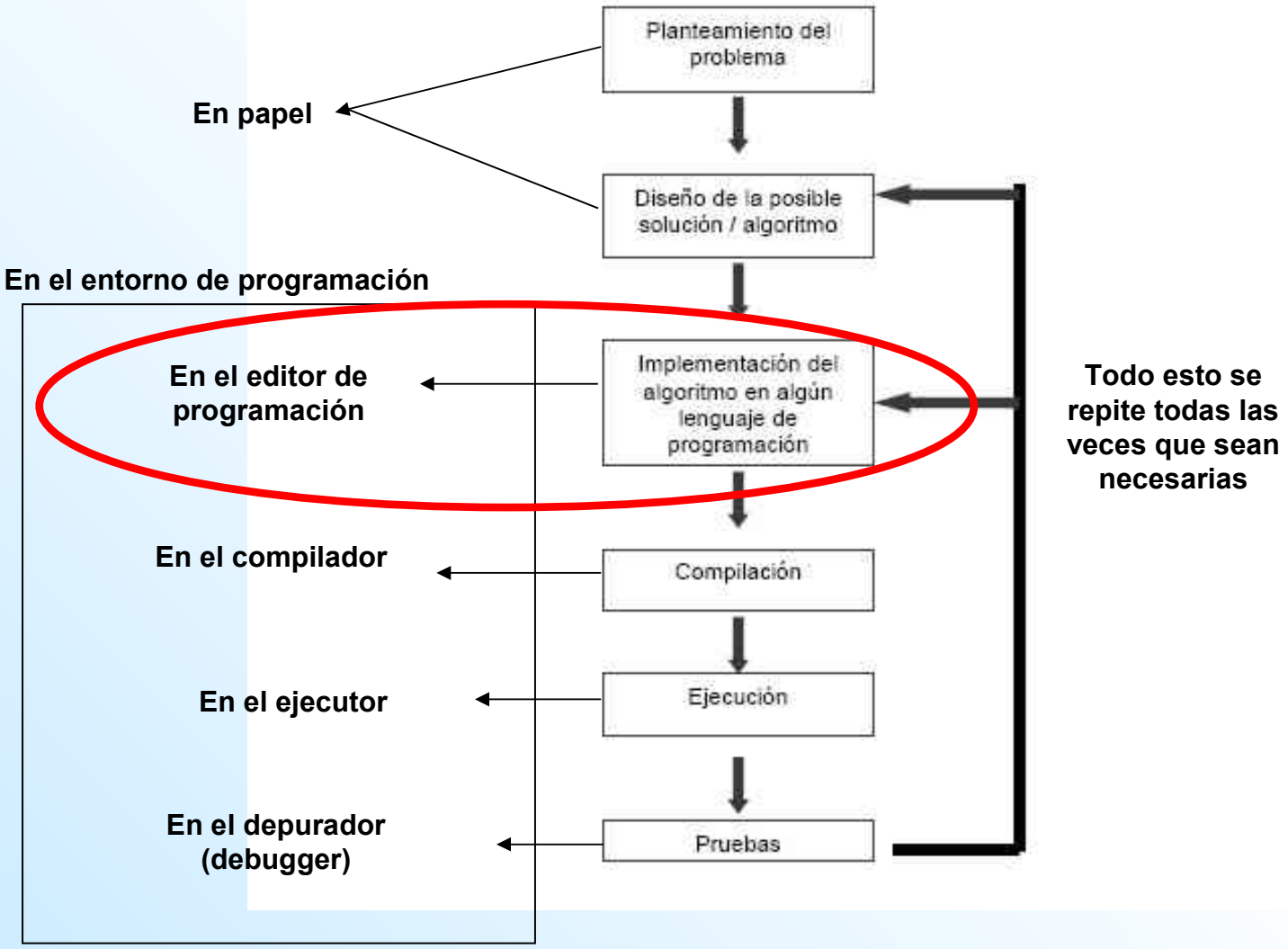
Pseudocódigo: Ejemplo de diagramas de flujo

Programa que espera 2 segundos para encender una bombilla conectada a la salida A del LOGO y luego la deja encendida permanentemente



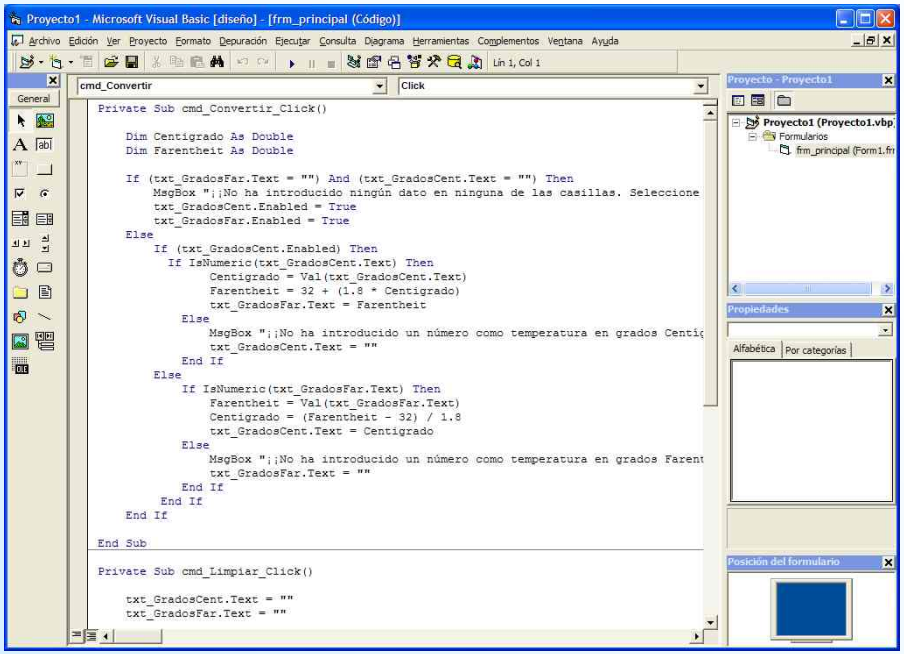


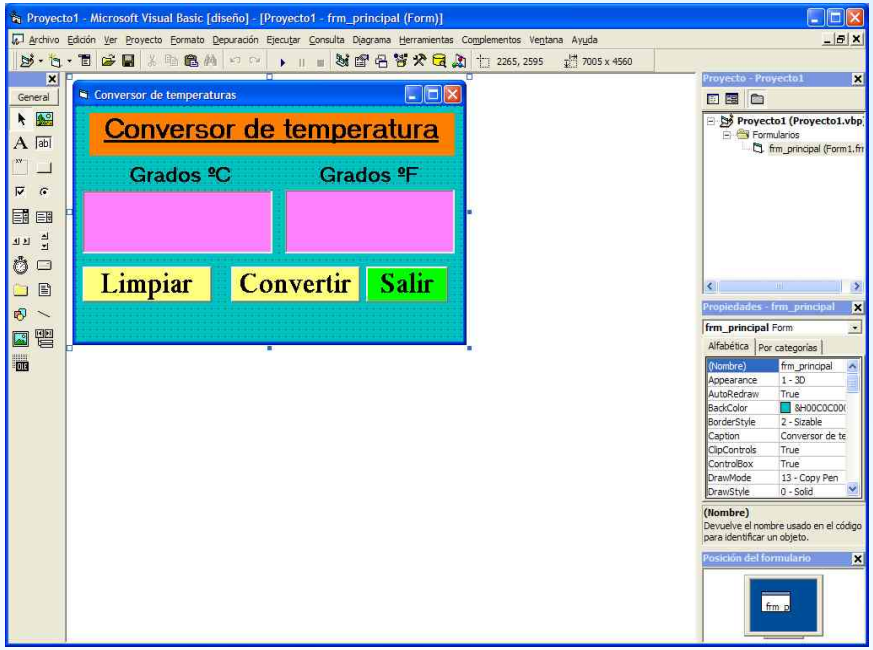
Fase de programación: Diseño gráfico



Objetos gráficos

Casi todos los lenguajes de programación actuales tiene entornos que permiten primero diseñar la parte gráfica del programa, por ejemplo Visual Basic, para luego programar cada objeto con lo que se desee

