

MACROS EN EXCEL



Julio César Martínez Sánchez

jcms2665@gmail.com

ÍNDICE GENERAL

Introducción	1
1. Programación	2
1.1. Programación Estructurada	3
1.2. Programación Orientada a Objetos	5
2. Elementos básicos de la programación	7
2.1. Funciones	7
2.2. Argumentos o variables	8
2.3. Operadores lógicos	9
2.4. Bucles de programación	9
2.4.1. Bucle FOR	9
2.4.2. Bucle WHILE	10
2.4.3. Bucle REPETIR	10
2.4.4. Selección IF	11
3. Pseudocódigo	11
3.1. Implementación de Variables	12
3.2. Implementación de Código	12
4. Visual Basic	16
4.1. Aspectos Generales	16
4.2. Aplicaciones	20
Bibliografía	24

Índice de esquemas

Esquema 1 Implementación de un programa	2
Esquema 2 Programación estructurada	4
Esquema 3 Programación orientada a objetos	6
Esquema 4 Descripción de objetos y pseudocódigo	14

Introducción

Visual Basic for Applications es un lenguaje de programación que se incluye en las aplicaciones de Microsoft Office. Este vínculo permite automatizar procesos y desarrollar aplicaciones que se pueden ejecutar en Word, Excel, PowerPoint y Access. Dichas aplicaciones ayudan para agilizar y facilitar el desarrollo de las tareas cotidianas.

El objetivo de este documento es introducir a los asistentes a la lógica de la programación orientada a objetos, que es el fundamento teórico para el desarrollo de macros. Dicha introducción de desarrolla en cuatro etapas: en la primera se presentan las nociones básicas de la programación, en particular se enfatiza en aquella que es orientada a objetos; en la segunda se detallan los elementos que intervienen en la programación; en la tercera se muestra el desarrollo de programas en pseudocódigo, que es el paso previo a una implementación real; finalmente, en la cuarta se desarrollan las aplicaciones en Visual Basic.

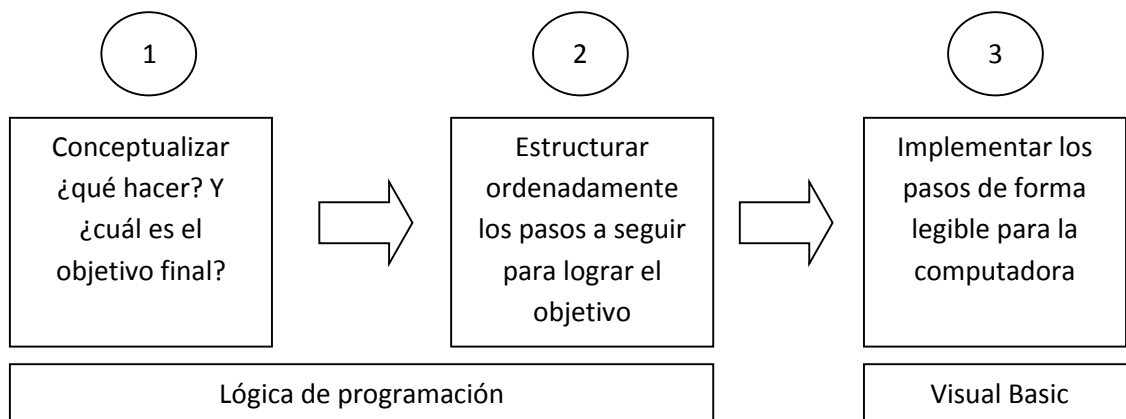
1. Programación

La *programación* es un proceso mediante el cual se busca modelar un comportamiento real. Para lograrlo se requiere de un conjunto de instrucciones (o comandos) que se ejecutan de manera ordenada para desarrollar una tarea en particular; a dicho conjunto se le denomina *programa*. Éste debe de seguir ciertas reglas, las cuales son definidas por el *lenguaje de programación* en donde se esté implementando.

La diferencia entre programa y lenguaje de programación es que el primero es la parte conceptual que determina ¿qué hacer?, mientras que el segundo hace la implementación con base en ciertas reglas preestablecidas¹.

El proceso para desarrollar un programa inicia definiendo ¿qué se quiere realizar? y ¿cuál es el resultado esperado?; el siguiente paso es definir una secuencia ordenada de los pasos que se requiere para llegar a la meta planteada; el último paso es implementar dicha secuencia en una forma que sea legible para la computadora, es decir, hacer uso de los lenguajes de programación establecidos para poder comunicarse con la máquina. Conceptualmente, los pasos a seguir se muestran en el esquema 1.

Esquema 1 Implementación de un programa



¹ Por ejemplo, supongamos que en una convención internacional y se pide a los asistentes que para la siguiente reunión todos lleven preparado un taco. Para que esto sea posible, se requiere explicar paso a paso y de forma ordenada ¿qué hacer?, desde preparar una tortilla hasta cómo cortar la carne (programa). Sin embargo, el problema es que no todos hablan de la misma forma. Entonces, la solución es establecer un idioma que todos entiendan, que puede ser el Inglés; así, habría que ajustarse a las reglas gramaticales de este idioma para describir los pasos a seguir para preparar el taco (lenguaje de programación).

Los pasos fundamentales son definir qué hacer y estructurar ordenadamente los pasos a seguir, 1 y 2 respectivamente. Si bien este taller tiene como objetivo el desarrollo de las herramientas necesarias para el diseño de macros (programas) en Excel, en este documento se busca introducir a los asistentes a la lógica de programación que es la base teórico-conceptual que sirve de fundamento para el diseño de programas en cualquier lenguaje de programación.

Existen dos tipos de programación: la estructurada y la orientada a objetos. Si bien poseen características similares, la segunda tiene una mayor flexibilidad y es más fácil de utilizar. Estas características han propiciado que los nuevos lenguajes tengan una estructura orientada a objetos, incluyendo Visual Basic.

1.1. Programación Estructurada

La Programación Estructurada (PE) surge a finales de los años 1970 con el trabajo de Corrado Böhm y Giuseppe Jacopini². En él se establece que cualquier proceso se puede modelar a partir de tres estructuras de control:

- i) Secuencia: se refiere a una ejecución ordenada de instrucciones.
- ii) Selección: ejecución de una de dos instrucciones, según el valor de una variable booleana.
- iii) Iteración: ejecución de una instrucción mientras una variable booleana sea 'verdadera'. Esta estructura también se conoce como ciclo o bucle.

Para comprender éste tipo de programación considérese el siguiente ejemplo: supóngase que se quiere que se quiere hacer uso de la programación estructurada para modelar las actividades que se realizan durante una semana. Para hacerlo, conviene pensar en un día cualquiera y preguntarse por: ¿qué actividades se realizan de manera cotidiana?, ¿cuál de todas ellas es la primera?, ¿cuál sigue después?, ¿con qué frecuencia?, ¿cuál es la última actividad que se realiza en el día?, etc.

