Layla Hirsh Martínez

Resolución de problemas usando *Visual Basic for Applications* en Excel

FONDO EDITORIAL
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

RESOLUCIÓN DE PROBLEMAS USANDO *VISUAL BASIC FOR APPLICATIONS* EN EXCEL

LAYLA HIRSH MARTÍNEZ

RESOLUCIÓN DE PROBLEMAS USANDO *VISUAL BASIC FOR APPLICATIONS* EN EXCEL

FONDO EDITORIAL
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Resolución de problemas usando Visual Basic for Applications *en Excel*
Layla Hirsh Martínez

De esta edición:
© Fondo Editorial de la Pontificia Universidad Católica del Perú, 2012
Av. Universitaria 1801, Lima 32 - Perú
Teléfono: (51 1) 626-2650
Fax: (51 1) 626-2913
feditor@pucp.edu.pe
www.pucp.edu.pe/publicaciones

Diseño, diagramación, corrección de estilo y cuidado de la edición:
Fondo Editorial PUCP

Primera edición, noviembre de 2012
Tiraje: 500 ejemplares

Hecho el Depósito Legal en la Biblioteca Nacional del Perú N° 2012-14564
ISBN: 978-612-4146-21-3
Registro de Proyecto Editorial: 31501361200674

Impreso en Tarea Asociación Gráfica Educativa
Pasaje María Auxiliadora 156, Lima 5, Perú

A mi Cucutita, mi abuela Blanca, con quien tengo los más lindos recuerdos de mi vida; te extraño muchísimo.

¡Llamando a Cucutita!

AGRADECIMIENTOS

A mis padres, responsables de mis metas y logros. A mi familia, por tenerme paciencia durante la elaboración de este libro.

A mis alumnos del curso Introducción a la Computación de la Pontificia Universidad Católica del Perú, ya que fue para ellos que elaboré los ejercicios que me llevaron a la creación de este libro.

Índice

PREFACIO

Visual Basic for Applications (VBA) es un lenguaje de programación simple, pero efectivo, que nos permite ampliar aplicaciones de Office. Con este lenguaje y sin necesidad de un gran conocimiento en programación se pueden llevar a cabo aplicaciones más sencillas y amigables para, de esta manera, personalizarlas según la necesidad de cada usuario.

El poder automatizar tareas repetitivas, acelerar labores diarias y crear nuevas funcionalidades en herramientas de uso cotidiano, como lo son Microsoft Word, Outlook, Access, Excel, PowerPoint y Publisher, es una ventaja para el usuario que no está dedicado al desarrollo de softwares de una manera avanzada.

Actualmente, en la vida diaria se utilizan diferentes aplicaciones de oficina: en las minas se usa Excel para manejar las estadísticas y la planificación de volado, en una fábrica se aplica para gestionar el inventario o la producción. Asimismo, en una construcción, se emplea para utilizar o administrar los espacios o los recursos, más aún, en un hogar, sirve para manejar el presupuesto familiar.

Obviamente, cuando mencionamos ejemplos como los anteriores pensamos en aplicaciones informáticas, grandes, complejas y desarrolladas por terceros, pero ¿qué ocurre, en cambio, cuando una empresa es pequeña y no puede contratar a terceros?, ¿o cuando lo que necesita es automatizar una tarea cotidiana y pequeña pero compleja o trabajosa? En este contexto, VBA es una herramienta adicional que, tal vez, no solucionará nuestras vidas, pero sí las puede hacer un poco más sencillas, y lo más importante es que no es necesario ser un genio en computación para entender cómo hacer las cosas con este lenguaje.

¿Qué hará este libro por usted?

Este libro le permitirá no solo manejar el lenguaje VBA, sino también crear un proceso para la solución de problemas. La idea principal es que, sobre la base de

problemas sencillos, el usuario común se familiarice con el lenguaje VBA para que pueda elaborar fácilmente aplicaciones productivas, a través del manejo de esta herramienta en Excel, así como de las celdas, las hojas y los libros de este.

Si bien usted no se convertirá en un experto en Excel, sí estará capacitado para elaborar aplicaciones personalizadas, las cuales difícilmente se podrían realizar usando solamente las opciones básicas de la aplicación.

¿Cómo está organizado el libro?

En la actualidad, los libros que se encuentran en el mercado sobre el tema están más orientados al uso avanzado de Excel o de las herramientas de Microsoft Office que en el manejo de VBA; es por ello que, en el presente libro, se hace más hincapié en el lenguaje que en la aplicación.

Capítulo 1: Resolución de problemas

Si usted no sabe cómo modular un problema o identificar los datos que debe incluir en su aplicación, con este capítulo aprenderá cómo «planificar» su aplicación, utilizando las tres primeras fases para solucionar problemas: definición, análisis y diseño.

Capítulo 2: Subprogramas en VBA

En este capítulo empezará a familiarizarse con el lenguaje VBA. Conocerá sus dos estructuras básicas, funciones y procedimientos, la manera de definirlas, sus partes y, sobre todo, su manejo.

Capítulo 3: Datos y expresiones

A través de este capítulo recordará los distintos tipos de datos que conoce y cómo son manejados en este lenguaje de programación. Adicionalmente, aprenderá a utilizar constantes, variables y expresiones, partes necesarias para una programación eficiente.

Capítulo 4: Estructuras algorítmicas selectivas

De manera general, estas estructuras son necesarias para utilizar condiciones simples y complejas. Cuando sepa cómo manejarlas, usted será capaz de emplear más directa y naturalmente todas las condiciones que requiera.

Capítulo 5: Estructuras algorítmicas iterativas

Anteriormente se mencionó que mediante el uso de VBA usted podría repetir las tareas que necesite. Justamente, estas estructuras serán las que le permitirán reproducir instrucciones tantas veces como lo requiera, sin necesidad de volver a copiar el código. Bastará que coloque las instrucciones que desea rehacer dentro de estas estructuras para que las acciones indicadas se realicen todas las veces que quiera.

Capítulo 6: Editor de VBA

Para manejar Excel, sus celdas, los errores y el editor de VBA son necesarios los conocimientos que se presentan en este capítulo.

En síntesis, todos los capítulos contienen, en primer lugar, la teoría, seguida de ejercicios prácticos en distintos niveles y, finalmente, otros ejercicios, para que usted pueda poner en práctica los conocimientos aquí adquiridos.

PRÓLOGO

En el primer ciclo del año 2008, dicté por primera vez el curso Introducción a la Computación, H117, en la Pontificia Universidad Católica del Perú, con 77 alumnos. Desde ese mes de marzo hasta la actualidad ha hecho falta un libro de texto que facilite el estudio de los alumnos. Si bien en marzo de 2010 se publicó la primera edición de un material para la docencia titulado *Ejercicios propuestos* y en el segundo ciclo de ese año, su segunda edición, en ninguno de estos materiales se explicaba qué pasos se debían seguir o cómo desarrollar los ejercicios. Es por ello que este libro intenta facilitar el aprendizaje del lenguaje de programación VBA.

Por tanto, el libro está inicialmente dirigido a estudiantes universitarios con conocimientos básicos o mínimos de las herramientas Office que quieran desarrollar pequeñas aplicaciones para usar dichas herramientas de manera más sencilla mediante VBA.

No obstante, cualquier persona, de cualquier edad o profesión, puede leerlo y aprender este lenguaje para mejorar su conocimiento de Office.

Para finalizar, espero que este, mi primer libro, pueda ser leído y comprendido fácilmente por todos.

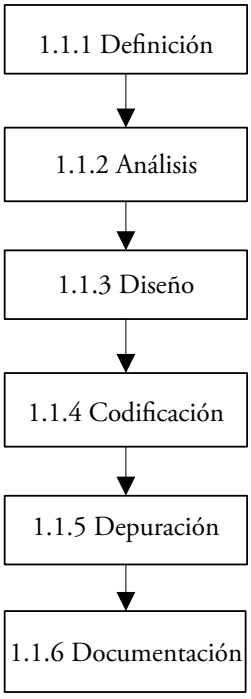
RESOLUCIÓN DE PROBLEMAS*

En este capítulo veremos cuál es el proceso de resolución de problemas; es decir, analizaremos cómo el correcto desarrollo de las fases de dicho proceso nos llevará a una solución óptima.

1.1. Proceso de resolución de problemas

El proceso de resolución de problemas se divide en seis partes o fases:

Ilustración 1.1 Fases de la resolución de problemas



* Todas las ilustraciones y tablas del libro han sido elaboradas por la autora.

Para recordar todas estas fases utilizaremos un acrónimo:

Deadiscodedo

A lo largo de este capítulo emplearemos el siguiente ejemplo para explicar cada fase:

Procesar compra de cliente

Como se puede deducir, en el ejemplo se desea implementar un programa que emita la boleta de venta de un producto. Ahora bien, si observamos este proceso en la vida real, nos daremos cuenta de que, usualmente, se siguen los siguientes pasos:

- Se obtiene el precio unitario y la cantidad de productos.
- Se calcula el subtotal de la boleta.
- Se calcula el monto por IGV.
- Se calcula el monto total de la boleta.
- Se emite (o imprime) la boleta.

1.1.1. Definición

¿En qué consiste el problema?

Esta primera fase consta de cuatro partes:

- Explicación: se expone de manera general lo que se busca hacer.
- Datos de entrada: se menciona cuáles serán las entradas[1] utilizadas.
- Salidas: se indica cuáles serán las salidas[2] obtenidas o calculadas.
- Fórmulas: se señala, si es que las hay, las fórmulas que se utilizarán.

Así, para el ejemplo *Procesar compra de cliente*, la definición será la siguiente:

- Explicación: calcular el total y emitir la boleta de venta.
- Datos de entrada: cantidad y precio del producto.
- Salidas: monto total.

> *No consideramos «salidas» ni al subtotal ni al monto de IGV, ya que ambos son valores intermedios que nos servirán para calcular, finalmente, la salida, pero que no queremos imprimir en la boleta.*

[1] Las entradas son los datos variables que se dan en el planteamiento del problema y que serán utilizados a lo largo de este.

[2] Valores que se desean obtener, los valores objetivos y que, finalmente, queremos escribir o imprimir en pantalla.

- Fórmulas[3]:

$$Subtotal = cantidad * precio$$

$$montoIGV = subtotal * 0.18$$

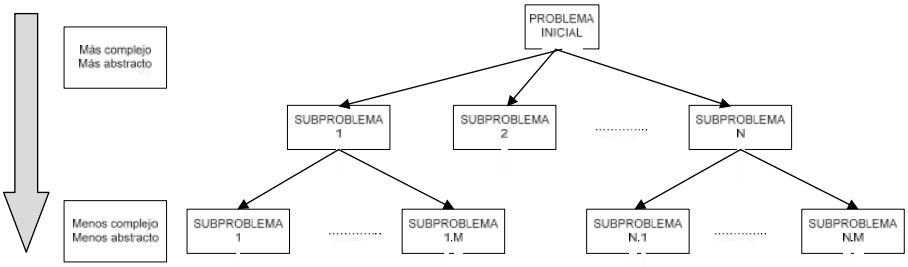
$$montoTotal = subtotal + montoIGV$$

1.1.2. Análisis

¿Qué debo hacer para resolver el problema?

En esta fase se utiliza la técnica de diseño descendente o *top-down*, en la cual una tarea grande se divide en pequeñas tareas o módulos. Es decir, se trata de un proceso en el que un problema se refina —o descompone— en una serie de pasos sucesivos o estructuras jerárquicas[4]. A este tipo de proceso inicial se le denomina «diagrama de módulos».

Ilustración 1.2 Diagrama de módulos



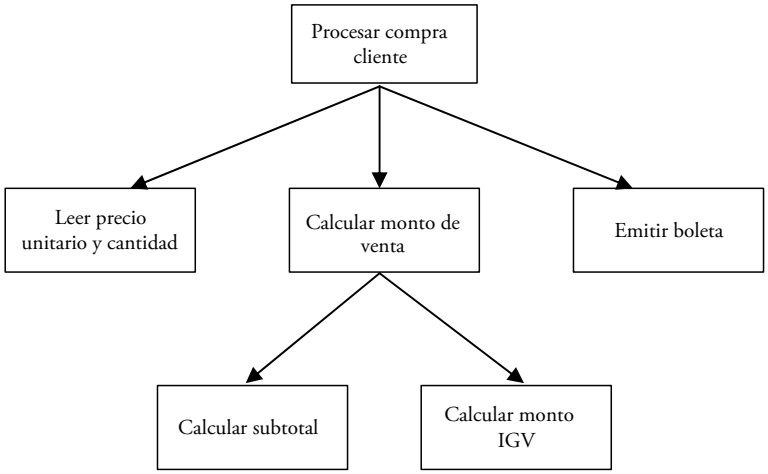
En algunos casos se puede hablar de análisis e incluir a la definición en él. No obstante, en el presente libro de texto los asumiremos como dos fases diferentes.

[3] No se utilizarán tildes en las fórmulas.

[4] Como ninguna persona piensa igual que otra, cada análisis será diferente, de modo que, posiblemente, se obtendrá infinidad de soluciones.

Seguidamente, para el ejemplo *Procesar compra de cliente*, un posible análisis sería:

Ilustración 1.3 Diagrama de módulos del ejemplo «Procesar compra de cliente»



Ahora bien, usualmente, y a modo de ayuda, para la creación del diagrama de módulos, nos podemos basar en las fórmulas halladas en la fase de la definición, agruparlas de distintas maneras e, incluso, establecer una dependencia entre ellas.

Finalmente, si lo que queremos es refinar aún más el diagrama, se pueden agregar opcionalmente los módulos encargados de leer los datos de entrada y mostrar las salidas.

1.1.3. Diseño

¿Cómo planteo el algoritmo*[5]*?

En esta tercera fase se diseña el algoritmo utilizando diversas representaciones:

- diagrama de flujo
- pseudocódigo
- lenguaje natural
- diagrama NS
- otros.

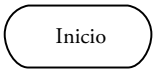
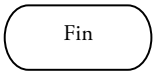
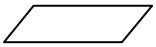
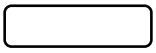
[5] «Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema». En Real Academia Española (RAE) (2001). *Diccionario de la lengua española.* Recuperado de http://lema.rae.es/drae/?val=algoritmo

No obstante, en este material estudiaremos únicamente dos representaciones: diagrama de flujo y pseudocódigo.

> *En el diagrama de flujo tendremos el uso de símbolos o distintas figuras geométricas, mientras que en el pseudocódigo tendremos solamente texto.*

A continuación, se detallan las acciones y los símbolos que se utilizarán en ambas representaciones

Tabla 1.1 Cuadro de acciones

Acción	Diagrama de flujo	Pseudocódigo
Iniciar el diagrama de flujo o el pseudocódigo		Inicio
Finalizar el diagrama de flujo o el pseudocódigo		Fin
Leer		Leer
Escribir o imprimir		Escribir o imprimir
Procesar		
Asignar		

Ahora bien, dentro de un proceso como el diagrama de flujo o el pseudocódigo se pueden tener tres tipos de instrucciones:

a) Procedimientos

Procesos encargados de devolver más de un valor o ninguno. Los procedimientos pueden mostrar, leer o modificar y devolver varios valores. Por tanto, instrucciones como «leer datos» y «mostrar salidas» siempre serán procedimientos. De este modo, según la representación que escojamos, este tipo de instrucciones se utiliza de la siguiente manera[6]:

[6] El nombre del procedimiento no debe incluir espacios en blanco y los parámetros deben estar separados por comas.

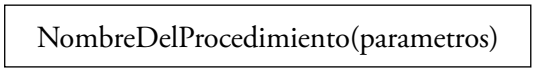
- Usando pseudocódigo[7]:

  NombreDelProcedimiento(parametros)

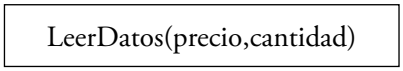
  Por ejemplo:

  LeerDatos(precio,cantidad)

- Usando diagrama de flujo:

  Por ejemplo:

b) Funciones

Procesos encargados de devolver un único valor. De acuerdo con la representación escogida, este tipo de procesos se utiliza de la siguiente manera[8]:

- Usando pseudocódigo:

  variable←NombreDeLaFuncion(parametros)

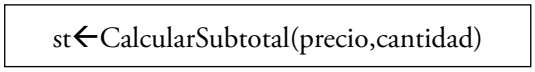
  Por ejemplo:

  st←CalcularSubtotal(precio,cantidad)

- Usando diagrama de flujo:

  ejemplo:

[7] No se utilizarán tildes en los diagramas de flujo ni en los pseudocódigos que se presenten a lo largo del presente libro.

[8] El nombre de la función no debe incluir espacios en blanco y los parámetros deben estar separados por comas.

c) Operaciones básicas de asignación

Instrucciones encargadas de guardar en una variable el resultado de operaciones matemáticas u otro tipo de operaciones[9]. Sobre la base de la representación, a este tipo de instrucciones se denomina de la siguiente manera[10]:

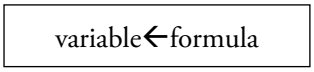
- Usando pseudocódigo:

variable←*formula*

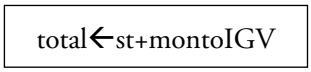
Por ejemplo:

total←*st+montoIGV*

- Usando diagrama de flujo:



Por ejemplo:



1.1.3.1. Diagrama de flujo

Como ya hemos adelantado, los diagramas de flujo son representaciones gráficas que utilizan distintos símbolos. Y dado que estos últimos pertenecen a una estructura, es preciso recordar que todas las fases de los diagramas deben concordar entre ellas; el diseño siempre debe estar en consonancia con el análisis.

Por ejemplo, si en el análisis se tiene el módulo «leer datos», entonces dicho proceso debe de estar en el diseño correspondiente y viceversa.

A continuación, veremos el diseño del diagrama de flujo para el ejemplo *Procesar compra de cliente*.

Antes de comenzar, es preciso señalar que todos los módulos establecidos en la fase análisis deberán contar con su diagrama de flujo respectivo. Por tanto, dado que existen tres niveles de módulos para el ejemplo *Procesar compra de cliente*, el primer diagrama que deberá diseñarse será el del módulo principal; es decir, el nivel más complejo. Seguidamente, se diseñarán los diagramas de flujo de los módulos del segundo nivel (que en este caso son tres) para, por último, elaborar los correspondientes a los módulos del tercer nivel (que en este caso son dos).

---

[9] Las operaciones y las fórmulas matemáticas pueden ser muy sencillas, como la mostrada en el ejemplo, o de lo más complicadas, como las instrucciones booleanas, entre otras.

[10] El nombre de la variable no debe incluir espacios en blanco y los parámetros deben estar separados por comas.

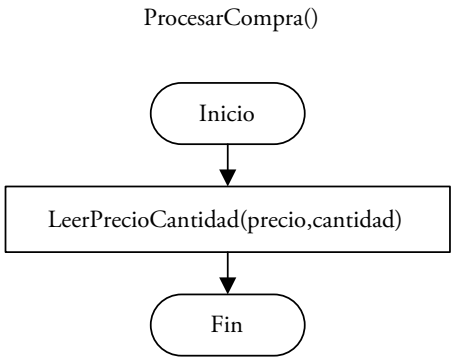
Diagrama de flujo del módulo principal «Procesar Compra Cliente»

–Paso 1

Se coloca el título del diagrama, aludiendo al módulo principal de la fase análisis, «Procesar compra cliente» (ver Ilustración 1.3), el cual se reformula en el diagrama de flujo como «ProcesarCompra»[11]. De esta forma, se ha obtenido el nombre del diagrama de flujo del módulo principal.

Acto seguido, se inicia el diseño a partir de los tres módulos obtenidos del principal en la fase análisis (ver Ilustración 1.3)[12]. Así, se crea el proceso correspondiente al módulo «Leer precio unitario y cantidad». Pero, para realizar dicho proceso, primero se necesita saber cuál es su objetivo y/o cuántos valores se calcularán en él. Además, debemos identificar las variables necesarias para su ejecución adecuada. En este caso, si lo que se quiere es «leer el precio y la cantidad», precisamente, estos dos componentes —«precio» y «cantidad»— serán sus parámetros. Como el proceso no calculará nada, sino que devolverá los valores modificados, se tratará de un «procedimiento» y el primer proceso del diagrama de flujo del módulo principal, ahora denominado «LeerPrecioCantidad», quedará así:

Ilustración 1.4 Módulo principal «ProcesarCompra»: módulo 1



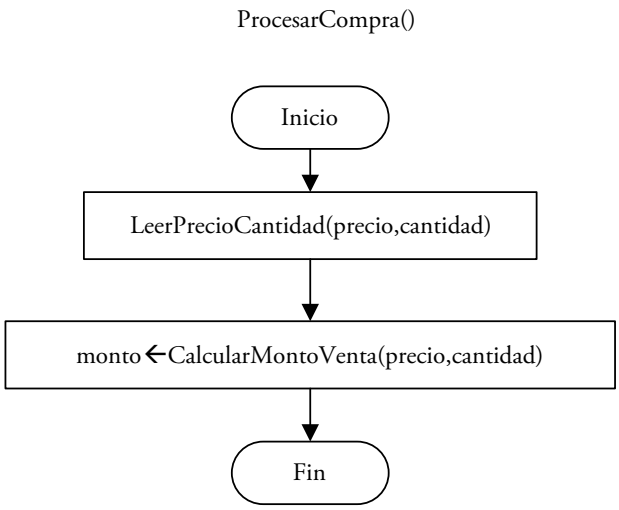
[11] El nombre de cada proceso en el diagrama de flujo no tiene que ser exactamente igual al del módulo correspondiente en el diagrama de módulos, pero sí debe mostrar la misma idea. Sin embargo, dicho nombre debe estar conformado por un conjunto de letras entre las que no deben de haber espacios en blanco. Además, debe empezar por una letra y ser descriptivo.

[12] De existir más niveles, estos se ignoran. Los únicos módulos que deberán tenerse en cuenta serán los presentes en el siguiente nivel.

–Paso 2

Después, se evalúa al módulo «Calcular Monto de Venta» de la fase análisis. Como este devolverá un único valor, se tratará de una función, en tanto que para calcularlo necesitaremos los mismos parámetros que en el paso anterior; esto es, el precio y la cantidad. Así, al diagrama del procedimiento anterior (ver Ilustración 1.4) se le añadirá la función «CalcularMontoVenta» y el diagrama del módulo principal será:

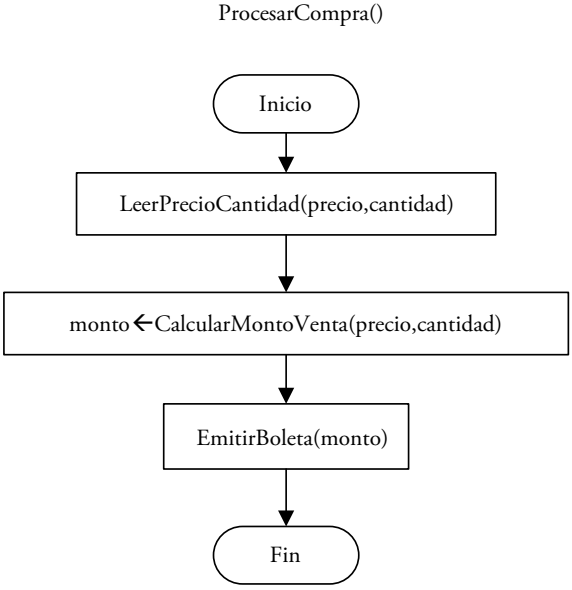
Ilustración 1.5 Módulo principal «ProcesarCompra»: módulo 2



–Paso 3

El siguiente paso consiste en verificar que los parámetros colocados se hayan leído o calculado previamente o se calculen dentro del proceso (en este caso, los dos parámetros ya fueron leídos). Luego, se evalúa al tercer módulo del segundo nivel del análisis: «Emitir boleta». Si bien este no devolverá ningún valor, sí lo imprimirá y, además, el valor escrito o impreso será el monto. De esta manera, al diagrama de flujo anterior (ver Ilustración 1.5) se le añadirá el procedimiento «EmitirBoleta» y el diagrama del módulo principal, finalmente, quedará de este modo:

Ilustración 1.6 Módulo principal «ProcesarCompra»: módulo 3



–Paso 4

Como ya se ha dicho en el paso anterior, debemos verificar que los parámetros colocados ya se hayan leído y/o calculado, o se lean o calculen dentro del proceso (en este caso, el parámetro «monto» ya fue calculado). Asimismo, como este es el diagrama del módulo principal, revisaremos que se lean todos los datos de entrada y se escriban las salidas. Y, dado que estas acciones sí se llevan a cabo, el siguiente paso supone realizar el diseño de los siguientes módulos de la fase análisis, verificando siempre que se devuelvan o modifiquen los valores adecuados.

> *Podríamos llamar a los procesos (funciones o procedimientos) con un nombre completamente diferente al de los módulos de la fase análisis, pero eso dificultaría la implementación del diseño en sí.*
>
> *De otra parte, los parámetros que se coloquen en dichos procesos deben responder a la pregunta ¿qué necesito para calcular el valor o los valores por devolver? O ¿qué necesito para realizar esta acción? Por ejemplo, para la interrogante ¿qué necesito para imprimir la boleta?, la respuesta será el «monto total»; entonces, ese será el parámetro del procesamiento «EmitirBoleta». Si, por ejemplo, se quisiera calcular el total, se necesitará como parámetros al precio unitario y a la cantidad del producto. Lo mismo sucede con la lectura: si la pregunta es ¿qué datos son los que necesito leer?, la respuesta será «todos los datos de entrada hallados en*

la fase de la definición»; por tanto, esos serán los parámetros del proceso «LeerPrecioCantidad».

Diagrama de flujo del módulo «Leer precio unitario y cantidad»

–Paso 1

A continuación, se presentan los pasos para elaborar el diagrama de flujo del módulo «Leer precio unitario y cantidad». Así pues, en este proceso se leerán los valores de «precio» y «cantidad» y se asignarán a las variables correspondientes (aquí es preciso señalar que, como se trata de un diagrama de flujo, utilizaremos el paralelogramo para realizar dicha lectura —ver Tabla 1.1–). Luego, el diagrama del procedimiento será el siguiente:

Ilustración 1.7 Procedimiento «LeerPrecioCantidad»

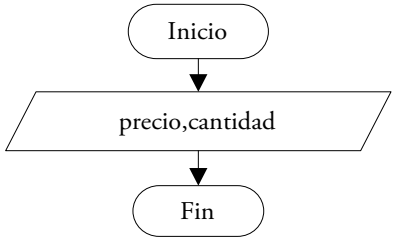
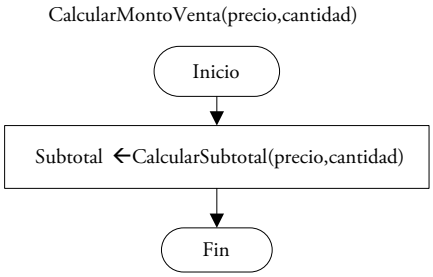


Diagrama de flujo del módulo «Calcular monto de venta»

–Paso 1

Seguidamente, elaboraremos el diagrama del módulo «Calcular Monto de Venta», que, como ya se ha observado, es una función, debido a que devuelve un único valor. Empero, lo más importante de dicha función es que tiene dos módulos en el siguiente nivel (ver Ilustración 1.3). Por ello, la evaluación de este módulo es similar a la realizada para el principal. Así, este diagrama se iniciará con el módulo «Calcular subtotal», el cual necesita de los parámetros «precio» y «cantidad» para devolver un único valor (por tanto, se trata de una función), es decir, «el subtotal»:

Ilustración 1.8 Función «CalcularMontoVenta»: módulo 1

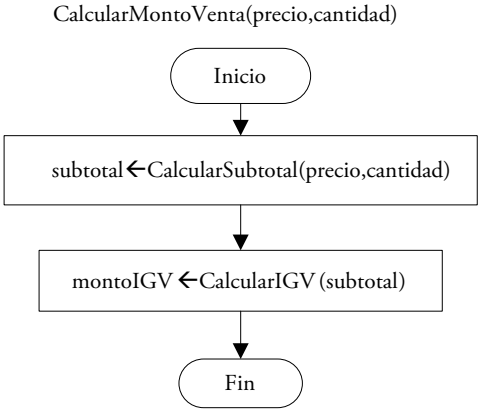


Acto seguido, se debe verificar que los parámetros colocados se hayan obtenido previamente —como ocurre en este caso, puesto que los dos parámetros del módulo «Calcular monto de venta» han sido leídos fuera de este módulo—. De ser así, se podrán utilizar directamente.

–Paso 2

De igual forma, se repite el proceso en el siguiente módulo «Calcular monto IGV», el cual, a su vez, devolverá un único valor, es decir, «montoIGV», y para ello necesita del parámetro «subtotal» (ya calculado), de modo que al diagrama de flujo anterior se le añadirá una nueva función y quedará así:

Ilustración 1.9 Función «CalcularMontoVenta»: módulo 2



Al igual que en el último paso del diagrama del módulo principal, una vez incluidos todos los módulos de la función «CalcularMontoVenta», debemos verificar que el valor que buscábamos sea devuelto; es por ello que se lleva a cabo la asignación del valor por devolver. Ahora bien, como se trata de una función, se utiliza el nombre de esta para retornar el valor:

Ilustración 1.10 Función «CalcularMontoVenta»

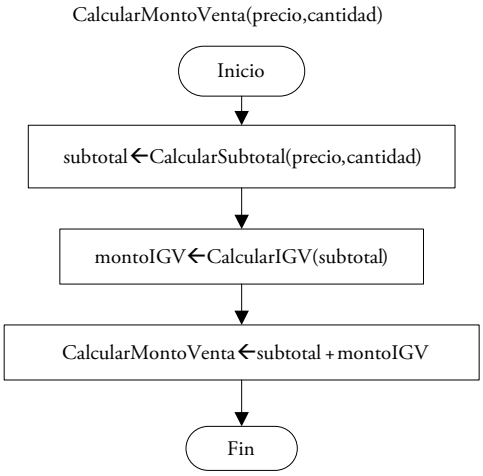


Diagrama de flujo del módulo «Emitir boleta»

–Paso 1

Luego, se desarrolla el diagrama de flujo del módulo «Emitir boleta». Al igual que en los pasos previos, como en este proceso se imprimirá o escribirá el valor del monto total, será necesario colocar el símbolo correspondiente a la impresión (ver Tabla 1.1). Es por ello que el diagrama de este procedimiento quedará como se muestra a continuación:

Ilustración 1.11 Procedimiento «EmitirBoleta»

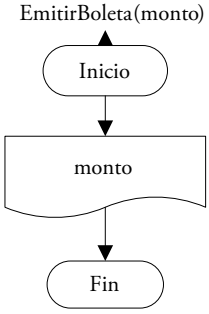


Diagrama de flujo del módulo «Calcular subtotal»

–Paso 1

Ahora bien, ya se han llevado a cabo cuatro de los seis módulos planteados en la fase análisis que pertenecen al primer y segundo nivel. Por tanto, en este paso se iniciará el diseño del tercer nivel con el diagrama correspondiente al módulo «Calcular

subtotal». Se debe tener en cuenta que en este nivel, al no haber submódulos, solo se pueden utilizar operaciones básicas al lado derecho de la asignación. Como se podrá observar (ver Ilustración 1.12), se siguen los mismos pasos anteriores, se devuelve el único valor por calcular y se verifica que se cuente con los valores necesarios para poder aplicar la fórmula.

Ilustración 1.12 Función «CalcularSubtotal»

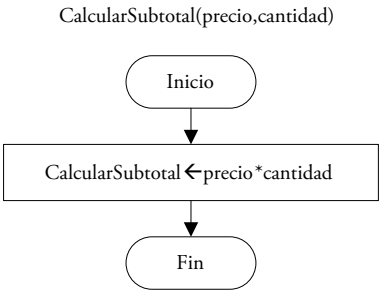
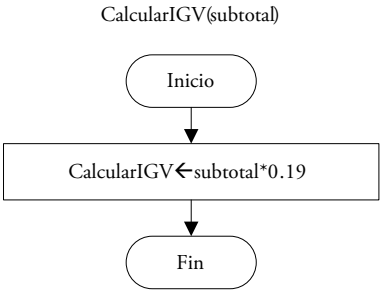


Diagrama de flujo del módulo «Calcular monto IGV»

–Paso 1

Para finalizar, se sigue el mismo proceso que en los pasos anteriores en el módulo «Calcular monto IGV», pero, esta vez, teniendo en cuenta que «0.19» es considerado un *valor constante*, por lo que se colocará directamente sin hacer uso de una variable:

Ilustración 1.13 Función «CalcularIGV»



1.1.3.2. Pseudocódigo

Por otro lado, el pseudocódigo, a diferencia de los diagramas de flujo, no se representa de manera gráfica, sino que solamente utiliza texto y, por tanto, necesita de una menor cantidad de pasos. A continuación, se mostrará el pseudocódigo del ejemplo anterior.

Paso 1

Inicio ProcesarCompra()
LeerPrecioCantidad(precio,cantidad)
monto←CalcularMontoVenta(precio,cantidad)
EmitirBoleta(monto)
Fin ProcesarCompra

Paso 2

Inicio LeerPrecioCantidad(precio,cantidad)
Leer precio,cantidad
Fin LeerPrecioCantidad

Paso 3

Inicio CalcularMontoVenta(precio,cantidad)
subtotal ←CalcularSubtotal(precio,cantidad)
montoIGV←CalcularIGV(subtotal)
CalcularMontoVenta ←subtotal+montoIGV
Fin CalcularMontoVenta

Paso 4

Inicio EmitirBoleta(monto)
Escribir monto
Fin EmitirBoleta

Paso 5

Inicio CalcularSubtotal(precio,cantidad)
*CalcularSubtotal←precio*cantidad*
Fin CalcularSubtotal

Paso 6

Inicio CalcularIGV(subtotal)
*CalcularIGV←subtotal*0,19*
Fin CalcularIGV

En ciertos casos se opta por colocar solo la palabra «Fin» al finalizar el proceso. En el presente libro para no confundir el bloque de «Fin» se colocará inmediatamente después de esta palabra el nombre del proceso que culmina.

1.1.4. Codificación

¿Cómo elaboro la programación?

En esta fase se realiza la implementación del algoritmo, pero usando un lenguaje de programación, que, como ya hemos anunciado en el prefacio, será VBA (para más detalle ver el capítulo 2).

1.1.5. Depuración

¿El programa resuelve correctamente el problema planteado?

En esta fase es necesario que se hagan las pruebas para verificar que la solución propuesta es la correcta. De lo contrario, se deberá realizar un seguimiento a las fases anteriores con el fin de hallar el error. Como se verá más adelante, esta depuración se suele hacer en la computadora, pero también es posible hacerla por escrito.

1.1.6. Documentación

¿Qué documentos necesito para que el programa sea accesible?

Finalmente, existen dos tipos de documentación: la interna y la externa.

1.1.6.1. Documentación interna

Este tipo de documentación, que trata de responder a la interrogante: «Si otra persona viera el programa, ¿lo entendería?», sirve para que el programador o el desarrollador entienda el *código fuente*; es decir, las instrucciones de ejecución del programa. En otras palabras, se le llama documentación *interna* porque sus lectores finales son personas que participan, de una u otra manera, en la elaboración del programa. Por consiguiente, se desarrolla a través de comentarios, los cuales deben ser descriptivos y generales. No obstante, no hay que olvidar que en niveles superiores de programación el código fuente se explica por sí solo, siempre y cuando los estándares de programación sean utilizados correctamente.

1.1.6.2. Documentación externa

Usualmente este tipo de documentación, que trata de responder a la pregunta: «Si un usuario empleara la aplicación, ¿podrá hacerlo correctamente?», implica un manual de usuario que incluye las indicaciones necesarias para utilizar el programa o la aplicación. En otras palabras, se le llama *externa* porque sus lectores finales son personas que no elaboran el programa.

1.2. Ejercicios desarrollados

En esta sección se desarrollarán ejercicios utilizando los pasos explicados previamente.

1.2.1. María quiere besar a José

María, como siempre, quiere darle un beso a José, quien se encuentra al borde del muelle. Es la primera vez que intenta alcanzarlo usando una lancha de motor, así que necesita tomar nota de los resultados. Como dicha lancha se encuentra en reposo ($v_0 = 0$ y $x_0 = 0$), María planea acelerar esta vez a 3 m/s² durante 0.13 min. Felizmente, luego del intento, logra su objetivo y besa a José.

María, obviamente, ha quedado encantada con lo ocurrido y ha decidido repetir su hazaña mañana, pero necesita saber a qué distancia del muelle deberá colocar la lancha y cuál será la velocidad media que deberá utilizar. Como somos amigos(as) de María hemos decidido ayudarla elaborando una pequeña aplicación en VBA.

Pero, antes de ayudarla, debemos tener en cuenta las siguientes fórmulas:

$$V = Vo + a * t \ldots\ldots\ldots (velocidad)$$
$$\bar{v} = (v + Vo)/2 \ldots\ldots\ldots (velocidad\ media)$$
$$x = Xo + \bar{v} * t \ldots\ldots\ldots (distancia)$$

Donde:
–«V_0» es igual a velocidad inicial
–«a» es igual a aceleración
–«t» es igual a tiempo
–«X_0» es igual a distancia inicial

Además, las unidades de medida de cada uno de los valores serán:

$$a \rightarrow m/s\text{^}2 \qquad v \rightarrow m/s \qquad x \rightarrow m \qquad t \rightarrow s$$

1.2.1.1. Paso 1

Hallamos la definición. Para ello, luego de leer el problema, identificamos la idea general de este (la explicación); después, los datos dados en el enunciado («de entrada»), los valores que debemos calcular («salidas») y, posteriormente, las fórmulas que necesitamos para calcularlos («fórmulas»).

- Definición

  Explicación: hallar la distancia y la velocidad media necesarias para poder darle un beso a José.

  Datos de entrada: aceleración (a) y tiempo en minutos (Tm)

  Salidas: distancia (dist) y velocidad media (Vm)

  Fórmulas:

$$V=Vo+a*t \ldots\ldots (velocidad)$$

$$\bar{v}=(v+Vo)/2\ldots\ldots (velocidad\ media)$$

$$x=Xo+\bar{v}*t\ldots\ldots (distancia)$$

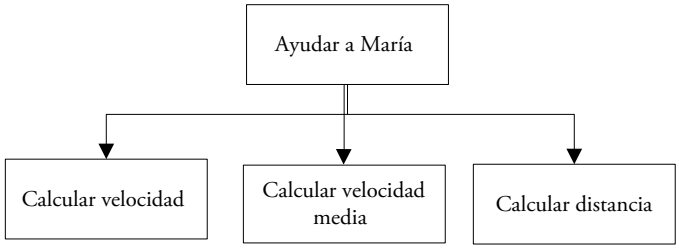
$$t=tm*60\ldots\ldots (tiempo\ en\ segundos)$$

1.2.1.2. Paso 2

Realizamos el análisis mediante un diagrama de módulos. Una manera muy sencilla de identificar los módulos que elaboraremos es utilizar las fórmulas halladas en la fase anterior. Por consiguiente, contamos con cuatro módulos: tres generados por las fórmulas y uno por el programa principal. En caso quisiéramos generar más módulos, podríamos agregar los de «LeerDatos» y/o «MostrarSalidas». Por el contrario, si lo que queremos es tener menos módulos, podríamos juntar dos o más fórmulas, o no considerar algunas:

- Análisis

Ilustración 1.14 Diagrama de módulos del ejercicio desarrollado «María quiere besar a José»



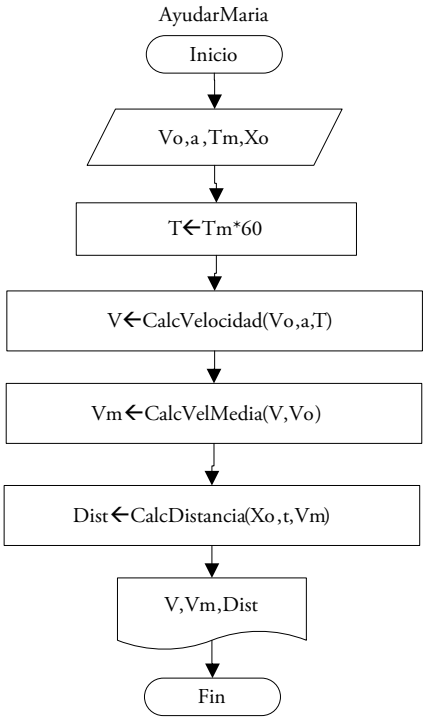
1.2.1.3. Paso 3

Seguidamente, realizamos el diseño, ya sea usando un diagrama de flujo o un pseudocódigo. Para este ejercicio utilizaremos el primero. Así, partimos del módulo principal y colocamos los módulos correspondientes, en este caso, «CalcVelocidad», «CalcVelMedia» y «CalcDistancia» (de acuerdo con la Ilustración 1.14), con sus

respectivas llamadas y parámetros. Luego, analizamos la procedencia de dichos parámetros; es decir, leemos los datos de entrada necesarios con los que, después, calcularemos los valores y escribiremos todas las salidas.