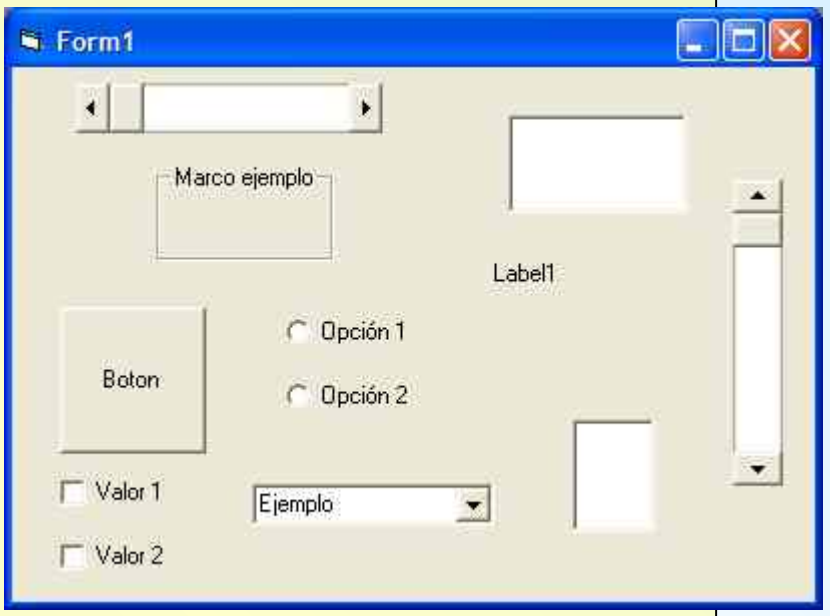




En el ejemplo de arriba, la parte de la derecha es la parte gráfica del programa y la parte de la izquierda es la parte del código de cada objeto gráfico

Objetos gráficos: Tipos

Hay muchos objetos gráficos en Visual Basic, pero los que vas a usar son los siguientes:

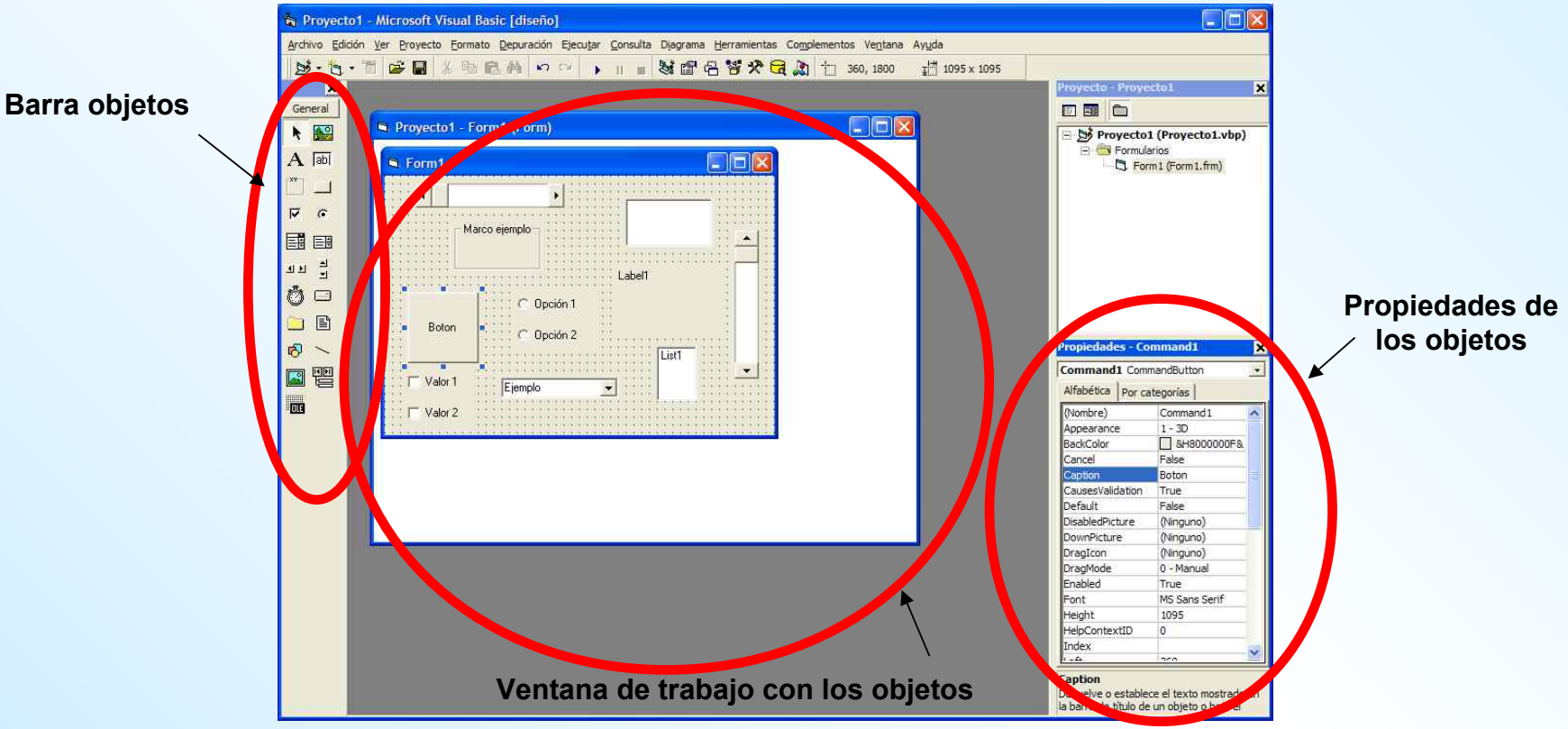


- **Formularios (Form – frm)**
- **Botones (CommandButton – cmd)**
- **Etiquetas (Label – lbl)**
- **Ventanas de texto (TextBox – txt)**
- **Marco (Frame – fra)**
- **Casilla de opción (OptionButton – opt)**
- **Casilla de selección (CheckBox – chk)**
- **Menú desplegable (ComboBox – cmb)**
- **Lista de selección (ListBox - lst)**
- **Barra de desplazamiento horizontal (VScrollBar – vsb)**
- **Barra de desplazamiento vertical (VHScrollBar – vsb)**

Para insertarlos en el programa, simplemente se arrastran con el ratón a la ventana de trabajo y se configuran sus propiedades

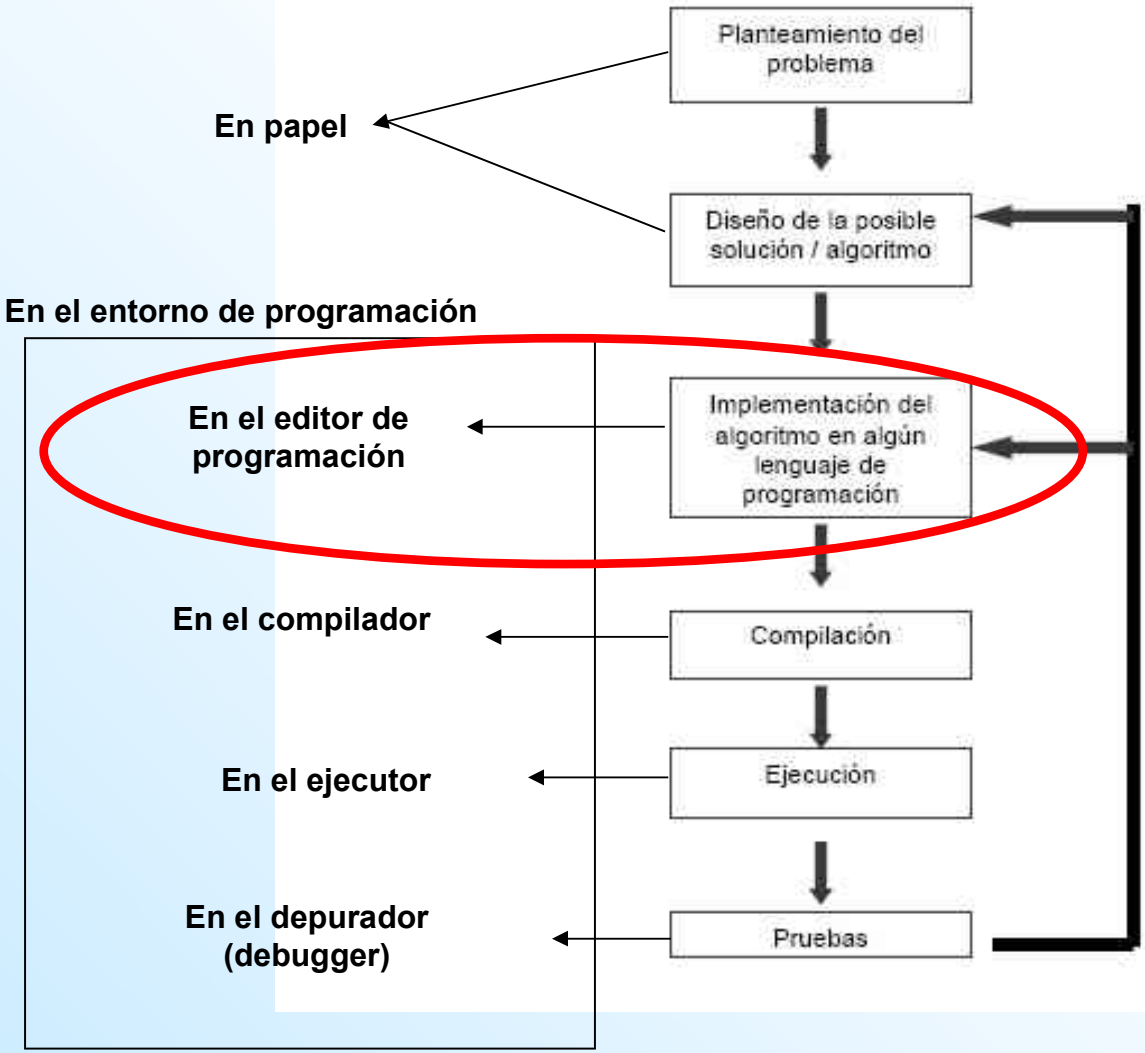
Objetos gráficos: Propiedades – Ajuste inicial