Ya que se debe partir de algún punto, es factible suponer que el despertar por las mañanas es el punto de partida para las actividades del día. Así, luego de haberse despertado el siguiente paso es determinar si se trata de un día laboral –o no-. Si es laboral, entonces lo más probable es que se dedique unos minutos al arreglo personal y se vaya al trabajo; al finalizar la jornada laboral, se pueden hacer otras actividades, pero invariablemente se regresará a casa en algún momento, se tomará un descanso, la cena y se terminará el día yendo a dormir.

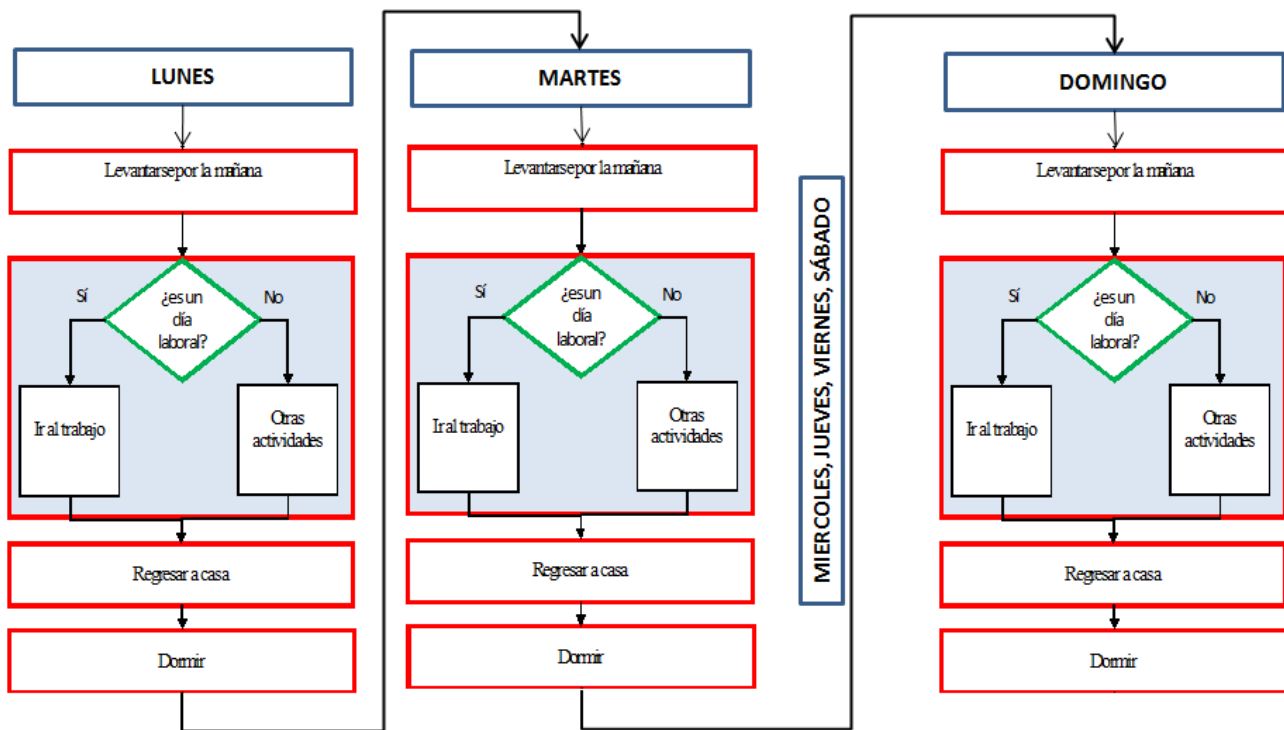
² Para mayor detalle ver: Corrado Böhm y Giuseppe Jacopini, "Flow diagrams, Turing Machines and Languages with only Two Formation Rules". Comm. of the ACM, 9(5): 366-371,1966. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.9119&rep=rep1&type=pdf>

Al día siguiente, luego de despertar, nuevamente se identificará si es un día laboral –o no-. Si lo es, se repetirá la rutina anteriormente descrita. La excepción se da cuando se trata de un día no laboral (sábado o domingo); ya que se puede optar por realizar otras actividades como salir a pasear, permanecer en casa u otra. Sin embargo, al final del día también se regresará a casa para descansar y dormir.

Planteado de esta manera se distinguen algunas actividades que intervienen en el desarrollo de las actividades de una semana como: despertar por las mañanas; verificar qué día de la semana es; el arreglo personal; ir al trabajo o realizar otras actividades; regresar a casa, descansar y dormir.

Con estos elementos que se ha descrito, se puede construir una rutina que puede ser programable. No perder de vista que la intención final es modelar el comportamiento que se sigue una persona durante una semana.

Esquema 2 Programación estructurada



En este esquema, la *secuencia* se refiere a los cuadros en rojo que son actividades que se realizan de forma ordenada, cuyo flujo comienza en la mañana y termina en la tarde, para cada día de la semana. La *selección* es el rombo verde, en donde se determina la trayectoria a seguir, a saber: si se trata de un día laboral, entonces se irá al trabajo; caso contrario, se realizarán otras actividades. Mientras que la *iteración* se trata del conjunto de acciones que se encuentran en el rectángulo con

fondo azul. Cabe hacer notar que el flujo que se sigue en el día lunes se repite para el martes, miércoles y así sucesivamente hasta el domingo aun cuando sean muy parecidos.

1.2. Programación Orientada a Objetos

De manera formal, la Programación Orientada a Objetos (POO) es una forma de programar en donde las instrucciones hacen referencia a elementos de entorno los cuales se denominan "objetos". La noción de objetos es similar a los del mundo real: una mesa, casa, auto, etc. La diferencia es que en programación se trata de entidades que tiene:

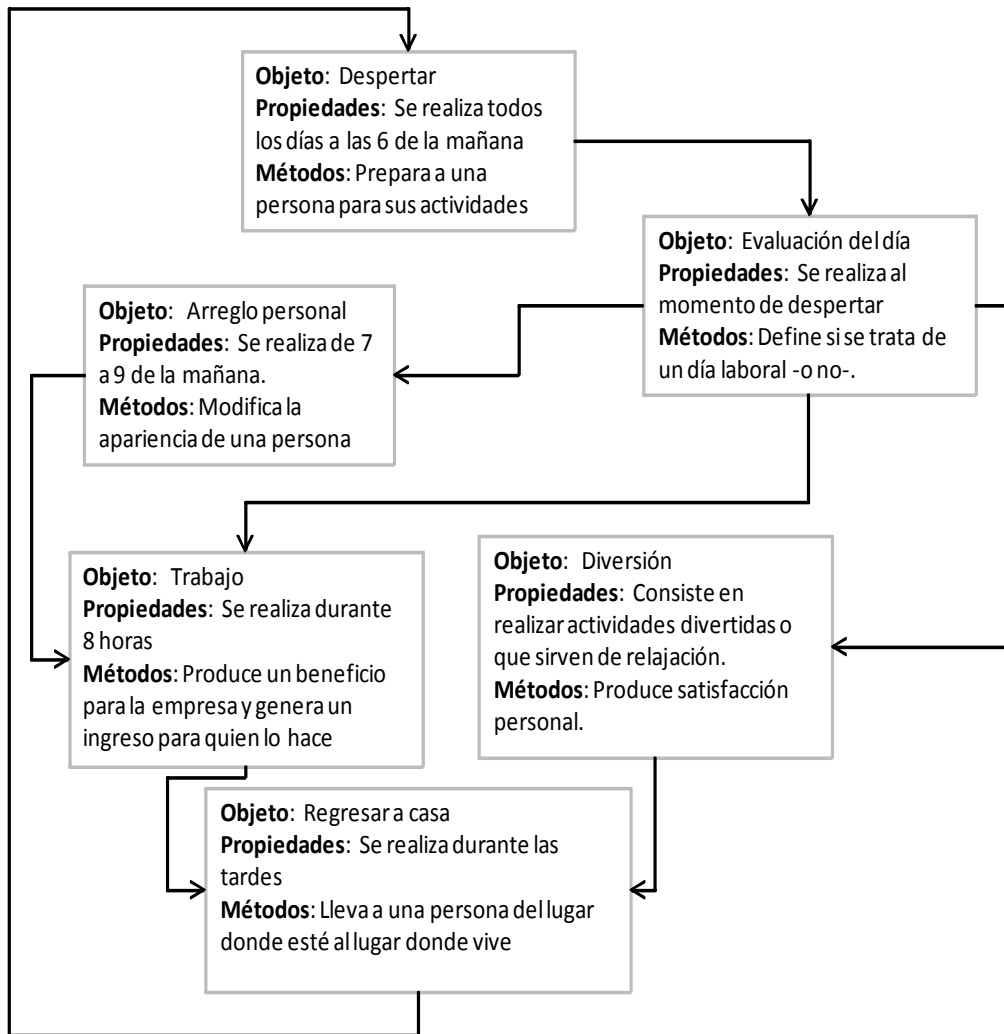
- i) Propiedades o atributos (datos).
- ii) Comportamiento o funcionalidad (métodos).

Por ejemplo, si se considerara como un *objeto* un automóvil, entonces las *propiedades* de este objeto son sus características: el tamaño, el color, el peso; mientras que los *métodos* serían las funciones: moverse rápido, modificar el clima con el aire acondicionado, producir smog.

Con esta noción de objeto se puede dar el siguiente paso, que es el desarrollo de una rutina con base en la creación de objetos. Para ello, se retoma el ejemplo anterior de las actividades que se realizan durante una semana, pero a diferencia de la programación estructurada en donde el flujo se iba definiendo etapa por etapa, acá se definen objetos que engloban procesos completos.

Así, se propone crear un objeto que se denomine "despertar" el cual tenga la *propiedad* que se realice todos los días a las 6 de la mañana y que posea el *comportamiento* tal que prepare a una persona para sus actividades diarias. Para completar el comportamiento es necesario crear más objetos que engloben a otras actividades que se realizan durante el día. En el esquema siguiente se describen dichos objetos y su interacción.

Esquema 3 Programación orientada a objetos



Como se puede observar, la rutina general se compone de objetos que se relacionan unos con otros y dan origen a una estructura más general. A dichos objetos se puede recurrir sin necesidad de programarlos nuevamente. Obsérvese el caso del objeto “evaluación del día”, de donde se puede generar tres tipos de flujos distintos, según se trate de un día laboral o se requiera –o no- de arreglo personal. Éstas son justo las características más significativas que distinguen a este tipo de programación, la flexibilidad de usar los objetos según se requiera.

Una analogía más, para fijar ideas en cuanto a este tipo de programación se puede obtener a partir de retomar el ejemplo del automóvil. Si se diera el caso de que dejara de funcionar el estéreo, basta

con sustituirlo para que regrese a su estado original, pero no hay necesidad de reemplazar todo el carro. Así es en la programación orientada a objetos, cuando hay un código que no funciona, no se desecha el programa completo sólo se corrige lo que haga falta. Una ventaja adicional, es que al contar con objetos independientes sólo es necesario definirlos una vez y recurrir a ellos cada vez que se haga falta.

2. Elementos básicos de la programación

La literatura señala que existe una amplia variedad de elementos que se pueden considerar al momento de realizar un programa; en gran medida depende del lenguaje de programación que se utilizando. Sin embargo, los elementos más generales que dan forma a un programa son: variables; operaciones básicas aritmético-lógicas; funciones; y bucles de programación. A continuación se explican cada uno de ellos.

2.1. Funciones

Se trata de un conjunto de códigos organizados de tal manera que realizan una tarea específica y son utilizadas para dividir grandes problemas en tareas más simples y sencillas. Dichas funciones permiten organizar el código de una manera legible y puede ayudar a reducir significativamente la cantidad de código en un programa. La estructura general es la siguiente:

Función *Nombre* (*Variables a utilizar*)
Procedimiento
Fin de la función

Las características que debe de tener toda función para llevar a cabo una tarea son:

- 1) nombre: sirve para distinguir una función de otra, además que sirve para identificar su función rápidamente
- 2) variables: son los insumos que usa la función para poder realizar la tarea
- 3) procedimiento: son las acciones que ejecuta la función

2.2. Argumentos o variables

Normalmente, las funciones operan sobre ciertos *argumentos* que son pasados a éstas en forma de variables. Formalmente, dichos argumentos se definen como un espacio en el sistema de almacenaje (memoria principal de una computadora), cuya principal característica es que poseen un nombre simbólico asociado a dicho espacio.

Por ejemplo, para saber la hora en varios momentos del día, se puede definir una variable que almacene dicha información. Así, conforme pasa el tiempo, su valor se va ajustando dependiendo la hora. Es importante hacer notar que las funciones pueden intercambiar variables (argumento) y así interactuar unas con otras.

El dato que almacena la variable (argumento) puede ser de diversos tipos:

- Lógico: es aquel que puede representar valores de forma lógica binaria, esto es 2 valores, que normalmente representan falso o verdadero.
- Entero: puede representar a los números enteros: Flotante: representan números reales: 3.1416, 3.222, 849349.222,...
- Carácter: corresponde a letras del alfabeto: "a", "b", "c",..."z"
- Cadena: es una secuencia de caracteres: "jsadflk", "jlasjdfjk",....

TIPO DE DATO	DEFINICIÓN	EJEMPLO
Lógico	Es aquel que puede representar valores de forma lógica binaria	Falso o verdadero
Entero	Puede representar a los números enteros	1, 2, 3,..., 1000,....
Flotante	Representan números reales	3.1416, 3.222, 849349.222,...
Carácter	Corresponde a letras del alfabeto	"a", "b", "c",..."z"
Cadena	Es una secuencia de caracteres	"Libro", "árbol",....

Es importante definir correctamente el tipo de dato que almacena cada variable, ya que un error muy común que se suele definir un tipo de dato al inicio del programa pero pero al momento de usarlo se considera con otras características. Por ejemplo, se define una variable con un dato entero y en el código se pide asignarle un número con decimales.