- Diseño

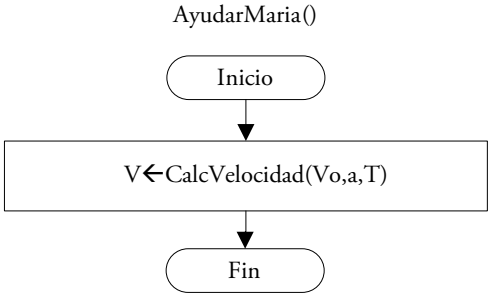
Ilustración 1.15 Diseño del diagrama de flujo: Proceso principal «AyudarMaria»



Paso 3.1

Ahora desarrollaremos uno por uno los pasos para obtener el diseño mostrado en la figura anterior. En primer lugar, colocaremos el primer módulo «CalcVelocidad» y responderemos a dos preguntas: ¿cuántos valores devuelve? y ¿qué necesitamos para calcular los valores por devolver? Luego, con las respuestas a estas interrogantes determinaremos que este módulo es una función, pues devuelve un único valor, «la velocidad», y que, además, necesitamos a «la velocidad inicial», «la aceleración» y «el tiempo en segundos». Con todos estos datos, el diagrama quedará como se muestra:

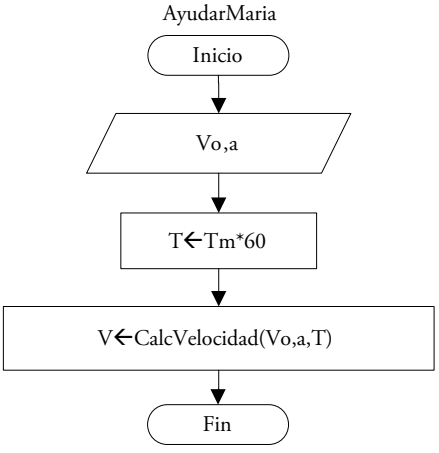
Ilustración 1.16 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.1



Paso 3.2

En segundo lugar, evaluamos si tenemos o necesitamos calcular los parámetros hallados. En este caso, «la velocidad inicial» y «la aceleración» son datos de entrada, pero «el tiempo en segundos» necesita ser calculado, ya que el tiempo dado al inicio estaba en minutos. En consecuencia, el diagrama resultará de la siguiente manera:

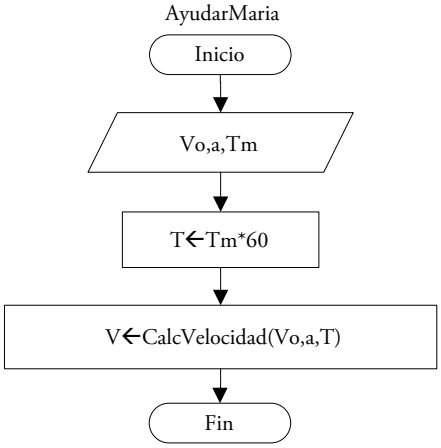
Ilustración 1.17 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.2



Paso 3.3

Como ya se ha mencionado, por cada proceso agregado debemos evaluar las variables y/o los parámetros. En este caso, aparece «Tm», el tiempo en minutos que necesita modificarse de alguna manera para obtener el tiempo en segundos. Así, dado que es un dato de entrada, debe figurar junto con los otros datos similares al inicio del diagrama:

Ilustración 1.18 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.3

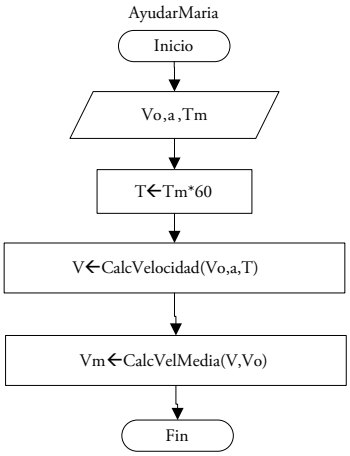


Hemos colocado el paralelogramo para leer (ver Tabla 1.1) porque no existe un módulo «leer datos», y lo mismo ocurre con la conversión del tiempo, cuyo símbolo es el de un proceso que, en este caso, se define como una operación básica de asignación. De esta forma, el análisis coincide con el diseño planteado.

Paso 3.4

Luego, seguimos los mismos pasos anteriores pero con el siguiente módulo: «CalcVelMedia». Como este último devuelve un único valor (función), necesita de los parámetros «velocidad» y «velocidad inicial». Así, dado que el primer parámetro ya había sido calculado en la función anterior y que el segundo es un dato de entrada (por lo que debe de estar dentro del paralelogramo), se pueden usar directamente. Por tanto, el diagrama será el siguiente:

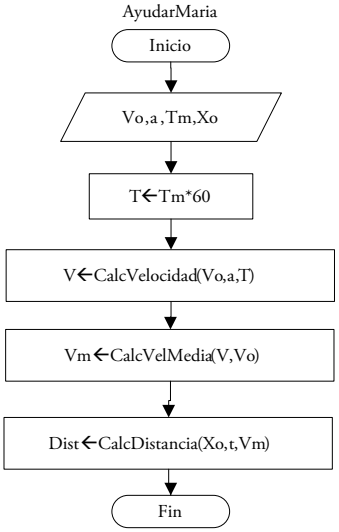
Ilustración 1.19 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.4



Paso 3.5

Posteriormente, elaboramos el último módulo, «calcDistancia», que devuelve un único valor (función), para lo cual, necesita de «X_0», «la velocidad media» y «el tiempo en segundos»; donde «X_0» es un dato de entrada (por lo que debe de estar dentro del paralelogramo), mientras que «la velocidad media» y «el tiempo en segundos» ya fueron calculados en los procesos anteriores. De este modo, podemos utilizar estos tres parámetros directamente:

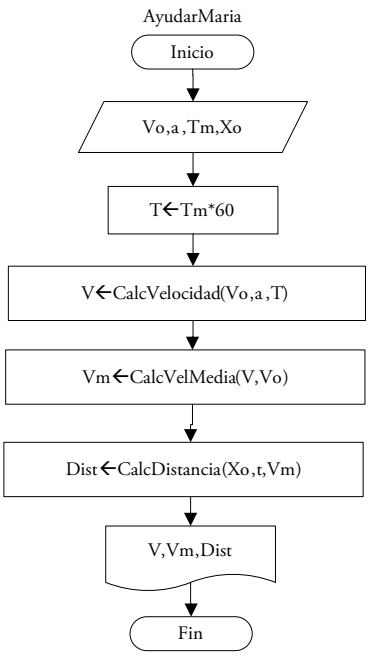
Ilustración 1.20 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.5



Paso 3.6

Como ya hemos incluido todos los módulos del segundo nivel (tres, según el diagrama de módulos), ahora debemos verificar que hayamos escrito las salidas y leído los datos de entrada colocados en la definición. Por consiguiente, el diseño final será el que ya habíamos elaborado en la Ilustración 1.15 y que, por razones didácticas, repetiremos aquí:

Ilustración 1.21 Diseño del diagrama de flujo: Proceso principal «AyudarMaria». Paso 3.6



Paso 3.7

Ahora bien, en esta última etapa, repetiremos los pasos del 3.1 al 3.5 hechos para el módulo o proceso principal, pero para cada uno de los módulos restantes del diseño, que, como ya hemos visto, son funciones. En este punto, recordemos que como la función siempre devuelve un solo valor debemos colocar el nombre de esta y asignarle a cada una el valor de retorno.

Ilustración 1.22 Diseño del diagrama de flujo: Función «CalcVelocidad»

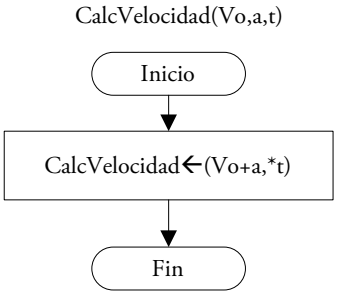


Ilustración 1.23 Diseño del diagrama de flujo: Función «CalcVelMedia»

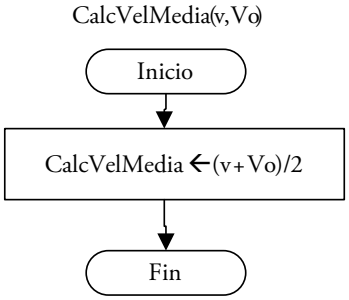
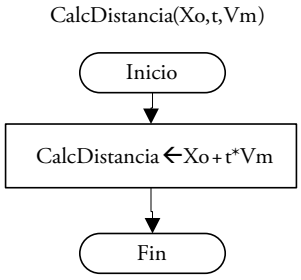


Ilustración 1.24 Diseño del diagrama de flujo: Función «CalcDistancia»



1.2.2. Satélite artificial

Un satélite artificial se mueve alrededor de un planeta, describiendo una órbita circular de 419 kilómetros en un tiempo de 42,47 horas. Se quiere que calculemos la fuerza gravitatoria que actúa sobre el satélite, así como las energías cinética y potencial de este en su órbita. Como no conocemos la masa del planeta, no podemos aplicar la ley de gravitación universal, pero sabemos que para que un cuerpo se mantenga en una órbita, el valor de su fuerza centrípeta debe coincidir con el valor de la fuerza dada por esta ley. Por ello, podemos decir que:

Fuerza gravitatoria = Fuerza centripeta
*Fuerza centripeta = m*v^2/r*
*v = (2*Pi*r)/T*

Donde:
–«v» es la velocidad a partir del radio de la órbita medida en m/s
–«r» es el radio en m
–«m» es la masa en kg
–«T» es el periodo medido en s

Adicionalmente, sabemos que:

*Ec = 0.5*m*v^2*
*Ep= –(v^2*r)*m/r*

Donde:
–«Ec» es la energía cinética
–«Ep» es la energía potencial

Ilustración 1.25 Ejercicio desarrollado: «Satélite artificial»

	A	B	C	D	E
1	**Datos de entrada**			**Salidas**	
2	Masa del satélite	300		Fuerza gravitatoria	0.21
3	Radio de la órbita circular	419		Energía cinética	44896.20
4	Tiempo	42.27		Energía potencial	-89792.39

1.2.2.1. Paso 1

Hallamos la definición. Luego de leer el problema, identificamos la idea general de este (la explicación), los datos dados en el enunciado (datos de entrada), los valores que se deben calcular (salidas) y las fórmulas necesarias para obtener dichos valores (fórmulas).

- Definición

  Explicación: hallar la fuerza gravitatoria, la energía cinética y la energía potencial.

  Datos de entrada: masa del satélite en kg, el radio de la órbita en km y el tiempo en h.

  Salidas: fuerza gravitatoria, energía potencial y energía cinética.

Fórmulas:

$$Fuerza\ gravitatoria=m*v\^2/r$$

$$v=(2*Pi*r)/T$$

$$Ec=0.5*m*v\^2$$

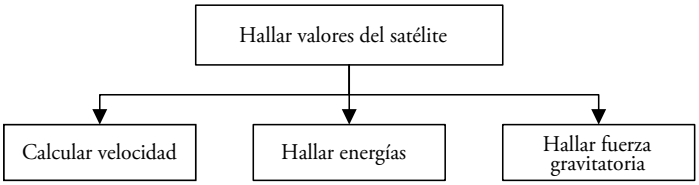
$$Ep=-(v\^2*r)*m/r$$

1.2.2.2. Paso 2

Realizamos el análisis con el diagrama de módulos. Una forma muy sencilla de identificar los módulos es utilizar las fórmulas halladas en la fase anterior. De esta manera tenemos cinco módulos, cuatro generados por las fórmulas y uno por el programa principal. Pero estas fórmulas pueden agruparse de distintas maneras; en este caso, manejaremos solo cuatro módulos, incluido el principal, puesto que hemos agrupado las fórmulas que calculan a las energías:

- Análisis

Ilustración 1.26 Diagrama de módulos del ejercicio desarrollado «Satélite artificial»



1.2.2.3. Paso 3

Elaboramos el diseño, ya sea usando diagrama de flujo o pseudocódigo. En este ejercicio utilizaremos el pseudocódigo. Así, partimos del módulo principal y, debajo de este, colocaremos a los módulos correspondientes, en este caso, «CalcVelocidad», «HallarEnergias» y «HallarFuerza», con sus respectivas llamadas y parámetros. Luego, analizaremos la procedencia de esos parámetros; es decir, leeremos los datos de entrada que serán necesarios para calcular y mostrar los valores que se nos piden o calcularemos directamente otros valores.

De este modo iniciaremos el pseudocódigo del módulo principal con la palabra «Inicio», seguida del nombre de este y sus respectivos parámetros, y lo terminaremos con la palabra «Fin», seguida únicamente del nombre del módulo.

Paso 3.1

Iniciamos el diseño con el primer submódulo del módulo principal, esto es, «CalcVelocidad». Como se puede observar, este solo devolverá un valor y, por tanto, es una función que, además, para ser calculada necesita según la fórmula, el radio en metros y el tiempo en segundos. De otra parte, debemos subrayar que el módulo principal no tiene parámetros, por lo que luego de su nombre hemos colocado paréntesis sin contenido.

- Diseño

Inicio HallarValoresSatelite()
V←CalcVelocidad(r,T)
Fin HallarValoresSatelite

Paso 3.2

A continuación, evaluamos si tenemos o necesitamos calcular los parámetros de la función anterior. En este caso, como ya se ha visto en el paso previo, necesitamos el radio en metros y el tiempo en segundos, pero en nuestros datos de entrada ambos parámetros se encuentran en kilómetros y en horas, respectivamente. En consecuencia, será necesario calcular estos dos valores por medio de operaciones básicas de asignación y se utilizará tanto el radio en km como el tiempo en horas, que eran datos de entrada, por lo que serán leídos directamente sin el uso de un procedimiento «LeerDatos» (pues dicho módulo no existe). Con esto, el pseudocódigo resultará así:

Inicio HallarValoresSatelite()
Leer radioenkm,Thoras
*r←radioenkm*1000*
*T←Thoras*3600*
V←CalcVelocidad(r,T)
Fin HallarValoresSatelite

Paso 3.3

Es momento de que incluyamos al segundo módulo del proceso principal, «HallarEnergias», el cual devolverá dos energías: la cinética y la potencial, por lo que se tratará de un procedimiento. Y, como podemos observar en las fórmulas (ver Paso 1 de este ejercicio), si necesitamos la masa en kg, la velocidad en m/s y el radio en m, estos serán los parámetros junto con las dos energías. De esta manera, el pseudocódigo obtenido será:

```
Inicio HallarValoresSatelite()
    Leer radioenkm,Thoras
    r←radioenkm*1000
    T←Thoras*3600
    V←CalcVelocidad(r,T)
    HallarEnergias(m,V,r,Ec,Ep)
Fin HallarValoresSatelite
```

Paso 3.4

Acto seguido, evaluamos los parámetros que hemos colocado en el procedimiento anterior: la masa en kg es un dato de entrada, por lo que es necesario leerla (es decir, se añade a la operación de lectura); la velocidad y el radio en metros ya se calcularon previamente, y, por último, las energías son las variables por devolver. De este modo, el pseudocódigo quedará como se muestra a continuación:

```
Inicio HallarValoresSatelite()
    Leer radioenkm,Thoras,m
    r←radioenkm*1000
    T←Thoras*3600
    V←CalcVelocidad(r,T)
    HallarEnergias(m,V,r,Ec,Ep)
Fin HallarValoresSatelite
```

Paso 3.5

Para incluir el último módulo, «HallarFuerza», necesitamos (como ya hemos leído y calculado a la masa en kg), la velocidad y el radio en m. Luego, como solo queremos devolver un valor (fuerza) se tratará de una función. Finalmente, debemos escribir las salidas, pero, como no hay un módulo para ello, se escriben directamente sin el uso de un procedimiento adicional. Por tanto, el pseudocódigo resultará de la siguiente manera:

```
Inicio HallarValoresSatelite()
    Leer radioenKm,Thoras,m
    r←radioenKm*1000
    T←Thoras*3600
    V←CalcVelocidad(r,T)
    HallarEnergias(m,V,r,Ec,Ep)
    F←HallarFuerza(m,v,r)
    Escribir F,Ec,Ep
Fin HallarValoresSatelite
```

En este punto verificaremos que se hayan escrito todas las salidas mencionadas en la definición y que estas hayan sido calculadas previamente. Como las condiciones anteriores se cumplen, el desarrollo del módulo principal no sufrirá ningún cambio.

1.2.2.4. Paso 4

Luego de diseñar el diagrama del módulo principal, realizamos los pasos anteriores para cada uno de los módulos restantes.

```
Inicio CalcVelocidad(r,T)
    CalcVelocidad←(2*3.1416*r)/T
Fin CalcVelocidad

Inicio HallarFuerza(m,v,r)
    HallarFuerza←(m*v^2)/r
Fin HallarFuerza
```

Paso 4.1

Debemos tener en cuenta que el orden de los parámetros de un mismo subprograma (otro nombre para los módulos distintos del principal) debe coincidir entre un módulo y otro. Además, al igual que en el diagrama de flujo, cuando se devuelve un valor utilizamos una función, pero, si se devuelven dos o más valores, debemos usar un procedimiento que, como ocurre en este caso, tenga a dichos parámetros en los paréntesis luego de su nombre:

```
Inicio HallarEnergias(m,V,r,Ec,Ep)
    Ec←0.5*m*v^2
    Ep←-(v^2*r)*m/r
Fin HallarEnergias
```

1.2.3. ¿Una bolsa de aire protege realmente a un conductor?

Estime qué tan rápido se debe inflar la bolsa de aire para proteger efectivamente a un conductor en el caso de que sufra una colisión frontal. Al respecto, se sabe que la rapidez inicial del auto es 100 km/h y que recorre 1 m. Para el cálculo de la aceleración y el tiempo que demorará utilice las siguientes fórmulas:

$$a=-Vo/2x$$
$$t=(t-Vo)/a$$

Donde:
–«V_0» es la rapidez inicial en m/s
–«x» es la distancia recorrida en m
–«V=0» es la velocidad en la que el auto llega al reposo
–«a» es la aceleración previamente calculada en m/s²

Intente describir el desarrollo del diagrama de flujo propuesto siguiendo los pasos de los ejercicios desarrollados previamente.

- Definición

  Explicación: calcular qué tan rápido se debe inflar la bolsa de aire para proteger efectivamente al conductor.

  Datos de entrada: rapidez inicial en km/h, distancia recorrida en m (x), velocidad en reposo.

  Salidas: tiempo que demorará (t).

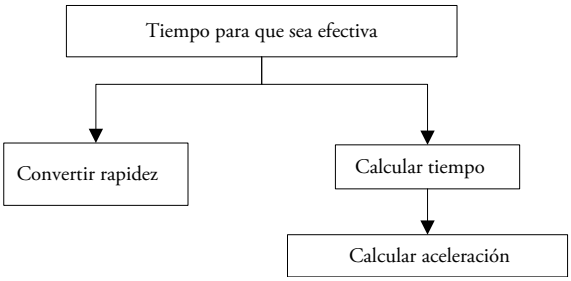
  Fórmulas:

$$t=(V-Vo)/a$$

$$a=-Vo/2x$$

- Análisis

Ilustración 1.27 Diagrama de módulos del ejercicio desarrollado «Bolsa de aire»



- Diseño

Ilustración 1.28 Diagrama de flujo del ejercicio desarrollado «Bolsa de aire»: Subprograma principal «CalcularTiempoEfectivo»

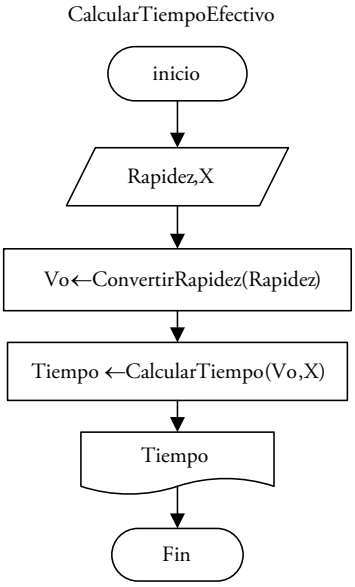


Ilustración 1.29 Diagrama de flujo del ejercicio desarrollado «Bolsa de aire»: Función «ConvertirRapidez»

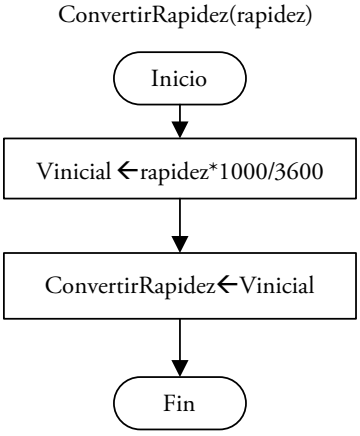


Ilustración 1.30 Diagrama de flujo del ejercicio desarrollado «Bolsa de aire»: Función «CalcularAceleracion»

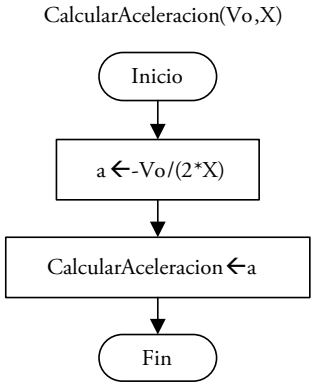
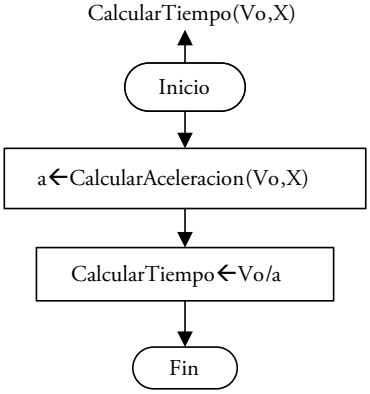


Ilustración 1.31 Diagrama de flujo del ejercicio desarrollado «Bolsa de aire»: Función «CalcularTiempo»



1.3. Ejercicios propuestos

Elabore la definición, el análisis y el diseño de los siguientes ejercicios.

1.3.1. Pintar fachada

Se desea pintar la fachada (techo y paredes frontales) de la casa en 2D que se muestra a continuación y se sabe que sus ventanas son cuadradas, así como los anchos y altos de techo, puerta, pared y ventanas. Por tanto, se le pide hallar el área en cuestión.

Ilustración 1.32 Ejercicio propuesto: «Pintar fachada»

1.3.2. Cálculo de raíces

Dada la ecuación de segundo grado *ax^2+bx+c*, se calcula el discriminante:

disc=b^2-4ac

Por el planteamiento del problema asuma que el único caso posible es que el discriminante sea mayor a 0 y calcule el valor de las dos raíces utilizando las siguientes fórmulas:

x1= (-b+(disc^0.5))/2a

x2= (-b-(disc^0.5))/2a

1.3.3. Temperatura en grados Fahrenheit

Se conoce la temperatura en grados Fahrenheit y se debe convertir a grados Celsius. Para ello utilizaremos la siguiente fórmula:

°C=(5(°F-32))/9*

1.3.4. Ventas de vendedor

Cada vendedor tiene un monto por venta anual en el primer y segundo semestre. Se desea calcular el promedio y la venta total de cada uno de los vendedores.

Ilustración 1.33 Ejercicio propuesto: «Ventas de vendedor»

	A	B	C	D	E
1	Venta anual	**Semestre 1**	Semestre 2	Promedio	Venta total
2	Juan Pérez	228,000.00	13,000.00	120,500.00	241,000.00
3	María Morales	298,321.00	1,429.00	149,875.00	299,750.00

1.3.5. Margen bruto

Se quiere calcular la venta neta y el margen bruto del producto, sobre la base de los datos mostrados en la siguiente hoja de cálculo.

Ilustración 1.34 Ejercicio propuesto: «Margen bruto»

	A	B
1	**Resultados por productos**	**Producto**
2	Ventas	50
3	Devoluciones	5.5
4	Ventas netas	44.5
5	Amortizaciones	10.5
6	Costos de producción	9.00
7	Margen bruto	25

1.3.6. Jugadora empuja un disco

Una jugadora empuja un disco de 250 g que inicialmente está en reposo de manera que una fuerza horizontal constante de 6N actúa sobre él durante una distancia de 50 cm.

Las fórmulas necesarias para los cálculos son:

$$Ko=1/2*(m*(Vo\text{^}2)$$
$$W=F*d$$
$$K=W+Ko$$
$$v=(2K/m)\text{^}0.5$$

Ilustración 1.35 Ejercicio propuesto: «Jugadora empuja un disco»

	A	B	
1	**Datos de entrada**		
2	Masa	250	gramos
3	F	6	N
4	Distancia	0.0005	km
5	Vo	0	m/s
6			
7	**Salidas**		
8	Energía cinética	3	joules
9	Rapidez(v)	4.9	m/s
10	Ko	0	joules
11	Trabajo efectuado(W)	3	joules

1.3.7. Persecución

Un automóvil, que viaja con rapidez constante en m/s, pasa un letrero panorámico detrás del cual se encuentra escondido un patrullero. Un segundo después de que el automóvil con exceso de velocidad pasa al letrero, el patrullero inicia la persecución con una aceleración constante en m/s^2. Sabiendo cuánto tiempo le toma al patrullero alcanzar al automóvil, se quiere saber qué tan rápido va el patrullero en ese momento y cuál es el valor de X_{auto}. Para ello, se dan las siguientes fórmulas:

$$Xauto=Xo+Vo*t+(1/2)*a\_auto*t\text{^}2$$

$$Vpatrullero=Vo+a\_patrullero*t$$

Ilustración 1.36 Ejercicio propuesto: «Persecución»

	A	B	C	D	E	F
1		**Auto**	Patrullero		Salidas	
2	Xo	24	0		Xauto	429.6
3	Vo	24	0		Vpatrullero	50.7
4	a	0	3			
5	Tiempo de alcance	16.9				

Se le pide:

- Elaborar la definición.
- Elaborar el análisis con al menos cuatro módulos, incluido el principal.
- Elaborar el diseño, el cual debe coincidir con el análisis.

1.3.8. Inmueble

Cuando se solicita un préstamo hipotecario, el monto de la cuota total se calcula sobre la base de la suma del monto de amortización, los intereses, el seguro de desgravamen, el seguro de todo riesgo y los portes. María quiere comprarse un departamento (inmueble), pero desea saber cuánto le pagaría al banco si solicita un préstamo.

El banco ha calculado el valor de la cuota a partir de una tasa anual de interés, gracias a la cual María puede amortizar una cantidad mayor del préstamo en ciertos meses.

Ilustración 1.37 Ejercicio propuesto: «Inmueble»

	A	B
1	Costo del inmueble	250,000.00
2	Valor de una cuota	2,141.10
3	Duración del préstamo (años)	20
4	Porcentaje de seguros y otros	0.04
5	Cantidad de pagos hechos en un año	12
6	Cantidad de pagos hechos durante el préstamo	240
7	Monto total por pagar	513,864.03
8	Ganancia del banco	243,309.47
9	Porcentaje de ganancia	97%

Como María quiere saber cuántos pagos hará, y cuáles serán la ganancia del banco, el pago por seguros y portes, el monto total por pagar y, finalmente, el porcentaje de ganancia del banco, decide realizar todas las fases del desarrollo de problemas, teniendo en cuenta que:

- La ganancia del banco es el monto correspondiente a la diferencia entre el costo del inmueble y lo pagado a lo largo del préstamo (monto total por pagar), sin considerar el monto correspondiente a seguros y otros.
- El porcentaje de ganancia se refiere al costo del inmueble y no considera los montos de seguro y otros.
- El monto de cuota incluye los de los seguros y otros.

Elaborar la definición; el análisis, con al menos cinco módulos incluido el principal, y el diseño en diagrama de flujo.

1.3.9. Blanca y César

Blanca Genny y César Moisés se conocieron en enero de 1941. A los pocos días y por motivos de trabajo, César se fue a Tumbes, alejándose, así, del amor de su vida. Lamentablemente, en aquellas épocas las comunicaciones eran escasas, por lo que no les quedó más que comunicarse por cartas y telegramas.

Fue así que durante varios meses ellos solo se comunicaron por ese medio. Incluso se hicieron novios pues César le propuso matrimonio a Blanca.

A lo largo de este periodo, los telegramas y las cartas se enviaban con bastante regularidad, pero existían fechas especiales en las que se mandaban una carta y un telegrama adicional, como los cumpleaños, la Navidad y el Año Nuevo.

Además, César envió una carta especial a la madre de Blanca en la que le pedía la mano de su hija.

Con toda esta información, se quiere saber cuántas cartas envío César, cuánto gastó en el despacho de cartas y telegramas, respectivamente y, finalmente, cuál fue el gasto total de toda la remisión.

Tenga en cuenta que el año tiene doce meses y asuma que cada mes tiene treinta días.

Ilustración 1.38 Ejercicio propuesto: «Blanca y César»

	A	B	C	D	E
1	**Datos de entrada**			Salidas	
2	Periodo de tiempo entre carta y carta (días)	20		Cartas enviadas durante el noviazgo	63
3	Periodo de tiempo entre telegrama y telegrama (días)	30		Monto gastado en el envío de cartas a Blanca	220.5
4	Años de noviazgo	3		Monto gastado en el envío de telegramas a Blanca	247.5
5	Precio de envío de carta	3.5		Monto total gastado	471.5
6	Precio de envío de telegrama	5.5			

Se le pide elaborar la definición y el análisis (en diagrama de módulos) necesarios para solucionar el problema (mínimo con cuatro módulos).

SUBPROGRAMAS EN VBA

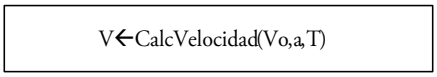
En este capítulo estudiaremos los dos tipos de subprogramas que tiene VBA, así como sus diferencias y manejo.

2.1. Funciones y procedimientos

2.1.1. Funciones

Son un tipo de subprogramas utilizado para calcular y devolver un único valor. Por ejemplo, si quisiéramos calcular la suma, el producto o la resta de dos números, en cualquiera de estos casos, se devolvería un solo valor; por lo que deberíamos usar una función para implementar la operación seleccionada. De este modo, en el diseño, la función se llama o invoca desde un proceso, tal y como se muestra:

- En diagrama de flujo:



- En pseudocódigo:

V←CalcVelocidad(Vo,a,T)

Ahora bien, en la codificación la función se llamará o invocará de la siguiente manera:

V=CalcVelocidad(Vo,a,T)

Por otro lado, la estructura de una función en VBA presenta la siguiente forma:

```
Function <NombreFuncion>(parametro1,parametro2,...)
    <instrucciones del lenguaje>
    <NombreFuncion>=<valor de retorno>
End Function
```

Donde:

- <NombreFuncion>: por ejemplo, «CalcularSuma», «CalcularProducto» o «CalcularResta». Como puede observarse, los nombres de las funciones deben ser descriptivos, empezar siempre por una letra y no pueden contener símbolos o caracteres especiales[1], ni espacios en blanco.
- (parametro1,parametro2,...): son las variables que se necesitan para calcular el valor por devolver; siempre estarán separadas por comas. Sus nombres deben ser descriptivos, empezar siempre por una letra y no pueden contener símbolos o caracteres especiales, ni espacios en blanco.
- <instrucciones del lenguaje>: son las pautas que se llevarán a cabo dentro de la función, con el fin de obtener el valor por devolver. Por ejemplo, la suma, la multiplicación o la resta de dos parámetros o, incluso, el uso de otras funciones o procedimientos para calcular valores intermedios.
- <NombreFuncion> = <valor de retorno>: esta es la instrucción más importante de la función porque con ella se devuelve el valor calculado dentro de ella.
- El resto de palabras reservadas (*Function* y *End Function*) siempre deben estar presentes en la implementación de la función con el objetivo de delimitarla.

Devolver no es lo mismo que mostrar, imprimir ni escribir. Si se le debe a alguien una cantidad de dinero y se le muestra el dinero correspondiente, ¿se puede asumir que la deuda está saldada? Obviamente no.

2.1.1.1. Ejemplos de funciones

a) Se quiere implementar una función que calcule la suma de dos edades.

```
Function CalcularSumaEdades(edad1,edad2) (1)
    SumaEdad=edad1+edad2                 .....(2)
    CalcularSumaEdades=SumaEdad          .....(3)
End Function                             .....(4)
```

- Explicación
  - En la línea 1 observamos que la función se llama «CalcularSumaEdades» y, como el nombre es descriptivo, podemos concluir que dentro de esta se sumarán dos o más edades. Luego, si vemos los parámetros entre

[1] Por ejemplo: #, $, %, &, ¿, ?, *, @, entre otros.

paréntesis, «edad1» y «edad2», podemos intuir que se sumarán dos edades y que la función debe devolver el resultado de dicha suma.

- En la línea 2 usamos una variable temporal o local (SumaEdad) que guarda el resultado de la adición de las dos edades. De este modo, como empleamos una asignación, basta que coloquemos la variable en la que se guardará el valor seguida de un signo igual y, luego de este, el valor (fórmula) que se desea guardar.
- En la línea 3 devolvemos el valor de la función «CalcularSumaEdades», por medio del nombre de la función y asignándole a este el valor guardado en la línea anterior.
- En la línea 4 se finaliza la función.

b) Se sabe que cada caramelo tiene un precio y que se quiere comprar una determinada cantidad, por lo que se deberá implementar una función que ayude al cálculo del monto por pagar.

```
Function CalcularMonto(cantidad,precio)  .....(1)
    CalcularMonto=cantidad*precio        .....(2)
End Function                             .....(3)
```

- Explicación
  - En la línea 1 observamos que la función se llama «CalcularMonto» y, como el nombre es descriptivo, se puede llegar a la conclusión de que dentro de esta se calculará el monto resultante de multiplicar la cantidad por el precio y que la función debe devolver el resultado de dicho producto.
  - En la línea 2 se devuelve el valor de la función, que resulta de multiplicar la cantidad por el precio.
  - En la línea 3 se finaliza la función.

c) Se quiere efectuar una función que calcule el valor de M:

$$M(x)=(x^2+3x-2)/(4+12x)$$

```
Function CalcularM(x)                    .....(1)
    Numerador=x^2+3*x-2                  .....(2)
    Denominador=4+12*x                   .....(3)
    CalcularM=Numerador/Denominador      .....(4)
End Function                             .....(5)
```

- Explicación
  - En la línea 1 vemos que la función se llama «CalcularM» y, como el nombre es descriptivo, concluimos que dentro de esta se calculará el valor de M^2.

De ahora en adelante *dividiremos para vencer*: si el caso lo amerita —y lo permite— en lugar de tener una sola línea con toda la función, se optará por fraccionarla en partes[3].

- En la línea 2 usamos una variable temporal o local (numerador) en la cual se guarda el cálculo del numerador de la función M.
- En la línea 3 empleamos una variable temporal o local (denominador) en la cual se guarda el cálculo del denominador de la función M.
- En la línea 4 retornamos el valor de la función, pero como en las dos líneas anteriores calculamos el numerador y el denominador, respectivamente, ahora devolveremos el resultado de su división.
- En la línea 5 se finaliza la función.

2.1.1.2. Ejercicios propuestos

a) Realizar un subprograma que devuelva el valor de la función P:

$$P(y)=(y\text{^}3+13y-9)/(18*9y)$$

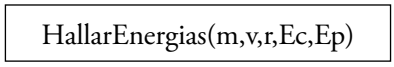
b) Efectuar un subprograma que calcule la edad de una persona sabiendo el año actual y el de su nacimiento.

c) Se tienen tres subtotales y se desea calcular el total. Implementar una función que lo calcule.

2.1.2. Procedimientos

Son un tipo de subprogramas usado para devolver más de un valor o ninguno. Por ejemplo, si deseamos mostrar uno o varios valores, hallar la suma y el producto de dos valores, o el producto y la potencia de dos números, emplearíamos procedimientos. En el diseño, el procedimiento se llama desde un proceso así:

- En diagrama de flujo:



[2] El circunflejo eleva un valor x a una potencia y (ver capítulo 3).

[3] El asterisco multiplica el valor previo por el valor posterior a dicho signo (ver capítulo 3).

• En pseudocódigo:

HallarEnergias(m,v,r,Ec,Ep)

Luego, la llamada en codificación sería:

Call HallarEnergias(m,v,r,Ec,Ep)

o

HallarEnergias m,v,r,Ec,Ep

Por otro lado, la estructura de un procedimiento sería:

Sub <NombreProcedimiento>(parametro1,parametro2,...)
<instrucciones del lenguaje>
End Sub

Donde:

- <NombreProcedimiento>: por ejemplo, «MostrarValores», «LeerDatos», «HallarSumayProducto» o «HallarProductoyPotencia». Los nombres deben ser descriptivos, empezar siempre por una letra y no pueden contener símbolos o caracteres especiales, ni espacios en blanco.
- (parametro1,parametro2,...): son los que necesitará y/o devolverá el procedimiento, y siempre estarán separados por comas. Podemos diferenciarlos como parámetros formales de entrada y de salida, en tanto los primeros solo se usan, mientras que los segundos se modifican dentro del procedimiento[4] para luego ser devueltos.

  Por otro lado, sus nombres deben ser descriptivos, empezar siempre por una letra y no contener símbolos o caracteres especiales, ni espacios en blanco.
- <instrucciones del lenguaje>: son las pautas que se realizarán dentro del procedimiento; por ejemplo, «escribir los parámetros», «leer los datos» o «hallar la suma», «hallar el producto de dos parámetros», «hallar la multiplicación» y «hallar la potencia de un número elevado a otro»; teniendo en cuenta que para ser un procedimiento debe devolver más de un valor o ninguno.

2.1.2.1. Ejemplos de procedimientos

a) Se quiere implementar un procedimiento que lea los dos valores de la hoja de Excel.

[4] Ver capítulo 3.

Ilustración 2.1 Ejemplo de procedimiento: «LeerDatos»

	A	B
1	**Nombre de variable**	**Valor**
2	Cantidad	10
3	Precio	0.50

```
Sub LeerDatos(cantidad,precio,producto)        ……..(1)
    cantidad=Range("B2")                       ……..(2)
    precio=Range("B3")                         ……..(3)
End Sub                                        ……..(4)
```

- Explicación
  - En la línea 1 observamos que el procedimiento se llama «LeerDatos» y como el nombre es descriptivo concluimos que dentro de este se leerán los datos.
  - En las líneas 2 y 3 usamos una propiedad[5] de Excel que nos permite leer el valor en la celda mencionada. Luego, guardaremos dicho valor en la variable local o temporal correspondiente.
  - En la línea 4 se finaliza el procedimiento.

b) Generar un subprograma que muestre el valor de un total, como el siguiente:

Ilustración 2.2 Ejemplo de procedimiento: «MostrarTotal»

	A	B
6	**Salidas**	
7	Nombre de la variable	Valor por mostrar
8	Total	5

```
Sub MostrarTotal(total)          ……..(1)
    Range("B8")=total            ……..(2)
End sub                          ……..(3)
```

- Explicación
  - En la línea 1 vemos que el procedimiento se llama «MostrarTotal» y como el nombre es descriptivo deducimos que dentro de este se exhibirá el valor del total.

[5] Ver capítulo 6.

- En la línea 2 empleamos una propiedad[6] de Excel con la que podemos escribir el valor de una variable, en este caso denominada «Total», y colocarlo en la celda indicada.
- En la línea 3 se finaliza el procedimiento.

De este modo, no hay ningún valor modificado y no se calcula ni devuelve nada; solo se escribe un valor.