Los objetos gráficos se deben configurar al principio para que tengan el aspecto que queramos. Esto se hace en la ventana de propiedades del Visual Basic (a la derecha), teniendo seleccionado el objeto deseado:



Cada objeto tiene muchas propiedades para ajustar: Color, texto, apariencia, etc. Hay que consultar la ayuda para conocer las de cada uno

Fase de programación: Implementación del código



Una vez que hemos diseñado y ajustado las propiedades iniciales de los objetos gráficos, hay que pasar a programar el código de lo qué se hace con cada objeto al ejecutar el programa.

Pasamos a ver qué necesitamos saber para hacerlo

Datos: Tipos de datos

Debido a que el programa va a manejar diferente tipo de información, tiene que ser capaz de distinguirla, por ejemplo, un número de una letra, un número decimal, de un número entero o un carácter de un conjunto de caracteres. Por eso los lenguajes de programación manejan diferentes tipos de datos, y tengo que decirle al programa qué es es cada uno de ellos al principio para que lo sepa antes de empezar

Cada lenguaje de programación maneja sus propios tipos de datos, aunque todos manejan los siguientes:

- Números binarios -> 0/1 (BOOLEAN)
- Números enteros -> 3, 4, 6 (INTEGER, LONG)
- Números reales (decimales) -> 3.67, 8.90 (SINGLE, DOUBLE)
- Caracteres -> a, b, c (BYTE)
- Cadenas de caracteres -> Esto es una cadena de caracteres (STRING)
- Combinados (ARRAY y STRUCT) -> Varios de los anteriores unidos

Datos: Variables

Normalmente todos estos tipos de datos que vamos a manejar en el programa no van a valer siempre lo mismo, sino que a lo largo de la ejecución del programa van a cambiar de valor. Por tanto, los datos que varían en el programa se llaman variables.

Estas variables pueden llamarse de cualquier forma, pero no pueden llamarse con una palabra reservada.

Estas variables hay que declararlas en algún punto del programa para que las conozca. En VB, se declaran así:

Dim Nombre de la variable As Tipo de la variable

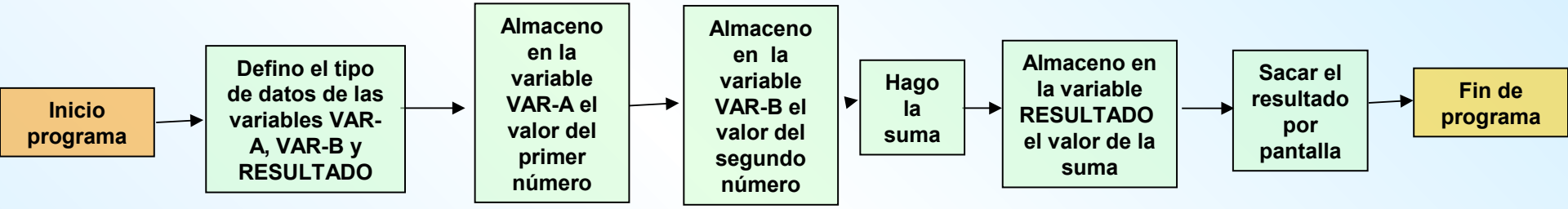
Por ejemplo: Dim VAR-A As Integer

Por ejemplo, imagina que vas a hacer un programa que al pulsar un pida que escribas en el teclado 2 números y al pulsar un botón da el resultado de la misma por pantalla.

El programa tiene primero que guardar el valor de esos números en 2 variables de la memoria, sumarlos y depositar el resultado en otra variable que saque por pantalla, es decir, para hacer este programa necesitamos 3 variables.

Datos: Ejemplo del uso de variables

El ejemplo de la suma con pseudocódigo y digrama de flujo



El ejemplo de la suma en código de VB

```
Dim VAR-A As Integer
Dim VAR-B As Integer

VAR-A = txt_Operando1.Text

VAR-B = txt_Operando2.Text

RESULTADO = VAR-A + VARB

MsgBox “Resultado: ” & RESULTADO
```

Constantes

Hay variables que siempre van a tener el mismo valor a lo largo del programa. Ese tipo de variables se les llama constantes.

Por ejemplo, imagina que vas a hacer un programa en el que vas a usar el número PI constantemente para hacer operaciones matemáticas.

En vez de escribir en cada operación 3,14159…, defines PI = 3,14159 al principio del programa y ya sólo escribes PI en cada operación.

Esto simplifica la escritura del programa y lo hace más claro y sencillo.

Estas constantes también hay que declararlas en algún punto del programa para que las conozca. Normalmente al principio. En VB, se declaran así:

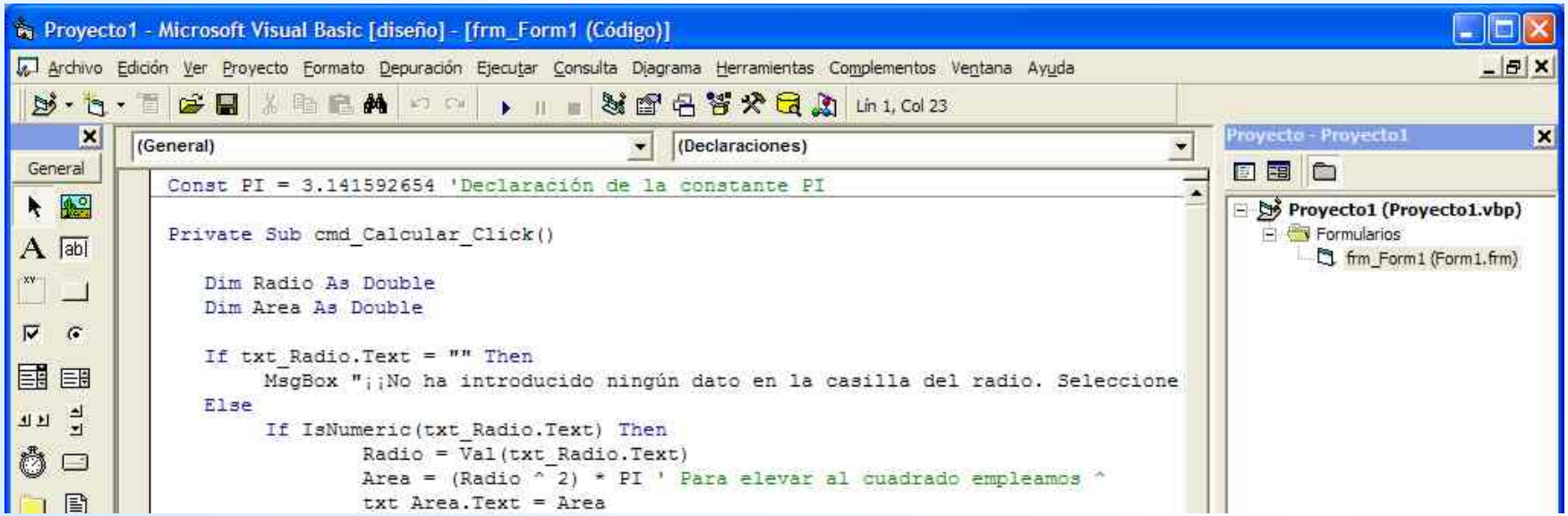
Const Nombre de la variable = Valor

Por ejemplo: Const PI = 3.141592654

Comentarios

Aunque los programas no tienen porqué tener comentarios, es recomendable muchas veces que los tengan, para explicar lo que se va haciendo en cada línea o en grupos de línea

¡¡ Los comentarios no forman parte del programa ni se compilan y normalmente van en otro color para que sepa que no es parte del programa. En Visual Basic, se colocan en verde con un apóstrofe delante (‘)!!