2.3. Operadores lógicos

Los operadores lógicos proporcionan un resultado a partir de que se cumpla -o no- una cierta condición. Los más usuales en programación son:

OPERADOR	SÍMBOLO	EJEMPLO	SIGNIFICADO
NO	<>	<>A	todo lo que no es A
Y	and	A & B	A y B a la vez
O	or	A B	O bien A, o bien B, o bien los dos
IGUAL	=	A = B	A debe ser igual a B
MENOR QUE	<	A < B	A debe ser menor que B
MAYOR QUE	>	A > B	A debe ser mayor que B

2.4. Bucles de programación

Un bucle o ciclo en programación es una sentencia que se realiza repetidas veces un conjunto de instrucciones, hasta que la condición asignada deje de cumplirse. La utilidad es hacer una acción repetidas veces sin tener que escribir varias veces el mismo código; esto ahorra tiempo, deja el código más claro y facilita su modificación en el futuro.

2.4.1. Bucle FOR

Es una estructura en donde se indica el número de iteraciones (o repeticiones) que se deben de realizar. Para poder trabajar con este bucle se requiere de una variable de control, que generalmente es la letra *i*. Su utilidad es que sirve como iterador, es decir, que se incrementa con cada repetición del ciclo. Este incremento, también tiene que ser definido (letra *s*) e indica cuántas unidades se incrementa el iterador. Por último, al tratarse de un ciclo finito, también se tiene que hacer explícito dónde parar, lo cual se suele simbolizar con la letra *n*. La estructura general de este bucle es la siguiente:

para $i \leftarrow x$ hasta n a incrementos de s hacer
instrucciones
fin para

2.4.2. Bucle WHILE

En esta estructura la intención es repetir un bloque de código mientras una condición se mantenga verdadera. Para saber si dicha condición sigue presente se realiza una comparación, generalmente usando los operadores lógicos: igual, diferente, menor o igual, mayor o igual, menor y mayor. La estructura general de este bucle es la siguiente:

mientras condición hacer
instrucciones
fin mientras

2.4.3. Bucle REPETIR

Comprueba la condición de finalización al final del cuerpo del bucle. Si ésta es cierta, entonces continua con el resto del programa. Si bien es muy similar a la estructura anterior, en muchas ocasiones esto resulta más adecuado ya que ejecuta las instrucciones en la primera iteración. Sin embargo, depende del programador optar por usar una u otra según sus necesidades.

repetir
instrucciones
hasta que condición

2.4.4. Selección IF

Se trata de una estructura de control que permite redirigir un flujo según la evaluación de una condición simple, es decir, si ésta es falsa o verdadera. Es la estructura más usada en la programación en Visual Basic, por lo que conviene tener claro su funcionamiento. La estructura general es la siguiente:

Sí condición entonces

Sentencia 1

En otro caso

Sentencia 2

Fín Sí

En la primera línea de código se evalué la condición, si es verdadera, se ejecuta el bloque de sentencias 1, de lo contrario, se ejecuta el bloque 2.

3. Pseudocódigo

Hasta este momento se ha definido cuáles son los elementos de un programa, el siguiente paso es interactuar entre dichos elementos y desarrollar la habilidad de programar en abstracto. Es decir, pensar qué pasos se requieren para modelar un cierto fenómeno, qué características se deben de tener y el resultado deseado. Pensar en abstracto es la forma básica de implementar los programas, esto en términos de la programación formal recibe el nombre de *pseudocódigo*.

Se propone realizar un ejercicio más pensando en abstracto (pseudocódigo). La propuesta es retomar el ejemplo de las actividades que se realizan durante una semana e implementarlo con base en los elementos que hasta el momento se han descrito.

Antes de iniciar, conviene detenerse un momento y hacer una lluvia de ideas respecto a todos los elementos que intervienen en las actividades de la vida diaria. O cómo se sugiere en el primer acápite pensar en las preguntas: ¿qué hacer?, ¿cuál es el objetivo final?, ¿Qué pasos son primero?, etc. Con ayuda de los diagramas anteriores se pueden identificar los siguientes elementos:

- Saber si se tiene trabajo –o no-.
- Saber la hora del día en varios momentos.
- Saber con exactitud el día de la semana.

Estos son los aspectos generales que intervienen en la toma de decisión de las actividades que se realizan durante el día. Ahora, se tiene que pensar en qué variables intervienen en el proceso y definir el tipo de dato que tendrán.

3.1. Implementación de Variables

Como primer paso, en todo programa se requiere declarar las variables. Es decir, los argumentos mediante los cuales va a funcionar el modelo. Con base en los tres puntos anteriormente descritos, se proponen las siguientes variables:

Nombre de la variable	Descripción	Tipo de dato
Contrato	Identifica si una persona tiene –o no- trabajo.	Binario. Si se tiene trabajo contrato=1, de lo contrario contrato=0
T	Contiene la hora del día	Entero.
Día	Contiene el día de la semana	Entero.
Flag	Indicando el flujo general	Cadena. Ya que anota

3.2. Implementación de Código

Para proceder a implementar el programa es necesario pensar estructural y ordenadamente los pasos a seguir, con el entendido de que cada actividad puede ser considerada como un objeto. Para iniciar, considere las actividades de un día normal con más detalle:

- 1) El día inicia cuando la persona se despierta.
- 2) Luego de levantarse, el siguiente paso es identificar qué día de la semana es, ya que de ello dependerá las actividades que se van a realizar. Si se trata de un día laboral, lo más normal es que se decida ir al trabajo.
- 3) Independientemente si se decide ir al trabajo o a otro lugar, lo más común es que antes de salir de la casa, se tome unos minutos para bañarse, cambiarse, etc. Dicho en otras palabras, se procede a dedicarle un tiempo al arreglo personal. Sin embargo, ésta actividad no puede durar mucho tiempo, pues el ingreso al trabajo es a una hora determinada.
- 4) Una vez que se haya completado el arreglo personal entonces ya es factible dirigirse al lugar de trabajo, o bien, emprender otras actividades en caso de que se trate de un día no laboral.
- 5) Luego de transcurrido el día, independientemente si se estuvo en el trabajo o se realizaron otras actividades, lo más común es que al anochecer, se regrese a casa para descansar y dormir.

Esta primera fase constituye una descripción general de cómo se podría modelar las actividades que se realizan en un día normal. El siguiente paso, previo a la implementación, es retomar cada punto e ir profundizando en los detalles. Esas cuestiones que en primera instancia pasaron desapercibidas pero que pueden ser importantes.

Además, se debe de tener presente de que esto es un intento de modelar una situación real, por lo que hay situaciones que salen fuera de control de la programación; cuando esto suceda se tendrán que hacer *supuestos*. Si bien no hay literatura al respecto y se alude al criterio del programador, lo recomendable es pensar en situaciones generales y no en particulares. Es decir, definir los criterios que puedan dar respuesta a la mayor cantidad de casos posibles.