Por otro lado, a los procedimientos que no tienen parámetros los llamamos «programas principales» o «macros», y estos son los que podemos relacionar con botones de formularios[7]. Por ejemplo:

```
Sub Prueba()
  Call LeerDatos(cantidad,precio)
  MontoTotal=CalcularMonto(cantidad,precio)
  Call MostrarTotal(MontoTotal)
End Sub
```

2.2. Tipos de parámetros

Existen dos tipos de parámetros: los que se utilizan en la definición o la creación de un subprograma, llamados *parámetros formales*, y los que se usan al llamar o invocar a un subprograma, y que de ahora en adelante conoceremos como *parámetros actuales* o *reales*. A continuación, se detallan estos tipos.

2.2.1. Parámetros formales

Son los que, durante la creación de un subprograma, se colocan entre paréntesis y al costado del nombre de este. Por ejemplo, en la siguiente función los parámetros formales son la cantidad y el precio de la línea 1:

```
Function CalcularMonto(cantidad,precio)      ……..(1)
    Monto= cantidad*precio                   ……..(2)
    CalcularMonto=Monto                      ……..(3)
End Function                                 ……..(4)
```

2.2.2. Parámetros reales o actuales

Son los empleados al momento de la llamada, uso o invocación de un subprograma, ya sea función o procedimiento. Por ejemplo, en el siguiente procedimiento los

[6] Ver punto 6.1.

[7] Ver capítulo 6.

parámetros actuales de «LeerDatos» son el precio y la cantidad; de «CalcularMonto», el precio y la cantidad, y para «MostrarTotal», el monto total:

```
Sub Prueba()
    Call LeerDatos(cantidad,precio)
    MontoTotal=CalcularMonto(cantidad,precio)
    Call MostrarTotal(montoTotal)
End Sub
```

Ahora veremos un caso práctico: volvamos al ejemplo *Procesar compra de cliente*.

```
Sub ProcesarCompra()
    Call LeerPrecioCantidad(precio,cantidad)
    monto=CalcularMontoVenta(precio,cantidad)
    Call EmitirBoleta(monto)
End Sub

Function CalcularMontoVenta(precio,cantidad)
    subtotal=CalcularSubtotal(precio,cantidad)
    montoIGV=CalcularIGV(subtotal)
    CalcularMontoVenta=subtotal+montoIGV
End Function

Function CalcularSubtotal(precio,cantidad)
    CalcularSubtotal=precio*cantidad
End Function

Function CalcularIGV(Subtotal)
    CalcularIGV=Subtotal*0.19
End Function

Sub EmitirBoleta(monto)
    Range("A3")=monto
End Sub

Sub LeerPrecioCantidad(precio,cantidad)
    precio=Range("A1")
    cantidad=Range("A2")
End Sub
```

2.2.3. Paso de parámetros

Por defecto, VBA asume que los parámetros formales se dan por referencia; esto significa que lo que reciben en realidad los subprogramas son las direcciones de memoria en donde se encuentran estos parámetros. Para fines prácticos esto implica que si se realiza algún cambio en la variable estos parámetros permanecerán en ella.

Existen dos posibles maneras de pasar los parámetros: por referencia (ByRef) y por valor (ByVal). No obstante, es importante señalar que estos modificadores solo se pueden colocar dentro de los parámetros formales.

2.2.3.1. Por valor

En este tipo de paso de parámetros usamos la palabra reservada, *ByVal*, solo, como ya dijimos, para los formales, ya sea para utilizarlos o devolverlos sin modificaciones, o solo para imprimirlos.

2.2.3.2. Por referencia

Para este caso empleamos la palabra reservada, *ByRef*, cuando se quiere modificar el valor del parámetro formal y, a su vez, devolverlo modificado.

Los ejemplos (que incluyen ByVal, ByRef y tipos de datos) de esta sección se verán en el siguiente capítulo.

2.3. Resumen

Seguidamente, mostramos ejemplos en los que se identifican las partes de los subprogramas:

Ilustración 2.3 Resumen del procedimiento «ProgramaPrincipal»

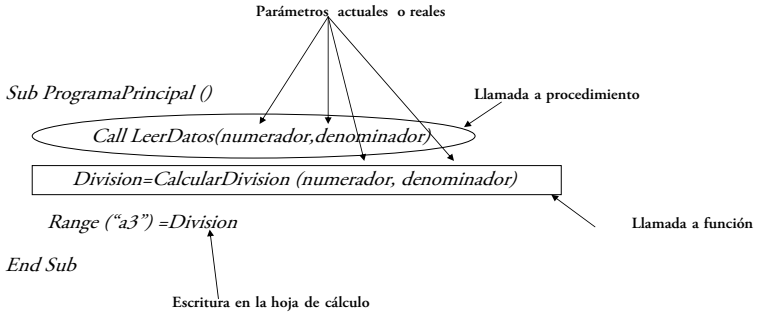


Ilustración 2.4 Resumen de la función «CalculaDivision»

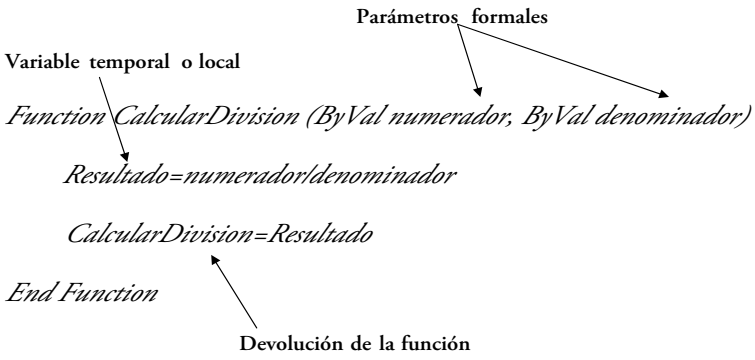
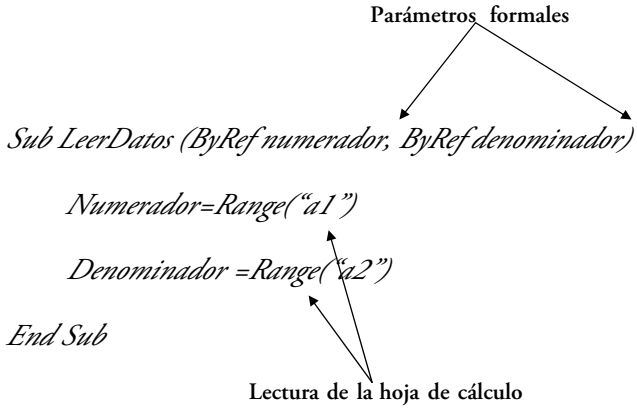


Ilustración 2.5 Resumen del procedimiento «LeerDatos»



2.4. Ejercicios desarrollados

En este punto se plantea la codificación de algunos de los ejercicios desarrollados en el capítulo anterior.

2.4.1. María quiere besar a José

María, como siempre, quiere darle un beso a José, quien se encuentra al borde del muelle. Es la primera vez que intenta alcanzarlo usando una lancha de motor, así que necesita tomar nota de los resultados. Como dicha lancha se encuentra en reposo ($v_0 = 0$ y $x_0 = 0$), María planea acelerar esta vez a 3 m/s² durante 0.13 min. Felizmente, luego del intento, logra su objetivo y besa a José.

María, obviamente, ha quedado encantada con lo ocurrido y ha decidido repetir su hazaña mañana, pero necesita saber a qué distancia del muelle deberá colocar la lancha y cuál será la velocidad media que deberá utilizar. Como somos amigos(as) de María hemos decidido ayudarla elaborando una pequeña aplicación en VBA.

Pero, antes de ayudarla, debemos tener en cuenta las siguientes fórmulas:

$$V=Vo+a*t \ldots\ldots\ldots(velocidad)$$

$$\bar{v}=(v+Vo)/2\ldots\ldots\ldots(velocidad\ media)$$

$$x=Xo+\bar{v}*t\ldots\ldots\ldots(distancia)$$

Donde:
–«V_0» es igual a velocidad inicial
–«a» es igual a aceleración
–«t» es igual a tiempo
–«X_0» es igual a distancia inicial

Además, las unidades de medida de cada uno de los valores serán:

$$a \rightarrow m/s\text{^}2 \qquad \bar{v} \rightarrow m/s \qquad x \rightarrow m \qquad t \rightarrow s$$

Como ya se ha mencionado anteriormente, la codificación es la fase siguiente al diseño, por lo que antes de realizar este ejercicio recomendamos revisar, en el capítulo 1, las tres primeras fases de este problema y utilizar ese diseño para implementar la codificación.

Ahora bien, debemos partir del módulo principal, traduciendo los símbolos o los textos de dicha fase al código en VBA y colocando las constantes donde sea conveniente.

```
'Se define la constante para manejar los segundos por minuto
Const segxmin = 60

'Se inicia el procedimiento principal
Sub AyudarMaria()
'se leen los datos de entrada,
    v0 = InputBox("Ingrese la v0")
    a = InputBox("Ingrese la a")
    tm = InputBox("Ingrese el tiempo")
    x0 = InputBox("Ingrese x0")
    'se convierte el tiempo
    t = tm*segxmin
    'se llama a la función que calcula la velocidad (ojo: en esta fase ya no
```

```
        'se usa la flecha para la asignación; ahora se usa el símbolo «igual»)
        V= CalcVelocidad(v0,a,t)
        'se llama a la función que calcula la velocidad media
        Vm= CalcVelmedia(v,v0)
        'se llama a la función que calcula la distancia
        dist = CalcDistancia(x0,vm,t)
        'se muestran las salidas
        Call MsgBox("Salidas: d=" & dist & "vm=" & vm)
'se finaliza el procedimiento
End Sub

'Se procede a hacer la misma conversión con cada uno de los diagramas o bloques
del pseudocódigo
'se inicia la función con sus respectivos parámetros
Function CalcVelocidad(v0,a,t)
'se devuelve el valor calculado
        CalcVelocidad = v0+a*t
'se finaliza la función
End Function

'se inicia la función con sus respectivos parámetros
Function CalVelMedia(v,v0)
'se devuelve el valor calculado
        CalVelMedia = (v+v0)/2
'se finaliza la función
End Function

'se inicia la función con sus respectivos parámetros
Function CalcDist(x0,vm,t)
'se devuelve el valor calculado
        CalcDist = x0+vm*t
'se finaliza la función
End Function
```

2.4.2. Satélite artificial

Un satélite artificial se mueve alrededor de un planeta, describiendo una órbita circular de 419 kilómetros, en un tiempo de 42,47 horas. Se quiere que calculemos la fuerza gravitatoria que actúa sobre el satélite, así como las energías cinética

y potencial de este en su órbita. Como no conocemos la masa del planeta, no podemos aplicar la ley de gravitación universal, pero sabemos que para que un cuerpo se mantenga en una órbita el valor de su fuerza centrípeta debe coincidir con el valor de la fuerza dada por esta ley. Por ello, podemos decir que:

Fuerza gravitatoria=Fuerza centripeta

*Fuerza centripeta=m*v^2/r*

*v=(2*Pi*r)/T*

Donde:

–«v» es la velocidad a partir del radio de la órbita medida en m/s

–«r» es el radio en m

–«m» es la masa en kg

–«T» es el periodo medido en s

Adicionalmente, sabemos que:

*Ec=0.5*m*v^2*

*Ep=-(v^2*r)*m/r*

Ilustración 2.6 Ejercicio desarrollado: «Satélite artificial»

	A	B	C	D	E
1	**Datos de entrada**			Salidas	
2	Masa del satélite	300		Fuerza gravitatoria	0.21
3	Radio de la órbita circular	419		Energía cinética	44,896.20
4	Tiempo	42.27		Energía potencial	-89,792.39

```
'se inicia el procedimiento
Sub CalcValoresDelSatelite()
    'se leen los datos de entrada
     radio = Range("B3")
     masa = Range("B2")
     Tiempo = Range("B4")
     'Se convierte el tiempo
     T=(Tiempo*3600)
'se calcula la velocidad por medio de una función
     v =CalcVelocidad(radio,t)
'se calculan las energías por medio de un procedimiento
     Call HallarEnergias(v,radio,masa,Ep,Ec)
```

```
'se calcula la fuerza por medio de una función
    Fg = CalcFg(masa,v,radio)
    'se escriben las salidas en la hoja de cálculo
    Range("E2") = Fg
    Range('E4") = Ep
    Range("E3") = Ec
End Sub

'Se calcula la fuerza por medio de una función
Function CalcFg(masa,velocidad,radio)
    CalcFg = (masa velocidad^2)/radio
End Function

'Se calculan las energías a través de un procedimiento, los valores por devolver
también se colocan como parámetros formales
Sub HallarEnergias(velocidad,radio,masa,Ep,Ec)
    Ec = (masa*velocidad^2)/2
    Ep = (velocidad^2*radio*masa)/radio
End sub

'Se calcula la fuerza por medio de una función
Function  CalcVelocidad(radio,t)
    CalcVelocidad=(2*Pi*radio/t)
End function
```

2.4.3. ¿Una bolsa de aire protege realmente a un conductor?

Estime qué tan rápido se debe inflar la bolsa de aire para proteger efectivamente a un conductor en el caso de que sufra una colisión frontal. Al respecto, se sabe que la rapidez inicial del auto es 100 km/h y que recorre 1 m. Para el cálculo de la aceleración y el tiempo que demorará utilice las siguientes fórmulas:

$$a=-Vo/2x$$

$$t=(t-Vo)/a$$

Donde:

–«V_0» es la rapidez inicial en m/s
–«x» es la distancia recorrida en m
–«V =0» es la velocidad en la que el auto llega al reposo
–«a» es la aceleración previamente calculada en m/s²

```
Sub CalcularTiempoEfectivo()
    rapidez=Range("A1")
    X=Range("A3")
    V0=ConvertirRapidez(rapidez)
    a=CalcularAceleracion(v0,x)
    t=(-v0)/a
    Range ("A4")=t
End Sub

Function ConvertirRapidez(rapidez)
    Vinicial=rapidez*1000/3600
    ConvertirRapidez=Vinicial
End Function

Function CalcularAceleracion(v0,x)
    acele=-v0/(2*x)
    CalcularAceleracion=acele
End Function
```

2.4.4. Disco duro

Un disco duro tiene una cabeza lectora ubicada a 3 centímetros de su eje de rotación. Además, se sabe que el disco gira a 7200 revoluciones por minuto. ¿Cuál es la velocidad angular del disco? ¿Cuál es la rapidez lineal de un punto sobre el disco justo debajo de la cabeza lectora?

Utilice las siguientes fórmulas:

$$f=revoluciones/segundos$$

$$w=2*Pi*f$$

$$v=r*w$$

Donde:

–«f» es la frecuencia

–«w» es la velocidad angular radianes/seg

–«v» es la rapidez lineal en m/seg

```
Const sxmin=60
Const cmxmetro=100
```

```
Sub CalcularValoresEnDiscoDuro()
    Call LeerDatos(rpm,rcm)
    w=CalcularVelocidadAngular(rpm)
    v=CalcularRapidezLineal(rcm,w)
    Call Mostrar(w,v)
End Sub

Sub Mostrar(w,v)
    Range("A3")=w
    Range("A4")=v
End Sub

Function CalcularVelocidadAngular(rpm)
    f=rpm/sxmin
    CalcularVelocidadAngular=2*Application.Pi*f
End Function

Function CalcularRapidezLineal(rcm,w)
    r=rcm/cmxm
    CalcularRapidezLineal=r*w
End Function

Sub LeerDatos(rpm,rcm)
    rpm=Range("A1")
    rcm=Range("A2")
End Sub
```

2.4.5. Pintar fachada

Se desea pintar la fachada (techo y paredes frontales) de la casa 2D que se muestra a continuación y se sabe que sus ventanas son cuadradas, así como los anchos y altos de techo, puerta, pared y ventanas. Por tanto, se le pide hallar el área en cuestión.

Ilustración 2.7 Ejercicio desarrollado: «Pintar fachada»

Además, se sabe que las abreviaturas para cada medida y objeto son:
–«h» para la altura
–«a» para el ancho
–«p» para puerta
–«pp» para pared

Se pide implementar en VBA:

a) Un subprograma «CalcularAreaTotalCasa» que lea, calcule y muestre el área.

```
Sub CalcAreaFachada()
    Call LeerDatos(hp,hpp,ht,av,ap,app,at)
    areatotal = CalcAreaTotal(hp,hpp,ht,av,ap,app,at)
    Call Mostrar(areatotal)
End Sub
```

b) Un subprograma «LeerDatos» que permita leer los datos de entrada.

```
Sub LeerDatos(hp,hpp,ht,av,ap,app,at)
    hp = Range("B2")
    hpp = Range("B3")
    ht = Range("B4")
    av = Range("B5")
```

```
    ap = Range("B6")
    app = InputBox("Escriba el ancho de puerta", "Ingreso de datos")
    at = Range("B8")
End Sub
```

c) Un subprograma que calcule el área total por pintar con los siguientes parámetros: altura de la puerta, la pared, el techo, la ventana y el ancho de la puerta, la pared y el techo.

```
Function CalcAreaTotal(hp,hpp,ht,av,ap,app,at)
    Aventana = av^2
    Apuerta = app*hpp
    Atecho = (ht*at)/2
    Apared = (ap*hp) - (2*Aventana) - Apuerta
    CalcAreaTotal = apared + atecho
End Function
```

d) Un subprograma que muestre el área total hallada, teniéndola, a su vez, como parámetro.

```
Sub Mostrar(areaTotal )
    Range("B9") = areaTotal & "metros cuadrados"
End Sub
```

2.4.6. Temperatura en grados Fahrenheit

Se conoce la temperatura en grados Fahrenheit y se debe convertir a grados Celsius. Para ello utilizaremos la siguiente fórmula:

$$°C = (5*(°F-32))/9$$

```
Sub CalcularTemp()
    Call LeerDatos(gF)
    gC = CalcGC(gF)
    Call mostrar(gC)
End Sub

Sub LeerDatos(gF )
    gF = InputBox("Ingrese la temperatura", "Ingreso de datos")
End Sub
Function CalcGC(gF )
    num = 5*(gF-32)
    CalcGC = num/9
End Function
```

```
Sub mostrar(gC)
    Call MsgBox("La temperatura es:" & gC , 0, "Salidas")
End Sub
```

2.4.7. Cálculo de raíces

Dada la ecuación de segundo grado *ax^2 + bx + c,* se calcula el discriminante:

disc = 4ac

Por el planteamiento del problema, asuma que el único caso posible es que el discriminante sea mayor a 0 y calcule el valor de las dos raíces con las siguientes fórmulas:

x1=(-b+disc^0.5)/2a x2=(-b-disc^0.5)/2a

Ilustración 2.8 Ejercicio desarrollado: «Cálculo de raíces»

	A	B
1	Coeficientes	
2	A	1
3	B	5
4	C	1
5		
6	Raíces	
7	X1	-3
8	X2	-2

```
Sub CalcularRaices()
    a = Range("B2")
    b = Range("B3")
    c = Range("B4")
    Disc = CalcDiscriminante(b,a,c)
    X1 = CalculaRaiz1(Disc,a,b)
    X2 = CalculaRaiz2(Disc,a,b)
    Range("B7") = X1
    Range("B8")= X2
End Sub

Function CalcDiscriminante(b,a,c)
    CalcDiscriminante= b-(4*a*c)
End Function
```

```
Function CalculaRaiz2(Disc,a,b)
    CalculaRaiz2 = (-b+Sqr(Disc))/(2*a)
End Function

Function CalculaRaiz1(Disc,a,b)
    CalculaRaiz1 = (-b-Sqr(Disc))/(2*a)
End Function
```

2.4.8. Ventas de vendedor

Cada vendedor tiene un monto por venta anual en el primer y segundo semestre. Se desea calcular el promedio y la venta total de cada uno de los vendedores.

Ilustración 2.9 Ejercicio desarrollado: «Ventas de vendedor»

	A	B	C	D	E
1	**Venta anual**	**Semestre 1**	**Semestre 2**	**Promedio**	**Venta total**
2	Juan Pérez	228,000.00	130,000.00	179,000.00	358,000.00
3	María Morales	2,984,566.00	154,863.00	1,569,714.50	3,139,429.00

```
Sub VentaVendedores()
    Sem1Vend1 = Range("B2")
    Sem1Vend2 = Range("B3")
    Sem2Vend1 = Range("C2")
    Sem2Vend2 = Range("C3")
    VentaTotalVend1 = CalcVentaTotal(Sem1Vend1,Sem2Vend1)
    VentaTotalVend2 = CalcVentaTotal(Sem1Vend2,Sem2Vend2)
    PromVend1 = CalculaPromedio(VentaTotalVend1)
    PromVend2 = CalculaPromedio(VentaTotalVend2)
    Range("D2") = PromVend1
    Range("D3") = PromVend2
    Range("E2") = VentaTotalVend1
    Range("E3") = VentaTotalVend2
End Sub
Function CalculaPromedio(total)
    CalculaPromedio = (total)/2
End Function
```

```
Function CalcVentaTotal(Semestre1,Semestre2)
    CalcVentaTotal = (Semestre1+Semestre2)
End Function
```

2.4.9. Margen bruto

Se quiere calcular la venta neta y el margen bruto del producto, sobre la base de los datos mostrados en la siguiente hoja de cálculo.

Ilustración 2.10 Ejercicio desarrollado: «Margen bruto»

	A	B
1	**Resultados por productos**	**Producto**
2	Ventas	50
3	Devoluciones	5.5
4	Ventas netas	44.5
5	Amortizaciones	10.5
6	Costos de producción	9.00
7	Margen bruto	25

```
Sub MargenBruto()
    Call LeerDatos(ventas,devoluciones,amort,costo)
    VentasNetas = CalculaVentas(ventas,devoluciones)
    MargenB = CalcMargen(VentasNetas,amort,costo)
    Range("B4") = VentasNetas
    Range("B7") = MargenB
End Sub

Sub LeerDatos(ventas,devoluciones,amort,costo)
    ventas = Range("B2")
    devoluciones = Range("B3")
    amort = Range("B5")
    costo = Range("B6")
End Sub
Function CalcMargen(VentasNetas,amort,costo)
    CalcMargen = VentasNetas-amort-costo
End Function
```

```
Function CalculaVentas(ventas,devoluciones)
    CalculaVentas = ventas-devoluciones
End Function
```

2.5. Ejercicios propuestos

2.5.1. Importaciones y exportaciones del año

Se quiere calcular el total de las importaciones y exportaciones, los porcentajes sobre el total y la diferencia de porcentajes de cada periodo.

Ilustración 2.11 Ejercicio propuesto: «Importaciones y exportaciones del año»

	A	B
1	**Periodo**	**2010**
2	Exportaciones	24,543
3	% sobre el total	0.66
4	Importaciones	12,543
5	% sobre el total	0.34
6	Diferencia importación/exportación	0.32
7	Total del periodo	37,086

2.5.2. Presupuesto

Dado el siguiente presupuesto:

Ilustración 2.12 Ejercicio propuesto: «Presupuesto»

	A	B	C	D
1		**Cantidad**	**Precio**	**Subtotal**
2	Pasajes	10	0.80	8.00
3	Almuerzos	5	4.50	22.50
4	Copias	30	0.07	2.10
5			Total	32.60

Se pide implementar en VBA:

a) Un subprograma que lea la cantidad de pasajes, almuerzos y copias, así como los precios de cada uno.

b) Un subprograma que calcule el subtotal a partir del precio y la cantidad.

c) Un subprograma que calcule el presupuesto de la semana (total).
d) Un subprograma que muestre los subtotales y el total presupuestado.
e) Un subprograma que lea los datos de entrada, calcule los subtotales, el total, y lo muestre haciendo uso de los subprogramas anteriores que considere convenientes.

2.5.3. Promedio del curso

Se tienen las notas parciales de un curso y se quiere obtener el promedio final a través de la siguiente fórmula:

$$(NF=(PP*2+PL*2+E1*2+E2*4))/10$$

Donde:
–«PL» es el promedio de los laboratorios
–«PP» es el promedio de las prácticas
–«E1» es el examen parcial
–«E2» es el examen final

Ilustración 2.13 Ejercicio propuesto: «Promedio del curso»

	A	B
1	**Concepto**	**Nota**
2	Promedio Laboratorio	15
3	Promedio Prácticas	14.7
4	Examen Parcial	12
5	Examen Final	17
6	Promedio del curso	15.14

a) Un subprograma que calcule el promedio final mediante los siguientes parámetros de entrada: promedios de práctica y laboratorio, así como de los dos exámenes.
b) Un subprograma «LeerDatos» que perciba los datos de la hoja de Excel.
c) Un subprograma «Mostrar» que escriba en la hoja de Excel el promedio final del curso.
d) Un subprograma principal que lea los datos de entrada, calcule y muestre el promedio final, y utilice los subprogramas anteriores.

2.5.4. Presupuesto y duración de película

Se quiere saber cuál sería el presupuesto necesario para ver las películas mostradas y cuál el tiempo necesario para verlas, en segundos, minutos y horas, respetivamente.

Ilustración 2.14 Ejercicio propuesto: «Presupuesto y duración de película»

	A	B	C
1	**Película**	**Duración en horas**	**Costo de entrada**
2	Iron Man	2.4	20.00
3	El señor de los anillos	3.1	25.00
4	Harry Potter	2.6	15.00
5			
6	**Salidas**		
7	Presupuesto	60	
8	Tiempo total en segundos	29,160	
9	Tiempo total en minutos	486	
10	Tiempo total en horas	8.1	

2.5.5. Comisión

Calcular el monto por comisión y el total ganado por cada uno de los vendedores. Se sabe que cada uno recibe el 2% de las ventas.

Ilustración 2.15 Ejercicio propuesto: «Comisión»

	A	B	C	D	E
1	**Vendedor**	**Ventas**	**Comisión**	**Sueldo base**	**Total**
2	Juan Vargas	4,600.00	92.00	90.00	182.00
3	Pedro Torres	2,430.00	28.60	97.00	125.60

2.5.6. Presupuesto con y sin descuento

Calcular los presupuestos con y sin descuento, de acuerdo con la siguiente tabla:

Ilustración 2.16 Ejercicio propuesto: «Presupuesto con y sin descuento»

	A	B	C	D	E	F
1	**Producto**	**Precio unitario**	**Cantidad**	**Descuento**	**Subtotal**	**Subtotal con descuento**
2	Silla	92.4	4	10%	369.60	332.64
3	Escritorio	299.3	3	5%	897.90	853.01
4				Total sin IGV	1,267.50	1,185.65
5				Monto IGV	240.83	225.27
6				Total con IGV	1,508.33	1,410.92

Para ello se pide implementar en VBA:

a) Un subprograma «LeerDatos» que lea todos los datos de entrada.

b) Un subprograma «CalcularSubtotales» que reciba por parámetros al precio unitario, la cantidad de un producto y el porcentaje de descuento, y que calcule y devuelva los subtotales con y sin descuento.

c) Un subprograma «CalcularMontoConIGV» que reciba por parámetro un subtotal, y calcule y devuelva el monto con IGV sobre ese subtotal.

d) Un subprograma «Factura» que a partir del llamado a los subprogramas anteriores, lea, calcule los subtotales y los montos, y muestre todas las salidas.

DATOS Y EXPRESIONES

Un dato es un símbolo o un conjunto de símbolos que la computadora procesa y que por sí solo puede no tener sentido. No obstante, luego de ser procesado, se convierte en información útil.

3.1. Tipos de datos y sus operaciones

El tipo de dato se refiere a la forma en la cual se interpretarán las secuencias de bits que representan a los datos, así como las operaciones que podrán realizarse sobre estos.

La asignación de tipos de datos permite administrar de una mejor manera los recursos, así como los espacios en la memoria.

3.1.1. Clasificación de tipos de datos

Los tipos de datos se pueden clasificar en a) numéricos, b) alfanuméricos y c) lógicos.

3.1.1.1. Numéricos

Expresan valores enteros o reales, con signo positivo o negativo: byte, integer, long, single, double.

3.1.1.2. Alfanuméricos

Se representan por un conjunto de caracteres, números y/o símbolos especiales, pero siempre van entre comillas dobles; por ejemplo, el tipo string.

3.1.1.3. Lógicos

Representan solo dos valores: VERDADERO (true) o FALSO (false); por ejemplo, el tipo boolean.

3.1.2. Tipos de datos

3.1.2.1. Para manejar números enteros

- Tipo: byte

  Rango de valores posibles: entre 0 y 255

  Caso de uso: para números enteros positivos entre 0 y 255

  Ejemplo: la edad de una persona

- Tipo: integer

  Rango de valores posibles: entre -32.768 y 32. 767

  Caso de uso: para números enteros positivos y negativos dentro del rango anterior

  Ejemplo: el año actual

- Tipo: long

  Rango de valores posibles: entre -2.147.483.648 y 2.147.483.647

  Caso de uso: para números enteros positivos y negativos dentro del rango de valores posibles

  Ejemplo: el número de DNI

3.1.2.2. Para manejar números decimales

- Tipo: single

  Rango de valores posibles: desde -3,402823E38 a -1,401298E-45 para valores negativos y desde 1,401298E-45 a 3,402823E38 para valores positivos

  Caso de uso: para números decimales positivos y negativos dentro del rango de valores posibles

  Ejemplo: el precio de un producto

- Tipo: double

  Rango de valores posibles: desde 1,79769313486231E308 a -4,94065645841247E-324 para valores negativos y desde 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos

Caso de uso: para números decimales positivos y negativos dentro del rango de valores posibles

Ejemplo: el precio de un producto

3.1.2.3. Para manejar valores booleanos

- Tipo: boolean

  Rango de valores posibles: true y false

  Caso de uso: para valores de verdad y/o falsedad

  Ejemplo: la afirmación de ser o no mayor de edad

3.1.2.4. Para manejar cadenas de texto

- Tipo: string

  Rango de valores posibles: cadenas de longitud fija en un intervalo de 0 a aproximadamente 63.000 caracteres

  Caso de uso: para cadenas que pueden incluir números, letras y/o símbolos pero siempre entre comillas dobles

  Ejemplo: el nombre de una persona

3.1.3. Ejemplos de tipos de datos

¿Qué tipo de datos usaría para los siguientes valores?

- La dirección de un local: string
- La edad de una persona: byte
- El nombre de una persona: string
- La placa de un auto: string
- El código de un alumno: single
- La cantidad de cuadrados que hay en un *sudoku* de 9 x 9: byte
- El número de DNI: long
- La estatura de una persona en centímetros: byte
- La estatura de una persona en metros: single
- La estatura de una persona en pies: single
- El precio de un caramelo: single
- El precio de un auto: single
- La pregunta ¿es usted mayor de edad?: boolean
- El número de extremidades que posee un ser humano: byte

- El resultado de restar 40 y 80: integer
- El nombre de un canal de televisión: string
- La talla de un pantalón (por ejemplo, *medium*): string
- La talla del calzado: single
- La marca de auto: string
- La pregunta ¿aún es alumno PUCP?: boolean
- La cantidad de dedos que tiene una persona byte
- La pregunta ¿es alumno de EEGGCC?: boolean
- Edad en la que se cumple la mayoría de edad: byte
- La cantidad de cilindros de un auto: byte
- La cantidad de ruedas de una bicicleta: byte

3.1.4. Operaciones sobre los tipos de datos numéricos

Existen operaciones básicas sobre datos numéricos con sus respectivas precedencias.

Cabe resaltar que tanto la división entera como el módulo son operaciones sobre números enteros únicamente.

- Operación: potencia
  Jerarquía: 1
  Signo: ^
  Ejemplo: x = a ^ b
- Operación: negación
  Jerarquía: 2
  Signo: -
  Ejemplo: x = -a
- Operación: multiplicación
  Jerarquía: 3
  Signo: *
  Ejemplo: x = a*b
- Operación: división
  Jerarquía: 3
  Signo: /
  Ejemplo: x = a/b

- Operación: división entera
  Jerarquía: 4
  Signo: \
  Ejemplo: x = a\b
- Operación: cociente o módulo
  Jerarquía: 5
  Signo: mod
  Ejemplo: x = a mod b
- Operación: suma
  Jerarquía: 6
  Signo: +
  Ejemplo: x = a+b
- Operación: resta
  Jerarquía: 6
  Signo: -
  Ejemplo: x = a-b

3.1.5. Ejemplos de tipos de datos numéricos

20\4+9=14
2^2*3\6=2
4^2-4 *mod* 6 = 12
4*3-4/2 = 10
12 *mod* 10*0,5 = 2
47 *mod* 5+2 =
4^2 + 16\4*2 = 18
4*3 + 2*4 *mod* 5*2 = 20
9*2 - 1^2 = 17
2*8 + 9 + -3*2 = 19
5*2+2^2-1=9
4+1^2*4=8
18 *mod* 3+5*2=16
10/2+4^2=21

3.1.6. Operaciones sobre los tipos de datos lógicos

Existen operaciones básicas sobre datos lógicos a las que se aplican las reglas de lógica ya conocidas:

- Operación: conjunción
  Jerarquía: and
- Operación: disyunción lógica
  Jerarquía: or
- Operación: negación
  Jerarquía: not

3.1.7. Operaciones sobre los tipos de datos alfanuméricos

Existen operaciones básicas sobre datos alfanuméricos. A continuación los mostramos junto con un ejemplo y su respectiva salida.

- **LCase(cadena)**

  Función con la que se obtiene la cadena entregada como parámetro (cadena) en minúsculas.

  –Ejemplo:

```
Sub PP()
    Dim cad as String, Lowercase as String
    cad= "Hola Mundo"
    Lowercase = LCase(cad)
    Range("A1")= Lowercase
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «hola mundo».

- **UCase(cadena)**

  Función con la que se obtiene la cadena entregada como parámetro (cadena) en mayúsculas.

  –Ejemplo:

```
Sub PP()
    Dim cad as String, Uppercase as String
    cad= "Hola Mundo"
    Uppercase = UCase(cad)
    Range("A1")= Uppercase
End Sub
```

– Salida:

En la celda A1 se ha escrito: «HOLA MUNDO».

- **Mid(cad,inicio,Nrocar)**

Función con la que se elabora una cadena formada por una cantidad de caracteres (Nrocar) desde la posición (inicio) de la cadena original (cad).

–Ejemplo:

```
Sub PP()
    Dim UnaCadena as String, MiCadena as String
    UnaCadena  = "Hola Mundo"
    MiCadena = Mid(MiCadena,  6, 2)
    Range("A1")= MiCadena
End Sub
```

–Salida:

En este caso se asignan dos caracteres de la cadena original empezando desde la posición 6.

En la celda A1 se ha escrito: «Mu».

- **Left(cadena,NroCar)**

Función con la que se adquiere una cadena formada por una cantidad de caracteres (Nrocar), empezando por el carácter que está más a la izquierda de la cadena original (cad).

–Ejemplo:

```
Sub PP()
    Dim UnaCadena as String, MiCadena as String
    UnaCadena  = "Hola Mundo"
     MiCadena = Left(UnaCadena, 2)
    Range("A1")= MiCadena
End Sub
```

–Salida:

En la celda A1 se ha escrito: «Ho».

- **Right(cadena,Nrocar)**

Función con la que se produce una cadena formada por una cantidad de caracteres (Nrocar), empezando por el carácter que está más a la derecha de la cadena original (cad).

–Ejemplo:

```
Sub PP()
    Dim UnaCadena as String, MiCadena as String
    UnaCadena  = "Hola Mundo"
    MiCadena = Right(UnaCadena, 3)
    Range("A1")= MiCadena
End Sub
```

–Salida:

En la celda A1 se ha escrito: «ndo».

- **Space(cantidad)**

  Función con la que se obtiene una cadena compuesta por una cantidad (cantidad) de espacios en blanco.

  –Ejemplo:

```
Sub PP()
    Dim MiCadena as String
    MiCadena = "Hola" & Space(5) & "Mundo"
    Range("A1")= MiCadena
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «Hola Mundo».

- **Chr(codigocar)**

  Función con la que se produce el carácter correspondiente al codigocar.

  Las letras del alfabeto empiezan en 65 y 97, «A» y «a», respectivamente.

  –Ejemplo:

```
Sub PP()
    MiCaracter = Chr(65)
    Range("A1")= MiCaracter
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «A».

- **String(cantidad, carácter)**

  Función con la que se desarrolla una cadena formada por la repetición de un carácter (carácter) una determinada cantidad de veces (cantidad).

–Ejemplo:

```
Sub PP()
    MiCadena = String(4, "p")
    Range("A1")= MiCadena
End Sub
```

–Salida:

En la celda A1 se ha escrito: «pppp».

- **LTrim(cadena)**

  Función con la que se realiza una nueva cadena formada por la cadena original (cadena), sin los espacios en blanco que esta pueda contener a la izquierda.

  –Ejemplo:

```
Sub PP()
    Cadena="  Hola Mundo"
    MiCadena = LTrim(cadena)
    Range("A1")= MiCadena
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «Hola Mundo».

- **RTrim(cadena)**

  Función con la que se obtiene una nueva cadena formada por la cadena original (cadena), sin los espacios en blanco que esta pueda contener a la derecha.

  –Ejemplo:

```
Sub PP()
    Cadena="Hola Mundo    "
    MiCadena = RTrim(cadena)
    Range("A1")= MiCadena & "."
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «Hola Mundo».

- **StrComp(cadOrig,cad,1)**

  Función que devuelve un valor numérico que representa una comparación entre cadenas. Si se desea ignorar las mayúsculas o minúsculas se usa 1 como tercer parámetro; si no se las quiere ignorar, se emplea el 0.

–Si la primera cadena (cadOrig) es igual que la segunda el valor devuelto es 0.

–Si la primera cadena (cadOrig) es mayor que la segunda el valor devuelto es 1.

–Si la primera cadena (cadOrig) es menor que la segunda el valor devuelto es -1.

Esta comparación de mayor y menor es parecida a la hallada en un diccionario o una guía telefónica, donde, por ejemplo, «auto» sería menor que «carro» debido a que se encuentra en una posición anterior con respecto al segundo término.

–Ejemplo:

```
Sub PP()
    Cadena1="Hola"
    Cadena2="hola"
    Cadena3="Mundo"
    SonIguales= StrComp(Cadena1,Cadena2,1)
    SonDiferentes1= StrComp(Cadena2,Cadena3,1)
    SonDiferentes2= StrComp(Cadena3,Cadena2,1)
    Range("A1")= SonIguales
    Range("A2")= SonDiferentes1
    Range("A3")= SonDiferentes2
End Sub
```

–Salida:

–En la celda A1 se ha escrito: «0».

–En la celda A2 se ha escrito: «-1».

–En la celda A3 se ha escrito: «1».

- **Trim(cadena)**

  Función con la que se genera una nueva cadena formada por la original (cadena), sin los espacios en blanco que esta pueda contener a la izquierda y derecha.

  –Ejemplo:

```
Sub PP()
    Cadena="    Hola Mundo    "
    MiCadena = Trim(cadena)
    Range("A1")= MiCadena & "."
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «Hola Mundo».

- **StrReverse(cadena)**

  Función con la que se produce una nueva cadena formada por la original, pero invertida.

  –Ejemplo:

  ```
  Sub PP()
      Cadena= "Hola"
      MiCadena = StrReverse(cadena)
      Range("A1")= MiCadena
  End Sub
  ```

  –Salida:

  En la celda A1 se ha escrito: «aloh».

- **Len(cadena)**

  Función con la que se obtiene la longitud de la cadena.

  –Ejemplo:

  ```
  Sub PP()
      Cadena="Hola"
      Tamaño=Len(Cadena)
      Range("A1")=Tamaño
  End Sub
  ```

  –Salida:

  En la celda A1 se ha escrito: «4».

- **Asc(carácter)**

  Función con la que se adquiere el código correspondiente al carácter (las letras del alfabeto empiezan en 65 y 97, «A» y «a», respectivamente).

  –Ejemplo:

  ```
  Sub PP()
      MiCaracter = Asc("A")
      Range("A1")= MiCaracter
  End Sub
  ```

  –Salida:

  En la celda A1 se ha escrito: «65».

- **Replace(cadenaOrig,cadenaAreemplazar,cadenaDereemplazo)**

  Función con la que se produce una nueva cadena con la estructura de la original (cadenaOrig), sustituyendo a esta última (cadenaAreemplazar) y colocando a la nueva en su lugar (cadenaDereemplazo).