Operadores

Para trabajar y hacer operaciones con las variables o los diferentes operandos en cada línea del programa, se utilizan operadores (que también utilizan palabras reservadas que no se pueden usar para otras cosas).

Hay diferentes tipos de operadores:

- Matemáticos
- Lógicos
- Comparación
- Asignación

Operadores matemáticos

Permiten realizar todo tipo de operaciones matemáticos. Cada lenguaje puede tener algunos suyos propios, pero todo ellos tienen siempre:

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)

Por ejemplo, si queremos sumar en Visual Basic el valor de las variables VAR_A y VAR_B, habrá que escribir:

VAR_A + VAR_B

Operadores lógicos

Permiten realizar operaciones del álgebra de bool (operaciones lógicas). Cada lenguaje podrá tener algunos más o menos, pero todo ellos tienen siempre:

- **INVERSOR (Not)**
- **Y (And)**
- **O (Or)**

INVERSOR

A	S
0	1
1	0

AND

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Por ejemplo, si se quiere hacer en Visual Basic la operación AND de las variables VAR_A y VAR_B, habrá que escribir:

VAR_A And VAR_B

Operadores de comparación

Permiten realizar operaciones de comparación entre variables o expresiones. Cada lenguaje puede tener algunos más o menos, pero todo ellos tienen siempre:

- Igual que (=)
- Distinto de (<>)
- Mayor que (>)
- Menor que (<)
- Mayor o igual que (>=)
- Menor o igual que (<=)

Por ejemplo, si queremos comparar respecto a superioridad en Visual Basic el valor de las variables VAR_A y VAR_B, habrá que escribir:

VAR_A > VAR_B

Operador de asignación

Este operador sirve para asignar a una variable el resultado de cualquier otra operación con operandos o cualquier valor constante. En todos los lenguajes de programación. Este operador es el signo de igual (=):

¡¡ IMPORTANTE: Una expresión de un programa (una línea o sentencia), siempre se lee de derecha a izquierda, por tanto las operaciones se hacen de derecha a izquierdA !!

Por ejemplo, vamos a asignar todas las operaciones que hemos hecho en Visual Basic en las transparencias anteriores a diferentes variables:

Resultado_1 = VAR_A + VAR_B

Resultado_2 = VAR-A And VAR_B

Resultado_3 = VAR_A > VAR_B

Por ejemplo, si queremos asignar un valor fijo a la variable VAR-C, pondremos:

- VAR_C = 5
- VAR_C = “pepe”

Sentencias

Cada una de las líneas de código se llama sentencia.

En realidad, hasta ahora sólo hemos visto un tipo de sentencia, la que se realiza con el operador de asignación, el resto de operadores que hemos visto sólo nos permitían hacer diferentes tipos de operaciones.

El programa lo que hace es ejecutar las sentencias (líneas de código) por orden, según vayan apareciendo.

Por ejemplo, un programa completo que realiza la suna de las 2 variables podría ser el siguiente:

```
VAR_A = 5

VAR_B = 6

RES = VAR_A + VAR_B
```

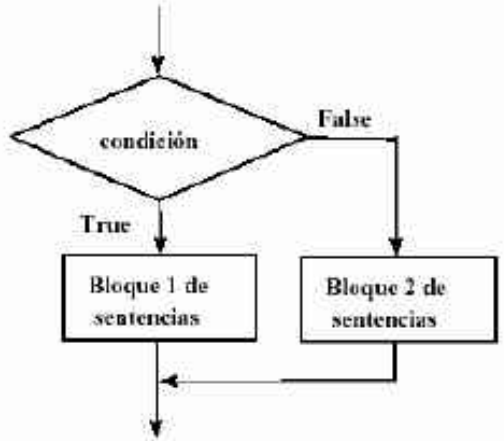
Pero esto no es suficiente para la mayoría de los programas. Necesitamos tener sentencias que me permitan decidir si hago una parte del programa u otra, o sentencias que permitan repetir varias sentencias para por ejemplo hacer un contador.

Por eso, todos los lenguajes de programación tienen sentencias que permiten hacer esto, Estas sentencias se llaman sentencias de control del flujo del programa

Secuencia condicional

Permite seleccionar el camino que va a seguir el programa, en función de que se cumpla o no una o varias condiciones.

Pseudocódigo	Diagrama de flujo del programa

SI <condicion> ENTONCES
 	
//Bloque para verdadero
Sentencias 1
SINO
//Bloque para falso
Sentencias 2
 	
FIN
	

EJEMPLO en VISUAL BASIC

```
If Numero = 0 Then
        txt_Sol.Text = "0“
Else
        txt_Sol.Text = Sqr (Numero)
End If
```

Secuencia de condición múltiple

Permite elegir entre varias sentencias en función de una o varias condiciones.

```
SEGÚN <expresion> HACER
        SEA <valor1>
                <Sentencias para valor1>
        SEA <valor2>
                <Sentencias para valor2>
        ...

        EN OTRO CASO
                <Sentencias>
FIN
```

EJEMPLO en VISUAL BASIC

```
Select Case Numero
      Case 1
            txt_NIF.Text = “SI"
      Case 2
            txt_NIF.Text = “NO"
      Case Else
            txt_NIF.Text = “IMPOSIBLE”
End Select
```

Secuencia repetitiva o bucle

Permite repetir una parte del programa hasta o mientras se cumpla una condición.

MIENTRAS <condicion> Sentencias FIN	REPETIR Sentencias HASTA <condicion>

EJEMPLO en VISUAL BASIC

```
Do While Numero < 0
        txt_Sol.Text = "Imposible"
        Exit Sub
Loop
```

```
Do
        txt_Sol.Text = "Imposible"
        Exit Sub
Loop While Numero < 0
```

Funciones y procedimientos

Muchas veces en el programa, tenemos que hacer muchas veces lo mismo. Para evitar tener que escribir lo mismo muchas veces, las instrucciones o sentencias que se deben realizar de manera conjunta se agrupan en las llamadas funciones o procedimientos.

¡¡ Las hay del sistema y las hay creadas por el usuario !!

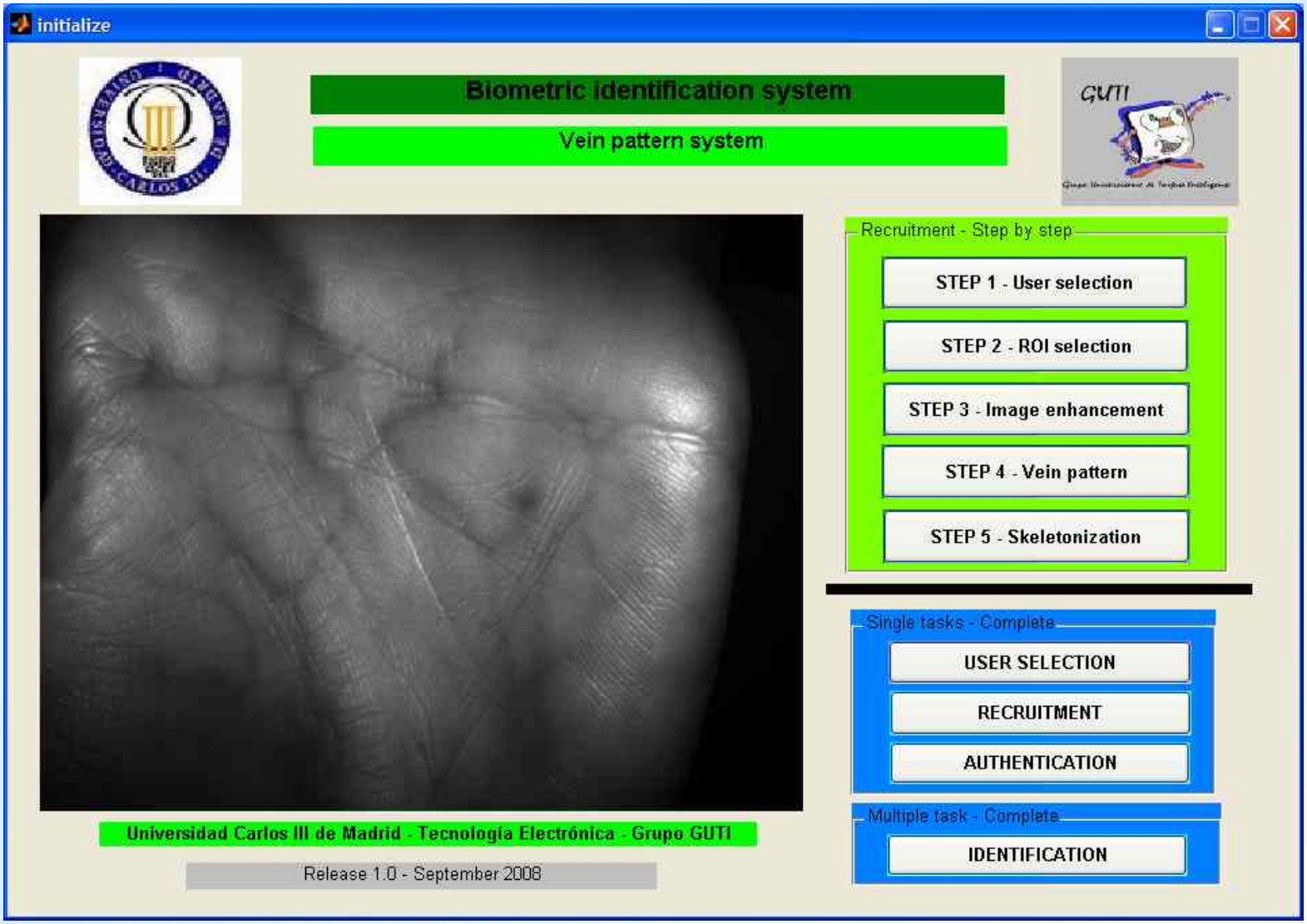
Por tanto, una función o procedimiento no es más que un grupo de instrucciones que agrupamos dentro de un grupo y que podemos llamar desde donde queramos del programa y las veces que seamos