Retomando el ejemplo, en esta segunda revisión se busca solventar y definir aquellos detalles que no se consideraron al inicio:

- En el punto 1, se menciona que el día inicia cuando una persona despierta. ¿a qué hora sucede? Si bien esto es arbitrario, lo cierto es que sucede en algún momento en la mañana. Para poder realizar una programación se debe de considerar una hora en específico, o bien un lapso de tiempo en el cual la persona se despierte. Pensando en una generalidad, se propone considerar un lapso de tiempo entre las 6am y las 7am.
- Identificar el día de la semana es un proceso inmediato, el cual supondremos que no requiere de un periodo de tiempo para realizarlo.
- Sin embargo, el arreglo personal si está limitado. Sobre todo si se trata de un día laboral, entonces se tiene que realizar en un cierto periodo de tiempo. Si se supone que la entrada al trabajo es a las 8:30, entonces éste se tiene que realizar entre las 7am y 8am.

Considerando que después de que se realice será necesario trasladarse al trabajo o bien emprender otras actividades.

- Si se acude al trabajo se tiene un horario establecido de 8 horas, por lo que no hay discusión, y se supondrá que si se decide realizar otras actividades también se ocupará el mismo periodo de tiempo. No obstante, el regreso tiene un horario variable que dependerá de las actividades y el lugar donde se encuentre la persona. Sin embargo, el regreso considera el traslado de un lugar a otro por lo que debe de haber un momento determinado. Si bien es imposible considerar caso por caso, se debe de considerar un criterio general, para lo cual se supone que si son entre las 9 pm y las 10 pm entonces se procede a regresar a casa.

Una vez que se tiene definido qué se quiere modelar, el siguiente paso es escribir el código. Para ello se debe de tener presente que se va a hacer uso de la programación orientada a objetos, entonces hay que crearlos.

Como se mencionaba al inicio, un objeto es una estructura que posee propiedades y métodos. En términos de programación se puede entender como un bloque de instrucciones que posee variables y rutinas (bucles) que sirven para ejecutar una acción determinada. Así, la propuesta es considerar los siguientes objetos:

Esquema 4 Descripción de objetos y pseudocódigo

OBJETO	PSEUDOCÓDIGO	DESCRIPCIÓN
A	<p><i>Función Despierta (variable t)</i></p> <p><i>Si t >= 6 am Y t < 7 am entonces</i></p> <p><i>Levántase de la cama para iniciar el día</i></p> <p><i>flag = "listo"</i></p> <p><i>Caso contrario</i></p> <p><i>Seguir durmiendo</i></p> <p><i>Termina Si</i></p> <p><i>Termina la función</i></p>	<p>Nombre: Despierta</p> <p>Variables: Tiempo</p> <p>Propiedad: Se realiza entre las 6 am y las 7 am</p> <p>Método: Prepara a una persona para iniciar el día</p>
B	<p><i>Función Arreglo (variable t)</i></p> <p><i>Si t >= 7 am y t < 8 am entonces</i></p> <p><i>Modificar la apariencia de la persona</i></p> <p><i>flag = "listo"</i></p> <p><i>Caso contrario</i></p> <p><i>No modificar la apariencia</i></p> <p><i>Termina Si</i></p> <p><i>Termina la función</i></p>	<p>Nombre: Arreglo</p> <p>Variables: Tiempo</p> <p>Propiedad: Se realiza entre las 7 am y las 8 am</p> <p>Método: Modifica la apariencia de una persona</p>

C	<p>Función Actividad (variable día)</p> <p><i>Si día=Lu ó Ma ó Mi ó Ju ó Vi entonces</i> <i>ejecuta Arreglo (t)</i> <i>Ve al trabaja por 8 horas</i></p> <p>Caso contrario <i>ejecuta Arreglo (t)</i> <i>Diversión</i></p> <p><i>Termina Si</i> <i>Termina la función</i></p>	<p>Nombre: Actividad</p> <p>Variables: Día de la semana</p> <p>Propiedad: Se realiza una vez definido el día</p> <p>Método: Lleva a una persona al trabajo, o hacia otra actividad si es fin de seman</p>
D	<p>Función Regreso (variable t)</p> <p><i>Si t >= 9 pm Y t <= 10 pm entonces</i> <i>Traslado del lugar donde esté a la casa</i></p> <p>Caso contrario <i>Mantenerse en el lugar</i></p> <p><i>Termina Si</i> <i>Termina la función</i></p>	<p>Nombre: Regreso</p> <p>Variables: Tiempo</p> <p>Propiedad: Se realiza entre las 9 am y las 10 am</p> <p>Método: Regresa a una persona de donde este a su casa.</p>

Ya que se tienen los objetos diseñados, el último paso es crear uno más que sirva para ensamblarlos. La característica de éste último es que hace referencia a otros objetos y tiene la condición más general, que es validar si la persona trabaja –o no-.

E	<p>Repetir mientras contrato=1</p> <p><i>ejecuta Despierta (t)</i> <i>Si flag = "Listo" entonces</i> <i>ejecuta Actividad (día)</i></p> <p>Caso contrario <i>ejecuta Trabajo (día)</i></p> <p><i>Termina Si</i> <i>ejecuta Regreso (t)</i> <i>Termina repetir</i></p>	<p>Nombre: General</p> <p>Variables: Contrato, tiempo y día</p> <p>Propiedad: Ensambla los demás objetos</p> <p>Método: Modela las actividades de una semana</p>
---	--	--

Elemento del programa
Bucles
Sentencias o código que define el programador
Operadores lógicos
Variables
Funciones

Con estos elementos ya se puede implementar un programa dado que se han definido los objetos, rutinas y procedimientos a seguir. El ejemplo que se ha desarrollado hasta este momento donde se buscaba modelar el comportamiento que se tiene en una semana concluye en este acápite. En el siguiente se explican las características más generales de la aplicación de VB que se encuentra disponibles en Excel y se construye un ejemplo a partir de situaciones reales que se presentan en el área de Estadísticas Sociodemográficas pero que son aplicables a las demás áreas.

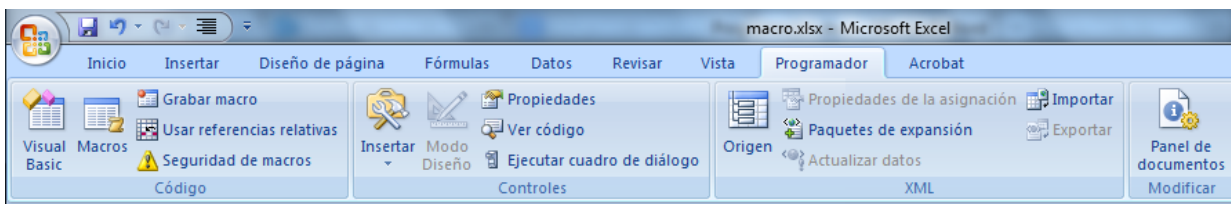
4. Visual Basic

Formalmente, un *lenguaje de programación* es un lenguaje artificial que puede ser usado para controlar el comportamiento de una computadora. Este se compone de un conjunto de reglas y semánticas que permiten expresar instrucciones que luego serán interpretadas. El objetivo de este taller es conocer el lenguaje de programación denominado como Visual Basic y en particular sus aplicaciones para Office.