–Ejemplo:

```
Sub PP()
    Cadena="Hola Mundo"
    MiCadena = Replace(cadena, "Hola", "Chau")
    Range("A1")= MiCadena
End Sub
```

–Salida:

En la celda A1 se ha escrito: «Chau Mundo».

- **InStr(inicio, cadBusqueda,caracter)**

  Función que busca un carácter (carácter) desde una posición (inicio) de la cadena original (cadBusqueda) hasta el final y lo devuelve a la posición en donde lo encontró. Por defecto, la posición de inicio es 1.

  –Ejemplo:

```
Sub PP()
    Cadena="Hola"
    Posición=InStr(1,Cadena, "o")
    Range("A1")= Posición
End Sub
```

  –Salida:

  En la celda A1 se ha escrito: «2».

3.1.8. Ejemplos de tipos de datos alfanuméricos

3.1.8.1 Información de alumno

Ilustración 3.1 Ejemplo «Información de alumno»

	A	B	C	D	E
1	Nombre completo	Apellido	Edad	Unidad	Salida
2	Juan Felipe	Martínez Ponce	19	Estudios Generales Ciencias	Juan Martínez, de 19 años, estudia en Generales Ciencias

a) Realice un subprograma que devuelva la unidad a partir del parámetro «nombre completo de la unidad».

```
Function ObtUnidad(ByVal unidadComp As String, ByVal LongEst As Byte)_
As String
    Dim LongUnidad As Byte
    espBlanco=Instr(1,unidadComp, " ")
    LongUnidad = Len(unidadComp) - espBlanco
    ObtUnidad = Right(unidadComp, LongUnidad)
End Function
```

b) Elabore un subprograma que devuelva una cadena como la que se muestra en la celda E2, que tenga como parámetros el nombre y el apellido completos, así como la edad y la unidad completas. Utilice el subprograma anterior y funciones de cadenas.

```
Function ObtSalida(ByVal NComp As String, ByVal ApComp As String,_
ByVal edad As Byte,  ByVal unidadComp As String) As String
    Dim PrimerNombre As String, PrimerApe As String, unidad As String
    'ubicamos la posición del espacio en blanco
    espBlancoNomb=Instr(1,Ncomp," ")
    'el nombre está en los caracteres de la izquierda hasta la posición anterior
    'al espacio en blanco previamente hallado
    PrimerNombre = Left(Ncomp, espBlancoNomb-1)
    espBlancoAp=Instr(1,Apcomp," ")
    PrimerApe = Left(Apcomp, espBlancoAp-1)
    'como obtener la unidad es un poco más complicado, se utilizará una
     'función que encapsulará la complejidad del proceso
    unidad = ObtUnidad(unidadComp)
    ObtSalida = PrimerNombre & " " & PrimerApe & "de" & edad & "años,
     estudia en" & unidad
End Function
```

c) Implemente un subprograma principal que lea, obtenga y muestre la salida de la ilustración 3.1.

```
Const LongEspacio As Byte = 1

Sub Ejercicio1()
    Dim NC As String, LNC As Byte, APC As String, LA2 As Byte
    Dim edad As Byte, unidad As String, Lest As Byte
    Dim salida As String
'se leerán los datos de entrada
    NC = Hoja1.Range("A2")
    LNC = Sheets(«Hoja1»).Range("B2")
    APC = Range("C2")
    LA2 = Range("D2")
    edad = Range("E2")
    unidad = Range("F2")
    Lest = Range("G2")
'se hallará la salida por medio de una función
    salida = ObtSalida(NC, LNC, APC, LA2, edad, unidad, Lest)
'se escribe la salida hallada
```

```
    Hoja1.Range("H2") = salida
End Sub
```

3.2. Constantes, variables y expresiones

En general, las tres deben cumplir con las siguientes reglas:

- Deben ser descriptivas.
- No deben empezar por números.
- No deben incluir símbolos o caracteres especiales.

3.2.1. Constantes

«Dicho de una cosa: persistente, durable»[1]. Es por ello que utilizaremos este tipo de datos para manejar valores que permanecerán invariables dentro de un programa; por ejemplo, el IGV, el valor de π, la cantidad de días que tiene un año, entre otros.

De este modo, para definir una constante podemos utilizar cualquiera de las siguientes estructuras:

Const <NombreConstante> = <valorNumerico>
Const <NombreConstante> as <tipoDeDato> = <valorNumerico>

Si tuviéramos que definir varias constantes podemos separarlas por comas o utilizar líneas diferentes:

Const <NombreConst> = <valor>,<NombreConst> = <valor>

Por ejemplo, para definir la constante tipo de cambio o el valor de π:

Const Pi=3.1416, TC as single =2.8

3.2.2. Variables

«Que varía o puede variar»[2]. Existen ciertos valores que deberán guardarse en una variable. Como su mismo nombre lo indica, estos valores podrán modificarse durante la ejecución del programa.

En VBA, si las variables no se definen con dim, entonces se inician con un valor de vacío y son, por defecto, de tipo variant. Por el contrario, si se definen con dim, principian con un valor correspondiente al tipo definido.

---

[1] Real Academia Española (RAE) (2001). *Diccionario de la lengua española.*Recuperado de http://lema.rae.es/drae/?val=constante

[2] Real Academia Española (RAE) (2001). *Diccionario de la lengua española.*. Recuperado de http://lema.rae.es/drae/?val=variable

Así, los valores iniciales para los siguientes tipos de datos son:

- byte - 0
- boolean - false
- integer - 0
- long - 0
- single – 0.0
- double – 0.0
- string – "" (cadena vacía)
- variant – vacío

3.2.2.1. Variables locales

Las variables locales son aquellas declaradas dentro de una función (Function) o un procedimiento (Sub) de la siguiente manera:

Dim nombreVariable as <tipo de dato>

De este modo, si deseamos declarar muchas variables lo haremos así:

Dim nombreVariable as tipo de dato, nombreVariable as tipo de dato_ ,nombreVariable as tipo de dato

Una buena práctica es que siempre declaremos las variables en la línea siguiente al encabezado de cada función o procedimiento.

3.2.2.2. Variables globales

Las variables globales son aquellas que son declaradas fuera de una función (Function) o un procedimiento (Sub):

Dim nombreVariable as <tipo de dato>

Luego, si queremos declarar muchas variables lo haremos de esta forma:

Dim nombreVariable as tipo de dato, nombreVariable as tipo de dato_, nombreVariable as tipo de dato

Las variables globales se definen al inicio del módulo y son conocidas en cualquier función o procedimiento de este sin necesidad de pasarlas como parámetros formales. No obstante, es preciso mencionar que en este libro no se utilizarán variables globales.

3.2.3. Expresiones

Combinación de constantes, variables y/o funciones, interpretada (evaluada) de acuerdo con las normas particulares de precedencia y asociación para un lenguaje de programación en particular.

3.2.4. Conversiones entre tipos

Existen casos en los que necesitamos convertir valores de un tipo de dato a otro. Para ello, utilizaremos funciones de conversión. Las más usadas en VBA son:

- CStr(Var): convierte cualquier tipo de dato en una cadena.
- CInt(Var): convierte cualquier tipo de dato en un valor entero.
- CLng(Var): convierte cualquier tipo de dato en un valor entero largo.
- CSng(Var): convierte cualquier tipo de dato en un valor de precisión simple.
- CDbl(Var): convierte cualquier tipo de dato en un valor de precisión doble.
- CBool(Var): convierte cualquier tipo de dato en un valor booleano.
- CDate(Var): convierte cualquier tipo de dato en un valor de fecha.
- CByte(Var): convierte cualquier tipo de dato en un valor de byte.

3.3. Ejercicios desarrollados

Una vez realizados la definición, el análisis y el diseño, llevamos a cabo la fase de codificación. Luego de desarrollar el algoritmo debemos completar los tipos de datos; para ello, debemos recordar los siguientes puntos:

- Cada uno de los parámetros formales debe tener su propio tipo de dato, además de tener su propio modificador ByVal o ByRef.
- Todas las funciones tienen tipo de dato de retorno que hace referencia a la clase de dato que devolverá.
- Dentro de cada programa/subprograma se declaran con dim todas las variables que no son parámetros formales.
- Los valores que permanecerán constantes a lo largo del programa se definen como tales, pero fuera de cualquier subprograma.

A continuación, desarrollaremos algunos ejercicios sobre este punto.

3.3.1. Temperatura en grados Fahrenheit

Se conoce la temperatura en grados Fahrenheit y se debe convertir a grados Celsius.

Para ello utilizaremos la siguiente fórmula:

$$°C=(5*(°F-32))/9$$

Luego de leer el texto, identificamos el dato de entrada, la salida y la fórmula que utilizaremos. Como es bastante sencilla, se plantea la siguiente solución:

```
Sub CalcularTemp()
    Call LeerDatos(gF)
    gC = CalcGC(gF)
    Call mostrar(gC)
End Sub
```

Repasamos los cuatro puntos anteriores y observamos que el único que afecta al procedimiento anterior es declarar con dim todas las variables que no son parámetros formales, con lo cual, este proceso queda finalmente así:

```
Sub CalcularTemp()
    Dim gF As Byte, gC As Single
    Call LeerDatos(gF)
    gC = CalcGC(gF)
    Call mostrar(gC)
End Sub
```

Luego, implementamos todos los módulos. Así, empezamos por «LeerDatos», y vemos que la única regla que afecta a este procedimiento es el modificador y el tipo de datos en los parámetros formales:

```
'el parámetro es ByRef, ya que el valor de gF se devolverá modificado
Sub LeerDatos(ByRef gF As Byte)
    gF = InputBox("Ingrese la temp", "Ingreso de datos")
End Sub
```

Después, elaboramos el siguiente módulo «CalcGC», en el cual se aplicará la fórmula dada. Como se trata de una función, entonces, tendrá tipo de retorno. Además, por tener variables que no son parámetros formales, habrá declaración de dim y, como solo se usarán sus parámetros formales, utilizaremos ByVal:

```
Function CalcGC(ByVal gF As Byte) As Single
    Dim num As Integer
    num = 5 * (gF - 32)
    CalcGC = num / 9
End Function
```

Para finalizar, solamente faltaría que desarrollemos el procedimiento «Mostrar», en el cual tendremos un único parámetro formal que no será modificado, por lo que usaremos ByVal.

```
Sub mostrar(ByVal gC As Single)
    Call MsgBox("La temperatura es:" & gC, 0, "Salidas")
End Sub
```

3.3.2. Comisión

Calcular el monto por comisión y el total ganado por el vendedor. Se sabe que cada uno recibe una comisión del 2% de las ventas.

Ilustración 3.2 Ejercicio desarrollado: «Comisión»

	A	B	C	D	E
1	**Vendedor**	**Ventas**	**Comisión**	**Sueldo base**	**Total**
2	Juan Vargas	4,600.00	92.00	90.00	182.00
3	Pedro Torres	2,430.00	28.60	97.00	125.60

```
'Del texto obtenemos un valor que será definido como una constante
Const Porc = 0.02
'Se tienen dos salidas y dos datos de entrada para cada uno de los 'vendedores.
En este caso, se optará por 'calcular la comisión usando una 'función. Luego, se
debe calcular el total usando otra función y, 'finalmente, se deben mostrar las
salidas por medio de un procedimiento. Ojo: lo que se busca es 'reutilizar 'las
funciones con los datos de cada vendedor.
Sub Comision()
    Dim VentasVend1 As Integer, VentasVend2 As Integer
    Dim SueldoBaseVend1 As Byte, SueldoBaseVend2 As Byte
    Dim ComisionVend1 As Single, ComisionVend2 As Single
    Dim TotalVend1 As Single, TotalVend2 As Single
    VentasVend1 = Range("B2")
    VentasVend2 = Range("B3")
    SueldoBaseVend1 = Range("E2")
    SueldoBaseVend2 = Range("E3")
    ComisionVend1 = CalcComision(VentasVend1)
    ComisionVend2 = CalcComision(VentasVend2)
    TotalVend1 = CalcTotal(ComisionVend1, SueldoBaseVend1)
    TotalVend2 = CalcTotal(ComisionVend2, SueldoBaseVend2)
```

```
    Call Mostrar(ComisionVend1, ComisionVend2, TotalVend1,
    TotalVend2)
End Sub

'Esta función trabaja con valores generales
Function CalcComision(ByVal VentasVend As Integer) As Single
    CalcComision = (VentasVend * Porc)
End Function

'Esta función trabaja con valores generales
Function CalcTotal(ByVal ComisionVend As Single, ByVal _
 SueldoBaseVend As Byte) As Single
    CalcTotal = (ComisionVend + SueldoBaseVend)
End Function

'Este procedimiento escribirá las salidas en la hoja de calculo
Sub Mostrar(ByVal ComisionVend1 As Single, ByVal ComisionVend2 _
As Single, ByVal TotalVend1 As Single, ByVal TotalVend2 As Single)
    Range("C2") = ComisionVend1
    Range("C3") = ComisionVend2
    Range("E2") = TotalVend1
    Range("E3") = TotalVend2
End Sub
```

3.3.3. Pintar fachada

Se desea pintar la fachada (techo y paredes frontales) de la casa en 2D que se muestra a continuación y se sabe que sus ventanas son cuadradas, así como los anchos y altos de techo, puerta, pared y ventanas. Por tanto, se le pide hallar el área en cuestión.

Ilustración 3.3 Ejercicio desarrollado: «Pintar fachada»

Además, se sabe que las abreviaturas para cada medida y objeto son:
–«h» para la altura
–«a» para el ancho
–«p» para puerta
–«pp» para pared

Se pide implementar en VBA:

a) Un subprograma «CalcularAreaTotalCasa» que lea y calcule el área total de la fachada de la casa, y la muestre.

```
Sub CalcAreaFachada()
    Dim hp As Byte, hpp As Single, ht As Single
    Dim hv  As Single, ap As Single, app As Single,_
    at As Single,areaTotal As Single
    Call LeerDatos(hp, hpp, ht, av, ap, app, at)
    areaTotal = CalcAreaTotal(hp, hpp, ht, av, ap, app, at)
    Call Mostrar(areaTotal)
End Sub
```

b) Un subprograma «LeerDatos» que permita leer los datos de entrada.

```
Sub LeerDatos(ByRef hp As Byte, ByRef hpp As Single, ByRef ht As Single,_
    ByRef av As Single, ByRef ap As Single, ByRef app As Single,_
    at As Single)
```

```
    hp = Range("B2")
    hpp = Range("B3")
    ht = Range("B4")
    av = Range("B5")
    ap = Range("B6")
    app = InputBox("Escriba el ancho de puerta", "Ingreso de datos")
    at = Range("B8")
End Sub
```

c) Un subprograma que calcule el área total por pintar a partir de los siguientes parámetros: altura de la puerta, la pared, el techo, y el ancho de la puerta, la pared, el techo y la ventana.

```
Function CalcAreaTotal(ByVal hp As Byte, ByVal hpp As Single,_
    ByVal ht As Single, ByVal av As Single, ByVal ap As Single,_
    ByVal app As Single, ByVal at As Single) As Single
    Dim Atecho As Single, Apared As Single
    Atecho = (ht*at) / 2
    Apared = (ap*hp) - (2*av^2) –(app*hpp)
    CalcAreaTotal = Apared + Atecho
End Function
```

d) Un subprograma que muestre el área total hallada, en tanto se cuenta con el área total como parámetro.

```
Sub Mostrar(ByVal areaTotal As Single)
    Range("B9") = areaTotal & "metros cuadrados"
End Sub
```

3.3.4. Promedio del curso

Se tienen las notas parciales de un curso y se quiere obtener el promedio final a partir de la siguiente fórmula:

$$(NF=(PP*2+PL*2+E1*2+E2*4))/10$$

Donde:

–«PL» es el promedio de los laboratorios
–«PP» es el promedio de las prácticas
–«E1» es el examen parcial
–«E2» es el examen final

Ilustración 3.4 Ejercicio desarrollado: «Promedio del curso»

	A	B
1	**Concepto**	**Nota**
2	Promedio Laboratorio	15
3	Promedio Prácticas	14.7
4	Examen Parcial	12
5	Examen Final	17
6	Promedio del curso	15.14

a) Un subprograma principal que lea, calcule y muestre el promedio final. Utilice los subprogramas anteriores.

```
Sub Principal()
    Dim PP As Single, PL As Single, EP As Byte
    Dim EF As Byte, PF As Single
    Call LeerDatos(PP, PL, EP, EF)
    PF = CalcularPromedioFinal(PP, PL, EP, EF)
    Call Mostrar(PF)
End Sub
```

b) Un subprograma que calcule el promedio final en tanto tiene como parámetros de entrada los promedios de práctica, laboratorio y exámenes.

```
Function CalcularPromedioFinal(ByVal PP As Single, _
    ByVal PL As Single, ByVal EP As Byte, ByVal EF As Byte) As Single
    Dim SumaNotas As Single
    SumaNotas = PP*2+PL*2+EP*2+EF*4
    CalcularPromedioFinal = SumaNotas /10
End Function
```

c) Un subprograma «Mostrar» que imprima el promedio final del curso.

```
Sub Mostrar(ByVal PF As Single)
    Range("B6") = PF
End Sub
```

d) Un subprograma «LeerDatos» que lea los datos de la hoja de Excel.

```
Sub LeerDatos(ByRef PP As Single, _
    ByRef PL As Single, ByRef EP As Byte, ByRef EF As Byte)
    PP = Range("B2")
    PL = Range("B3")
    EP = Range("B4")
    EF = Range("B5")
End Sub
```

3.3.5. Código de barras

Es temporada navideña y usted decide empezar un negocio de tazas. Cada taza tiene un código de barras que contiene la información de cada comprador.

Entonces, para crear este código, usted decide elaborar un programa en VBA que, a partir de los siguientes datos: el país, la fecha de nacimiento (dd/mm/aaaa), el nombre y el apellido del comprador, y el género, devuelva una cadena de trece caracteres que represente un código de barras con la siguiente estructura (de izquierda a derecha):

- Los primeros dos dígitos son el código del país.
- Los siguientes seis dígitos contienen la fecha de nacimiento en el formato mmddaa.
- Los cuatro dígitos posteriores contienen el código ASCII de las iniciales del primer nombre y del primer apellido en mayúsculas NNAA.
- Finalmente, el último dígito corresponde el género: 1 para masculino y 0 para femenino.

Ilustración 3.5 Ejercicio desarrollado: «Código de barras»

	A	B
1	**Datos de cliente**	
2	País	Perú
3	Fecha de nacimiento	24/12/1989
4	Nombre y apellido	Pedro López
5	Género	Masculino
6		
7	**Salida**	
8	Código de barras	5012248980761

A partir de la información brindada se le pide implementar en VBA:

a) Un subprograma que lea las entradas, halle el código del país, la fecha, el código de las iniciales, el género, el código de barras correspondiente y muestre la salida:

```
Sub PP()
    Dim nombre As String, iniNom As Byte, iniAp As Byte
    Dim pais As String, fecha As String, genero As String
    Dim cod As String, fech As String, sexo As Byte
    pais = Range("B2")
```

```
    fecha = Range("B3")
    nombre = Range("B4")
    genero = Range("B5")
    cod = CodPais(pais)
    fecha = HallaFecha(fecha)
    Call HallaCodIniciales(nombre, iniNom, iniAp)
    sexo = HallaGenero(genero)
    Range("B8") = cod & fech & iniNom & iniAp & sexo
End Sub
```

b) Un subprograma que con el parámetro «país» devuelva el código del país correspondiente. Tenga en cuenta que para calcular el código debe restar los códigos ASCII de las dos primeras letras con los códigos ASCII de las dos segundas y aplicarles valor absoluto: *CodPais* = *CodAscii*(*Let*3) + *CodAscii*(*Let*4) – *CodAscii*(*Let* 1) – *CodAscii*(*Let* 2)|

```
Function CodPais(ByVal pais As String) As Byte
    Dim letra1 As String, letra2 As String, letra3 As String
    Dim  letra4 As String
    letra1 = Mid(pais, 1, 1)
    letra2 = Mid(pais, 2, 1)
    letra3 = Mid(pais, 3, 1)
    letra4 = Mid(pais, 4, 1)
    CodPais = Abs(Asc(letra3) + Asc(letra4) - Asc(letra1) - Asc(letra2))
End Function
```

c) Un subprograma que a partir del parámetro «fecha» en el formato dd/mm/aaaa devuelva la fecha en el formato mmddaa, en el cual «aa» corresponde a los dos últimos dígitos del año:

```
Function HallaFecha(ByVal fecha As String) As String
    Dim dd As String, aa As String, mm As String
    dd = Mid(fecha, 1, 2)
    mm = Mid(fecha, 4, 2)
    aa = Mid(fecha, 9, 2)
    HallaFecha = mm & dd & aa
End Function
```

d) Un subprograma que en tanto recibe por parámetro el nombre completo (nombre y apellido) devuelva el código ASCII de la primera letra del nombre y el código ASCII de la primera letra del apellido.

```
Sub HallaCodIniciales(ByVal nombre As String, ByRef iniNom As Byte, _
ByRef iniAp As Byte)
    Dim espb As Byte
    iniNom = Asc(Left(nombre, 1))
    espb = InStr(1, nombre, " ")
    iniAp = Asc(Mid(nombre, espb + 1, 1))
End Sub
```

e) Un subprograma que con el parámetro «sexo» devuelva 1 si el sexo es masculino y 0 si es femenino.

```
Function HallaGenero(ByVal genero As String) As String
    HallaGenero = StrComp(genero, "Femenino", 1)
End Function
```

3.3.6. Importaciones y exportaciones del año

Calcular el total de las importaciones y exportaciones, los porcentajes sobre el total y la diferencia de porcentajes de cada periodo.

Ilustración 3.6 Ejercicio desarrollado: «Importaciones y exportaciones del año»

	A	B
1	**Periodo**	**2010**
2	Exportaciones	24,543
3	% sobre el total	0.66
4	Importaciones	12,543
5	% sobre el total	0.34
6	Diferencia importación/exportación	0.32
7	Total del periodo	37,086

```
Sub ImportacionesYExportaciones()
    Dim PorcDeExport As Single, PorcDeImport As Single
    Dim Export As Integer, Import As Integer, total As Long
    Dim DiferenciaPorc As Single
    Export = Range("B2")
    Import = Range("B4")
    total = CalcularTotal(Export, Import)
    PorcDeExport = CalcPorc(Export, total)
    PorcDeImport = CalcPorc(Import, total)
    DiferenciaPorc = CalcDif(PorcDeExport, PorcDeImport)
```

```
    Range("B3") = PorcDeExport
    Range("B5") = PorcDeImport
    Range("B6") = DiferenciaPorc
    Range("B7") = total
End Sub

Function CalcularTotal(ByVal Export As Integer, ByVal Import As Integer)_
As Long
    CalcularTotal = CLng(Export) + Import
End Function

Function CalcPorc(ByVal parcial As Integer, ByVal total As Long) As Single
    CalcPorc = (parcial / total)
End Function

Function CalcDif(ByVal PorcDeExport As Single, ByVal PorcDeImport As_
Single) As Single
     CalcDif = (PorcDeExport - PorcDeImport)
End Function
```

3.4. Resumen

A continuación, mostramos ejemplos en los que se identifican los conceptos desarrollados en este capítulo.

Ilustración 3.7 Resumen del procedimiento «ProgramaPrincipal»

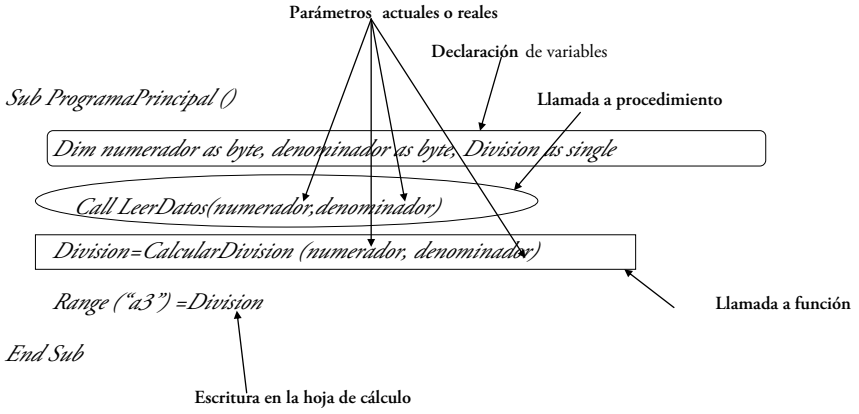


Ilustración 3.8 Resumen de la función «CalcularDivision»

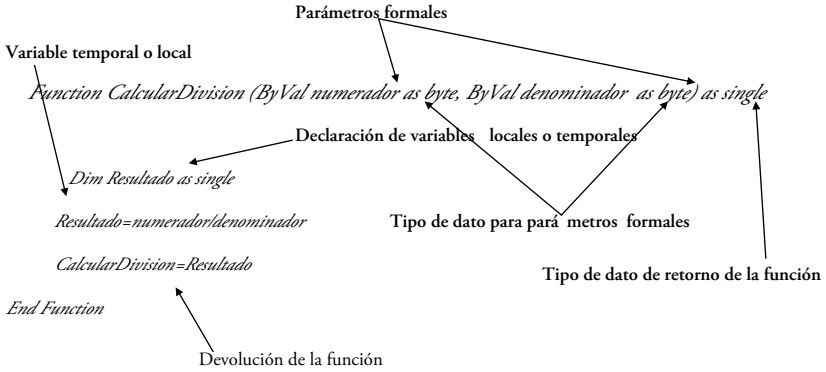
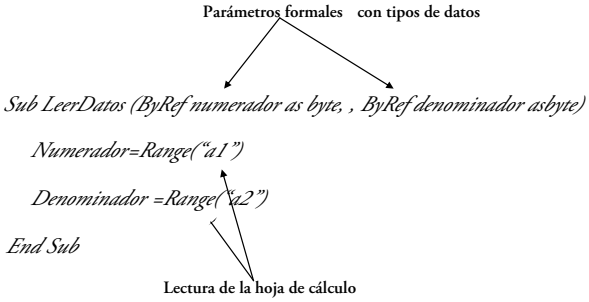


Ilustración 3.9 Resumen del procedimiento «LeerDatos»



3.5. Ejercicios propuestos

3.5.1. Nombre de dominio

Se quiere generar el enlace a la página web de una empresa, o también llamado nombre de dominio, para lo cual se cuenta con el nombre y el rubro de la empresa, el número de letras que se deben sacar del rubro, el país, el número de letras que se deben sacar del país, y una longitud máxima permitida que no debe exceder al enlace resultante.

Ilustración 3.10 Ejercicios propuesto: «Nombre de dominio»

	A	B	C	D	E	F	G	H
1	**Nombre de la empresa**	**Rubro**	Número de letras por sacar del rubro	País	Número de letras por sacar del país	Enlace	Longitud máxima permitida	Enlace aceptado
2	Pontificia Universidad Católica del Perú	Educación	3	Perú	2	www.pucp.edu.pe	60	VERDADERO

Se desea escribir un programa en VBA que halle el enlace y verifique si la longitud teórica es igual a la real, por lo que se solicita la implementación de:

a) Un subprograma principal que, a partir de los subprogramas anteriores, realice lo siguiente:
   - Lea los datos de entrada.
   - Obtenga las siglas de la empresa.
   - Cree el enlace, para lo cual se debe recordar que todas las letras de este deben ir en minúsculas.
   - Complete el enlace agregándole delante tres veces la letra «w» y un punto.
   - Verifique si el enlace es menor o igual a la longitud máxima permitida.
   - Y, finalmente, muestre el enlace hallado y la respuesta a la verificación hecha, tal como se muestra en la hoja de cálculo (ver Ilustración 3.10).

b) Un subprograma llamado «ObtenerSiglasEmpresa» que reciba por parámetros el nombre de la empresa y devuelva, por separado, las cuatro letras correspondientes a las siglas de dicho nombre.

c) Un subprograma llamado «CrearEnlace» que reciba como parámetros las cuatro letras correspondientes a las siglas de la empresa, el rubro, el número de letras que se debe sacar de este, el país, el número de letras que se debe sacar del país, y devuelva una cadena formada por las siglas de la empresa, los caracteres del rubro y del país, separados por puntos y en ese orden. Por ejemplo, para la hoja de cálculo mostrada, este subprograma deberá devolver la cadena: «pucp.edu.pe».

d) Un subprograma llamado «VerificarEnlace» que reciba por parámetros el enlace hallado y la longitud máxima permitida, y coteje si la longitud del enlace hallado es menor o igual que la longitud máxima permitida. Por ejemplo, para la hoja de cálculo mostrada, este subprograma deberá devolver el valor lógico «VERDADERO».

3.5.2. Presupuesto de fiesta

Una empresa de eventos debe organizar una fiesta de graduación. De este modo, para calcular el presupuesto utilizará una hoja de cálculo como la que se muestra:

Ilustración 3.11 Ejercicio propuesto: «Presupuesto de fiesta»

	A	B	C
1	**Concepto**	**Precio($)**	
2	Cena por invitado	24.00	
3	Video	100.00	
4	Música	150.00	
5	Local	900.00	
6			
7	Concepto		**Monto($)**
8	Costo total sin descuento		3,550.00
9	Costo total con descuento		3,195.00
10	Costo de entrada por alumno		31.95
11			
12	**Información adicional**		
13	Número de invitados		100
14	Descuento(%)		10

La persona a cargo desea, a partir de los costos de la cena por invitado, el video, la música y el alquiler del local, hallar el costo total con y sin descuento, así como el precio unitario por invitado.

3.5.3. Presupuesto de compras

Todos los martes son días de remate en una tienda por departamentos: la ropa tiene 20% de descuento, los zapatos están a mitad de precio y los accesorios tienen 10% de descuento. Además, las personas que paguen con tarjeta visa tendrán derecho a un 40% de descuento adicional sobre el precio total de las compras realizadas.

La finalidad de la hoja de Excel mostrada a continuación es saber cuál es el monto que gastaría una determinada persona por cada concepto (ropa, zapatos y accesorios); el monto total si paga con tarjeta visa (total con descuento por visa), y el monto total si cancela con otro medio de pago (total).

Además, dicha persona odia tener deudas con su tarjeta, por lo que paga parte del monto gastado (monto depositado) y desea saber cuál es el monto que debe cancelar cuando finalice el mes (monto pendiente de pago).

Ilustración 3.12 Ejercicio propuesto: «Presupuesto de compras»

	A	B	C	D
1				
2		**Concepto**	**Precio original**	**Subtotal con descuento**
3		Ropa	546.40	437.12
4		Zapatos	263.00	131.50
5		Accesorios	97.00	87.30
6			Total	655.92
7			Total con descuento por visa	393.55
8			Monto depositado	200.00
9			Monto pendiente de pago	193.55

3.5.4. Ejercicios de cadenas

Elaborar en VBA:

a) Un subprograma que reciba una cadena y la devuelva invertida en mayúsculas.

b) Un subprograma que reciba una cadena y la devuelva invertida en minúsculas.

3.5.5. Pago de boleta de matrícula

La administración de la Universidad incrementará los costos de los cursos que se lleven por segunda y tercera vez.

Óscar Véliz, alumno de la Universidad que se encuentra en la escala 2, se entera de esta posibilidad y, por ello, quiere calcular cuánto tendría que pagar mensualmente durante el siguiente ciclo. Para ello, se sabe que la primera boleta debe incluir el derecho de matrícula y que el pago de los créditos se divide en cuatro partes iguales.

Un dato adicional es que, a partir del próximo ciclo, los cursos que se lleven por segunda vez costarán el doble y los cursos que se lleven por tercera vez, el triple.

Además, Óscar conoce los nombres de los cursos en los que se matriculará, unos datos adicionales respecto a estos y cómo se obtienen las claves de ellos, pero no cuenta con las claves de dichos cursos, necesarias para llevar a cabo su matrícula.

Como el ciclo pasado Óscar llevó el curso de Introducción a la Computación, decide hacer una pequeña aplicación usando VBA sobre Excel, con la que calculará toda la información necesaria.

Finalmente, ante la preocupación de jalar otros cursos y tener que pagar más, decide enviar un correo a sus futuros profesores presentándose y consultándoles qué material podría revisar de antemano.

Ilustración 3.13 Ejercicio propuesto: «Pago de boleta de matrícula»

	A	B	C	D	E	F	G	H	I
1	**Nombre del curso**	**Créditos**	**Vez**	**Nivel**	**Contador**	**Especialidad**	**Profesor**	**Clave**	**Correo**
2	Introducción a la Computación	3	2	1	7	Informática	Vanessa Zegarra Martínez	INF117	vzegarram@pucp.edu.pe
3	Introducción a la Física Universitaria	4	1	0	9	Física	Maria Rosa del Río Torres	FIS009	mdel_riot@pucp.edu.pe
4	Total de créditos	7							
5	Pago de primera boleta	722.50							
6	¿Alcanza?	VERDADERO							

Por tanto, Óscar necesita implementar:

a) Un subprograma principal que lea los datos de entrada, halle la clave, el correo, calcule el total de créditos, el monto de la primera boleta y determine si el presupuesto de S/. 3400 con el que cuenta Óscar le alcanza o no para pagar el ciclo. Por último, este subprograma debe imprimir las salidas en la hoja de Excel.

b) Un subprograma en VBA, «HallaClave» que reciba por parámetros la especialidad, el nivel y el contador de un curso, y que halle y devuelva la clave correspondiente de este. Dicha clave se formará a partir de las tres letras iniciales de la especialidad, seguidas por el nivel repetido y, finalmente, el contador.

c) Un subprograma llamado «HallaCorreo» que reciba por parámetros el nombre del profesor, y halle y devuelva el correo con la siguiente estructura: la primera letra del nombre, seguida del primer apellido y, luego, la primera letra del segundo apellido seguida de «@pucp.edu.pe».

d) Un subprograma llamado «HallaTotalCreditosyMontoCuota» que reciba por parámetros el número de créditos de los dos cursos, la vez correspondiente de ambos, y que calcule y devuelva el total de créditos, así como el monto de la primera cuota.

3.5.6. Código de barras personalizado

Es temporada navideña y usted decide comprar tazas personalizadas para varios amigos. Cada taza tiene un código de barras que contiene la información de cada uno de ellos.

Ahora, es momento de envolver los regalos, pero no sabe a quién pertenece cada taza, por lo que decide elaborar un programa en VBA que, con un número del código de barras como parámetro, identifique la fecha de nacimiento y las iniciales del nombre, para poder colocar las dedicatorias correctas.

Ahora bien, el código de barras se crea de la siguiente manera (de izquierda a derecha):

- Los primeros dos dígitos son el código del país.
- Los siguientes seis dígitos contienen la fecha de nacimiento en el formato ddmmaa.
- Los cuatro dígitos posteriores contienen el código ASCII de las iniciales del primer nombre y del primer apellido en mayúsculas NNAA.
- Finalmente, el último dígito corresponde al género: 1 para masculino y 0 para femenino.

A partir de la información brindada se le pide:

a) Plantear la definición y el análisis del problema con al menos cuatro módulos (incluido el principal).

b) Implementar en VBA lo trazado en la parte a).

Nota: el código de barras debe ser leído y manejado como un número y no como una cadena.

3.6. Ejercicio de repaso

Se acerca el día de la madre y Juan ha estado ahorrando durante semanas para comprarle a su mamá varios regalos. Él cuenta con varios presupuestos parciales para cada uno de ellos y quiere saber, por medio de una aplicación en VBA, si dichos presupuestos le alcanzarán y cuál será el monto que pagará por cada uno de los tres regalos. Además, si le da los tres regalos, quiere saber cuál sería el total por los tres y cuánto necesitaría ahorrar para podérselos comprar a su mamá.

Para hallar estos montos, utilice el siguiente diagrama de módulos:

Ilustración 3.14 Ejercicio de repaso «Regalo a mamá»: diagrama de módulos

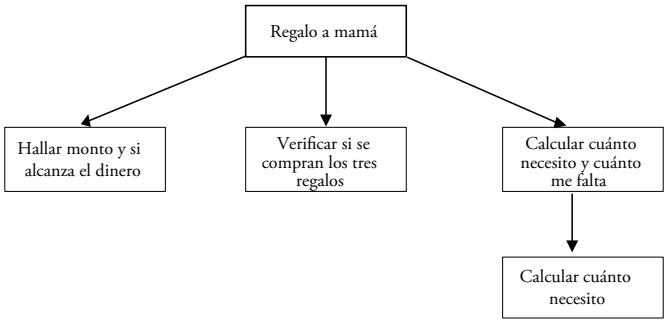


Ilustración 3.15 Ejercicio de repaso «Regalo a mamá»: hoja de Excel

	A	B	C	D	E	F
1	**Regalo**	**Precio**	**Descuento**	**Presupuesto parcial**	**Alcanza el presupuesto parcial**	**Monto**
2	Masajes relajantes	251.00	15%	240.00	VERDADERO	213.35
3	Televisor	1,300.00	10%	1,000.00	FALSO	1,170.00
4	Paquete de belleza en peluquería	400.00	5%	450.00	VERDADERO	380.00
5				¿Se le dan los tres regalos?	FALSO	
6					Total por pagar	1,763.35
7					¿Cuánto necesito ahorrar?	73.35

3.6.1. Definición

3.6.1.1. Explicación

- Verificar si a Juan le alcanzará el presupuesto para comprar los tres regalos y calcular cuánto debe ahorrar para lograrlo.
- Datos de entrada: precio, descuento y presupuesto para cada uno de los regalos (nueve datos de entrada).

3.6.1.2. Salidas

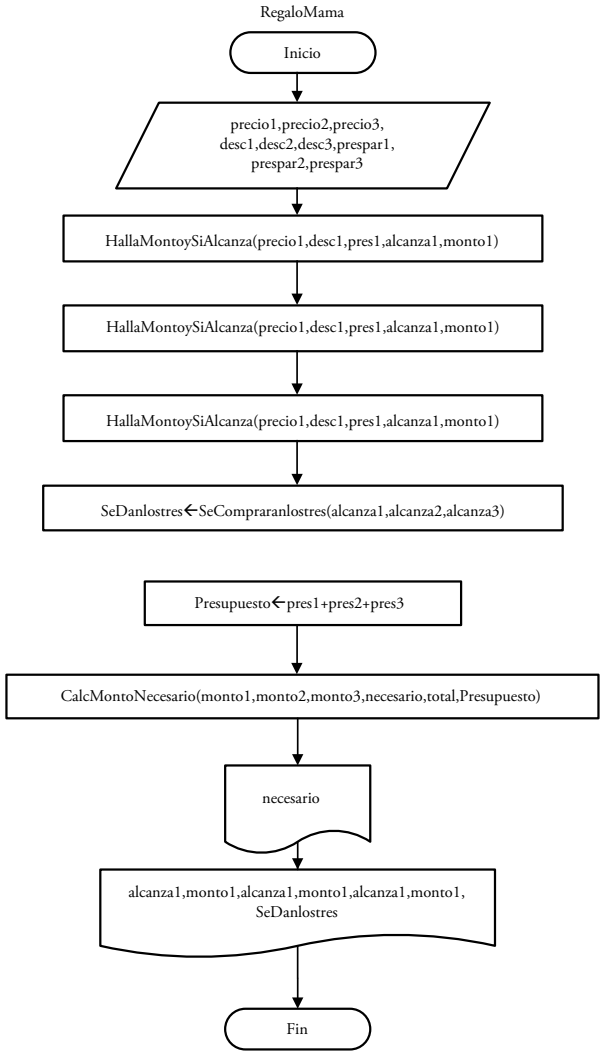
- Verificar, por un lado, si el presupuesto parcial de Juan alcanzará para comprar cada uno de los regalos y, por el otro, calcular el monto real de cada uno de ellos. Luego, determinar si se darán los tres regalos: para ello, se deberá calcular el monto total por pagar y el importe que Juan necesita ahorrar.