La diferencia entre una función y un procedimiento es que la función se llama y al ejecutarse devuelve uno o varios valores como resultado, mientras que el procedimiento se llama y no devuelve ningún valor al ejecutarse, simplemente hace una serie de acciones

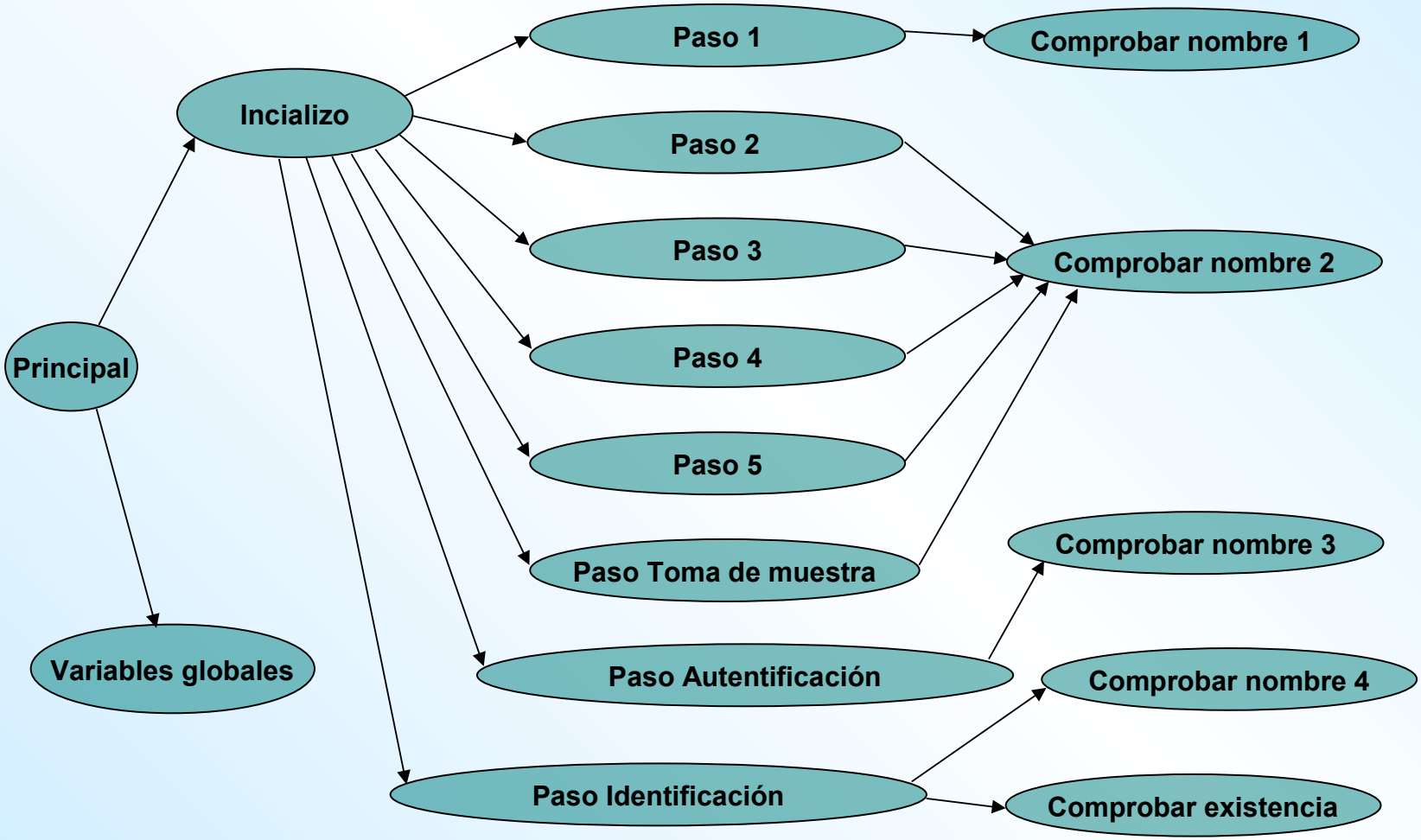
En las funciones y procedimientos se pueden indicar valores/parámetros de entrada, para que los utilice internamente

Para usar una función o procedimientos, hay que declararla previamente en algún sitio del programa con su nombre, parámetros de entrada y/o salida y su código. Si no hago esto, el compilador no va a entenderlo.

Funciones y procedimientos: Programa real



Funciones y procedimientos: Funciones del programa real



Funciones y procedimientos: Ejemplos en Visual Basic

En los programas que vamos a hacer, no vas a crear funciones o procedimientos propios, pero vas a usar algunas de ellas que ya han creado los programadores de Visual Basic y que simplemente las usas como indica la ayuda (no hace faltas que las declares en ningún sitio).

Ejemplo de funciones del sistema en VB que vas a tener que usar:

MsgBox (parámetros) -> Devuelve un mensaje por pantalla

Sqr (número) -> Devuelve la raíz cuadrada del número indicado

Val (variable de texto) -> Devuelve el valor numérico de una variable de texto

IsNumeric (valor de texto) -> Devuelve TRUE si el valor de texto se correponde con un número

QBColor (Color) = Devuelve el número del color indicado

Ejemplo de procedimiento en Visual Basic: Todo lo que se programa para cada objeto gráfico en VB se mete dentro de su procedimiento correspondiente (que crea VB automáticamente). Por ejemplo:

```
Private Sub Nombre del objeto_Acción()
   Sentencias
End Sub
```

Programación de las propiedades de los objetos gráficos

Para utilizar y cambiar las propiedades de los objetos gráficos al ejecutar el programa, hay que programarlas en el procedimiento de cada objeto.