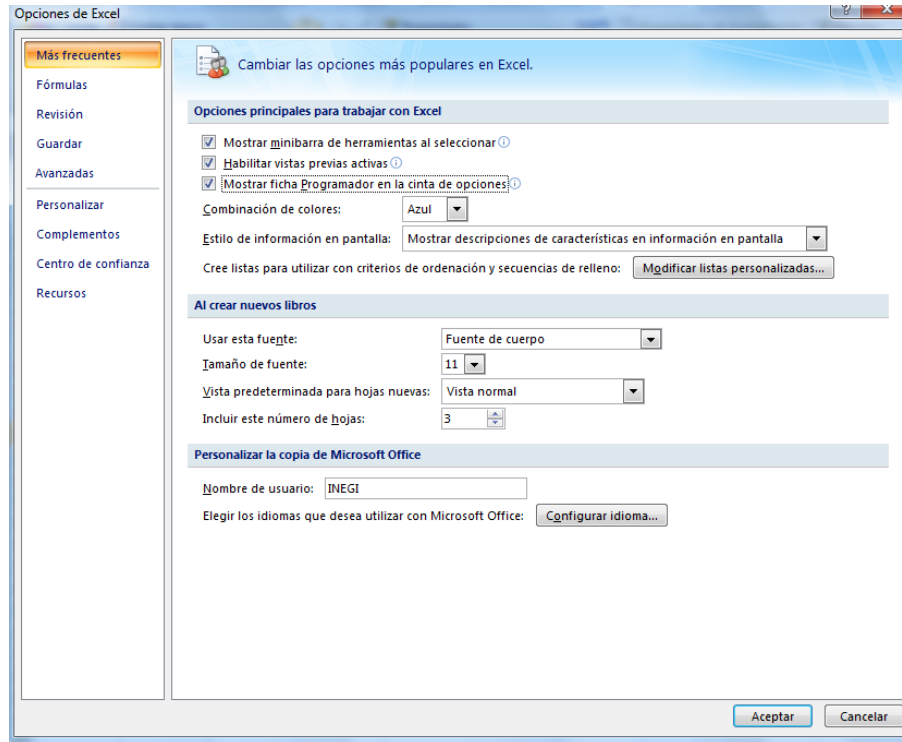
La estrategia metodológica es considerar dos etapas: en la primera, se describe las características más generales de VBA; en la segunda, se procede a la implementación de los programas con los elementos que se han desarrollado hasta este punto.

4.1. Aspectos Generales

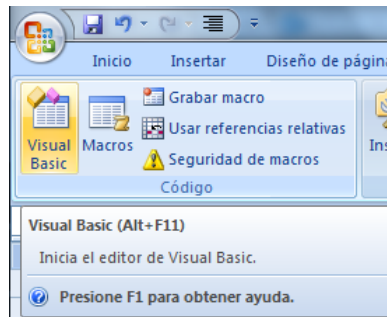
Para iniciar, el primer paso es tener habilitada la opción de *Programador* en la barra de menús.



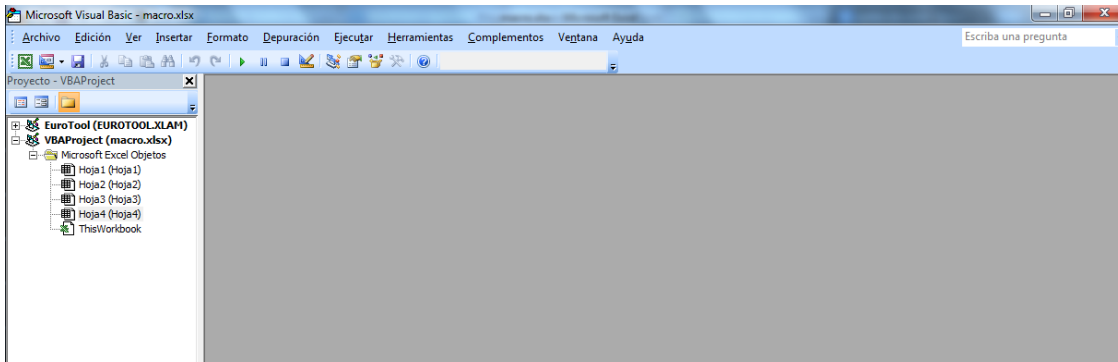
En caso de tenerlo inhabilitado, hay que dirigirse al botón de Inicio\Opciones de Excel y habilitar la opción *Mostrar ficha de Programador en la cinta de opciones*.



Para empezar con la captura de código, hay que ingresar a la opción *Visual Basic*

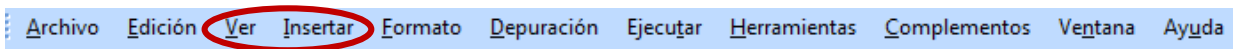


Ahí se abre el editor de Visual Basic que es el lugar donde se van a escribir las sentencias para su ejecución.

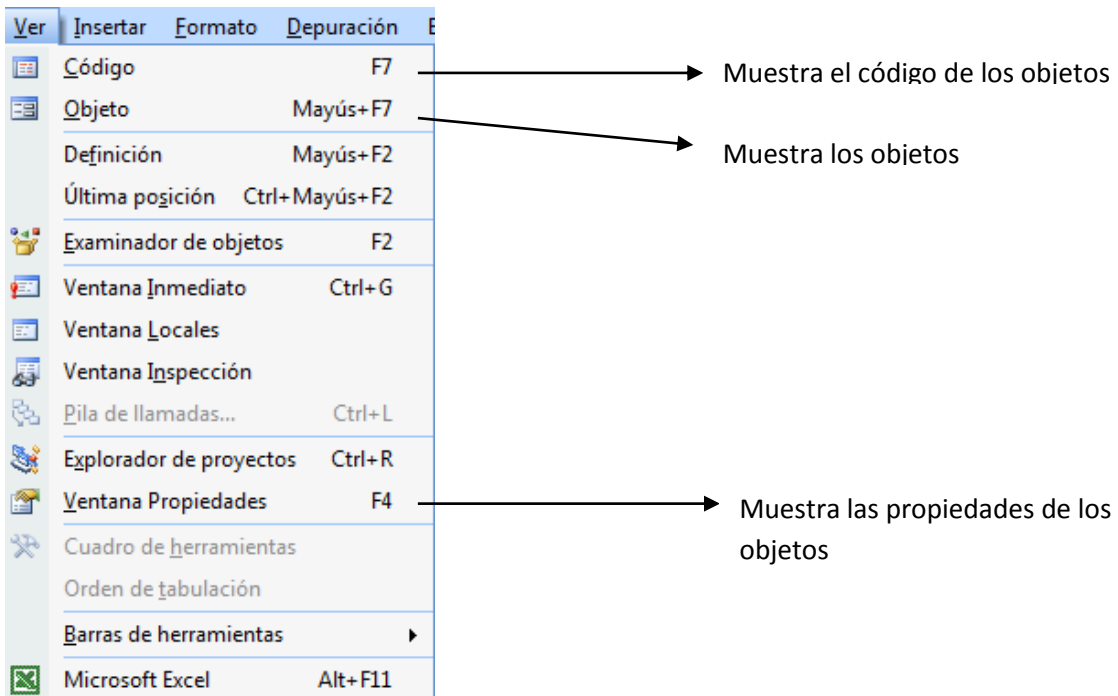


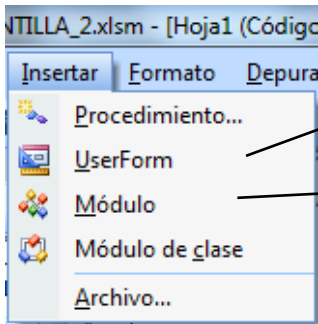
Esta pantalla consta de los siguientes elementos:

El menú del editor de Visual Basic:



Esta es la barra de menú del editor de Visual Basic donde se puede acceder a las diversas opciones y comandos del propio editor. Si bien los dos primeros menús, son similares a los que se encuentran en Word, los dos siguientes son particulares del visor de Visual Basic y tienen una gran funcionalidad. A continuación se explican los rasgos más importantes.