3.6.1.3. Fórmulas

- Monto = precio*(1-desc)
- Alcanza = monto<=presupuestoParcial
- Total por pagar = monto regalo 1 + monto regalo 2+ monto regalo3
- Presupuesto total = pres1+ pres2+ pres3
- Necesita ahorrar = total por pagar – presupuesto total

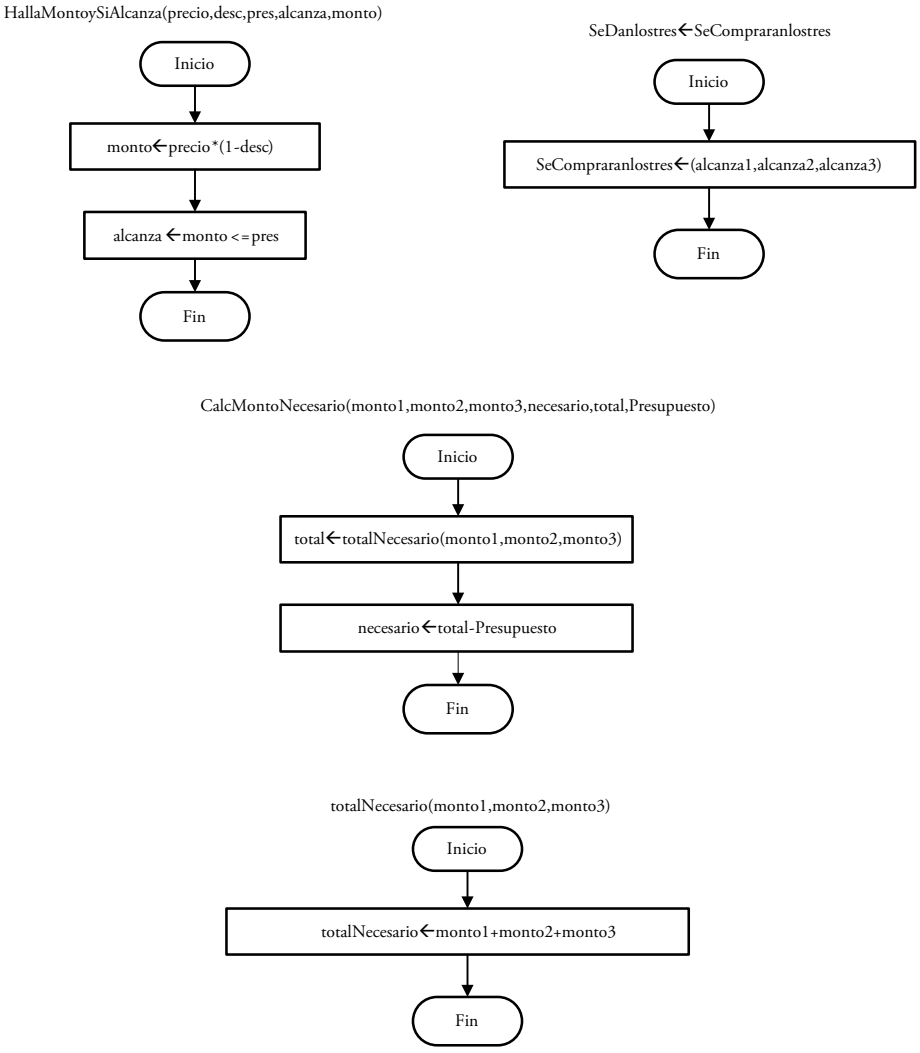
3.6.2. Diseño

Ilustración 3.16 Ejercicio de repaso «Regalo a mamá»: módulo principal



A continuación, sobre la base del diagrama de módulos y la definición, se plantea el diseño a través, en este caso, de los siguientes diagramas de flujo (ver Ilustración 3.18). Recuerde que en esta parte del ejercicio deberá utilizar las fórmulas dadas y realizar el desarrollo del diagrama de acuerdo con lo solicitado.

Ilustración 3.17 Ejercicio de repaso «Regalo a mamá»: diagramas de flujo



3.6.3. Codificación

```
Sub HallaMontoySiAlcanza(ByVal p As Single, ByVal desc As Single, ByRef
monto As Single, ByVal pres As Single, ByRef alcanza As Boolean)
        monto = p*(1 - desc)
        alcanza = pres >= monto
End Sub

Sub RegaloMama()
        Dim pmas As Single,descmas As Single,montoMas As Single,totalNec_
        As Single
        Dim  presmas As Single, alcanzaMas As Boolean, danlos3 As Boolean
        Dim ptel As Single, desctel As Single, montoTel As Single, prestel As Single
        Dim alcanzaTel As Boolean, ppaq As Single, descpaq As Single falta As Single
        Dim  montoPaq As Single, prespaq As Single, alcanzaPaq As Boolean
        pmas = Range("B2")
        descmas = Range("C2")
        presmas = Range("D2")
        ptel = Range("B3")
        desctel = Range("C3")
        prestel = Range("D3")
        ppaq = Range("B4")
        descpaq = Range("C4")
        prespaq = Range("D4")
    Call HallaMontoySiAlcanza(pmas, descmas, montoMas, presmas, _alcanzaMas)
    Call HallaMontoySiAlcanza(ptel, desctel, montoTel, prestel, alcanzaTel)
    Call HallaMontoySiAlcanza(ppaq, descpaq, montoPaq, prespaq,alcanzaPaq)
    danlos3 = Verifica(alcanzaMas, alcanzaTel, alcanzaPaq)
    Call HallaNecesitayFalta(montoMas, montoTel, montoPaq, totalNec, _
    presmas, prestel, prespaq, falta)
         Range("F7") = falta
        Range("E2") = alcanzaMas
        Range("E3") = alcanzaTel
        Range("E4") = alcanzaPaq
        Range("F2") = montoMas
        Range("F3") = montoTel
        Range("F4") = montoPaq
        Range("E5") = danlos3
        Range("F6") = totalNec
End Sub
```

```
Function Verifica(ByVal alcanzaMas As Boolean, ByVal alcanzaTel As Boolean,_
ByVal alcanzaPaq As Boolean) As Boolean
    Verifica = (alcanzaMas And alcanzaTel And alcanzaPaq)
End Function

Sub HallaNecesitayFalta(ByVal montoMas As Single, ByVal montoTel As Single,_
ByVal montoPaq As Single, _
ByRef totalNec As Single, ByVal presmas As Single, ByVal prestel As Single, ByVal
 prespaq As Single, ByRef falta As Single)
    totalNec = montoMas + montoTel + montoPaq
    falta = HallaNecesario(totalNec, presmas, prestel, prespaq)
End Sub

Function HallaNecesario(ByVal totalNec As Single, ByVal presmas As Single,_
ByVal prestel As Single, ByVal prespaq As Single) As Single
    HallaNecesario = (totalNec - presmas - prestel - prespaq)
End Function
```

ESTRUCTURAS ALGORÍTMICAS SELECTIVAS

Estas estructuras combinan instrucciones o sentencias individuales, pues nos permiten elegir entre dos o más alternativas o conjuntos de instrucciones. Anteriormente hemos utilizado instrucciones booleanas como la siguiente:

```
Function Verificar(byval nota as byte)as Boolean
    Verificar=(nota>=0 and nota<=20)
End Function
```

Sin embargo, estas instrucciones, a veces, dependiendo de la complejidad del problema, se vuelven muy complicadas y, en muchos casos, no nos dejan manejar ciertas condiciones o incluir algunas acciones. Es por todas estas razones que utilizaremos las estructuras selectivas.

4.1. Tipos de estructuras selectivas

Llamadas también «estructuras de control selectivas», pues se utilizan para tomar decisiones bajo ciertas condiciones, estas pueden ser simples, dobles, múltiples e incluso anidadas, según la complejidad de los caminos posibles.

4.1.1. Estructuras selectivas simples

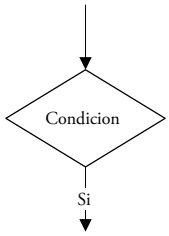
Este tipo de estructura permite un solo camino y funciona de la siguiente manera: si la condición es verdadera, se ejecuta el bloque de acciones; si es falsa, se sale de la estructura y se ejecutan las instrucciones siguientes.

4.1.1.1. Diseño de estructuras selectivas simples

Al igual que en el resto del libro, para el diseño de este tipo de estructuras se utilizarán diagramas de flujo y pseudocódigos.

Diagrama de flujo de estructuras selectivas simples

Ilustración 4.1 Diagrama de flujo de estructura selectiva simple



El diagrama mostrado (Ilustración 4.1) explica que si la condición se cumple, se seguirá el camino del sí; en caso de que no se cumpla, se seguirá el camino del no. Finalmente, ambos caminos se unirán para llegar a un fin o procesos comunes. En este caso el camino del «Sí» tiene dirección hacia abajo, pero podría ser lateral y en su lugar colocar el camino del «No». Incluso se podría dejar de colocar el camino del «No», ya que en este no se realizarán acciones, como se ha mostrado en la ilustración anterior.

Pseudocódigo de estructuras selectivas simples

En el diagrama de flujo anterior veíamos el símbolo del rombo con dos caminos posibles para manejar las decisiones; en el pseudocódigo, en cambio, se utiliza la siguiente estructura:

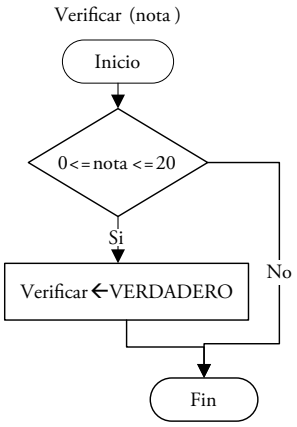
Si condición entonces
    Instrucciones
Fin si

En General, las condiciones son comparaciones formadas con operadores como «>»,«>=», «<», «<=», «<>», «=».

Ejemplos de diseño de estructuras selectivas simples

- **Diagrama de flujo:**

Ilustración 4.2 Ejemplo de diagrama de flujo de estructura selectiva simple



- **Pseudocódigo:**

```
Inicio Verificar(nota)
    Si 0<=nota<=20 entonces
        Verificar← VERDADERO
Fin si

    'si no se cumple la condición, esta función devolverá «FALSO», ya que el
    'valor de iniciación de un booleano es falso.
Fin Verificar
```

4.1.1.2. Codificación de estructuras selectivas simples

La estructura selectiva simple se codifica en VBA así:

```
If <condicion> then
    Instrucciones
End if
```

Estas estructuras pueden incluirse en cualquier parte de la programación y las instrucciones pueden ser de todo tipo, así como de cualquier cantidad. La única restricción de uso es que deben iniciar con «If» y finalizar con «End if».

Finalmente, por cada «If» debe haber un «End if», a menos que el «If», la condición y la (única) instrucción estén en una sola línea:

```
If condicion then Instruccion
```

Ejemplos de codificación de estructuras selectivas simples

```
Function Verificar(byval nota as byte) as Boolean
    If nota>=0 and nota<=20 then
        Verificar=true
    End if
End function
o
Function Verificar(byval nota as byte) as Boolean
    If nota>=0 and nota<=20 then  Verificar=true
End function
```

4.1.2. Estructuras selectivas dobles

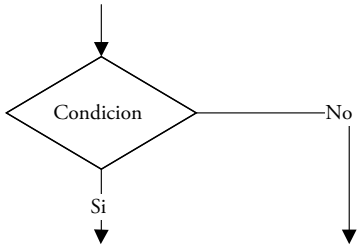
Con este tipo de estructuras podemos manejar dos posibles caminos y, en cada uno de ellos, llevar a cabo una o varias instrucciones.

4.1.2.1. Diseño de estructuras selectivas dobles

Para el diseño de este tipo de estructuras utilizaremos, como hemos hecho anteriormente, los diagramas de flujo y los pseudocódigos.

Diagrama de flujo de estructuras selectivas dobles

Ilustración 4.3 Diagrama de flujo de estructuras selectivas dobles



El diagrama mostrado (Ilustración 4.3) explica que si la condición se cumple se seguirá el camino del sí y se ejecutarán las instrucciones que se encuentren en él; en tanto que, si no se cumple, se seguirá el camino del no. A diferencia de las estructuras selectivas simples, en el camino del no de las dobles encontraremos instrucciones por realizar. Finalmente, los dos caminos se unirán para llegar a un fin o procesos comunes.

Pseudocódigo de estructuras selectivas dobles

En el diagrama de flujo veíamos el símbolo del rombo con dos caminos posibles para manejar las decisiones; en cambio, en el pseudocódigo, utilizaremos la siguiente estructura:

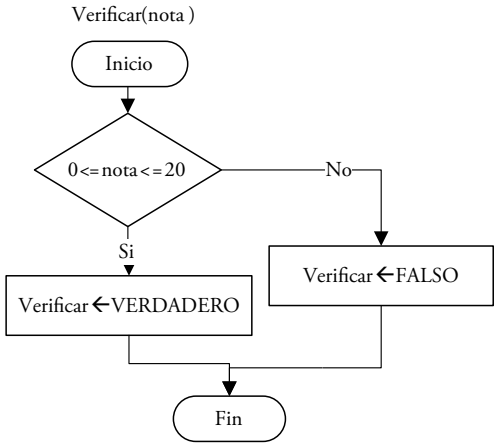
Si condicion entonces
Instrucciones
Sino
Instrucciones
Fin si

En general, las condiciones son comparaciones formadas con operadores como «>», «>=», «<», «<=», «<>», «=».

Ejemplos de diseño de estructuras selectivas dobles

- **Diagrama de flujo:**

Ilustración 4.4 Ejemplo de diagrama de flujo de estructura selectiva doble



- **Pseudocódigo:**

Inicio Verificar(nota)
Si 0<=nota<=20 entonces
Verificar←VERDADERO
Sino
Verificar←FALSO
Fin si
Fin Verificar

4.1.2.2. Codificación de estructuras selectivas dobles

La estructura selectiva doble se codifica en VBA de la siguiente manera:

```
If <condicion> then
    Instrucciones
Else
    Instrucciones
End if
```

Estas estructuras pueden incluirse en cualquier parte de la programación y sus instrucciones pueden ser de todo tipo, así como de cualquier cantidad. La única restricción de uso sobre esta estructura es que debe iniciarse con «If», seguido de la condición y el «then» correspondiente. Luego, irán las instrucciones que se realizan si se cumple dicha condición.

El «else» no tiene una condición, pero se sobreentiende que las instrucciones posteriores a este y previas al «End if» son las que se realizarán si la condición inicial no se cumple. Finalmente, el bloque de estructura deberá culminar con «End if».

Ejemplos de diseño de estructuras selectivas dobles

```
Function Verificar(byval nota as byte) as Boolean
    If nota>=0 and nota<=20 then
        Verificar=true
    Else
        Verificar=false
    End if
End function
```

Es necesario notar que en este caso la condición por evaluar: 0<=nota<=20, se modificó a: nota>=0 and nota<=20, debido a que estamos en la fase de la codificación y esta no permite ese tipo de uniones.

4.1.3. Estructuras selectivas múltiples

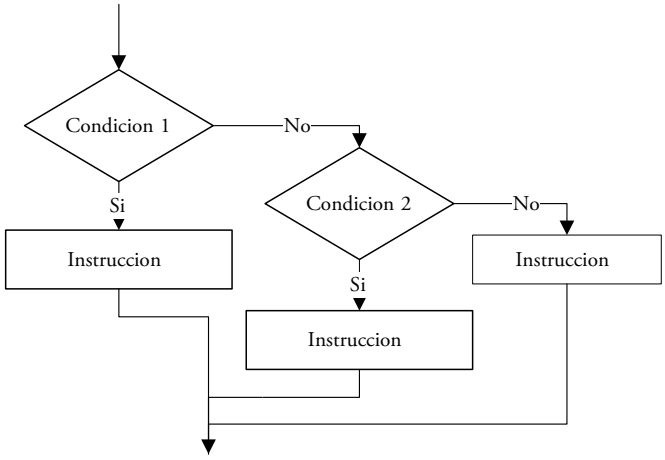
Con este tipo de estructuras podemos manejar tantos caminos posibles como se necesiten y, en cada uno de ellos, llevar a cabo una o varias instrucciones.

4.1.3.1. Diseño de estructuras selectivas múltiples

Al igual que en el resto del libro, para el diseño de este tipo de estructuras utilizaremos los diagramas de flujo y los pseudocódigos.

Diagrama de flujo de estructuras selectivas múltiples

Ilustración 4.5 Diagrama de flujo de estructura selectiva múltiple



El diagrama mostrado previamente (Ilustración 4.5) explica que, si la condición se cumple, se seguirá el camino del sí y se ejecutarán las instrucciones que se encuentren en dicho camino; en caso de que no se cumpla la condición, se evaluará la siguiente, para repetir el proceso anterior y, así sucesivamente, hasta que se cumpla alguna de las condiciones o se salga de la estructura.

Pseudocódigo de estructuras selectivas dobles

En el diagrama de flujo veíamos el símbolo del rombo con dos caminos posibles para manejar las decisiones; en cambio, en el pseudocódigo utilizaremos la siguiente estructura:

```
Si condicion entonces
    Instrucciones
Sino Si condicion entonces
    Instrucciones
Sino Si condicion entonces
    Instrucciones
Sino
    Instrucciones
Fin si
```

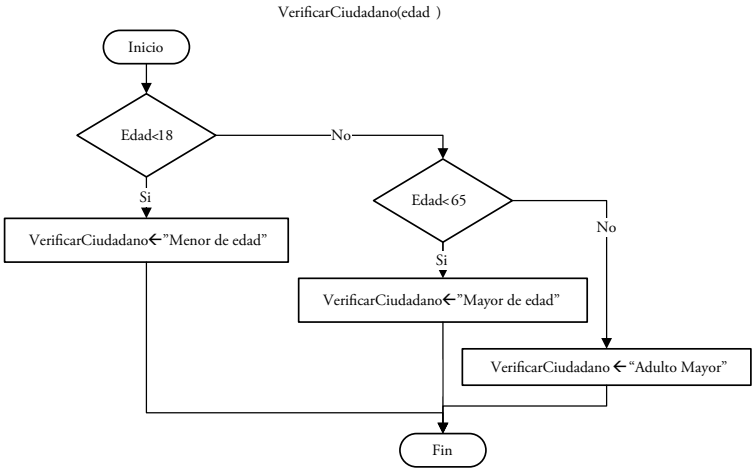
Es necesario recordar que puede haber muchos «Sino Si» adicionales, y que el «Sino» del final es opcional. Pero recordemos que siempre debemos concluir la estructura con el «Fin si».

> *En general, las condiciones son comparaciones formadas con operadores como «>», «>=», «<», «<=», «<>», «=».*

Ejemplos de diseño de estructuras selectivas múltiples

- **Diagrama de flujo:**

Ilustración 4.6 Ejemplo de diagrama de flujo de estructura selectiva múltiple



- **Pseudocódigo:**

```
Inicio VerificarCiudadano(edad)
    Si edad<18 entonces
        VerificarCiudadano ← "Menor de edad"
    Sino si edad<65 entonces
        VerificarCiudadano← "Mayor de edad"
    Sino
        VerificarCiudadano← "Adulto Mayor"
    Fin si
Fin Verificar
```

En este ejemplo se evalúa una primera condición. Si esta se cumple, se ejecuta la instrucción correspondiente; solo en el caso de que no se cumpla, se evalúa la que le sigue y, si esa tampoco se cumple, se lleva a cabo la siguiente instrucción.

Recordemos que solo una de las posibles condiciones puede cumplirse; por tanto, solo tomaremos uno de los caminos. Además, tengamos en cuenta que las condiciones pueden estar formadas por una o varias condiciones unidas por distintos operadores lógicos.

4.1.3.2. Codificación de estructuras selectivas múltiples

La codificación en VBA de la estructura selectiva múltiple es la siguiente:

```
If <condicion> then
   Instrucciones
Elseif <condicion> then
      Instrucciones
Else
      Instrucciones
End if
```

Estas estructuras pueden incluirse en cualquier parte de la programación y las instrucciones pueden ser de todo tipo, así como de cualquier cantidad. La única restricción de uso sobre esta es que debe iniciarse con «If», seguido de la condición y el «then» correspondiente; luego, irán las instrucciones que se realizan si se cumple dicha condición. Asimismo, cada uno de los «Elseif» debe ir seguido de una condición y del «then» correspondiente y, luego, las instrucciones que se deseen realizar en el caso de que esa condición sea verdadera. Finalmente, de manera opcional, se puede colocar un «else» y las instrucciones que se quieran ejecutar, en caso ninguna de las opciones anteriores haya sido verdadera. El bloque de estructura debe finalizar con «End if».

Ejemplos de codificación de estructuras selectivas múltiples

```
Function VerificarCiudadano(byval edad as byte) as String
   If edad<18 then
      VerificarCiudadano="Menor de edad"
   Elseif edad<65 then
      VerificarCiudadano="Mayor de edad"
   Else
      VerificarCiudadano="Adulto mayor"
   End if
End function
```

4.2. Ejemplos de diseño

Para identificar que la utilización de este tipo de estructuras es necesaria debemos observar que el problema presenta distintas opciones y que, de cumplirse alguna o más de una, ciertas instrucciones se llevarán a cabo.

4.2.1. Notas y mensajes

Se tiene una función que clasifica la nota del promedio de un alumno, tal y como se muestra a continuación:

Ilustración 4.7 **Ejemplo de diseño: «Notas y mensajes»**

Rango de nota	Mensaje
[20,17 [	Obtuvo una A
[17,14 [	Obtuvo una B
[14,10 [	Obtuvo una C
[10,7 [	Obtuvo una D
[7, 4 [	Obtuvo una E
[4, 0[	Obtuvo una F

Se debe asumir que la nota es válida, es decir, que tiene un valor entre 0 a 20.

Por ejemplo, si el alumno tuviera por nota 8, entonces la función debería devolver el mensaje: «Obtuvo una D». Si sacó 4, entonces obtendría el mensaje: «Obtuvo una F».

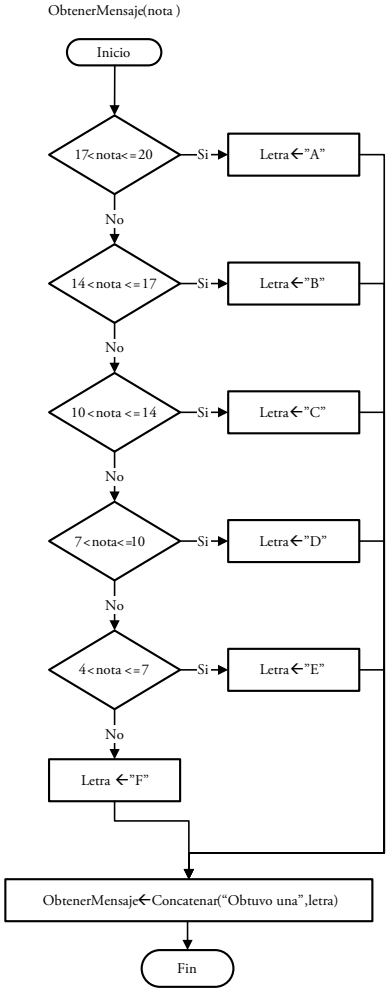
Se recibe la nota como parámetro formal y se devuelve el mensaje adecuado en una cadena.

- **Paso 1**

Evaluamos el primer rango: si este sí se cumple, entonces asignamos al mensaje el texto correspondiente; en caso contrario, examinamos el siguiente rango y así sucesivamente. En este caso, observamos que, luego de que hemos asignado el texto al mensaje, se sigue el camino hasta la instrucción en común; por tanto, devolvemos el valor por medio del nombre de la función y, finalmente, terminamos el diagrama.

4.2.1.1. Diagrama de flujo

Ilustración 4.8 Diagrama de flujo del ejemplo «Notas y mensaje»: Función «ObtenerMensaje»



Lo primero que se debe observar en este tipo de problemas es que se presentan varias opciones y que, luego del análisis, solo una se cumplirá. Una vez que esto suceda, las siguientes ya no serán evaluadas.

a) Primero es necesario que reconozcamos cuál o cuáles serán las variables por analizar; en este caso, «la nota».

b) Luego, debemos tener en cuenta qué valores podemos tomar, para luego tratar de factorizar condiciones o instrucciones. Por ejemplo, podríamos analizar la condición para cada uno de los valores posibles en el rango de 0 a 20, lo cual generaría veinte condiciones por verificar. Pero, en realidad, lo más recomendable es considerar las instrucciones que se generarían si las condiciones se cumplen. De modo que, si factorizamos adecuadamente las condiciones, solo tendremos seis por verificar en lugar de veinte.

c) Finalmente, identificamos si es necesario anidar o no condiciones: como solo hemos analizado una variable necesitaremos estructuras selectivas simples y, cada vez que una no se cumpla, verificaremos la siguiente.

4.2.1.2. Pseudocódigo

Ahora, desarrollaremos el ejemplo anterior pero usando el pseudocódigo. Existen distintas maneras de colocar las mismas condiciones y algunas de estas se muestran a continuación:

```
Inicio ObtenerMensaje(nota)
    Si 17<nota=<20 entonces
        mensaje←"A"
    Sino Si  14< nota entonces
        mensaje←"B"
    Sino Si nota=14 o nota=13 o nota=12 o nota=11  entonces
        mensaje←"C"
    Sino Si 7  <=nota entonces
         mensaje←"D"
    Sino Si 4<=  nota  entonces
        mensaje←"E"
    Sino Si 0  <=nota entonces
         mensaje←"F"
    Fin Si
    ObtenerMensaje ← Concatenar("Obtuvo una",mensaje)
Fin ObtenerMensaje
```

4.2.2. Verificación de nota válida

Se desea elaborar una función que verifique si una nota es válida o no, en tanto para ser válida, esta debe de estar entre el rango del 0 al 20.

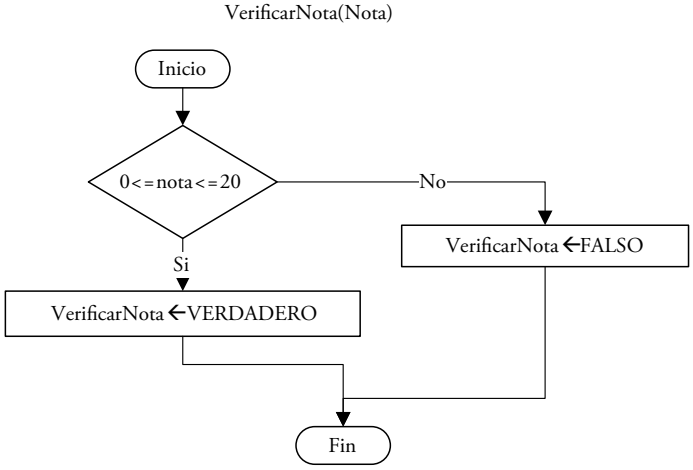
Por ejemplo, si la nota entregada fuera -34, esta no sería válida, por lo que la función «VerificarNota» devolvería el valor «FALSO»; en cambio, si la nota fuera 13, entonces devolvería el valor «VERDADERO».

Ahora bien, como podemos observar, este subprograma devuelve un solo valor y, por tanto, es una función. Asimismo, es evidente que como su único parámetro es la nota, dicha función devolverá un valor booleano de VERDADERO o FALSO. Consecuentemente, ninguno de estos valores estará entre comillas y serán considerados como constantes.

> *Existen únicamente dos posibles respuestas: VERDADERO y FALSO. La manera de plantear los caminos puede variar, pero lo importante es que todos los casos estén cubiertos.*

4.2.2.1. Diagrama de flujo

Ilustración 4.9 Diagrama de flujo del ejemplo «Notas y Mensajes»: Función «VerificarNota»



4.2.2.2. Pseudocódigo

```
Inicio VerificarNota(nota)
    Si 20 <= nota <= 0 entonces
        VerificarNota←VERDADERO
    Sino
        VerificarNota←FALSO
     Fin Si
Fin VerificarNota
```

4.3. Ejercicios desarrollados de diseño

4.3.1. Distribuidora de autos

En una empresa distribuidora de autos se asignan los sueldos a partir de las comisiones por ventas y de un mínimo de unidades vendidas. Ahora bien, se desea calcular el sueldo que se le debe pagar a un determinado vendedor. Asimismo, se sabe que si el vendedor no sobrepasa el monto mínimo, su sueldo será de S/. 550; por el contrario, si logra su meta, su sueldo se calculará a partir de las comisiones obtenidas en soles sobre el precio base.

Ilustración 4.10 Ejercicio desarrollado: «Distribuidora de autos»

	A	B	C	D	E	F
1	Nombre	Cantidad de ventas	Comisión	Mínimo requerido	Precio base	Sueldo
2	Juan Robles	4	5%	3	20,000.00	4,000.00

A partir de lo expuesto, se pide:

a) Elaborar un subprograma principal que lea los datos del vendedor, calcule su sueldo a través del siguiente subprograma y lo muestre:

```
Inicio SueldoVendedor()
    Leer cantVentas, comision, minReq, precioBase
    Sueldo←CalcSueldo(cantVentas, comision, minReq, precioBase)
    Escribir Sueldo
Fin SueldoVendedor
```

b) Implementar un subprograma que calcule el sueldo de cada vendedor, en tanto se tienen como parámetros la cantidad de ventas, la comisión, el mínimo requerido y el precio base. Utilizar los subprogramas siguientes:

```
Inicio CalcSueldo(cantVentas, comision, minReq, precioBase)
    Cumple ←VerificarCumplio(cantVentas,minReq)
    Si Cumple = true entonces
        Sueldo←SueldoPorComision(cantVentas,PorcCom,precioBase)
    Sino
        Sueldo←550
    Fin si
        CalcSueldo←Sueldo
Fin CalcSueldo
```

c) Desarrollar un subprograma que verifique si se cumplió con el mínimo requerido, la cantidad de las ventas y el mínimo requerido de un vendedor como parámetros de entrada:

```
Inicio VerificarCumplio(cantVentas,minReq)
    Si cantVentas>=minReq entonces
        VerificarCumplio←verdadero
    Sino
        VerificarCumplio←falso
    Fin si
Fin VerificarCumplio
```

d) Generar un subprograma que calcule el sueldo por comisiones de un vendedor, en tanto se cuenta con la cantidad de ventas, el porcentaje de comisión y el precio base como parámetros de entrada.

```
Inicio SueldoPorComision(cantVentas,PorcCom,precioBase)
    SueldoPorComision←(cantVentas*precioBase)* PorcCom
Fin SueldoPorComision
```

4.3.2. Costo de viaje

Una persona desea irse de viaje por vacaciones y, con el fin de calcular cual será el costo del pasaje, ha obtenido la siguiente información:

Ilustración 4.11 Ejercicio desarrollado: «Costo de viaje». Porcentajes y presupuestos de viaje por temporada

Edad de pasajero	Costo del pasaje
Entre 0 - 2	0.00
Entre 2 - más	2,000.00

Tipo de pasajero	Porcentaje incrementos
Niño sin compañía adulta	10%
Adulto con cuidados especiales	5%
Pasajero con dieta especial	3%

Tiempo de viaje	Porcentaje
Temporada alta	10%
Temporada baja	-10%

Las personas que tengan entre 0 a 2 años no pagan pasaje. Las personas con más de 2 años deben pagar un precio base de S/. 2000. Además, existen características especiales que podrían incrementar el precio base:

- Niños (mayores de 2 hasta los 10 años) sin compañía adulta deben pagar 10% más del precio base del pasaje.
- Adultos (de más de 10 años) con cuidados especiales debe pagar 5% más del precio base.
- Pasajeros con dieta especial deben pagar 3% más del precio base.

Asimismo, en la temporada alta el precio del pasaje se incrementa en un 10% y, en la baja, disminuye en un 10%. Adicionalmente, se debe tener en cuenta que la temporada alta incluye los meses de enero, febrero, julio y agosto; el resto de los meses están considerados dentro de la baja.

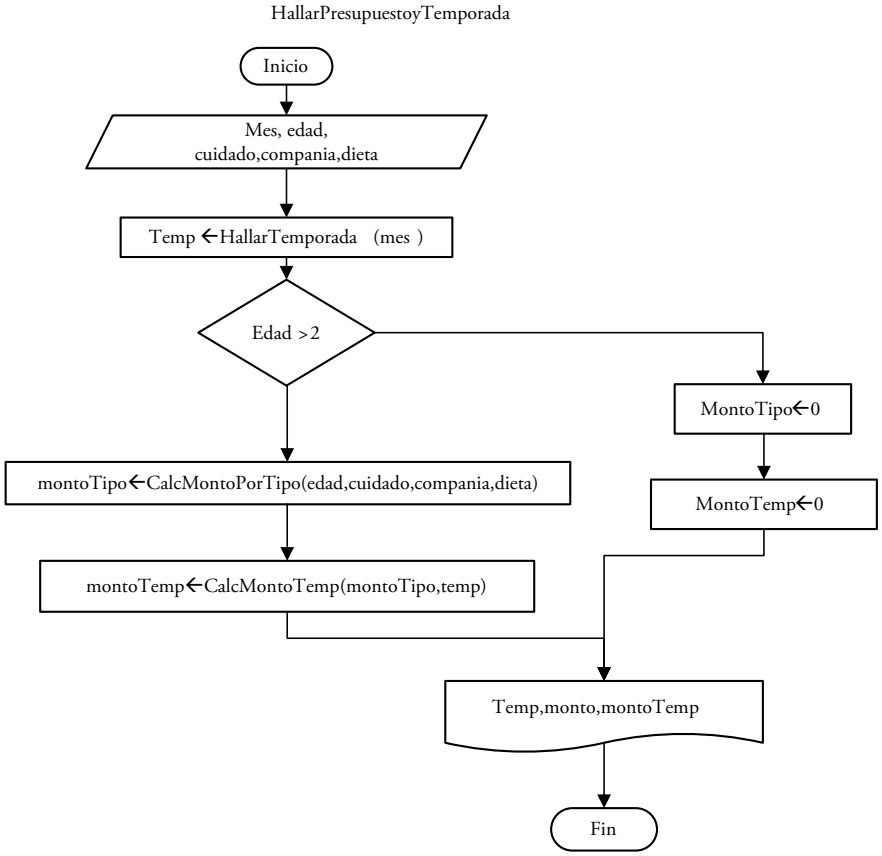
Por tanto, debemos hallar y mostrar el tipo de temporada de viaje (alta o baja); el precio, luego de aplicado el tipo de pasajero, y el precio final, después de aplicado el tipo de temporada, a través de un programa en VBA cuya hoja de cálculo es:

Ilustración 4.12 Ejercicio desarrollado: «Costo de viaje». Hoja de cálculo «Presupuesto de viaje por temporada»

	A	B	C	D
1	**Mes de viaje**	**Agosto**		
2	Edad de pasajero	Niño sin compañía	Adulto con cuidados especiales	Dieta especial
3	45	no	no	sí
4				
5				
6	Temporada de viaje	Alta		
7	Monto por pagar aplicando tipo de pasajero	2,060.00		
8	Monto por pagar aplicando tiempo de viaje	2,266.00		

4.3.2.1. Diagrama de flujo

Ilustración 4.13 Ejercicio desarrollado: «Costo de viaje». Diagrama de flujo del programa principal



Se leen los datos de entrada, se halla la temporada, luego, se evalúa la edad y, si la edad es menor o igual a 2, entonces los montos serán 0. De otra parte, si la edad es mayor a 2, entonces se calcula el monto por tipo y temporada. Finalmente, en cualquiera de los dos casos, se deben escribir los montos y la temporada.

Ilustración 4.14 Ejercicio desarrollado: «Costo de viaje». Diagrama de flujo de la función «HallarTemporada»

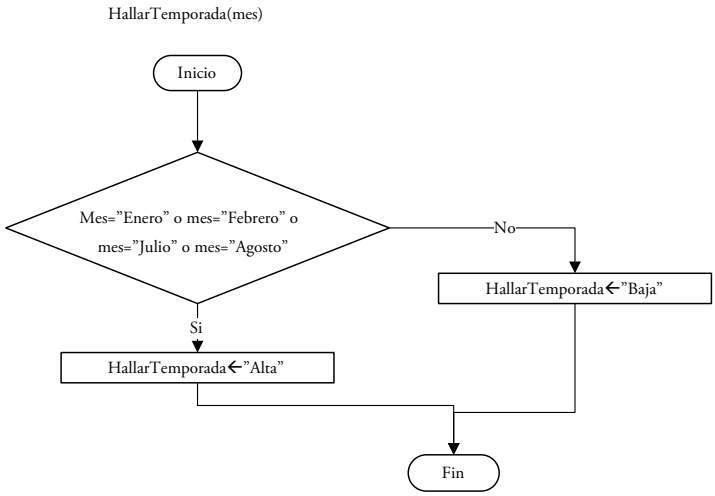


Ilustración 4.15 Ejercicio desarrollado: «Costo de viaje». Diagrama de flujo de la función «CalcMontoTemp»

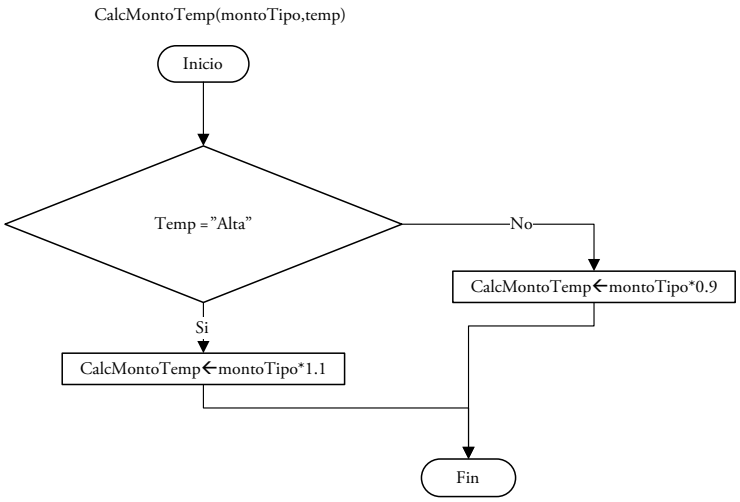
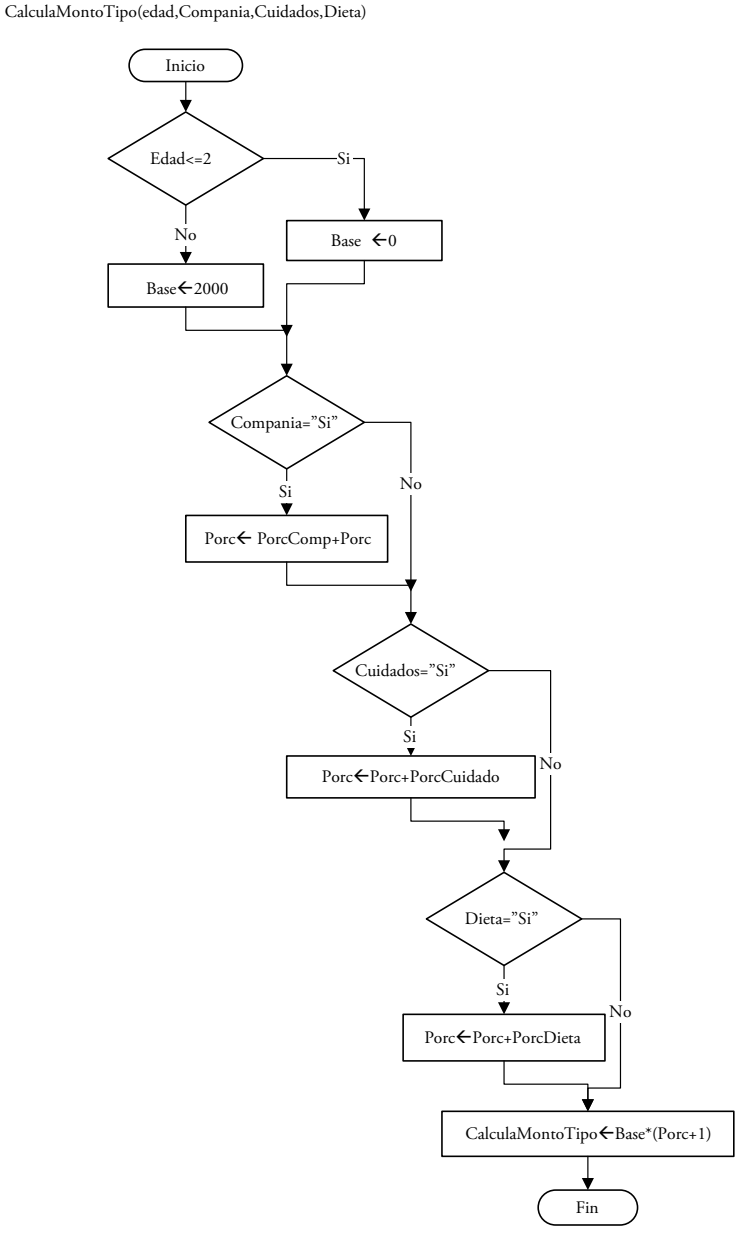


Ilustración 4.16 Ejercicio desarrollado: «Costo de viaje». Diagrama de flujo de la función «CalculaMontoTipo»



4.3.2.2. Pseudocódigo

```
Inicio PP
    Leer Edad,Compania, cuidados, dieta, mes
    montoTipo ← CalculaMontoTipo(edad,compania,cuidados,dieta)
    Temporada ← HallaTemporada(mes)
    MontoTiempo ← CalculaMontoTiempo(Temporada,montoTipo)
    Escribir  Temporada,montoTipo,MontoTiempo
Fin PP

Inicio CalculaMontoTipo(edad,compania,cuidados,dieta)
    si Edad > 2 entonces
        base ← 2000
    sino
        base ← 0
    fin si
    si Compania = "si" entonces
        porc ← PorcComp+porc
    fin si
    si cuidados = "si" entonces
        porc ← PorcCuidado+porc
    fin si
    si dieta = "si" entonces
        porc ← PorcDieta+porc
    fin si
    CalculaMontoTipo ← base*(porc+1)
Fin CalculaMontoTipo

Inicio HallaTemporada(mes)
  si mes = "Enero" O  mes = "Febrero" O  mes = "Julio" O mes = "Agosto" entonces
    HallaTemporada ← "Alta"
  sino
    HallaTemporada ← "Baja"
  Fin si
Fin  HallaTemporada
```

```
Inicio CalculaMontoTiempo(temporada,monto)
  si Temporada = "Alta" entonces
      CalculaMontoTiempo ←monto*(1+0.1)
  sino
      CalculaMontoTiempo ←monto*(1-0.1)
  fin si
Fin CalculaMontoTiempo
```

4.4. Codificación de estructuras selectivas

Ahora bien, esta es otra manera de programar las estructuras selectivas:

```
Select case <variable>
    Case <valor>: instrucciones
    Case <valor>: instrucciones
    .....
    Case else: instrucciones
End Select
```

Estas instrucciones permiten evaluar una única variable y distintos casos para esta. Así, las opciones para el caso se unen con comas y pueden incluir instrucciones como:

Case 1 to 10, is > 20, is >=30,40

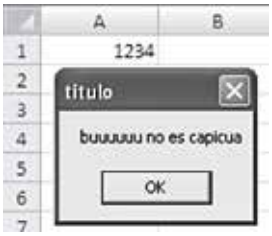
Se debe resaltar que solo una de las opciones entre comas se podrá cumplir, pues no hay manera de verificar que se cumplan todas las opciones del caso. No obstante, para entender mejor el manejo de esta estructura veremos algunos ejemplos.