Por ejemplo, si tengo un botón en VB llamado "cmd_Boton 1", se crea automáticamente un procedimiento para programar dentro de él todo lo que necesite al hacer clic sobre él:

```
Private Sub cmd_Boton_1_Click()
   Sentencias que necesite para el click (por ejemplo, cambiar la propiedad de algo)
End Sub
```

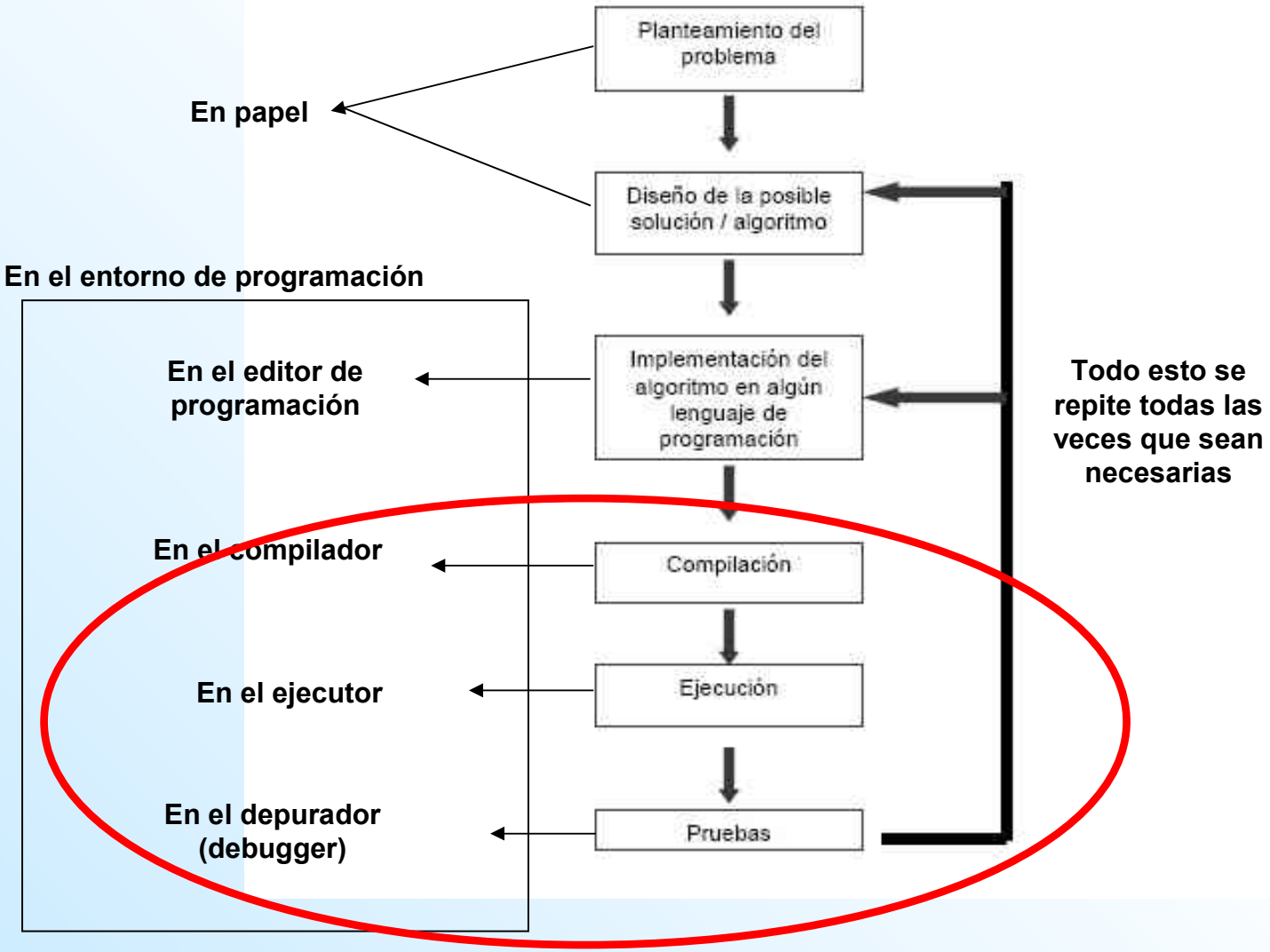
Una propiedad de un objeto cualquiera se programa de la siguiente manera:

Nombre del objeto.Nombre de la propiedad = Valor que quiero

Si ahora, al pulsar el botón "cmd_Boton1" , quiero ajustar la propiedad "Text" de la etiqueta "lbl_Etiqueta_1" para que ponga "Ejemplo", escribo dentro del procedimiento del botón:

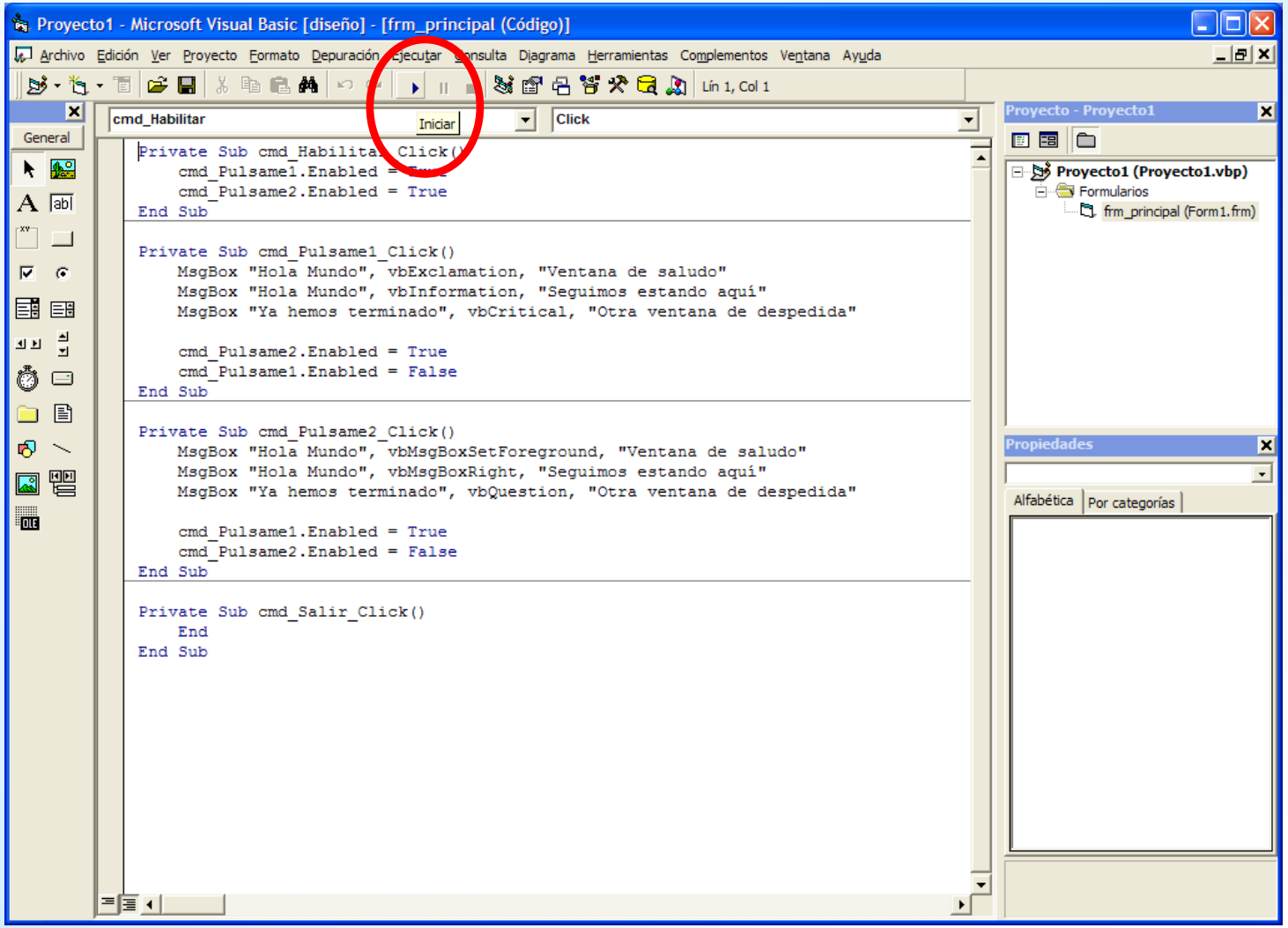
```
Private Sub cmd_Boton_1_Click()
    lbl_Etiqueta1.Caption = "Ejemplo"
End Sub
```

Fase de compilación, ejecución y depuración



Compilación y ejecución en Visual Basic

Hay que pulsar el botón INICIAR.



Pruebas en Visual Basic

Si al ejecutar el programa, no aparece ningún error y todo funciona perfectamente, ya hemos terminado y ahora habrá que generar el archivo ejecutable final (EXE) –> Lo vemos 5 transparencias después

Pero esto NUNCA ocurre, siempre hay fallos que tenemos que corregir. Puede haber 2 tipos de errores:

- **Errores sintácticos**
- **Errores de lógica del programa**

Los primeros son muy fáciles de descubrir, los segundos no tanto, pero en cualquier caso, NO SE PUEDE GENERAR EL ARCHIVO EXE hasta que estos errores estén resueltos.