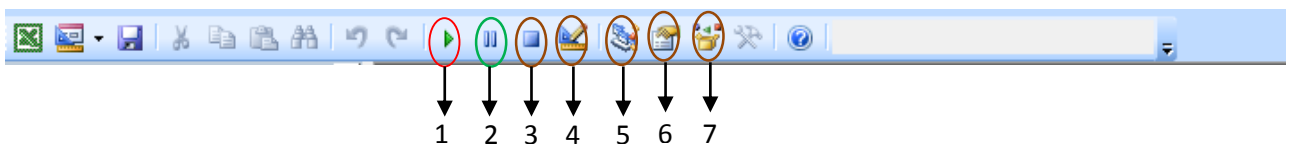




Inserta UserForm, que son las plantillas que sirven para hacer formularios

Inserta Módulos, que son los espacios para hacer la programación

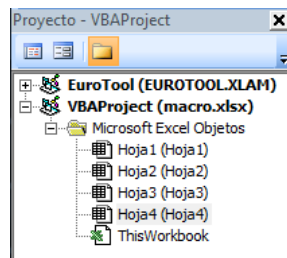
La barra de botones:



1. Ejecuta una macro
2. Hace una pausa en la ejecución de la macro
3. Detiene la ejecución de la macro
4. Cambia a la vista de diseño
5. Explorador de proyectos
6. Abre la ventana de propiedades de los objetos
7. Examinador de objetos

Esta es la barra de donde se puede acceder de una manera más rápida a las opciones más comúnmente utilizadas, como son la ejecución, la pausa, guardar, etc..

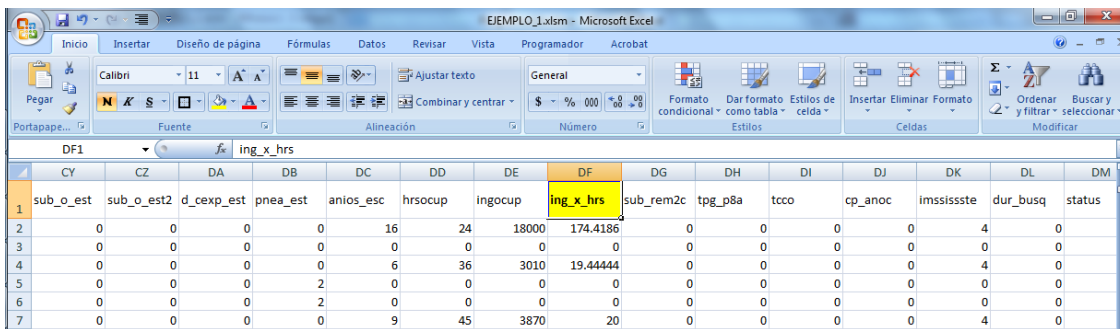
En la parte izquierda de la pantalla se puede ver un recuadro en la parte superior del proyecto. Ahí se encuentran los módulos, las hojas utilizadas en los procedimientos/funciones, etc.



4.2. Aplicaciones

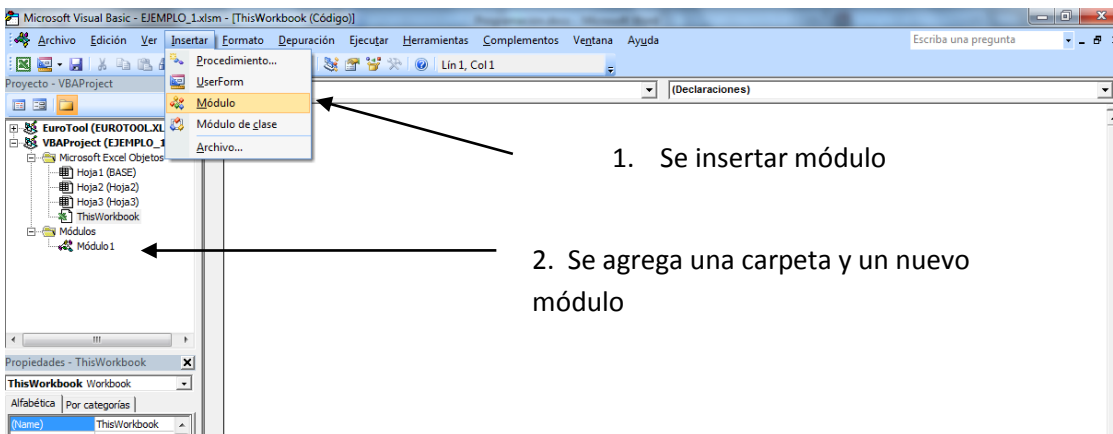
Luego de hacer una revisión de la lógica de programación y algunas de las características más generales del Editor de Visual Basic, el siguiente paso es unir la teoría con la práctica. Para ello, considérese un nuevo ejemplo:

*Supongamos que tenemos la base de datos del cuestionario sociodemográfico de la ENOE, y la intención es clasificar los datos de la variable **ing_x_hrs** (ingreso por horas) en tres categorías: alto, medio y bajo.*



	CY	CZ	DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM
	sub_o_est	sub_o_est2	d_cexp_est	pnea_est	anios_esc	hrsocup	ingocup	ing_x_hrs	sub_rem2c	tpg_p8a	tcco	cp_anoc	imssissste	dur_busq	status
1	0	0	0	0	16	24	18000	174.4186	0	0	0	0	0	4	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	6	36	3010	19.44444	0	0	0	0	0	4	0
4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	9	45	3870	20	0	0	0	0	0	4	0
7	0	0	0	0											

El primer paso es insertar un módulo, es decir, un espacio en el cual podemos introducir las variables y los procedimientos que se van a realizar.



Para ingresar al código, basta con hacer doble click en el ícono llamado *Módulo 1*. Una vez ahí, se puede proceder a escribir el código necesario.

En términos generales lo que necesitamos es escribir una rutina que evalúe cada renglón de la columna “DF” donde se encuentran los datos de la variable ing_x_hrs y evaluar si cumple – o no – con ciertos criterios preestablecidos.