4.5. Ejemplos de codificación

4.5.1. Es capicúa

Un número se considera capicúa si cuando se lee al derecho y al revés es el mismo número. Por ejemplo, 1991 es un número capicúa; en cambio 1242 no lo es.

Ilustración 4.17 Ejemplo de codificación: «Es capicúa»



Se tiene un número en la celda A1 y se quiere implementar un subprograma en VBA que muestre un mensaje que verifique si dicho número es o no capicúa, pero para hacer este análisis debe verificar primero que se trate de un número de cuatro cifras. Así, el procedimiento principal leerá el número original y, luego, verificará si es un número válido (de cuatro cifras). Después, si es válido, se evalúa si es capicúa por medio de una función: si es, se muestra el mensaje correspondiente, y lo mismo sucede si el número no es capicúa. Finalmente, si el número no es válido se muestra el mensaje respectivo:

```
Sub PP()
    numero = Range("A1")
    Esde4digitos = Verifica(numero)
    If Esde4digitos = True Then
        es = EsCapicua(numero)
        If es = "sí es capicúa" Then
            Call MsgBox("felicitaciones sí es capicúa")
        Else
            Call MsgBox("buuuuuu no es capicúa", 0, "título")
        End If
        Else
            Call MsgBox("el número no es de 4 dígitos")
    End If
End Sub
```

Seguidamente, la función «Verifica», devolverá «VERDADERO» si el número es de cuatro cifras y «FALSO» en caso contrario:

```
Function Verifica(ByVal numero As Integer) As Boolean
    If (numero > 999 And numero < 10000) Then
        Verifica = True
    Else
        Verifica = False
    End If
End Function
```

Luego, la función «EsCapicua» verificará si el número es capicúa y devolverá el mensaje correspondiente:

```
Function EsCapicua(ByVal numero As Integer) As String
    Dim diga As Byte, digb As Byte, digc As Byte, digd As Byte
    diga = numero \ 1000
    digb = (numero \ 100) Mod 10
    digc = (numero Mod 100) \ 10
```

```
    digd = numero Mod 10
    If diga = digd And digb = digc Then
        EsCapicua = "sí es capicúa"
    Else
        EsCapicua = "no es capicúa"
    End If
End Function
```

4.5.2. Tarjeta de crédito

Un cliente de un banco desea obtener una tarjeta de crédito, pero antes de otorgársela, el encargado debe hacer un análisis crediticio. Para ello, le da puntos al cliente a partir de las siguientes tablas:

Ilustración 4.18 Ejemplo de codificación «Tarjeta de crédito»: puntajes

Sueldo mínimo	Tipo de cliente	Puntos
700.00	Clásico	1
1,300.00	Clásico	2
2,500.00	Plata	3
5,000.00	Oro	4

Depósitos	Puntos
Hasta 3,000.00	3
Hasta 5,000.00	4
Hasta 7,000.00	5
Más de 7,000.00	6

Por ejemplo, si el cliente tiene un sueldo mínimo menor a S/. 700, entonces se le asigna un punto y «clásico» como tipo de cliente. Ahora bien, si tiene depósitos de hasta S/. 3,000, entonces se le dan tres puntos. Finalmente, estos puntos concedidos se suman y se verifica el siguiente cuadro:

Ilustración 4.19 Ejemplo de codificación «Tarjeta de crédito»: asignación de tarjeta

Tarjetas de crédito	Tipo de cliente	Puntaje
Clásica	Clásico	Máximo 5 puntos
VIP	Clásico	Más de 5 puntos
VIP	Plata	Máximo 8 puntos
Super VIP	Plata	Más de 8 puntos
Super VIP	Oro	Mínimo 5 puntos

De acuerdo con el ejemplo anterior, el cliente ya tendría cuatro puntos y, como se le asignó tipo «clásico», le correspondería una tarjeta clásica:

Ilustración 4.20 Ejemplo de codificación «Tarjeta de crédito»

	A	B	C	D
1	**Cliente**	**Sueldo**	**Depósito**	**Tipo de Tarjeta**
2	Juan Pérez	670.00	2,500.00	Clásica

```
'Las constantes se definen de las tablas dadas
Const Nivel1 = 700,Lim5000=5000, Nivel2 = 1300, Lim3000=3000
Const Nivel3 = 2500, Lim7000=7000
'Se define el programa principal, se leen los datos de entrada, se hallan los puntos
y el tipo de 'cliente, según el sueldo mínimo, mediante un procedimiento que
devolverá estos dos 'valores. Para calcular los puntos por el depósito se crea una
función y, finalmente, se crea una 'función para hallar el tipo de tarjeta.

Sub HallaTipoTarjeta()
    Dim sueldo As Integer, deposito As Integer, puntosbd As Byte,_
    puntosTot As Byte, tc As String, puntos As Byte, TipoTarjeta As String
    sueldo = Range("B2")
    deposito = Range("C2")
    Call calctcYp(sueldo, tc, puntos)
    puntosbd = HallaPuntos(deposito)
    puntosTot = puntosbd + puntos
    TipoTarjeta = hallaTT(puntosTot, tc)
    Range("D2") = TipoTarjeta
End Sub

'Este procedimiento devolverá dos valores, los cuales tienen modificador byref
Sub calctcYp(ByVal sueldo As Integer, ByRef tc As String, ByRef puntos As Byte)
    If sueldo <= Nivel1 Then
        puntos = 1
        tc = "clásico"
    ElseIf sueldo <= Nivel2 Then
        puntos = 2
        tc = "clásico"
    ElseIf sueldo <= Nivel3 Then
```

```
            puntos = 3
            tc = "plata"
        Else
            puntos = 4
            tc = "oro"
    End If
End Sub

'Esta función devolverá los puntos correspondientes al monto del depósito
Function HallaPuntos(ByVal deposito As Integer) As Byte
        If deposito <= Lim3000 Then
            HallaPuntos = 3
        ElseIf deposito <= Lim5000 Then
            HallaPuntos = 4
        ElseIf deposito <=Lim 7000 Then
            HallaPuntos = 5
        Else
            HallaPuntos = 6
    End If
End Function

'Esta función hallará el tipo de tarjeta correspondiente
Function hallaTT(ByVal puntosTot As Byte, ByVal tc As String) As String
        If tc = "clásico" And puntosTot <=5 Then
            hallaTT = "clásica"
        ElseIf tc = "clásico" And puntosTot > 5 Then
            hallaTT = "VIP"
        ElseIf tc = "plata" And puntosTot <= 8 Then
            hallaTT = "VIP"
        ElseIf tc = "plata" And puntosTot > 8 Then
            hallaTT = "SuperVip"
        ElseIf tc = "oro" And puntosTot >= 5 Then
            hallaTT = "SuperVip"
        End If
End Function
```

En Los tres subprogramas anteriores hemos usado estructuras selectivas. Como podemos ver, lo único que necesitamos es controlar adecuadamente el manejo de los rangos y los resultados correspondientes para cada uno de ellos.

4.5.3. Juego de dados

En este juego cada participante tienen que lanzar dos dados: si los dos son iguales, entonces el dado 1 se multiplica por 10 y se le suma 100; si el dado 1 es menor al dado 2, se multiplica el dado 2 por 10 y se le suma el dado 1; de ser mayor el dado 1, se le multiplica por 10 y se le suma el dado 2. Finalmente, gana el jugador con mayor puntaje.

Ilustración 4.21 Ejemplo de codificación: «Juego de dados»

	A	B
1	**Datos de entrada**	
2	Jugador1 dado1	5
3	Jugador1 dado2	1
4	Jugador2 dado1	4
5	Jugador2 dado2	4
6		
7	**Salidas**	
8	Puntaje jugador1	51
9	Puntaje jugador2	140
10	Ganador	Jugador2

```
'Se leerán los datos por medio de un procedimiento, luego, se validarán los dados,
y de ser estos 'válidos, se calcularán los puntajes para, finalmente, hacer el análisis
correspondiente y mostrar al 'ganador.
Sub PP()
  Dim j1d1 As Byte, j1d2 As Byte
  Dim  j2d1 As Byte
  Dim  j2d2 As Byte, puntaje1 As Byte
  Dim  puntaje2 As Byte
  Dim  ganador As String
     Call LeerDatos(j1d1,j1d2,j2d1,j2d2)
     esValida = Validar(j1d1,j1d2,j2d1,j2d2)
     If (esValida = True) Then
     Call CalcularPuntaje(j1d1,j1d2,j2d1,j2d2,puntaje1,puntaje2)
     If (puntaje1 > puntaje2) Then
          ganador = "Jugador 1"
     ElseIf (puntaje1 = puntaje2) Then
          ganador = "empate"
```

```
  Else
     ganador = "Jugador 2"
   End If
  Call Mostrar(puntaje1,puntaje2,ganador)
  Else
  Call MsgBox ("ERROR")
  End If
End Sub

Sub LeerDatos(ByRef j1d1 As Byte, ByRef j1d2 As Byte, ByRef j2d1 As Byte,_
ByRef j2d2 As Byte)
     j1d1 = Range("B2")
     j1d2 = Range("B3")
     j2d1 = Range("B4")
     j2d2 = Range("B5")
End Sub

'La función Validar evalúa que los cuatro dados tengan el valor adecuado, entre
1 y 6
Function Validar(ByVal j1dado1 As Byte, ByVal j1dado2 As Byte,ByVal_
j2dado1 As Byte, ByVal j2dado2 As Byte) As Boolean
  Dim valido As Boolean
  If (j1dado1 >= 1 And j1dado1 <= 6) And_
     (j1dado2 >= 1 And j1dado2 <= 6) And_
     (j2dado1 >= 1 And j2dado1 <= 6) And_
     (j2dado2 >= 1 And j2dado2 <= 6) Then
     valido = True
  Else
     valido = False
  End If
  Validar = valido
End Function

'El procedimiento calcular puntaje utiliza una función general para 'calcular el
puntaje de cada uno de los jugadores y luego devuelve 'dichos puntajes

Sub CalcularPuntaje(ByVal j1d1 As Byte, ByVal j1d2 As Byte, ByVal j2d1 As Byte,
  ByVal j2d2 As Byte, ByRef p1 As Byte, ByRef p2 As Byte)
     p1 = CalculaPuntJugador(j1d1,j1d2)
     p2 = CalculaPuntJugador(j2d1,j2d2)
End Sub
```

```
Function CalculaPuntJugador(ByVal dado1 As Byte, ByVal dado2 As Byte) As Byte
  If (dado1 = dado2) Then
      CalculaPuntJugador = dado1*10+100
  ElseIf (dado1 < dado2) Then
      CalculaPuntJugador = dado2*10+dado1
  Else
      CalculaPuntJugador = dado1*10+dado2
  End If
End Function

Sub Mostrar(ByVal p1 As Byte, ByVal p2 As Byte, ByVal ganador As String)
      Range("B8") = p1
      Range("B9") = p2
      Range("B10") = ganador
End Sub
```

4.5.4. Peso ideal

Para mantenerse en un peso ideal se debe consumir una dieta conformada por elementos de cada grupo de alimentos pero sin exceder las 1500 calorías al día.

A continuación, se muestran las posibles combinaciones de elementos para algunas de las distintas comidas diarias:

Ilustración 4.22 Ejemplo de codificación «Peso ideal»: combinaciones

Desayuno	**Merienda**	**Media mañana**
Lácteos	**Lácteos**	**Proteínas**
Hidratos de carbono	Hidratos de carbono	Hidratos de carbono
Frutas	-	-

En el cuadro anterior se dan distintas raciones de grupos alimenticios, pero, en el siguiente, se muestran los alimentos que podrían considerarse una ración de cada uno de los grupos mencionados:

Ilustración 4.23 Ejemplo de codificación «Peso ideal»: raciones

Grupo	**Raciones**			
Hidratos de carbono	80 g de pan	200 g de papa	120 g de pasta	120 g de arroz
Proteínas	100 g de carne	150 g de pescado		
Lácteos	1 taza de leche	2 yogures		
Frutas	150 g de melón	150 g fresas	50 g de higos	

Luego, y con los datos mostrados, se quiere determinar si los elementos seleccionados son los adecuados para las comidas de este cuadro:

Ilustración 4.24 Ejemplo de codificación «Peso ideal»

	A	B	C
1		**Desayuno**	**Merienda**
2		1 taza de leche	3 yogures
3		80 g de pan	300 g de papa
4		150 g fresas	
5	Elementos adecuados	VERDADERO	FALSO

Para ello se pide implementar en VBA:

a) Un subprograma «RacionAdecuada» que reciba por parámetros a las tres raciones de un tipo de comida y el tipo de comida, y que verifique si se trata de las raciones adecuadas. Para que la ración sea la conveniente, es necesario que cada uno de los tres elementos esté presente en el tipo de comida.

```
Function RacionAdecuada(ByVal TipoComida As String, ByVal Elem1 As_
String, ByVal Elem2 As String, ByVal Elem3 As String) As Boolean
    Dim Es1 As Boolean, Es2 As Boolean, Es3 As Boolean
    Es1 = EstaPresenteEn(TipoComida, Elem1)
    Es2 = EstaPresenteEn(TipoComida, Elem2)
    Es3 = EstaPresenteEn(TipoComida, Elem3)
    If ((Elem1 <> "-" And Es1) Or Elem1 = "-") And_
        ((Elem2 <>"-" And Es2) Or Elem2 = "-") And_
        ((Elem3 <>"-" And Es3) Or Elem3 = "-") Then
        RacionAdecuada = True
    Else
        RacionAdecuada = False
    End If
End Function
```

b) Un subprograma «EstaPresenteEn» que reciba por parámetros al tipo de comida y a una ración, y verifique si el elemento es ración de alguno de los tipos de comida.

```
Function EstaPresenteEn(ByVal TCom As String, ByVal Elem As String) As Boolean
    Dim TRac As String
    TRac = EsRacionDe(Elem)
    If TCom = TipoCom1 And (Rac = TipoRac1 Or TRac = TipoRac2_
    Or TRac = TipoRac3) Then
    EstaPresenteEn = True
    End If
    If TCom = TipoCom2 And (TRac = TipoRac1 Or TRac = TipoRac3) Then
    EstaPresenteEn = True
    End If
    If TCom = TipoCom3 And (TRac = TipoRac4 Or TRac = TipoRac3) Then
    EstaPresenteEn = True
  End If
End Function
```

c) Un subprograma «EsRacionDe» que reciba por parámetros a un elemento y que devuelva el grupo al que este pertenece.

```
Const EHid1 = "80 g de pan"
Const EHid2 = "200 g de papas"
Const EHid3 = "120 g de pasta"
Const EHid4 = "120 g de arroz"
Const EProt1 = "100 g de carne"
Const EProt2 = "150 g de pescado"
Const ELact1 = "1 taza de leche"
Const ELact2 = "2 yogures"
Const EFrut1 = "150 g de melón"
Const EFrut2 = "150 g de fresas"
Const EFrut3 = "50 g de higos"
Const TipoRac1 = "Hidrato de carbono"
Const TipoRac2 = "Frutas"
Const TipoRac3 = "Lácteos"
Const TipoCom1 = "Desayuno"
Const TipoCom2 = "Merienda"
Const TipoCom3 = "Media Mañana"
Const TipoRac4 = "Proteínas"
```

```
Function EsRacionDe(ByVal Elemento As String) As String
    Select Case Elemento:
    Case EHid1, EHid2, EHid3, EHid4: EsRacionDe = "Hidrato de carbono"
    Case EProt1, EProt2: EsRacionDe = "Proteína"
    Case ELact2, ELact1: EsRacionDe = "Lácteos"
    Case EFrut1, EFrut2, EFrut3: EsRacionDe = "Frutas"
    End Select
End Function
```

d) Un subprograma principal que lea las seis raciones y los dos tipos de comida, evalúe si la ración es adecuada o no mediante los subprogramas anteriores que crea convenientes y que, finalmente, escriba las respuestas.

```
Sub PP()
    Dim TipoComida1 As String, Elem1 As String, Elem2 As String
    Dim  Elem3 As String, Elem6 As String
    Dim TipoComida2 As String, Elem4 As String, Elem5 As String
    TipoComida1 = Range("B1")
    Elem1 = Range("B2")
    Elem2 = Range("B3")
    Elem3 = Range("B4")
    TipoComida2 = Range("C1")
    Elem4 = Range("C2")
    Elem5 = Range("C3")
    Elem6 = Range("C4")
    Rpta1 = RacionAdecuada (TipoComida1, Elem1, Elem2, Elem3)
    Rpta2 = RacionAdecuada (TipoComida2, Elem4, Elem5, Elem6)
    Range("B5") = Rpta1
    Range("B5") = Rpta2
End Sub
```

4.6. Ejercicios propuestos

4.6.1. Tutifruti

Ilustración 4.25 Ejercicio propuesto: «Tutifruti»

	A	B	C	D	E	F	G
1	**Tutifruti**						
2	**Letra**	**Cosa**	**Nombre**	**Lugar**	**Electrodoméstico**	**Animal**	**Puntos**
3	A	Auto	Ana	Austria	Aspiradora	Anaconda	5
4	P	Palo	Pablo	Perú		Gato	3

Implementar un subprograma que calcule el número de puntos obtenidos al jugar con una letra.

Los datos de entrada son la letra y la palabra correspondiente a cada una de las categorías. El subprograma debe verificar si la palabra se inicia con la letra adecuada o no; por cada palabra que se inicie con la letra apropiada se gana un punto adicional.

4.6.2. Estreno de película

Es el estreno de una película y un grupo de amigos desea ir al cine, pero para ello hacen un presupuesto. Se sabe que el costo de las entradas es de S/. 9.00, que irán en un auto, que la hora en el estacionamiento está a S/. 3.50 y que el costo del combo para dos personas es de S/. 15.00.

Ilustración 4.26 Ejercicio propuesto: «Estreno de película»

	A	B
1	**Datos de entrada**	
2	Horas de duración	3
3	Día de la semana	martes
4	Número de entradas	4
5		
6	**Subtotales por conceptos**	
7	Combo	30.00
8	Entradas	36.00
9	Estacionamiento	10.50
10		
11	**Salidas**	
12	Monto total	76.50
13	Monto por persona	19.13

Ahora bien, existen condiciones sobre el problema: en el auto solo entran cinco personas, así que este será el número máximo posible de amigos; además, se sabe que en el cine solo venden combos para dos personas, que todos los amigos tienen que comer y que todos compartirán el pago del estacionamiento.

Se le pide que implemente en VBA los subprogramas necesarios para obtener los subtotales por conceptos y las salidas mostradas.

4.6.3. Monto total con descuento

En las tiendas ABC se aplica un descuento sobre el monto total de cada compra, según el tipo de tarjeta que utilice el cliente.

Ilustración 4.27 Ejercicio propuesto «Monto total con descuento»: porcentajes de descuento

Tipo de tarjeta	Porcentaje de descuento
Visa	15%
Mastercard	10%
Ninguna	0

Elabore un subprograma que lea el monto total y el tipo de tarjeta para, posteriormente, calcular y mostrar el monto con descuento.

Ilustración 4.28 Ejercicio propuesto «Monto total con descuento»:

	A	B
1	Monto total	1,503.32
2	Posee tarjeta	Visa
3	Monto con descuento	1,277.82

4.6.4. Palabra con cinco vocales

Ilustración 4.29 Ejercicio propuesto: «Palabra con cinco vocales»

	A	B
1	Palabra	Incluye las 5 vocales
2	Murciélago	Sí
3	Australia	No

Se le pide desarrollar en VBA un subprograma que identifique si una palabra incluye las cinco vocales o no.

4.6.5. Horarios de universidad

Se acabó el año escolar y empezó la preparación para el ingreso a la universidad. *Mi Universidad,* una institución de prestigio, ofrece en su página web una pequeña aplicación que le permite al usuario saber si de dar el examen ingresaría o no.

Usted, como alumno del curso Introducción a la Computación, es contratado para hacer dicha aplicación.

Para ello, la universidad define las siguientes condiciones:

- El alumno debe de haber obtenido al menos el 30% del puntaje en cada una de las tres secciones.
- Debe de haber alcanzado un puntaje total mayor igual a 1300, si va a EEGGLL; mayor igual a 1800, si va a EEGGCC, y de 700 como mínimo, si va a Artes.
- Se sabe que el examen tiene tres secciones, que la sección de matemática tiene un total de 45 preguntas; la de lectura, un total de 35, y que la de redacción solo tiene un puntaje asignado que debe de estar entre 0 a 150.
- Por cada tres respuestas equivocadas de matemática se restan 50 puntos y cada respuesta correcta vale 50 puntos.
- Por cada tres respuestas equivocadas de lectura se restan 30 puntos y cada respuesta correcta vale 30 puntos.

Además, la aplicación debe permitir al usuario identificar en qué horario se matricularía de acuerdo con el puntaje obtenido:

Tabla 4.1 Ejercicio propuesto «Horarios de universidad»: porcentajes

112	Desde el mínimo puntaje de ingreso hasta un incremento del 15% de este para ingresar.
109	Desde un incremento del 15% hasta el 30% del puntaje mínimo para ingresar.
107	Desde un incremento del 30% hasta el 50% del puntaje mínimo para ingresar.
103	Desde un incremento del 50% hasta el 71% del puntaje mínimo para ingresar.
101	Desde un incremento del 71% hasta el 92% del puntaje mínimo para ingresar.

Se le pide implementar una aplicación con la que se obtenga la siguiente vista de Excel, pero solo para un postulante:

Ilustración 4.30 Ejercicio propuesto «Horarios de universidad»: porcentajes

	A	B	C	D	E	F	G	H
1		Respuestas correctas			Unidad	Puntaje total	Porcentaje adicional	Horario
2		Matemática	Lectura	Redacción				
3	Rosa López	25	10	40	EEGGLL	1006.67	Ninguno	No ingresó
4	Juan Pérez	42	31	150	EEGGCC	3090	72.00	H101

4.6.6. El regalo prometido

Debido a que usted ingresó a la universidad, su padre le ofreció un regalo. Usted definitivamente quiere una consola de juegos, pero no sabe cuál. Como necesita evaluar las posibilidades, realiza una pequeña aplicación. Para ello crea la siguiente hoja de cálculo:

Ilustración 4.31 Ejercicio propuesto: «El regalo prometido»

	A	B	C	D	E	F
1		Descripción de precios				
2	Nombre consola	Consola	Juegos	Imagen	Accesorios	Puntaje final
3	Play Station	Muy caro	Muy caro	5	3	10
4	Wii	Barato	Barato	4	4	14
5	xBox	Caro	Barato	5	3	13

Luego, para evaluar las consolas decide tener en cuenta los precios de estas y de los juegos de cada una, además de la calidad de imagen y la diversidad de accesorios que podría adquirir para dicha consola. De este modo, usted le asigna puntajes parciales a cada una para, finalmente, obtener los puntajes finales y seleccionar la que obtenga mayor puntaje:

Ilustración 4.32 Ejercicio propuesto «El regalo prometido»: puntajes

Descripción de precio	Puntaje
Muy caro	1
Caro	2
Barato	3

4.6.7. Préstamo

Ilustración 4.33 Ejercicio propuesto: «Préstamo»

Estado civil		Nro hijos		Se le puede dar un préstamo	
Casado (C)	No	Menos de 2	Sí	Si tiene más del 40% de respuestas positivas	Sí
Soltero (S) Viudo(V)	Sí	2 o más	No	Si tiene 40% o menos de respuestas positivas	No

	A	B	C	D	E	F	G
1		Mayor de edad	Estado civil	Nro hijos	Casa propia	Gana más de 3000	Se le puede dar el préstamo
2	Cliente	Sí	S	0	Sí	No	Sí

- Implemente un subprograma principal que a partir de la llamada a los subprogramas siguientes calcule y muestre el valor de la columna G.
- Desarrolle un subprograma «EstadoCivil» que tenga como parámetro el estado civil y que devuelva un «sí» o un «no» a partir de los valores mostrados en la tabla anterior.
- Elabore un subprograma «Nrohijos» que tenga como parámetro el número de hijos y que devuelva un «sí» o un «no» sobre la base de los valores mostrados en la tabla anterior.
- Realice un subprograma «Sumarse» que reciba los cinco datos de entrada y que devuelva la cantidad de «sí» que posee dicho cliente. Debe llamar a las funciones anteriores para calcular si el estado y el número de hijos dan «sí» o «no».

4.6.8. Seguro de auto

El importe del seguro de un automóvil depende de la categoría, el color, el año de fabricación y de si este es parte de la flota de una empresa. Las categorías de automóvil y los precios del seguro (según el color) se describen en la tabla siguiente:

Ilustración 4.34 Ejercicio propuesto: «Seguro de auto»: montos según color y categoría

Categoría	Color	Monto anual
SEDÁN	Rojo, Blanco,Negro	2100
	Otro	2000
HATCHBACK	Rojo, Blanco,Negro	2600
	Otro	2550
M1	Rojo, Blanco,Negro	3200
	Otro	3100

Además, si el año de fabricación es 2012, 2011 o 2010, el monto anual sufre un incremento del 5%; si es de algún otro año, de 2000 en adelante, tiene un incremento del 7%, y si es cualquier otro año, el incremento es del 15%.

Como en algunos casos los autos se registran en la flota de la empresa, estos tienen un descuento por precio al por mayor, con el que se puede lograr hasta un 30% de descuento del monto anual.

Ilustración 4.35 Ejercicio propuesto: «Seguro de auto»

	A	B
1	**Datos**	
2	Categoría	Sedán
3	Color	Plomo
4	Año de fabricación	2009
5	Parte de una flota	Sí
6	**Importe por pagar**	
7	Anual	**1498**
8	Mensual	124.83

A partir de la información brindada se le pide:

- Plantear la definición y el análisis del problema con al menos cuatro módulos (incluido el principal).
- Diseñar en pseudocódigo la solución del problema. El diseño debe coincidir con el diagrama de módulos planteado.

ESTRUCTURAS ALGORÍTMICAS ITERATIVAS

Existen ciertos casos en los que necesitamos repetir algunas instrucciones de código y, por tanto, escribimos varias veces las llamadas necesarias y leemos los distintos datos de entrada. No obstante, a pesar de que es posible proceder de esta manera si se trata de dos o tres repeticiones, esto no es eficiente cuando las repeticiones son 30, 100 o 1000, ya que escribir las llamadas necesarias para tantas sería bastante tedioso, sin mencionar la cantidad de memoria ocupada por las variables por definir. Por consiguiente, en lugar de hacer esto, debemos utilizar las estructuras algorítmicas iterativas. Así, gracias a estas instrucciones, podemos repetir secuencias de instrucciones un número determinado de veces de manera muy sencilla y con poco uso de memoria.

A veces hay subprogramas que necesitan ser invocados muchas veces y que, además, cuentan con valores particulares. De modo que, se deben leer datos, calcular valores y escribirlos, y, para ello, las instrucciones iterativas son ideales.

Seguidamente, debemos señalar que las estructuras iterativas, o también llamadas «instrucciones repetitivas», siempre deben tener tres partes:

- Iniciación
- verificación
- actualización.

Así, en la iniciación se asignan los primeros valores para las diferentes variables que se utilizarán. Luego, en la verificación, se incluye la condición «Fin», la cual, según el caso, detendrá la ejecución del código colocado dentro de la estructura iterativa. Si se maneja mal esta condición, es posible que se cree un bucle infinito, lo que significa que las instrucciones dentro del bloque iterativo se ejecutarán ilimitadamente. Y, finalmente, en la actualización, se confirman los valores que harán que la condición «Fin» se cumpla; por tanto, si no se actualizan las variables correctamente, probablemente se creará un bucle indefinido.

La manera más sencilla de desarrollar este tipo de estructuras es identificar qué instrucciones se deben repetir, para, posteriormente, hallar la manera general de llegar a la respuesta. En algunos casos, evaluar la solución nos permitirá saber si esta es la adecuada.

Incluso, podemos percibir este tipo de estructuras en nuestra vida cotidiana: por ejemplo, desde que comenzamos a vivir nuestra edad se actualiza cada año, y este bloque iterativo termina al morir.

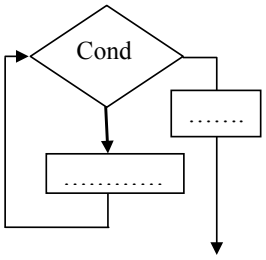
Otro ejemplo de la vida diaria podría ser el hecho de estudiar en la universidad: se inicia el bloque iterativo con una cantidad de créditos, en 0, y en cada ciclo actualizamos dicha cantidad. Finalmente, el bloque termina cuando completamos los créditos necesarios para culminar la carrera universitaria que hemos seleccionado.

Asimismo, existen bloques iterativos hasta en los juegos de rol de las distintas consolas, y uno de los más sencillos es el que maneja las «vidas»: al inicio del juego tenemos 3 o 10 vidas; luego mientras jugamos dentro del bloque iterativo, dicha variable disminuye hasta que nuestras vidas llegan a 0 y el juego termina.

5.1. Diseño de estructuras iterativas

Para este tipo de estructuras utilizaremos las siguientes acciones

Tabla 5.1 Cuadro de acciones: iteración

Acción	Símbolo	Pseudocódigo
Iteración		Hacer mientras(condición) Instrucciones Fin hacer Hacer hasta(condición) Instrucciones Fin hacer

5.2. Ejercicios desarrollados de diseño

5.2.1. Sumar los N primeros números positivos

Se tienen números desde el 1 hasta «N» y se quiere, por medio de una función que recibe el valor de «N» como parámetro, sumarlos. Para ello, la cuenta empezará en 1 y se acumulará dicho valor, luego, se incrementará en uno el contador y estos pasos se repetirán hasta que la cuenta llegue a «N». Aquí debemos recordar que para implementar adecuadamente una estructura iterativa es necesario que entendamos qué se quiere hacer, identifiquemos los pasos que deben repetirse, así como los valores iniciales y las condiciones de fin.

a) Valores que se deben iniciar:
   - Variable «i»: que empezará en 1 y que se incrementará de uno en uno hasta «N».
   - Variable «suma»: que servirá para acumular los valores; se iniciará en 0.

b) Condición de fin:
   - Cuando la variable «i» llegue al valor de «N» la iteración terminará. En cambio, mientras «i» sea menor igual a «N» o hasta que «N» sea mayor la iteración se realizará.

c) Instrucciones que se deben repetir:
   - Se acumula el valor de «i» en la variable «suma».
   - Se actualiza el valor de «i» incrementándolo en uno.

d) De este modo, los pasos que se deben seguir son:
   - Antes de iniciar el bucle, la variable «suma» vale 0 y la variable «i» vale 1. Luego, en la primera iteración, «suma» vale 1 e «i» se incrementa valiendo 2 y, así sucesivamente, hasta que «i» valga 8 y asumamos que «N» vale 7. Es en ese momento que la iteración se detiene ya que «i» es mayor que «N».

Ilustración 5.1 Ejercicio desarrollado: «Sumar los N primeros números positivos»

	A	B	C
1	Iteración	Suma	i
2	**0**	**0**	**1**
3	**1**	**1**	**2**
4	**2**	**3**	**3**
5	**3**	**6**	**4**
6	**4**	**10**	**5**
	5	**15**	**6**
	6	**21**	**7**
	7	28	8

Como ya se ha observado, en el caso que acabamos de mostrar, la iteración se repite mientras que «i» sea menor igual a «N»; en el caso de que no lo sea, la iteración termina.

Aquí es importante señalar que el flujo regresa justo antes de la condición, de tal manera que, luego de ejecutar las instrucciones de la iteración, debemos reevaluar si se llevará a cabo la repetición o no. Como se mencionó anteriormente, si manejamos mal la condición o no actualizamos la variable es posible que generemos un bucle infinito.

- **Paso 1**

Para comenzar debemos recordar que el diagrama de flujo de esta pequeña función tiene como parámetro formal a la variable «N». Luego, iniciamos el diagrama y después colocamos los procesos iterativos, sin olvidar que este bloque debe actualizarse.

- **Paso 2**

En este punto es preciso que tengamos en cuenta que el bloque iterativo se puede desarrollar siempre y cuando se cumpla una condición. Por tanto, al inicio del bloque, incluiremos la condición y, como es necesario reevaluar dicha condición, una vez terminado el bloque, el diagrama quedará como se muestra (ver Ilustración 5.2):

- **Paso 3**

Debemos iniciar todas las variables.

- **Paso 4**

Culminado el bloque iterativo, podemos realizar acciones adicionales, como, por ejemplo, devolver el valor de la función (ver Ilustración 5.2).

Ilustración 5.2 Ejercicio desarrollado
«Sumar los N primeros números positivos»: SumarNPrimeros

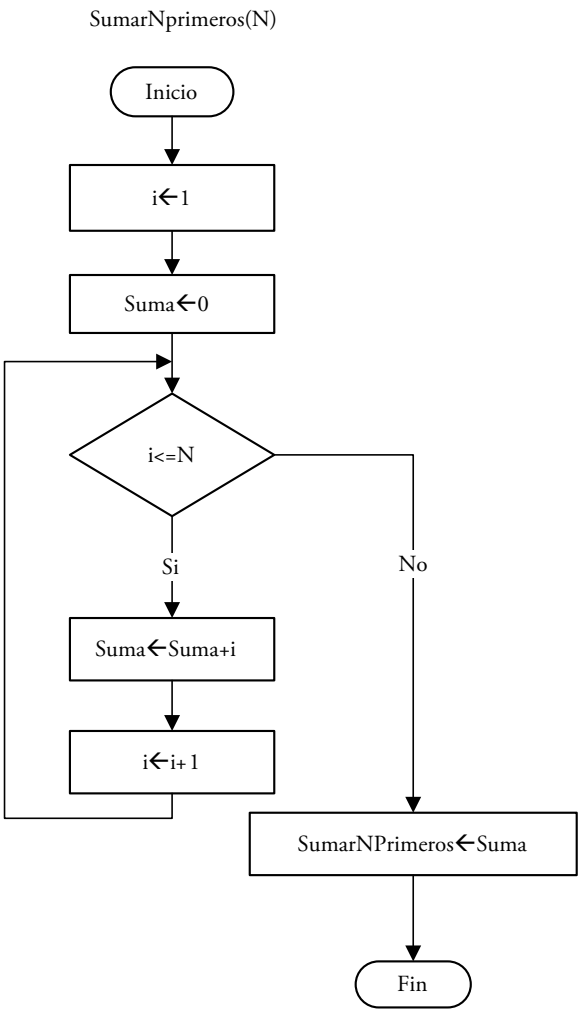
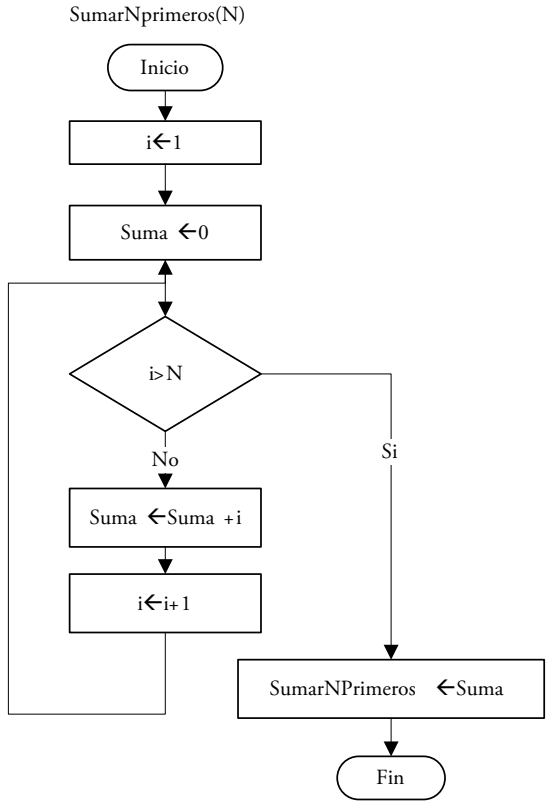


Ilustración 5.3 Ejercicio desarrollado:
«Sumar los N primeros números positivos»: SumarNPrimeros. Paso 2



De otra parte, en el pseudocódigo, las instrucciones serán las siguientes:

```
Inicio SumarNprimeros(N)
    i←1
    Suma←0
    Hacer mientras (i<=N)
        Suma←Suma+i
        i←i+1
    Fin Hacer
    SumarNprimeros←Suma
Fin SumarNprimeros
```

O si se cambia la condición:

```
Inicio SumarNprimeros(N)
    i←1
    Suma←0
```

```
    Hacer hasta que (i>N)
        Suma←Suma+i
        i←i+1
    Fin Hacer
    SumarNprimeros←Suma
Fin SumarNprimeros
```

5.2.2. Calcular el valor de factorial de un número

```
Inicio CalcFactorial(N)
    Fact←1
    Hacer mientras N>=1
        Fact←Fact*N
        N←N-1
    Fin Hacer
    CalcFactorial←Fact
Fin CalcFactorial
```

Ilustración 5.4 Ejercicio desarrollado «Calcular el valor de factorial de un número»: diagrama de flujo «CalcFactorial»

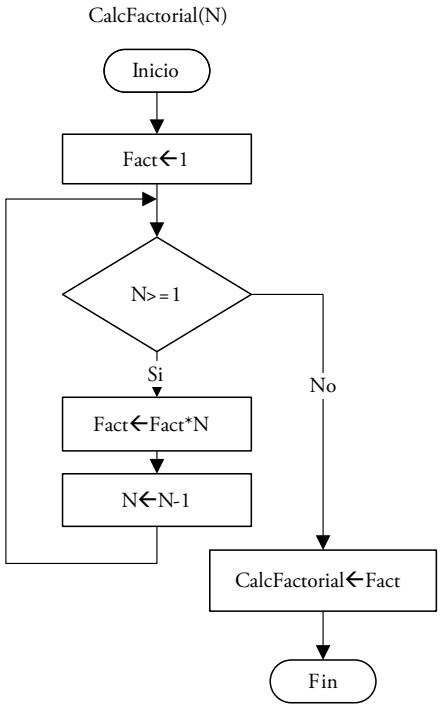


Ilustración 5.5 Ejercicio desarrollado «Calcular el valor de factorial de un número»: diagrama de flujo. Pasos «CalcFactorial»

Fact	N
1	4
4	3
12	2
24	1
24	0

5.2.3. Calcular la suma de los dígitos de un número

```
Inicio CalcSumaDig(N)
    Suma←0
    Hacer hasta que N=0
        Dig←Residuo(N,10)
        Suma←dig+Suma
        N←Cociente(N,10)
    Fin hacer
    CalcSumaDig←Suma
Fin CalcSumaDig
```

5.2.4. Calcular la suma de los dígitos pares de un número

```
Inicio CalcSumaDigPares(N)
    Suma←0
    Hacer hasta que N=0
        Dig←Residuo(N,10)
        Si Residuo(dig,10) =0 entonces
        Suma←dig+Suma
        Fin si
        N←Cociente(N,10)
    Fin hacer
    CalcSumaDigPares←Suma
Fin CalcSumaDigPares
```

5.2.5. Calcular la suma de los dígitos pares y la suma de los dígitos impares

```
Inicio CalcSumaDigParesImpares(N,SumaPar,SumaImpar)
    SumaPar←0
    SumaImpar←0
    Hacer hasta que N=0
        Dig←Residuo(N,10)
        Si Residuo(dig,10)=0 entonces
            SumaPar←dig+SumaPar
        Sino
            SumaImpar←dig+SumaImpar
        Fin si
        N←Cociente(N,10)
    Fin hacer
    CalcSumaDigPares←Suma
Fin CalcSumaDigPares
```

5.3. Ejercicios de diseño

5.3.1. ¿Viajaré a Italia?