Pruebas en Visual Basic: Errores sintácticos

Son los que se producen por escribir mal el código en el editor:

1) **No escribir correctamente las palabras reservadas**

* Por ejemplo, se escribe Dimm en lugar de Dim

2) **Utilizar nombres de variables o funciones que no existen**

* Por ejemplo, se declara la variable "contador_1" (Dim contador _1 As integer) y luego se escribe en una línea del programa: contador1 = a + b.
* Por ejemplo, se escribe en una línea del programa: MSgBox "Hola"

3) **Utilizar nombres de objetos gráficos o propiedades de los mismos que no existen o no son posibles**

* Por ejemplo, se inserta el botón llamado "cmd_boton" y luego se escribe:

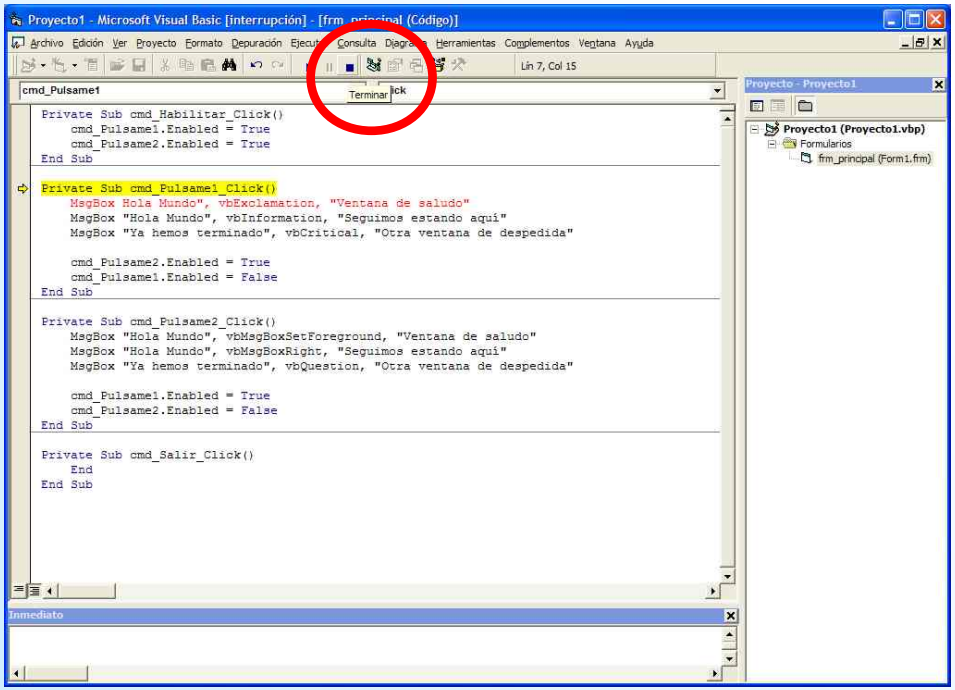
```
Private Sub cmd_Boton_Click()
    lbl_Etiqueta.Caption = "Ejemplo"
End Sub
```

* Por ejemplo, se inserta un label llamado "lbl_Etiqueta" y luego se escribe: Lbl_etiqueta.Text = "pepe", Lbl_etiqueta.Text = 234, Lbl_etiqueta.Text = pepe"

Pruebas en Visual Basic: Errores sintácticos

Los fallos sintácticos son muy fáciles de corregir porque cuando pulsas el botón INICIAR para probar, todo, en cuanto se ejecuta algo que no está bien, el programa para y saca una ventana de error como esta

Al pulsar sobre ACEPTAR, el programa salta a la línea o procedimiento donde está el fallo



Una vez resuelto el error, se pulsa sobre el botón TERMINAR y se vuelve a probar, todas las veces que sean necesarias hasta que no aparezca ningún error sintáctico

Pruebas en Visual Basic: Errores de lógica

Los errores de lógica son bastante difíciles de corregir porque, en este caso, la sintaxis está bien y el programa se ejecuta correctamente, pero no hace lo que yo quiero o lo hace mal

En este caso, tengo que volver a la fase de diseño y corregir lo que sea necesario para que al final lo haga bien

Ejemplo:

Imagina que diseñas un programa para sumar 2 números, declaras bien las variables, escribes todo bien, sacas el resultado por pantalla, pero resulta que tú has metido como operandos de la suma 3 y 5 y el resultado es -2. Hay que volver al diseño y al editor y comprobar línea a línea lo que está mal. ¿Qué puede haber pasado?

En este caso, está muy claro que paree que el diseño es correcto, pero me he equivocado al poner la operación de los operandos en el editor. Seguramente, en lugar de escribir,

Resultado = operando 1 + operando 2

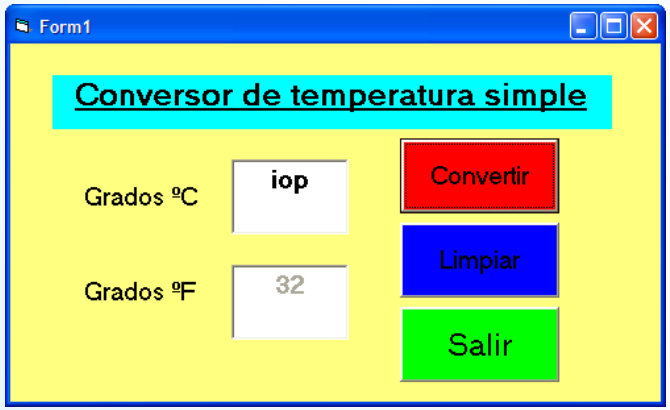
he escrito

Resultado = operando 1 - operando 2

Pruebas en Visual Basic: Errores de lógica

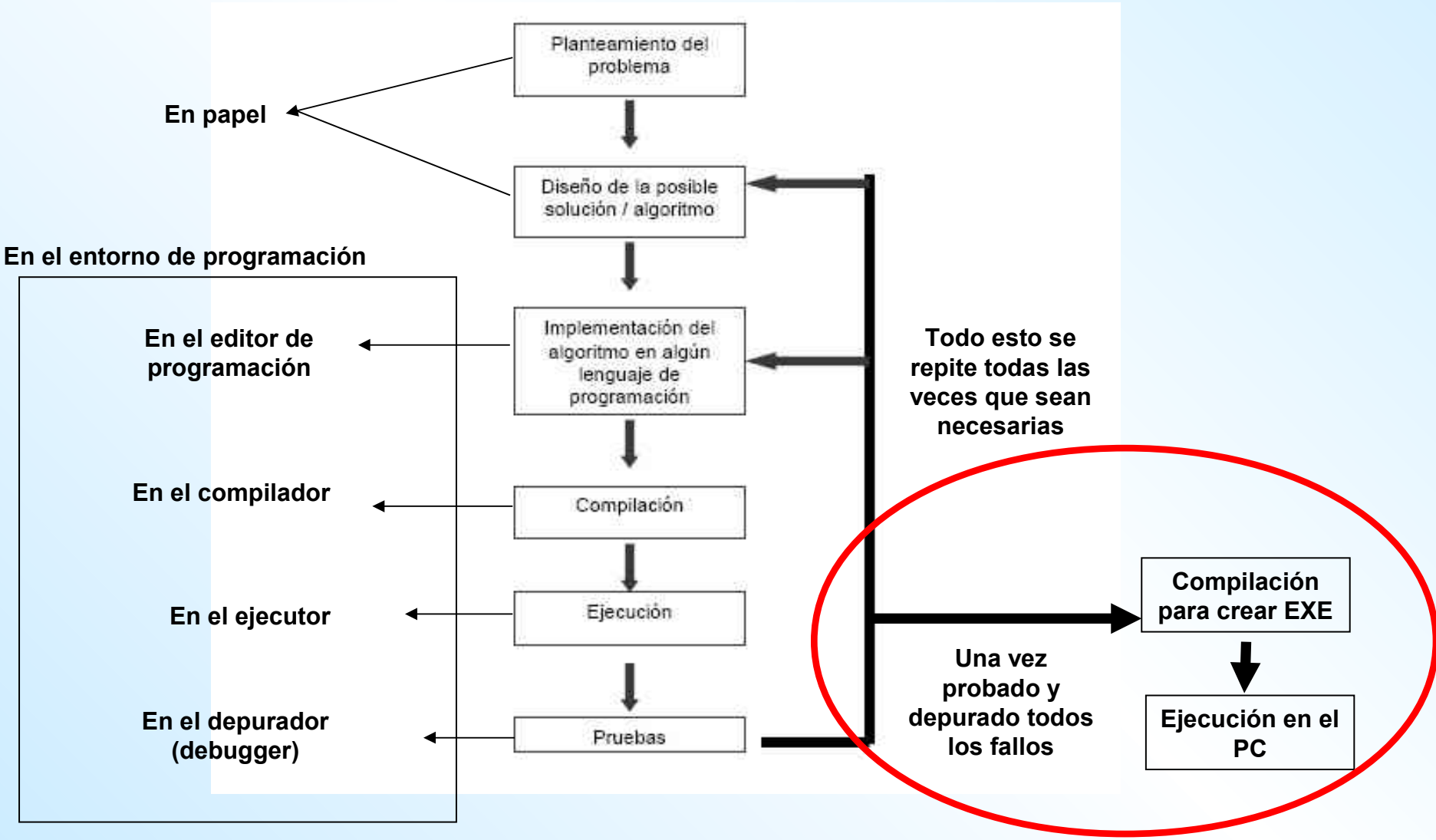
Otro ejemplo más complicado de resolver:

Imagina que diseñas un programa para convertir la temperatura de ºC a ºF y todo el correcto y funciona perfectamente, pero te das cuenta que si también metes letras en el campo de la temperatura, se hace la conversión, lo que es un claro error. Hay que volver al diseño y al editor y comprobar línea a línea lo que está mal. ¿Qué puede haber pasado?