Más específicamente, queremos clasificar la variable en tres niveles:

- Nivel bajo: aquellas personas que ganan entre 0 y 10
- Nivel medio: entre 11 y 50
- Nivel alto: más de 50

Conviene hacer notar varios aspectos:

- 1) El procedimiento debe evaluar cada celda, iniciando en la DF2 y terminando en la DF9873
- 2) En cada una de ellas, se debe de evaluar los valores para saber en qué nivel se encuentra

Así, necesitamos 2 tipos de rutinas: una que sirva para desplazarse desde la celda DF2 a la DF9873; y una condicional que sirva para clasificar el valor de cada celda. En términos de programación necesitamos hacer uso de un bucle **FOR** y de una condicional **IF**.

En *pseudocódigo* la rutina que se desea implementar es la siguiente:

para DF2 hasta DF9873 a incrementos de 1 en 1 hacer

Si valor está entre 0 y 10 entonces

Clasificar la celda como Nivel Bajo

sí valor está entre 11 y 50 entonces

Clasificar la celda como Nivel Medio

En otro caso


Clasificar la celda como Nivel Alto

Fin Si

fin para

Ya en el lenguaje de Visual Basic, estas se transforman en:

```
1      Sub ingreso()  
2      Dim i As Integer  
3      Sheets("BASE").Select  
4      For i = 3 To 9872  
5          If Cells(i, 110).Value >= 0 And Cells(i, 110).Value < 10 Then  
6              Cells(i, 119).Value = "NIVEL BAJO"  
7          ElseIf Cells(i, 110).Value >= 10 And Cells(i, 110).Value < 50 Then  
8              Cells(i, 119).Value = "NIVEL MEDIO"  
9          Else  
10             Cells(i, 119).Value = "NIVEL ALTO"  
11         End If  
12     Next  
13 End Sub
```

Los comandos 1 y 13 sirven para declarar la función llamada **ingreso**. Así, las líneas de códigos 2 a 12 se ejecutan cada vez que se corra la función. Lo cual se puede hacer con el ícono  que se encuentra en la barra de botones.

En la línea 2 se procede a declarar las variables que se van a necesitar (**Dim**) y se especifica el tipo de dato que va a almacenar dicha variable, que en este caso es un número entero (**As Integer**). Luego, en la línea 3 se especifica el objeto sobre el cual se va a trabajar que en este caso es la hoja llamada BASE. Cabe recordar que se está trabajando con base en la programación orientada a objetos, por lo que una hoja en Excel representa un objeto en VB. Así, dentro del objeto BASE (hoja de Excel) se recurre a la propiedad Select; de esta forma se especifica el lugar donde dará inicio la rutina.

En las líneas 4 y 12 se declara el bucle **For** que se había sugerido en el pseudocódigo; la forma de declararlo es recurrir a la variable *i*, declarada en la línea 2, y pedirle que inicie en el valor 3 y que se ejecute de 1 en 1 hasta (**To**) el valor 9872. Así se consideran todos los valores de la columna DF.

Entre las líneas 5 y 11 se encuentra la estructura de selección **If**, que sirve para evaluar el valor de las celdas de DF y clasificarlas de acuerdo con su valor.

Para poder desplazarse a lo largo de la columna DF, es necesario recurrir al objeto *cells(a,b)* que sirve para ubicar el valor de las celdas de Excel. Su funcionamiento es similar al de un eje de coordenadas *x*, *y* en donde el desplazamiento se efectúa dependiendo los valores de *a*, *b*; en donde “*a*” representa a las filas y “*b*” a las columnas. Por ejemplo, para ubicar a la celda B3 se tiene que

especificar `cells(3,2)`; para ubicar el valor de esta celda, se tiene que usar la propiedad `value`. Así, la el valor de la celda B3 se programa como `cells(3,2).value`

h

a

	A	B	C	D	E	F	G	H
1	r_def	loc	mun	est	ageb	t_loc	cd_a	ent
2	00	0001	003	41	00000	1	01	09
3	00	0001	003	41	00000	1	01	09
4	00	0001	007	22	00000	1	01	09
5	00	0001	007	22	00000	1	01	09
6	00	0001	007	22	00000	1	01	09
7	00	0001	007	22	00000	1	01	09
8	00	0001	007	22	00000	1	01	09
9	00	0001	016	33	00000	1	01	09
10	00	0001	010	42	00000	1	01	09
11	00	0001	010	42	00000	1	01	09
12	00	0001	003	43	00000	1	01	09

Retomando el ejemplo que se está mostrando, la intención es recorrer todas las celdas de la columna DF, por lo que se requiere conocer el valor de las celdas DF2, DF3, ..., DF9872. Usado el objeto `Cells(a,b)`, sería necesario conocer el valor de `Cells(2,110)`, `Cells(3,110)`, ..., `Cells(9872,110)`. Sin embargo, se observa que el único desplazamiento es en cuanto a las filas, ya que las columnas se mantienen fijas. Así conviene considerar `Cells(i,110)` en donde `i` varía desde 2 hasta 9872 según se especificó en la línea 4.

La línea 6, se ejecuta si la condición de la línea 5 se cumple. En caso de que esto suceda, en la columna DO (`Cells(i, 119)`) en usando el objeto `Cells(a,b)` se escribe el nivel en el cual se clasifica el valor.

En la línea 7 se evalúa el caso del Nivel Medio, en la 8 se especifica la rutina a seguir. Y ya que un valor tiene que ser clasificado en cualquiera de estas tres categorías, la columna 9 sirve para evaluar los casos que no fueron considerados en ninguna de los categorías anteriores. Así los casos que excedan a 50 se clasifican como Nivel Alto.

Bibliografía

Dijkstra, Edsger W. (1968), "Go To Statement Considered Harmful" (Letter to the Editor), Communications of the ACM 11(3) (March): 147-148. Disponible en:
https://files.ifi.uzh.ch/rerg/arvo/courses/kvse/uebungen/Dijkstra_Goto.pdf

Corrado Böhm y Giuseppe Jacopini, "Flow diagrams, Turing Machines and Languages with only Two Formation Rules". Comm. of the ACM, 9(5): 366-371, 1966. Disponible en:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.9119&rep=rep1&type=pdf>

Cómo programar en C/C++, H.M. Deitel, P.J. Deitel, Ed. Prentice Hall

Kernighan, Brian W. y Ritchie, Dennis M.: The C Programming Language (2nd Edition), Prentice Hall, 1988.

Programming, Principles and Practice using C++, Bjarne Stroustrup, Ed. Addison Wesley, 2009.

DESCRIPCIÓN	PÁGINAS
Teoría de la programación orientada a objetos.	http://en.wikibooks.org/wiki/Object_Oriented_Programming
Ejemplos de macros	http://chandoo.org/wp/excel-vba/examples/ http://analysistabs.com/excel-vba/most-useful-codes-examples-macros-how-tos-basics-advanced/ http://www.excelfunctions.net/Excel-Macro-Example.html http://www.aulaclie.es/macros-excel/index.htm
Código en Visual Basic que funciona en macros	http://www.planetsourcecode.com/vb/scripts/ShowCode.asp?txtCodeId=1653&lngWId=1
Foros de ayuda	http://stackoverflow.com/questions/3113368/theory-behind-object-oriented-programming