María ha ahorrado durante todo un año para viajar por Italia. Por ello, va a una agencia de viajes y pregunta cuánto gastará en los conceptos de hospedaje, comida y transporte en las diferentes ciudades seleccionadas.

Ahora bien, para saber si María realizará o no todas las visitas turísticas planeadas, solo se deben cumplir ciertas condiciones:

- El monto en hospedaje no puede ser mayor al 50% del monto total.
- El monto total debe ser menor igual al presupuestado.
- Se deben cumplir las dos condiciones anteriores para poder viajar; basta que una no se cumpla para que el viaje no se realice.

María, quien ha lleva el curso de Introducción a la Computación, ha creado una pequeña aplicación que le permitirá saber cuánto gastará en cada uno de los conceptos, cual será el presupuesto total y si puede viajar o no. Así, los datos que ingresa son el los siguientes:

El presupuesto y la cantidad de ciudades que quiere visitar. Luego, para cada uno de los destinos, ingresa las cantidades que consumirá en hospedaje, comida y transporte con el fin de obtener una respuesta positiva o negativa.

Por tanto, necesita hacer el diseño del problema anterior, con al menos cuatro módulos.

Al igual que en los ejemplos anteriores, tendremos que de la definición, luego, elaborar el análisis y, finalmente, realizar el diseño.

5.3.1.1. Definición

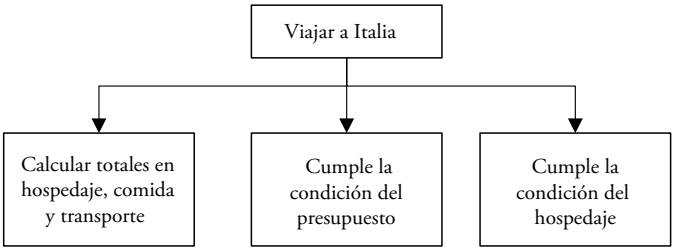
- Explicación: saber si María viajará o no.
- Datos de entrada: número de destinos, presupuesto, consumo en hospedaje, comida y transporte para cada uno de los destinos.
- Salidas: si viajará o no.
- Fórmulas

total =total transporte + total consumo + total hospedaje

viajará=(total<=pres) y (totalhospedaje<=pres*0.5)

5.3.1.2. Análisis

Ilustración 5.6 Ejercicio de diseño «¿Viajaré a Italia?»: diagrama de módulos



- Hemos propuesto un diagrama de módulos: el principal y tres más. En el primero de los módulos del segundo nivel, y por medio de un procedimiento que devolverá tres valores, calcularemos el monto total en hospedaje, comida y transporte. En este procedimiento leeremos cada uno de los conceptos para acumularlos en tres variables diferentes.
- Luego, en el segundo módulo, y por medio de una función, verificaremos si se cumple o no la condición de que el total en hospedaje sea menor al 50% del presupuesto.
- Finalmente, en el tercer módulo, y también por medio de una función, comprobaremos si el total es menor o igual al presupuesto.

5.3.1.3. Diseño

Ilustración 5.7 Ejercicio de diseño «¿Viajaré a Italia?»: diagrama de flujo «PuedeViajarHosp»

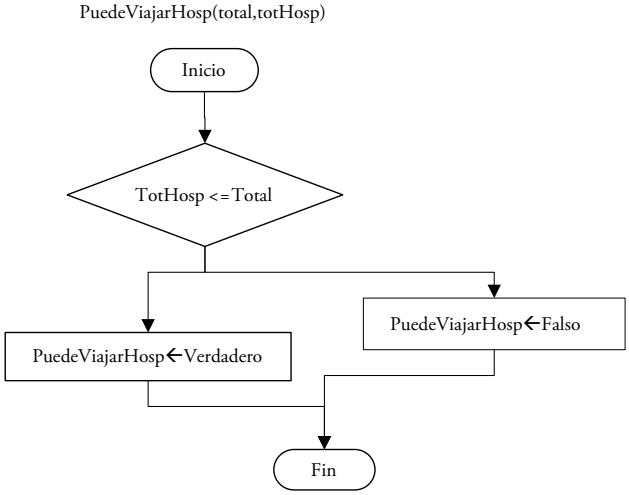


Ilustración 5.8 Ejercicio de diseño «¿Viajaré a Italia?»: diagrama de flujo del programa principal

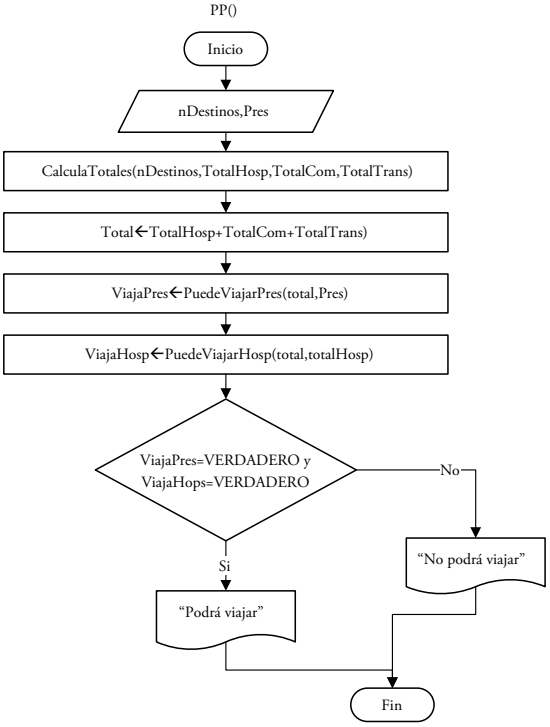


Ilustración 5.9 Ejercicio de diseño «¿Viajaré a Italia?»: diagrama de flujo «PuedeViajarPres»

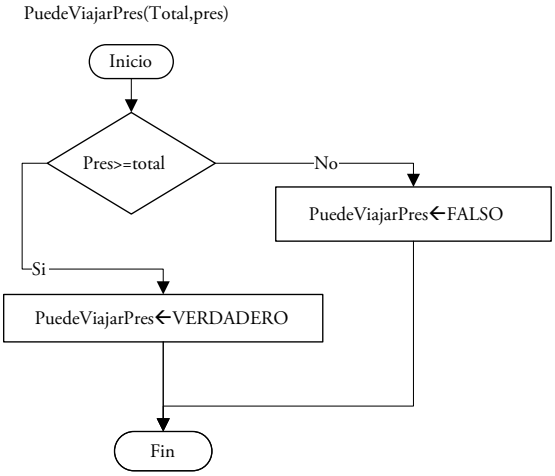
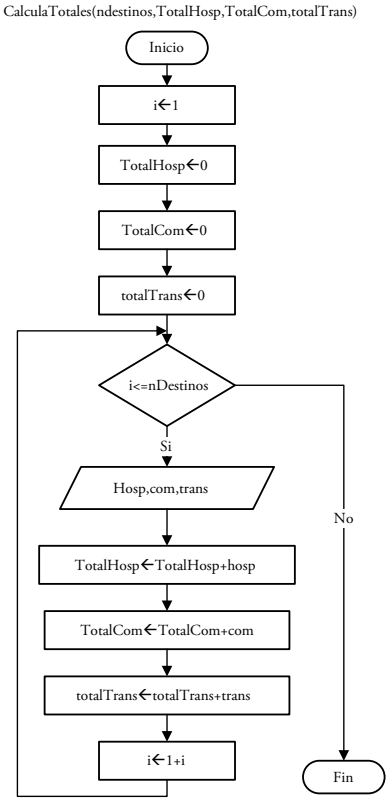


Ilustración 5.10 Ejercicio de diseño «¿Viajaré a Italia?»: diagrama de flujo «CalculaTotales»



5.3.2. Vamos a Italia

Se desea calcular el presupuesto para los fines de semana de un viaje a Italia. Para ello, se conocen los destinos que se visitarán, el lugar de origen y se sabe que siempre se debe regresar al lugar de origen los cinco días de una semana para luego pasar el fin de semana en el destino seleccionado. Se conoce, además, cuál es el gasto promedio en hospedaje y comida de los destinos, así como el costo del viaje.

Ilustración 5.11 Ejercicio de diseño «Vamos a Italia»: pseudocódigo

	A	B	C	D	E
1	**Cantidad de destinos**	4	**Presupuesto de viaje a Italia**		
2	Origen	Destino	Hospedaje	Comida	Costo viaje
3	Padua	Venecia	140	70	40
4	Padua	Roma	230	50	400
5	Padua	Milán	350	80	300
6	Padua	Florencia	100	70	370
7	Monto total	2200			

```
Inicio Principal
    leer N
    i←1
    hacer mientras i<=N
        Leer Num
        calcularCantidadyEmpieza(num,cant,empieza)
        Escribir cant,empieza
        i←i+1
    fin hacer
    CalcTotal(n,total,cantEmpieza)
    monto←total*1.17
    Escribir Total, monto
    porc←cantEmpieza/N
    si porc>=0.8 entonces
        Nmonto←monto*0.9
    Caso contrario
        Nmonto←monto
    fin si
    escribir Nmonto
Fin Principal
```

```
Inicio calcularCantidadyEmpieza(num,cant,empieza)
    dig←Residuo(num,10)
    si Residuo(Dig,2)=0 entonces
        empieza←VERDADERO
    Sino
        empieza←FALSO
    fin si
    hacer mientras Num>0
        dig←Residuo(Num,10)
        Num←Cociente(Num,10)
        si Residuo(Dig,2)=0 entonces
            cant←cant+1
        fin si
    fin hacer
Fin calcularCantidadyEmpieza

Inicio CalcTotal(n,total,cantEmpieza)
    i←1
    hacer mientras i<=N
        leer cant,emp
        total←total+cant
        si emp entonces
        cantEmpieza←cantEmpieza+1
    fin si
        i←i+1
    fin hacer
Fin CalcTotal
```

5.3.3. *Mi imprenta*: palabras codificadas

La empresa *Mi Imprenta* planea hacer una campaña por el día de la amistad en la que cada cliente puede personalizar sus tarjetas con mensajes cifrados de diferentes tipos y longitudes.

Ilustración 5.12 Ejercicio de diseño «*Mi imprenta*: palabras codificadas»: pseudocódigo

	A	B	C
1	**8**	**Cantidad de dígitos pares**	**Empieza por dígito par**
2	**778567726583**	**5**	**FALSO**
3	**7069767367736865686983**	**11**	**FALSO**
4	**807982**	**4**	**VERDADERO**
5	**6976**	**2**	**VERDADERO**
6	**687365**	**3**	**VERDADERO**
7	**6869**	**3**	**VERDADERO**
8	**7665**	**2**	**FALSO**
9	65777383846568	7	VERDADERO
10	Total	37	
11	Monto	43.29	38.96

Para ello deciden lo siguiente:

- Cada dígito par costará 1.17 y cada impar será gratis.
- Si el 80% o más de los mensajes cifrados empiezan por dígito par, se hará un descuento del 10% sobre el monto total.

```
Inicio Principal()
        Leer N
        montoTotal←HallarMonto(N)
        Escribir montoTotal
Fin Principal

Inicio HallarMonto(N)
        i←1
        suma←0
        hacer mientras(i<=N)
                leer hosp,com,costo
                suma←suma+hosp+com+costo
                i←i+1
        fin hacer
        HallarMonto←suma
Fin HallarMonto
```

5.4. Codificación de estructuras iterativas

5.4.1. Do - Loop

La estructura Do - Loop puede usarse de cuatro diferentes maneras:

```
a) Do While condicion
       Instrucciones
   Loop

b) Do Until condicion
       Instrucciones
   Loop

c)Do
       Instrucciones
   Loop While condicion

d) Do
       Instrucciones
   Loop Until condicion
```

En a) y b) las instrucciones se repiten como mínimo 0 veces; en cambio en c) y d), como mínimo, se repiten 1 vez, debido a que la condición se evalúa al final. Esto se debe a que en los dos primeros casos si la condición es falsa, entonces nunca se evaluarán las instrucciones, pero, en los dos últimos, se evalúan las instrucciones sin importar el valor de la condición.

Asimismo, en a) y c) las instrucciones se repetirán si la condición es verdadera; en cambio, en b) y d) se repetirán si la condición es falsa o hasta que la condición sea verdadera.

Por ejemplo, si se quisiera tener una función que sumara los N primeros números usando la opción a), el código sería el siguiente:

```
Function SumaNprimeros(ByVal N as Byte)as Long
    Dim Suma as Long
    Suma=0
    i=1
    Do while i<=N
        Suma=Suma+i
        i=i+1
        Loop
        SumaNprimeros=Suma
End Function
```

Las instrucciones dentro del Do - Loop son las que se repetirán y lo harán siempre que «i» sea menor igual a «N». Ni bien «i» sea mayor que «N», el intérprete se encargará de ir a la siguiente instrucción, al Loop, en este caso a SumaNprimeros=Suma.

Debemos recordar que si tenemos una función, al momento de acumular es necesario utilizar una variable local auxiliar, ya que si se usa la línea de instrucción «SumaNprimeros=SumaNprimeros+1» se caerá en un error de compilación.

Usando la opción b), el código será el siguiente:

```
Function SumaNprimeros(ByVal N as Byte)as Long
    Dim Suma as Long
    Suma=0
    i=1
    Do until  i>N
        Suma=Suma+i
        i=i+1
    Loop
    SumaNprimeros=Suma
End Function
```

Las instrucciones dentro del Do - Loop son las que se repetirán y lo harán hasta que «i» sea mayor que «N». Si «N» es menor o igual a «i» las instrucciones se repetirán. Ni bien «i» sea mayor que «N» el intérprete se encargará de ir a la siguiente instrucción, al Loop, que, en este caso, es «SumaNprimeros=Suma».

Así, las opciones c) y d) generarán pequeños cambios en la función:
–Usando la opción c)

```
Function SumaNprimeros(ByVal N as Byte)as Long
    Dim Suma as Long
    Suma=0
    i=1
    Do
        Suma=Suma+i
        i=i+1
    Loop until  i>(N+1)
    SumaNprimeros=Suma
End Function
```

–Usando la opción d)

```
Function SumaNprimeros(ByVal N as Byte)as Long
    Dim Suma as Long
    Suma=0
    i=1
    Do
        Suma=Suma+i
        i=i+1
    Loop until  i>(N+1)
    SumaNprimeros=Suma
End Function
```

5.4.2. For - Next

Se utiliza esta estructura si se sabe cuántas veces se repetirán las instrucciones, ya que el «For» necesita un inicio y un fin:

```
For i=<inicio> to <fin> Step <valor incremento>
    <instrucciones>
Next
```

Se puede colocar después del «Next» la variable que manejará el «For» (Next i), pero esto es opcional. Por tanto, queda como decisión del programador colocarla o no. Algo que debemos recordar es que el «For» solo nos permite evaluar una variable, así que si quisiéramos evaluar una segunda, sería necesario otro «For» u otra estructura.

En el caso anterior, las instrucciones se repetirán fin-inicio+1 veces, ya que el incremento de «i» va de uno en uno. Por ejemplo, si quisiéramos sumar los «N» números, desde 1 hasta «N», las instrucciones serían:

```
Function SumaNprimeros(ByVal N as Byte)as Long
Dim Suma as Long
Suma=0
For i=1 to N
        Suma=suma+i
Next
SumaNprimeros=Suma
End Function
```

Como se puede observar, el valor de «i» no se actualiza, ya que la misma estructura «For» se encarga de ello cuando llega al «Next».

Por otro lado, si quisiéramos sumar los «N» primeros números pares, bastaría con hacer el incremento de 2 en 2, empezando en 0:

```
Function SumaNprimerosPares(ByVal N as Byte)as Long
Dim Suma as Long
Suma=0
For i=0 to N Step 2
        Suma=suma+i
Next
SumaNprimerosPares=Suma
End Function
```

En una estructura «For» es posible colocar el inicio en cualquier valor y el fin también, pero la manera más sencilla de aplicar esta instrucción es colocando el inicio siempre en 1 y variando los límites de las instrucciones, ya que esto permitirá hacer referencias a filas y columnas. Además, debemos tener en cuenta que la condición por verificar en el «For» es que la variable, en este caso «i», sea menor igual al fin. Asimismo, cada vez que lleguemos a la instrucción «Next», dicha variable debe incrementarse en el valor colocado luego de la palabra «Step»; si no se ha colocado dicha palabra, entonces «i» deberá incrementarse de uno en uno. Finalmente, luego de que «i» incremente su valor, deberemos volver al «For», con el fin de evaluar la condición. Una vez que la condición ya no se cumpla, la instrucción que se ejecutará será la que esté después del «Next».

5.5. Ejercicios de codificación

5.5.1. Presupuesto

En este ejercicio, que ya vimos en el capítulo 2 (ver punto 2.5.2), se pide repetir la llamada al subprograma «CalcularSubtotal». Como en esta sección se desea utilizar estructuras iterativas, el enunciado ha sido modificado:

Ilustración 5.13 Ejercicio de codificación: «Presupuesto»

	A	B	C	D
1		Cantidad	Precio	Subtotal
2	Pasajes	10	0.8	8.0
3	Almuerzos	5	4.5	22.5
4	Copias	30	0.07	2.1
5			Total	32.6

Asimismo, se pide implementar en VBA:

a) Un subprograma que lea la cantidad y el precio de un concepto, en tanto recibe como parámetro a la fila en la que se encuentra el concepto.

```
Const cantidadConceptos = 3
Const col = 2
Sub LeerDatos(ByRef cant As Byte, ByRef precio As Single, ByVal fila As Byte)
  cant = Cells(fila, col)
  precio = Cells(fila, col + 1)
End Sub
```

b) Un subprograma que calcule el subtotal a partir del precio y la cantidad.

```
Function CalcularSubtotal(ByVal cant As Byte, ByVal precio As Single) As_
Single
  CalcularSubtotal = precio * cant
End Function
```

c) Un subprograma que llame a todos los subprogramas anteriores y que permita calcular el presupuesto de la semana. Tenga en cuenta que solo hay tres conceptos.

```
Sub Presupuesto()
  Dim i As Byte, total As Single, subtotal As Single
  Dim cantidad As Byte, precio As Single
  i = 1
  total = 0
  Do While i <= cantidadConceptos
      Call LeerDatos(cantidad,precio,i+1)
      subtotal = CalcularSubtotal(cantidad,precio)
      total = total+subtotal
      Range("D" & (i+1)) = subtotal
      i = i+1
```

```
        Loop
        Range("D" & (i+1)) = total
End Sub
```

En este ejemplo se llevará a cabo la iteración tres veces: desde que «i» sea 1 hasta que llegue a 3. Luego, las instrucciones que se repetirán serán: «lee los valores de la fila correspondiente», «calcula el subtotal para los valores leídos», «acumula el total», «muestra el subtotal» y, finalmente, «actualiza la variable i».

5.5.2. Tres tristes tigres

Se tiene una frase en una celda de la hoja de cálculo y se desea saber el número de veces que se repite cada letra del abecedario en ella.

Ilustración 5.14 Ejercicio de codificación: «Tres tristes tigres»

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	Frase	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	Tres tristes tigres comen trigo en un trigal	1	0	1	0	5	0	3	0	4	0	0	1	1	3	2	0	0	5	4	6	1	0	0	0	0	0

En este ejercicio se debe modular para no tener muchas estructuras iterativas en un mismo subprograma, por lo que se elaborará una función «HallaCantidad» que calculará las repeticiones de una única letra en la frase. Con esta función se recorrerá la frase letra por letra, no sin antes evaluar si la letra es la que se está buscando y, de ser así, se incrementará el contador iniciado en 1.

```
Const LetraZ = «Z»

Function hallacantidad(ByVal letra As String, ByVal cadena _
As String) As Byte
    Dim ncar As Byte, i As Byte, car As String, suma As Byte
    ncar = Len(cadena)
    suma = 0
    cadena = UCase(cadena)
    For i = 1 To ncar
        car = Mid(cadena,i,1)
        If car = letra Then
            suma = suma+1
        End If
```

```
    Next
    hallacantidad = suma
End Function
```

El problema pedía calcular las repeticiones de cada letra del abecedario, entonces, como ya se tiene la función anterior, lo único que haremos en el procedimiento mostrado a continuación será recorrer todo el abecedario y, con cada una de esas letras, llamaremos a la función anterior para saber cuántas veces se repite. Debemos tener en cuenta que para ir de letra en letra es necesario que calculemos el código ASCCI, que le sumemos 1 y, luego, que lo convirtamos a carácter. Por ejemplo, la letra «a» tiene código 97; por tanto, si a 97 le sumamos 1, obtenemos 98 y, si a este número lo convertimos en carácter, tendremos como resultado la letra «b». De este modo, podemos repetir dicha operación hasta que lleguemos, inclusive, a la «z».

```
Sub HallaRepeticiones()
    cadena = Range("A19")
    letra = "A"
    fila = 19
    col = 2
    Do While letra <= LetraZ
        Cells(fila - 1, col) = letra
        cant = hallacantidad(letra,cadena)
        Cells(fila, col) = cant
        col = col + 1
        letra = Chr(Asc(letra)+1)
    Loop
End Sub
```

5.5.3. Microondas nuevo

Una familia ha comprado un nuevo microondas que tiene una tabla para descongelar.

Se sabe que los valores para el descongelado empiezan en la tercera fila de la hoja de Excel y que hay seis alimentos posibles:

Ilustración 5.15 Ejercicio de codificación «Microondas nuevo»: descongelado

	A	B	C	D	E
1	**Tabla para descongelar**				
2	Alimento	Peso	Unidad-Peso	Tiempo	Unidad-Tiempo
3	Carne picada	450	g	6	min
4	Pollo en salsa	0.75	kg	12	min
5	Lasagna	300	g	0.33	hora
6	Chuletas	0.3	kg	0.33	hora
7	Lentejas	0.5	kg	0.5	hora
8	Bacalao	225	g	6	min

Descongelado Menu Semanal

Llegó el domingo y es hora de preparar la comida de la semana: se prepararán cinco platillos y se guardarán en el congelador para que, así, llegado el día de la semana determinado, se descongele y luego se caliente. Para ello, se cuenta con un menú semanal:

Ilustración 5.16 Ejercicio de codificación «Microondas nuevo»: menú semanal

	A	B	C	D	E
1	**Menú semanal**				
2	**Día**	**Alimento**	**Peso**	**Unidad-Peso**	**Instrucciones**
3	Lunes	Lentejas	1000	g	nada
4	Martes	Lasagnas	450	g	nada
5	Miércoles	Chuletas	700	g	nada
6	Jueves	Bacalao	450	g	nada
7	Viernes	Pollo en salsa	750	g	nada

Descongelado Menu Semanal

La información del menú semanal se encuentra a partir de la tercera fila de la hoja «Menú semanal» y solo hay cinco días de menú.

De este modo, a partir de la información de esta hoja, se deben completar las instrucciones para descongelar los platillos y:

Ilustración 5.17 Ejercicio de codificación
«Microondas nuevo»: menú semanal resuelto

	A	B	C	D	E
1	Menú semanal				
2	Día	Alimento	Peso	Unidad-Peso	Instrucciones
3	Lunes	Lentejas	1000	g	Hay que descongelar 60 minutos
4	Martes	Lasagna	450	g	Hay que descongelar 29.7 minutos
5	Miércoles	Chuletas	700	g	Hay que descongelar 46.2 minutos
6	Jueves	Bacalao	450	g	Hay que descongelar 12 minutos
7	Viernes	Pollo en salsa	750	g	Hay que descongelar 20 minutos

Descongelado | Menu Semanal

a) Implementar un procedimiento «UnificaMinutos» que reciba por parámetros al número de alimentos y la columna donde se encuentran los tiempos para descongelar, y que, accediendo a la hoja de Excel, unifique los tiempos a minutos, tanto los valores numéricos como la unidad de tiempo.

Como se sabe cuántos alimentos hay, se puede usar «For», ya que ese será el número de repeticiones. Lo primero que se hará repetidas veces es leer de la hoja Descongelado la unidad, para luego evaluar si se trata o no de «hora» y, de serlo, se deberá recalcular el tiempo y escribirlo.

```
Sub UnificaMinutos(ByVal cantAlimentos As Byte)
Dim i As Byte, unidad As String
    For i = 1 To cantAlimentos
    unidad = Sheets(«Descongelado»).Cells(i+2, colUnidadTiempo)
    If unidad = «hora» Then
        Sheets("Descongelado").Cells(i+2,colUnidadTiempo) = «min»
        Sheets("Descongelado").Cells(i+2,colUnidadTiempo-1) = _
Sheets("Descongelado").Cells(i+2,colUnidadTiempo-1) * MinxHora
    End If
    Next
End Sub
```

b) Elaborar un procedimiento «UnificaGramos» que reciba por parámetros el número de alimentos y la columna donde se encuentran los pesos para descongelar, y que, accediendo a la hoja de Excel, unifique los pesos a gramos, tanto los valores numéricos como la unidad de peso.

Como se sabe cuántos alimentos hay, se puede usar «For», ya que ese será el número de repeticiones. Lo primero que se hará repetidas veces es leer de la hoja «Descongelado» la unidad, para luego evaluar si se trata o no de «kilogramo» y, de serlo, se deberá recalcular el peso y escribirlo.

```
Sub UnificaGramos(ByVal cantAlimentos As Byte)
Dim i As Byte, unidad As String
        For i = 1 To cantAlimentos
            unidad = Sheets("Descongelado").Cells(i+2,colUnidadPeso)
            If unidad = "kg" Then
                Sheets("Descongelado").Cells(i+2,colUnidadPeso) = "g"
                Sheets("Descongelado").Cells(i+2,colUnidadPeso - 1) = _
Sheets("Descongelado").Cells(i+2, colUnidadPeso - 1) * grsxKg
        End If
    Next
End Sub
```

c) Realizar un subprograma «DevolverTiempoPeso» que con los parámetros de entrada del número de alimentos y al alimento por buscar devuelva el tiempo y el peso.

En este subprograma solo se llamará a los subprogramas anteriores una vez, pero antes hay que encontrar la fila correspondiente al alimento. Para lo cual sí es necesario recorrer el grupo de alimentos.

```
Sub DevolverPesoTiempo(ByVal cantAlimentos As Byte, ByVal alimentoBuscar_
As String, ByRef peso As Single, ByRef tiempo As Single)
        Dim alimento As String, fila As Byte,encontro As Boolean
        fila = filaInicio
        encontro = False
        Do Until (fila = (cantAlimentos + filaInicio)) Or (encontro = True)
            alimento = Sheets("Descongelado").Cells(fila,colAlimento)
            If alimento = alimentoBuscar Then
                peso = Sheets("Descongelado").Cells(fila,colUnidadPeso - 1)
             tiempo = Sheets("Descongelado").Cells(fila,colUnidadTiempo - 1)
                encontro = True
```

```
            End If
            fila = fila + 1
        Loop
End Sub
```

d) Desarrolle un subprograma «Instrucciones» que en tanto recibe por parámetros la cantidad de días y la cantidad de alimentos complete la hoja correspondiente al menú semanal. La unidad en el menú semanal siempre será la misma.

```
Se llamará al procedimiento anterior para cada uno de los alimentos y
se escribirán los valores hallados.

Sub Instrucciones(ByVal cantAlimentos As Byte, ByVal cantDias As Byte)
 Dim fila As Byte, peso As Single, tiempo As Single, pesoMenu As Single, i As_
Byte, AlimMenu As String, tiempoMenu As Single
    fila = filaInicio
    For i = 1 To cantDias
        AlimMenu = Sheets("Menú Semanal").Cells(fila,colAlimentoMenu)
        Call DevolverPesoTiempo(cantAlimentos,AlimMenu,peso,tiempo)
        pesoMenu = Sheets("Menú Semanal").Cells(fila,colPesoMenu)
        tiempoMenu = (pesoMenu/peso) * tiempo
        Sheets("Menú Semanal").Cells(fila,colInstrucciones) =_
         "hay que descongelar" & tiempoMenu & «minutos»
        fila = fila + 1
    Next
End Sub
```

e) Implemente un subprograma principal que llame a los subprogramas anteriores.

```
Const colUnidadTiempo = 5, MinxHora = 60, colInstrucciones = 5
Const colUnidadPeso = 3, colAlimento = 1
Const colAlimentoMenu = 2
Const grsxKg = 1000, filaInicio = 3
Const colPesoMenu = 3

Sub Principal()
    Dim cantAlimentos As Byte, cantDias As Byte
    cantDias = Range("H1")
    cantAlimentos = Range("H2")
```

```
    Call UnificaMinutos(cantAlimentos)
    Call UnificaGramos(cantAlimentos)
    Call Instrucciones(cantAlimentos,cantDias)
End Sub
```

5.5.4. Nota final del curso

Cada curso incluye una cantidad de laboratorios, prácticas y tres exámenes. El promedio de laboratorios se calcula sumando todas las notas, menos la mínima, y dividiéndola entre el número total de notas menos una. Seguidamente, el promedio de prácticas se calcula de la misma manera sin incluir la mínima nota. En el caso de los exámenes solo se pueden dar dos de los tres. En consecuencia, la fórmula para el promedio final es:

$$NF=(PromLab*2 + PromPrac*2+Ex1*2+ Ex2*4)/10$$

Ilustración 5.18 Ejercicio de codificación: «Nota final del curso»

	A	B	C	D	E	F
1	**Laboratorio**	**6**	**Promedio**	**15.0**		
2	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**
3	12	15	17	10	16	15
4	Práctica	5	Promedio	14.7		
5	P1	P2	P3	P4	P5	
6	16	9	13	17	13	
7	Ex1	12				
8	Ex2	17				
9	Ex3					
10	Promedio del curso	15				

```
Sub pp()
    Dim nalumnos As Byte, fila As Byte, prom As Single, npractica As Byte, pa
    As Single, pb As Single
    nalumnos = Range("D1")
    fila = 3
    i = 1
    col = 1
    Do While (i<=nalumnos)
```

```
        Call promalumno(fila,pa,pb,prom)
        Cells(fila,col+5) = pa
        Cells(fila,col+10) = pb
        Cells(fila,col+15) = prom
        fila = fila + 1
        i = i + 1
    Loop
End Sub

Sub promalumno(ByVal fila As Byte, ByRef pa As Single, ByRef pb As _
Single, ByRef pf As Single)
    Dim npractica1 As Byte, npractica2 As Byte, e As Byte, e1 As Byte
    Npractica1 = Range("B4")
    Npractica2 = Range("B1")
    inicioColPa = 2
    pa = calcpromedio(npractica1,fila,inicioColPa)
    inicioColPb = 7
    pb = calcpromedio(npractica2,fila,inicioColPb)
    col = 14
    e = Cells(fila,col)
    col = col + 1
    e1 = Cells(fila,col)
    pf = (2*pb+2*pa+2*e+4*e1)/10
End Sub

Function calcpromedio(ByVal npractica As Byte, ByVal fila As Byte, ByVal _
inicioCol As Byte) As Single
    Dim suma As Byte, col As Byte, nota As Byte, min As Byte
    suma = 0
    col = 1
    Do While (col<=npractica)
        nota = Cells(fila,col+inicioCol-1)
        suma = suma + nota
        col = col + 1
    Loop
    min = calcminimo(npractica,fila,inicioCol)
    suma = suma - min
    calcpromedio = suma / (npractica-1)
End Function
```

```
Function calcminimo(ByVal npractica As Byte, ByVal fila As Byte, ByVal inicioCol_
As Byte) As Byte
    Dim min As Byte, col As Byte, nota As Byte
    min = 20
    col = 1
    Do While (col<=npractica)
        nota = Cells(fila,col+inicioCol-1)
        If (nota<min) Then
            min = nota
        End If
        col = col+1
    Loop
    calcminimo = min
End Function
```

5.5.5. Revisiones técnicas

Las revisiones técnicas vehiculares se realizan según el último dígito de la placa de rodaje, de acuerdo con el siguiente cronograma:

Ilustración 5.19 Ejercicio de codificación «Revisiones técnicas»: cronograma

	A	B	C	D	E	F	G	H
1	Último dígito de la placa	Vehículos sujetos a revisión anual (particular)	Vehículos sujetos a revisión semestral (público/carga)		Vehículos sujetos a revisión semestral (material inflamable y sustancias peligrosas)			
2		1era Rev.	1era Rev.	2da Rev.	1era Rev.	2da Rev.	3ra Rev.	4ta Rev.
3	6	Agosto	Octubre	Abril	Agosto	Noviembre	Febrero	Mayo
4	7	Setiembre	Noviembre	Mayo	Setiembre	Diciembre	Marzo	Junio
5	2	Abril	Agosto	Febrero	Julio	Octubre	Enero	Abril
6	4	Junio	Setiembre	Marzo	Agosto	Noviembre	Febrero	Mayo
7	5	Julio	Octubre	Abril	Agosto	Noviembre	Febrero	Mayo
8	8	Octubre y noviembre	Noviembre	Mayo	Setiembre	Diciembre	Marzo	Junio
9	9	Diciembre	Diciembre	Junio	Setiembre	Diciembre	Marzo	Junio
10	0	Enero	Julio	Enero	Julio	Octubre	Enero	Abril
11	1	Febrero y marzo	Agosto	Febrero	Julio	Octubre	Enero	Abril
12	3	Mayo	Setiembre	Marzo	Julio	Octubre	Enero	Abril

Cronograma Datos

Se quiere completar el detalle de revisiones para cada uno de los vehículos mostrados a continuación:

Ilustración 5.20 Ejercicio de codificación «Revisiones técnicas»: datos

	A	B	C	D	E	F
1	**Dígitos de Placa**	**Tipo de auto**	**Revisión 1**	**Revisión 2**	**Revisión 3**	**Revisión 4**
2	**123**	**Particular**	**Mayo**	**No tiene**	**No tiene**	**No tiene**
3	543	Material inflamable	Julio	Octubre	Enero	Abril
4	2356	Público	Octubre	Abril	No tiene	No tiene
5	1287	Carga	Noviembre	Mayo	No tiene	No tiene
6	3276	Sustancias peligrosas	Agosto	Noviembre	Febrero	Mayo
7	2354	Público	Setiembre	Marzo	No tiene	No tiene

Cronograma Datos

Para ello, se le pide implementar en VBA:

a) Un subprograma «HallaCantidadColumnaRevision» que en tanto tiene por parámetro el tipo de autos devuelva la columna inicial de revisiones de la hoja Cronograma y la cantidad de revisiones correspondientes.

```
Const cantDigitos = 10
Const CantAutos = 6

Sub HallaCantidadColumnaRevision(ByVal tipo As String, ByRef col As
Byte, ByRef cant As Byte)
    If tipo = "Particular" Then
        col = 2
        cant = 1
    ElseIf tipo = «Carga» Or tipo = «Público» Then
        col = 3
        cant = 2
    ElseIf tipo = «Material inflamable» Or tipo = «Sustancias peligrosas»
    Then
        col = 5
        cant = 4
    End If
End Sub
```

b) Un subprograma «HallaFila» que reciba por parámetro al último dígito de la placa y devuelva la fila de trabajo correspondiente de la hoja Cronograma.

```
Function HallaFila(ByVal ultimoDig As Byte) As Byte
    Dim fila As Byte, col As Byte
    col = 1
    For fila = 1 To cantDigitos
        If Sheets(«Cronograma»).Cells(fila+2,col) = ultimoDig Then
            HallaFila = fila + 2
        End If
    Next
End Function
```

c) Un subprograma «HallaRevisiones» que llame a los dos subprogramas anteriores y que llene en la hoja Datos los meses en los que se llevarán a cabo las revisiones para un único auto.

```
Sub HallaRevisiones(ByVal tipo As String, ByVal placa As Long, ByVal
filaDatos As Byte)
    Dim col As Byte, cant As Byte
    ultDig = placa Mod 10
    fila = HallaFila(ultDig)
    colDatos = 2
    Call HallaCantidadColumnaRevision(tipo,col,cant)
    For i = 1 To cant
        Sheets(«Datos»).Cells(filaDatos,colDatos+i) =
Sheets(«Cronograma»).Cells(fila,col)
        col = col + 1
    Next
    For i = cant + 1 To 4
        Sheets(«Datos»).Cells(filaDatos, colDatos+i) = «No tiene»
        col = col + 1
    Next
End Sub
```

d) Un subprograma principal que, a través del llamado al subprograma anterior, halle las revisiones para cada uno de los seis vehículos.

```
Sub PP()
    Dim fila As Byte, col As Byte, i As Byte
    Dim placa As Long, tipo As String
```

```
    fila = 1
    col = 1
    For i = 1 To CantAutos
        placa = Sheets(«Datos»).Cells(fila+i,col)
        tipo = Sheets(«Datos»).Cells(fila+i,col+1)
        Call HallaRevisiones(tipo,placa,fila+i)
    Next
End Sub
```

5.6. Ejercicios propuestos

5.6.1. Productos en camiones

En un camión entran 100 cajas, ya sea de los productos 1, 2, 3, 4 o 5. Por tanto, se desea saber cuántos camiones se necesitarán para el transporte mensual de los productos y cuánto se gastará en ellos. Además, se sabe que el alquiler de cada camión cuesta S/. 230.

Nota: no se pueden llenar los camiones parcialmente.

Ilustración 5.21 Ejercicio propuesto: «Productos en camiones»

	A	B	C	D	E	F
1		Nro meses	4	Nro productos	5	
2						
3		Enero	Febrero	Marzo	Abril	Total del producto
4	Producto 1	32	43	6	65	146
5	Producto 2	26	34	54	7	121
6	Producto 3	74	54	76	457	661
7	Producto 4	35	35	843	23	936
8	Producto 5	188	75	65	54	382
9						
10	Total del mes	355	241	1044	606	
11	Nro camiones	4	3	11	7	
12	Costo alquiler	920	690	2530	1610	

a) Elabore un subprograma que calcule el total del producto, en tanto tiene como parámetros la fila en la que se encuentra el producto y la cantidad de meses.

b) Implemente un subprograma que calcule el total del mes, en tanto recibe como parámetros la columna correspondiente al mes y a la cantidad de productos.

c) Realice un subprograma que calcule la cantidad de camiones necesarios para transportar los productos y el costo del alquiler, en tanto tiene como parámetro el total del mes.

d) Genere un subprograma que llame a los subprogramas anteriores y que muestre todos los cálculos necesarios.