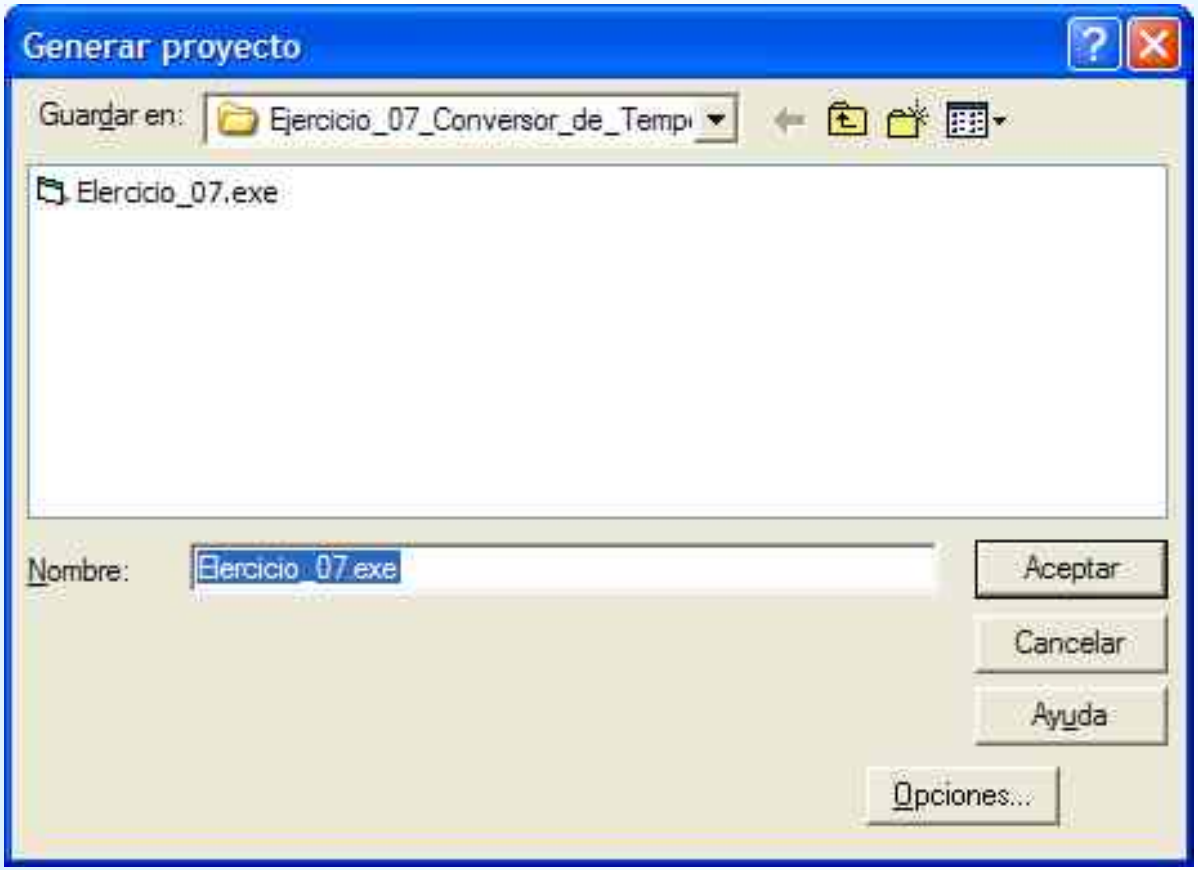
En este caso, está muy claro que el error es en el diseño, luego tengo que volver a pensar cómo hacer el programa, cambiar su diagrama de flujo y modificar lo que necesite en el editor del rograma

Fase de generación del programa EXE



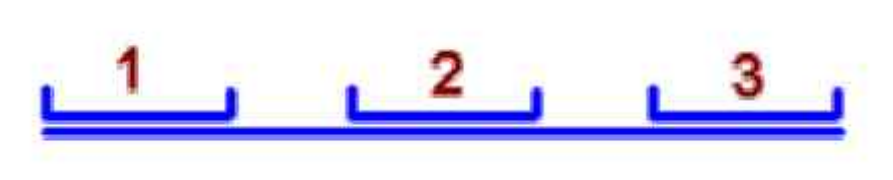
Fase de generación del programa EXE en VB

Hay que seleccionar la opción "Archivo -> Generar Nombre.Exe", se le da el nombre que se quiera y el lugar de archivo en el dicos duro y ya hemos terminado.



Ejercicio 1

Tenemos un robot en una plataforma móvil que se puede mover por tres posiciones (dibujo).



Y encima de una de estas tres posiciones pondremos un bloque (caja) llamada "A". Lo que queremos que realice nuestro Robot es lo siguiente:

El Robot siempre partirá de la posición 1, pero no conocerá donde está situado el bloque y este siempre debe estar en una de las 3 posiciones.

Nosotros deberemos hacer que es Robot se desplace por la cinta mirando si el bloque se encuentra en la nueva posición. Si el bloque está en la posición en la que se encuentra el Robot, este debe cogerlo. Lista de posibles instrucciones a utilizar:

BloqueEncima	Nos devolverá el nombre del bloque que está encima del Robot.
CogerBloque	El Robot cogerá el bloque.
MoverDer	Moverá el Robot a la derecha una posición.

Ejercicio 2

Tenemos otra vez el mismo robot montado en la plataforma móvil pero un poco ampliada ya que esta vez tenemos 5 posiciones.

Nosotros inicialmente podemos encontrar el Robot en una de las 4 posiciones iniciales y un bloque llamado A en una de estas 4 posiciones. (La colocación del robot y del bloque será aleatoria).

El bloque siempre estará situado en alguna posición. Nosotros queremos encontrar ese bloque, esté donde esté y llevarlo hasta la quinta posición.

Lista de posibles instrucciones a utilizar:

BloqueEncima	Nos devolverá el nombre del bloque que está encima del Robot.
CogerBloque	El Robot cogerá el bloque.
MoverDer	Moverá el Robot a la derecha una posición.
MoverIzq	Moverá el Robot a la izquierda una posición.
Posición	Nos devolverá la posición en la que se encuentra el Robot en el momento de hacer la "pregunta".

Ejercicio 3

Tenemos a nuestro robot subido en una plataforma móvil de 10 posiciones. El Robot SIEMPRE partirá de la posición 1.

Lo que queremos es que el robot se desplace por todas las posiciones hasta llegar a la 10, buscando dos bloques, llamados A y B, que podrán estar en cualquiera de las posiciones, exceptuando la última.

Cuando el robot encuentre el bloque A, deberá cogerlo y llevarlo a la primera posición, en cambio si encuentra el bloque B deberá cogerlo y llevarlo a la segunda posición. Antes de dejar el bloque deberás mirar si el lugar está ocupado por otro bloque.

Lista de posibles instrucciones a utilizar:

BloqueEncima	Nos devolverá el nombre del bloque que está encima del Robot.
CogerBloque	El Robot cogerá el bloque.
MoverDer	Moverá el Robot a la derecha una posición.
MoverIzq	Moverá el Robot a la izquierda una posición.
Parar	Detendrá el Robot.
Posición	Nos devolverá la posición en la que se encuentra el Robot en el momento de hacer la "pregunta".
SoltarBloque	El Robot dejará el bloque en la posición actual.