5.6.2. Presupuesto de viaje familiar

Un padre de familia desea calcular cuál tendría que ser el presupuesto familiar. Se tiene la siguiente hoja de Excel y quiere saber si se viajará o no. El padre ha dicho que si el máximo monto por concepto es menor a US$ 300 entonces sí viajarán. Asuma que el tipo de cambio es de S/. 3.1.

Ilustración 5.22 Ejercicio propuesto: «Presupuesto de viaje familiar»

	A	B	C	D	E	F	G
1							
2		Nro de miembros de familia	4				Total por miembro de familia
3			Pasaje	Comida	Diversión	Compras	
4		Padre	300	30	90	10	430
5		Madre	300	25	90	300	715
6		Hijo 1	150	10	120	50	330
7		Hijo 2	150	10	120	50	330
8		Total por concepto S/.	900	75	420	410	
9		Total por concepto US$	290.3225806	24.19354839	135.483871	132.2580645	
10		Viajamos	VERDADERO				

5.6.3. Números invertidos

Ilustración 5.23 Ejercicio propuesto: «Números invertidos»

	A	B
1	Número original	Número invertido
2	1256324	4763521
3	98765	56789
4	13579	97531

a) Elabore un subprograma que invierta los dígitos de un número.

b) Desarrolle un subprograma que halle el número invertido para cada uno de los números mostrados. Asuma que no sabe la cantidad de dígitos que tiene el número, pero sí que son tres los números por evaluar.

5.6.4. Vamos a Italia

Se quiere calcular el presupuesto para los fines de semana de un viaje a Italia. Para ello, se conocen los destinos que se visitarán, el lugar de origen y se sabe que siempre es necesario regresar a este lugar por los cinco primeros días de una semana para luego pasar el fin de semana en el destino seleccionado. Se conoce, además, cuál es el gasto promedio en hospedaje y comida de los destinos, así como el costo del viaje.

Ilustración 5.24 Ejercicio propuesto: «Vamos a Italia»

	A	B	C	D	E
1	Presupuesto de viaje a Italia				
2	**Origen**	**Destino**	**Hospedaje**	**Comida**	**Costo Viaje**
3	Padua	Venecia	140	50	40
4	Padua	Roma	230	50	200
5	Padua	Milán	350	50	300
6	Monto total	1410			

5.6.5. Código binario

Ilustración 5.25 Ejercicio propuesto: «Código binario»

	A	B	C	D	E
1	**Codigo binario**	**Letra**			
2	**1001000**	**H**			
3	1101111	o			
4	1101100	l		Mensaje	Hola María
5	1100001	a			
6	100000				
7	1001101	M			
8	1100001	a			
9	1110010	r			
10	11101101	í			
11	1100001	a			

a) Implemente un subprograma que, en tanto tiene como parámetro el código binario, devuelva la letra correspondiente. Verifique que el valor esté entre los rangos 65-90, 97-122 o que sea el valor 32. De no ser válido, devuelva el mensaje: «código inválido».

b) Realice un subprograma que lea los códigos y muestre las letras correspondientes. Debe hacer esto para todos los códigos de la hoja. Finalmente, debe mostrar el mensaje oculto en la celda E4.

5.6.6. Citas médicas

Ilustración 5.26 Ejercicio propuesto: «Citas médicas»

	A	B	C	D	E	F	G
1	**Hora**	**Lunes**	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**
2	**9**	**Juan Pérez**					
3	10						
4	11						
5	12						
6	1		Marco Rodríguez				
7	2						
8	3						
9	4						
10	5						
11	6						

Desarrollar un subprograma que permita el manejo de las citas médicas de un día.

Para este caso, existen dos posibilidades:

a) Un paciente llega y solicita una cita; usted deberá darle la primera que encuentre libre.

b) Un paciente pide una hora específica.

Los pasos que deben seguirse son los siguientes: un paciente pide una cita, seleccionando la opción 1 para cualquier hora y 2 para una hora específica. Luego, según la elección, se busca el día y la hora más cercana o la deseada, y se le solicita al paciente su nombre, para actualizar el horario de citas.

5.6.7. Cantidad y suma de dígitos

Ilustración 5.27 Ejercicio propuesto: «Cantidad y suma de dígitos»

	A	B	C
1	Número original	Cantidad de dígitos	Suma de dígitos
2	2345	4	14
3	93856	5	31
4	5847	4	24
5	438	3	15
6	10294	5	16

a) Elabore un subprograma que, en tanto recibe como parámetro el número original, calcule la cantidad de dígitos que este tiene y la suma de ellos.

b) Implemente un subprograma que intercambie los valores de dos filas, a partir de los parámetros (en filas) «origen» y «destino».

c) Genere un subprograma que ordene ascendentemente los valores, a través de los subprogramas anteriores.

d) Realice un subprograma que llene la hoja de Excel y ordene los datos mediante algunos subprogramas anteriores. Recuerde que solo tiene cinco números originales.

Ilustración 5.28 Ejercicio propuesto: «Cantidad y suma de dígitos»: ordenados

	A	B	C
1	Número original	Cantidad de dígitos	Suma de dígitos
2	2345	4	14
3	438	3	15
4	10294	5	16
5	5847	4	24
6	93856	5	31

5.6.8. *Mi imprenta*: palabras

La empresa *Mi Imprenta* planea hacer una campaña por el día de la amistad, en la que cada cliente pueda personalizar sus tarjetas con mensajes de diferentes tipos y longitudes.

Para ello deciden lo siguiente:

- Cada vocal costará 1.15 y cada consonante, 1.1.
- Si el 80% o más de las palabras empiezan por consonante se hace un descuento del 10% sobre el monto total.

Sobre la base de las condiciones anteriores, se obtiene la siguiente hoja de Excel:

Ilustración 5.29 Ejercicio propuesto «*Mi imprenta*»: palabras

	A	B	C	D
1	8	Cantidad vocales	Cantidad consonantes	Empieza por consonante
2	Muchas	2	4	VERDADERO
3	felicidades	5	6	VERDADERO
4	por	1	3	VERDADERO
5	el	1	2	FALSO
6	día	2	1	VERDADERO
7	de	1	2	VERDADERO
8	la	1	1	VERDADERO
9	amistad	3	4	FALSO
10	Total	16	23	
11	Monto	18.4	25.3	43.7

Se sabe que los datos empiezan a partir de la segunda fila.

Se le pide implementar en VBA:

a) Un subprograma principal que lea el número de palabras de la celda A1 y, luego, tantas veces como palabras tenga:
   - Lea la palabra.
   - Halle la cantidad de vocales mediante uno de los subprogramas siguientes.
   - Muestre la cantidad de vocales.
   - Encuentre la cantidad de consonantes con uno de los subprogramas.
   - Exhiba la cantidad de consonantes.

- Halle el total de vocales a partir de uno de los subprogramas.
- Averigüe el total de consonantes a través de uno de los subprogramas.
- Determine si aplica o no el descuento con uno de los subprogramas.
- Encuentre los tres montos mediante uno de los subprogramas.
- Muestre los totales y los montos.

b) Un subprograma «HallaMontos» que, en tanto recibe el total de cantidad de vocales, consonantes y la respuesta correspondiente a si aplica o no el descuento, devuelva el monto por vocales, por consonantes y el monto total aplicando el descuento, de ser aplicable.

c) Un subprograma «CalcCantVocales» que en tanto recibe la palabra devuelva la cantidad de vocales incluidas en ella.

d) Un subprograma «CalcCantConsonantes» que reciba por parámetros la cantidad de vocales de la palabra, así como la palabra, y devuelva la cantidad de consonantes que hay en ella.

e) Un subprograma «AplicaDescuento» que, en tanto tiene por parámetro la cantidad de palabras, diga, por medio de un valor de verdad o falsedad, si aplica o no el descuento.

5.6.9. *Mi imprenta*: palabras codificadas

La empresa *Mi Imprenta* planea hacer una campaña por el día de la amistad, en la que cada cliente pueda personalizar sus tarjetas con mensajes cifrados de diferentes tipos y longitudes. Para ello deciden lo siguiente:

- Cada número par costará 1.17 y cada impar, 1.2.
- Si el 80% o más de los mensajes cifrados empiezan por número par se hace un descuento del 10% sobre el monto total.

Sobre la base de las condiciones anteriores, se obtiene la siguiente hoja de Excel:

Ilustración 5.30 Ejercicio propuesto: «*Mi imprenta:* palabras codificadas»

	A	B	C	D
1	**8**	**Cantidad números pares**	**Cantidad números impares**	**Empieza por impar**
2	778567726583	5	7	VERDADERO
3	7069767367736865686983	11	11	VERDADERO
4	807982	4	2	FALSO
5	6976	2	2	FALSO
6	687365	3	3	FALSO
7	6869	3	1	FALSO
8	7665	2	2	VERDADERO
9	65777383846568	7	7	FALSO
10	Total	37	35	
11	Monto	43.29	42	85.29

Se sabe que los datos empiezan a partir de la segunda fila.

Se le pide elaborar en VBA:

a) Un subprograma principal que lea el número de palabras cifradas de la celda A1 y que luego, tantas veces como palabras cifradas tenga:
   - Lea la palabra cifrada.
   - Halle la cantidad de números pares mediante uno de los subprogramas siguientes.
   - Muestre la cantidad números pares.
   - Encuentre la cantidad de números impares con uno de los subprogramas.
   - Enseñe la cantidad de números impares.
   - Saque el total de números pares a través de uno de los subprogramas.
   - Determine el total de números impares a partir de uno de los subprogramas.
   - Establezca si aplica o no el descuento con uno de los subprogramas.
   - Obtenga los tres montos mediante uno de los subprogramas.
   - Exhiba los totales y los montos.

b) Un subprograma «HallaMontos» que, en tanto recibe el total de cantidad de números pares, el de números impares y la respuesta correspondiente

a si aplica o no el descuento, devuelva el monto por números pares, por números impares y el monto total aplicando el descuento, de ser aplicable.

c) Un subprograma «CalcCantNumPares» que en tanto reciba la palabra devuelva la cantidad de números pares incluidos en ella.

d) Un subprograma «CalcCantNumImpares» que con los parámetros de la cantidad de números pares de la palabra cifrada y la palabra en sí, devuelva la cantidad de números impares que hay en ella.

e) Un subprograma «AplicaDescuento» que, en tanto tiene por parámetro a la cantidad de palabras cifradas, diga, por medio de un valor de verdad o falsedad, si aplica o no el descuento.

5.6.10 Amigos a la playa

Es verano y un grupo de cinco amigos quiere aprovechar los días de sol; por tanto, deciden ir a la playa (tenga en cuenta que necesitarán regresar de ese lugar), pero aún no determinan cómo. Usted, alumno del curso de Introducción a la Computación, quiere ayudarlos, a través de una aplicación para que tomen la mejor decisión.

Ahora bien, se sabe que cada uno de los amigos tiene un vehículo, y que cada auto tiene sus propias características especiales: la cantidad de km que rinde por galón, el tipo de gasolina que consume, la capacidad de espacio adicional (como para llevar *coolers* con bebida y comida), además de tener o no aire acondicionado.

Ilustración 5.31 Ejercicio propuesto: «Amigos a la playa»

	A	B	C	D	E	F	G
1	**Amigo**	**Km recorridos por galón**	**Tipo de gasolina**	**Aire acondicionado**	**Tiene espacio**	**Monto utilizado**	**Puntaje total**
2	**Erick**	**50**	**90**	**Sí**	**Sí**	**63.8**	**8**
3	Vanessa	30	98	Sí	Sí	174.97	6
4	Juan Luis	35	95	No	No	117.66	2
5	José	40	90	Sí	No	79.75	5
6	Katia	50	90	No	Sí	63.8	6
7							
8	Se debe usar el carro de:		Erick				

De este modo, se necesita identificar el auto en el que viajarán teniendo en cuenta que tanto al espacio como al aire acondicionado se le asignarán un puntaje referencial y el puntaje por consumo de gasolina se relacionará con otro puntaje sobre la base del monto.

Adicionalmente, se sabe que la playa a la que desean ir está en el km 145.

Los valores correspondientes a precios y rangos se muestran a continuación (no olvide definir constantes para manejar estos valores):

Ilustración 5.32 Ejercicio propuesto: «Amigos a la playa»: puntajes

Tipo de gasolina	Precio
90	**11.0**
95	14.2
98	18.1

Puntaje	Debido a
2	**Sí tiene aire acondicionado**
0	No tiene aire acondicionado
3	Tiene espacio
0	No tiene espacio
4	De 0 a 50 soles por tramo
3	De más de 50 a 100 soles por tramo
2	De más de 100 a 150 soles por tramo
1	Más de 150 soles por tramo

Por consiguiente, usted debe implementar:

a) Un subprograma «HallaMontoUtilizadoParaUnVehiculo» que, en tanto recibe la cantidad de km recorridos por galón y el tipo de gasolina que consume, calcule y devuelva el monto utilizado por un vehículo.

b) Un subprograma «CalculaPuntaje» que en tanto tiene el monto utilizado verifique si cada vehículo tiene espacio y/o aire acondicionado, y devuelva el puntaje total obtenido por auto.

c) Un subprograma «HallaGanador» que con el puntaje total y el nombre de cada propietario devuelva el nombre del amigo cuyo vehículo se usará. Tenga en cuenta que dicho propietario debe ser el que obtenga mayor puntaje. Si hay puntajes iguales, se escogerá al primero de la lista con puntaje ganador.

d) Un subprograma principal que:
   - Para cada uno de los vehículos de los amigos:
     - Lea los km recorridos, el tipo de gasolina, si tiene o no aire acondicionado y si tiene o no espacio en el auto.

- Calcule el monto utilizado con uno de los subprogramas anteriores.
- Muestre el monto utilizado en la fila correspondiente.
- Deduzca el puntaje mediante uno de los subprogramas precedentes.
- Exprese el puntaje total en la fila correspondiente.

• Halle al ganador a partir de uno de los subprogramas previos.

• Declare al ganador.

5.6.11. Mensajes personales codificados

Usted y su mejor amigo quieren encontrar la manera de comunicarse en secreto; para ello, crean una pequeña aplicación que les permite cifrar distintos mensajes:

Ilustración 5.33 Ejercicio propuesto
«Mensajes personales codificados»: originales

	A	B	C	D	E	F	G	H	I
1	**Mensaje original**	5	**Long Mensaje**	8					
2	Hoy-es-un-lindo-día-para-cortar-flores-	**Hoy**	**es**	**un**	**lindo**	**día**	**para**	**cortar**	**flores**
3	Es-necesario-estudiar-para-aprobar-con-buena-nota-	Es	necesario	estudiar	para	aprobar	con	buena	nota
4	Tres-tristes-tigres-comen-trigo-en-un-trigal-	Tres	tristes	tigres	comen	trigo	en	un	trigal
5	Es-necesario-conocer-el-mundo-para-aprender-geografía-	Es	necesario	conocer	el	mundo	para	aprender	geografía
6	Es-necesario-aprovechar-el-verano-y-viajar-mucho-	Es	necesario	aprovechar	el	verano	y	viajar	mucho

Mensaje original | Mensaje cifrado

Ilustración 5.34 Ejercicio propuesto «Mensajes personales codificados»: codificados

	A
1	**Mensaje cifrado**
2	Hoy Es Tres Es Es
3	es necesario tristes necesario necesario
4	un estudiar tigres conocer aprovechar
5	lindo para comen el el
6	día aprobar trigo mundo verano
7	para con en para y
8	cortar buena un aprender viajar
9	flores nota trigal geografía mucho

Mensaje original | Mensaje cifrado

Se le pide elaborar, utilizando VBA sobre Excel (recuerde manejar las distintas hojas de cálculo):

a) Un subprograma «MostrarPalabras» que reciba como parámetros la fila donde se encuentra el mensaje y el mensaje original de esa fila, y obtenga y muestre las palabras separadas del mensaje.

b) Un subprograma «CreayMuestraMensajeCifrado» que reciba como parámetros la columna donde están las palabras separadas, la fila de impresión donde se colocará el mensaje cifrado y la cantidad de mensajes cifrados originales, y halle y muestre uno de los mensajes cifrados.

c) Un subprograma principal que:
   - Lea la cantidad de mensajes originales.
   - Por cada mensaje inicial:
     - Lea el mensaje original.
     - Halle y muestre las palabras mediante un subprograma anterior.
   - Lea la longitud de mensajes.
   - Y, para cada una de estas longitudes:
     - Cree y muestre el mensaje cifrado a través de un subprograma previo.

5.6.12. Tutifruti

Usted desea hacer una aplicación en VBA con la que pueda jugar Tutifruti; para ello, define lo siguiente:

Ilustración 5.35 Ejercicio propuesto: «Tutifruti»

	A	B	C	D	E	F	G
1	Tutifruti						
2	**Letra**	**Cosa**	**Nombre**	**Lugar**	**Electrodoméstico**	**Animal**	**Puntaje**
3	A	Anillo	Ana	Austria	Aspiradora	Anaconda	5
4	P	Palo	Pablo	Perú	-	Gato	3
5	M	Marco	María	Madrid	-	Mono	4
6	O	-	Óscar	-	-	Oso	2
7	L	-	Lucas	Lima	Licuadora	León	4
8							
9					Puntos totales	18	

Elabore un subprograma «CalcPuntaje» que calcule el número de puntos obtenidos al jugar una de las letras en tanto tiene por parámetros a la fila y la letra por evaluar. El subprograma debe verificar si la palabra se inicia con la letra adecuada o no. Si el resultado es afirmativo, esto significará que el jugador hizo un punto adicional.

Realizar un subprograma principal que lea cada letra; calcule, mediante el subprograma anterior, los puntos correspondientes para cada una; muestre el puntaje; acumule los puntajes parciales y, finalmente, muestre el puntaje total.

EDITOR DE VBA

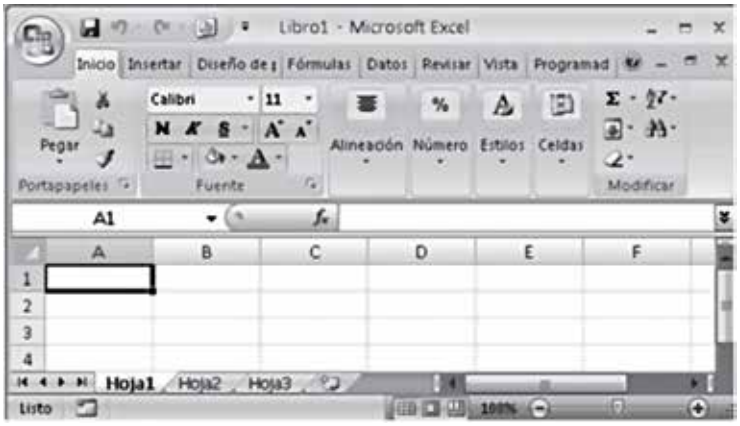
6.1. Acceso a las celdas

Existen dos propiedades que nos permiten acceder a las celdas de una hoja de Excel. A continuación, explicaremos cada una de ellas y veremos cuáles son las posibilidades para acceder a diferentes hojas dentro de un mismo programa.

El primer término que debemos conocer es el de *hoja activa*, que es aquella que se encuentra seleccionada al momento de ejecutar una macro o un subprograma principal. Es, además, la hoja que por defecto usan los módulos de VBA.

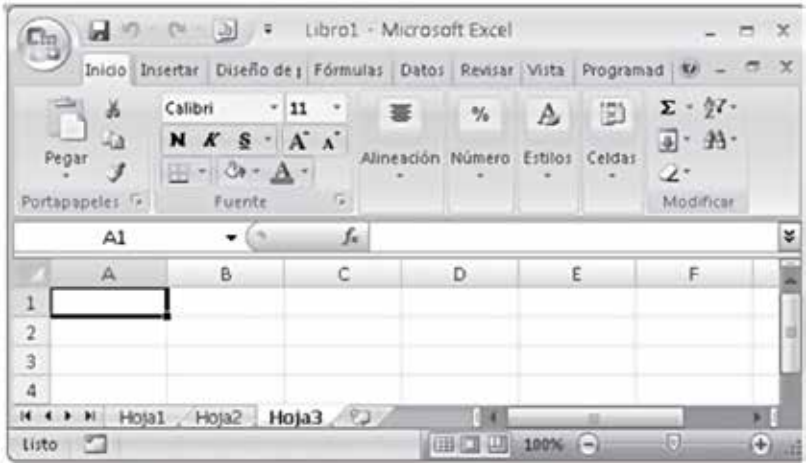
Por ejemplo, en la siguiente ilustración la hoja activa es la Hoja1:

Ilustración 6.1 «Hoja1»: hoja activa



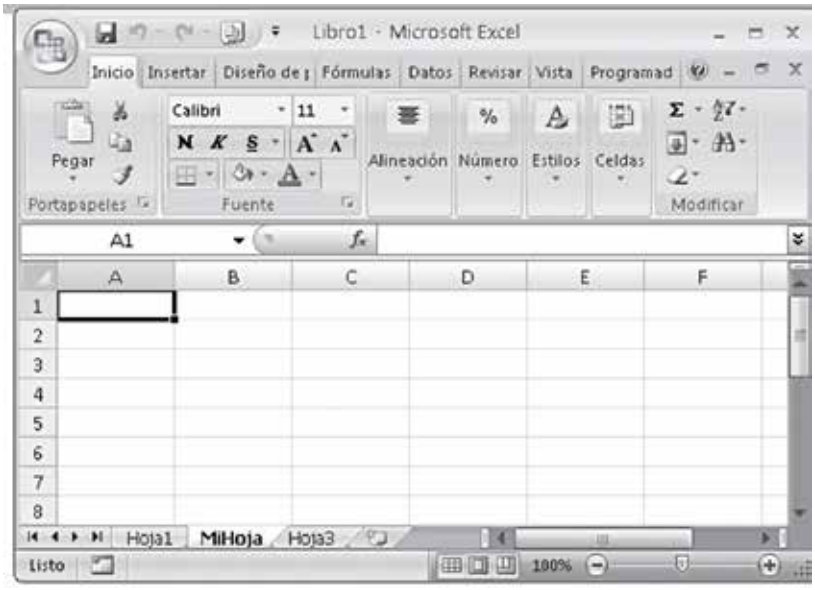
Seguidamente, en la ilustración la hoja activa es la Hoja3:

Ilustración 6.2 «Hoja3»: hoja activa



Como podemos observar en las ilustraciones anteriores, cada hoja tiene una etiqueta o nombre que la identifica. Por defecto, el nombre es «Hoja» + «i», donde «i» es el índice de la hoja, pero este puede cambiarse sin problemas. Por ejemplo, en la siguiente ilustración la «Hoja2» tiene un nuevo nombre, «MiHoja», y es la hoja activa:

Ilustración 6.3 «MiHoja»: hoja activa



Por tanto, si queremos cambiar el nombre, haremos clic derecho sobre la pestaña del nombre y seleccionemos la opción «cambiar nombre».

De otra parte, así como cada hoja tiene un nombre, cada libro o archivo tiene uno. Por ejemplo, en las ilustraciones anteriores el libro se llama «Libro1». De este modo, un punto muy importante cuando cambiamos el nombre del libro en VBA es que, como necesitamos activar macros, estos deben guardarse con las macros habilitadas. De lo contrario, el código escrito en los módulos se perderá o no se ejecutará. En consecuencia, cuando grabemos el libro seleccionaremos la opción «Libro de Excel habilitado para macros»:

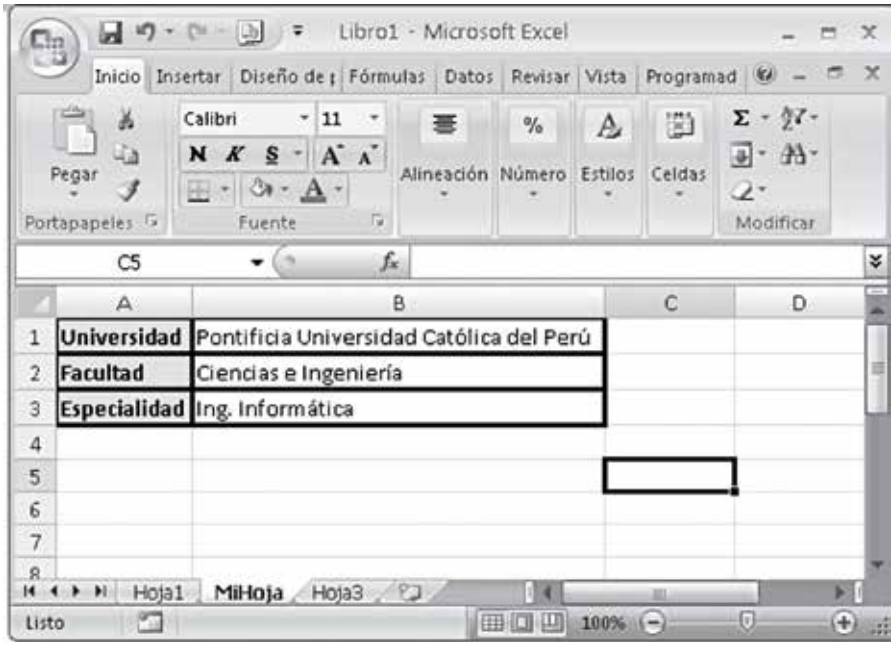
Ilustración 6.4 Guardar: Libro de Excel habilitado para macros

No obstante, si el diálogo de la ilustración anterior no aparece en nuestra pantalla, debemos asegurarnos de que la opción sea la adecuada cuando escribamos el nombre del archivo o libro:

Ilustración 6.5 Guardar como: Libro de Excel habilitado para macros

Ahora, a partir de los conceptos anteriores, podremos acceder a las celdas de Excel, para lo cual, como ya hemos mencionado, existen dos opciones: Range («COLUMNAFILA») y Cells («FILA,COLUMNA»).

Ilustración 6.6 MiHoja: Cells y Range



Por ejemplo:

- Si quisiéramos leer la facultad con Range la instrucción sería:

Facultad=Range("B2")

- Si deseáramos leer la facultad mediante Cells[1] la instrucción sería:

Facultad=Cells(2,2)

- Si decidiéramos leer la especialidad a través de Range la instrucción sería:

Especialidad=Range("B3")

- Finalmente, si quisiéramos leer la especialidad con Cells la instrucción sería:

Especialidad =Cells(3,2)

Ahora bien, como cada celda pertenece a una hoja en particular, si queremos referirnos específicamente a una, tendremos las siguientes posibilidades:

Especialidad = Sheets("MiHoja").Range("B3")
Especialidad = Sheets("MiHoja").Cells(3,2)

- Si trabajáramos en la Hoja1, por ejemplo, la podríamos llamar así:

Especialidad = Sheets("Hoja1").Range("B3")
Especialidad = Sheets("Hoja1").Cells(3,2)

- Además, por tratarse del nombre por defecto de la hoja, la podríamos llamar así:

Especialidad = Hoja1.Range("B3")
Especialidad = Hoja1.Cells(3,2)

Luego, así como la celda se encuentra en una hoja, esta se encuentra en un libro. Por consiguiente, también podemos aludir a este dentro de la programación:

Especialidad = Worksheets("Libro1").Sheets("Hoja1").Range("B3")
Especialidad = Worksheets("Libro1").Sheets("Hoja1").Cells(3,2)
Especialidad = Worksheets("Libro1").Hoja1.Range("B3")
Especialidad = Worksheets("Libro1").Hoja1.Cells(3,2)

Además, en tanto el nombre del libro se da por defecto, lo podemos llamar así:

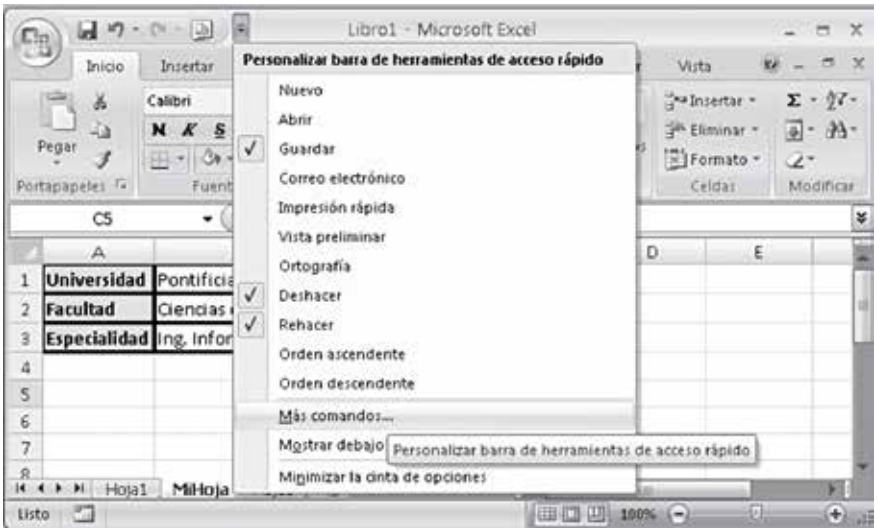
Especialidad = Libro1.Hoja1.Range("B3")
Especialidad = Libro1.Hoja1.Cells(3,2)

[1] En el caso de Cells, cuando se refiere a la columna, se reemplazan las letras por los valores numéricos correspondientes; por ejemplo, la columna A se convierte en la columna 1, y así sucesivamente.

6.2. Creación de botones

Existen opciones de formularios que nos permiten insertar botones a nuestras hojas de Excel y relacionarlos con macros previamente definidas. Para ello, debemos acceder a la opción «personalizar barra de herramientas de acceso rápido»:

Ilustración 6.7 Crear botones: Personalizar barra de herramientas

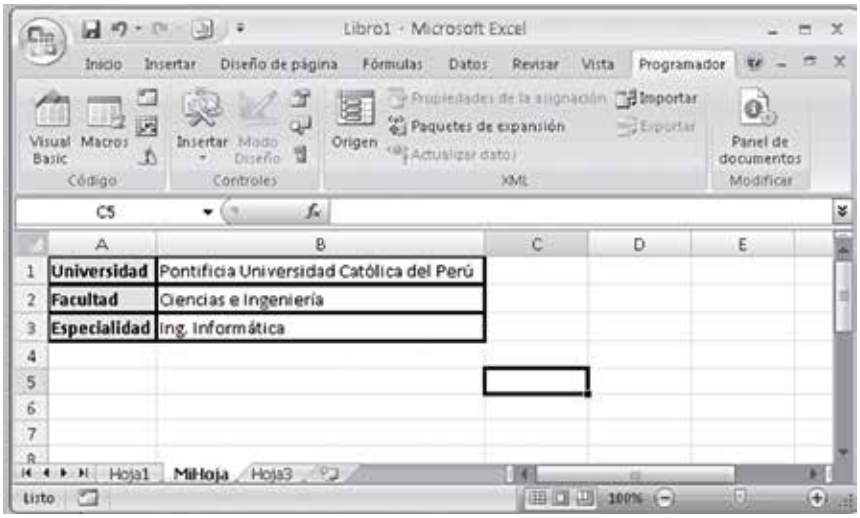


De este modo, si seleccionamos la opción «más comandos», se mostrará el siguiente diálogo:

Ilustración 6.8 Crear botones: Opciones de Excel

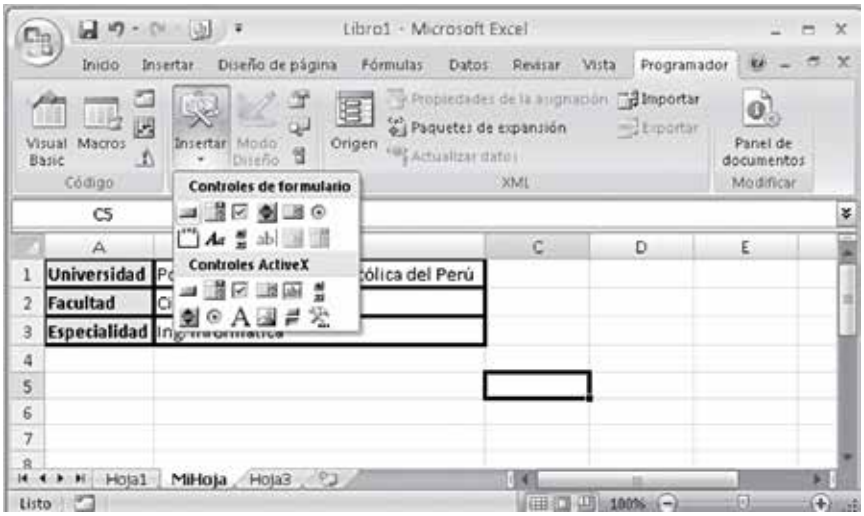
Luego, si escogemos la opción «Mostrar ficha Programador en la cinta de opciones», se mostrará dicha ficha:

Ilustración 6.9 Crear botones: Ficha programador



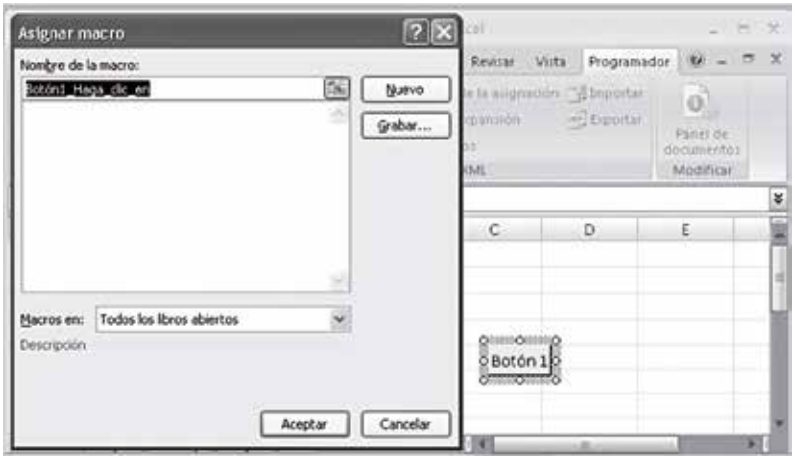
Ahora bien, para insertar un botón debemos seleccionar el control respectivo en la opción «controles de formulario»:

Ilustración 6.10 Crear botones: Insertar botón



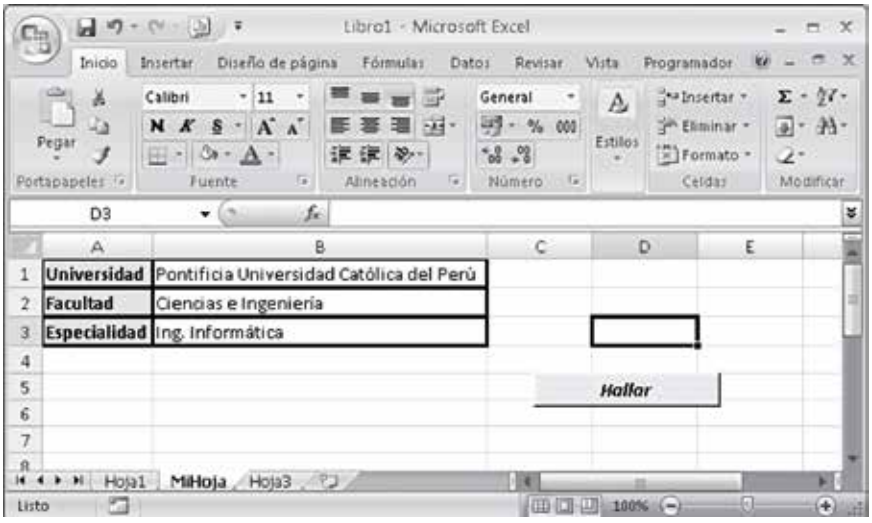
Una vez seleccionado el control, hacemos clic en cualquier lugar de la hoja de Excel y es allí donde se posiciona el botón. Luego, lo podemos relacionar con una macro o procedimiento sin parámetros.

Ilustración 6.11 Crear botones: Asignar macro



Asimismo, es importante señalar que al botón se le puede cambiar de tamaño, texto, entre otras opciones.

Ilustración 6.12 Crear botones

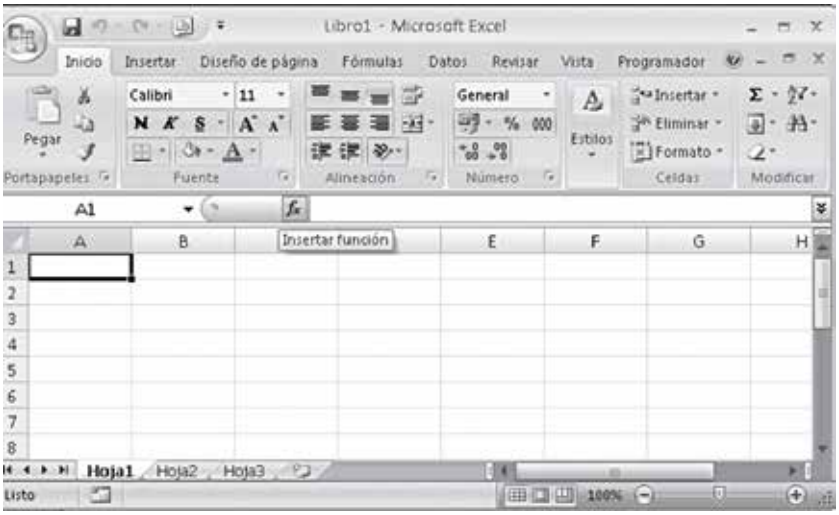


6.3. Prueba de funciones definidas por el usuario

Así como utilizamos una función de Excel en la hoja de cálculo, es posible que empleemos las funciones programadas en este libro en los módulos de VBA:

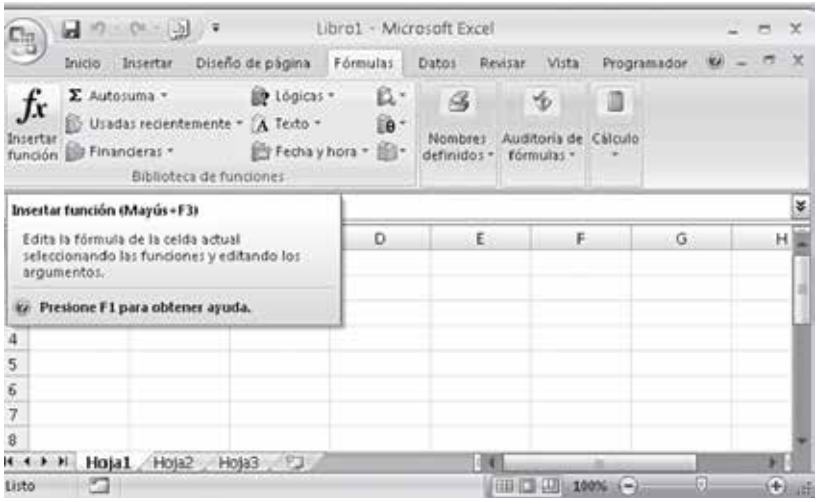
- En primer lugar, elegimos una celda e insertamos una función, seleccionando el signo «Fx»:

Ilustración 6.13 Insertar función



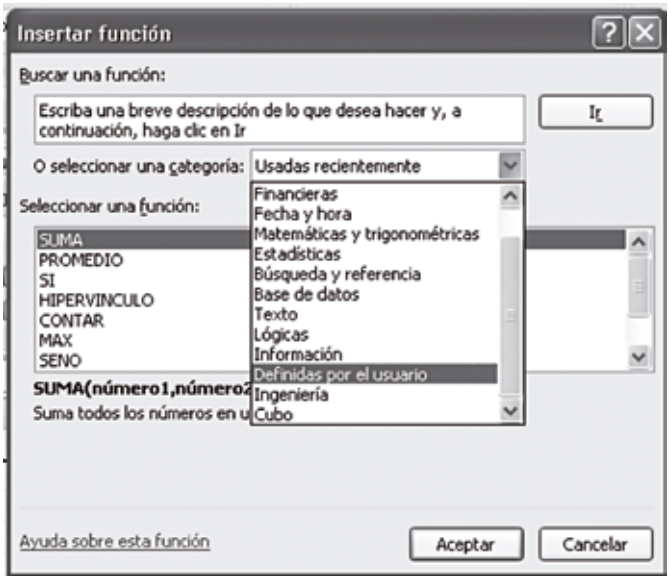
- O bien, podemos realizar esta acción escogiendo «Insertar función» desde la pestaña de fórmulas:

Ilustración 6.14 Insertar función: opciones



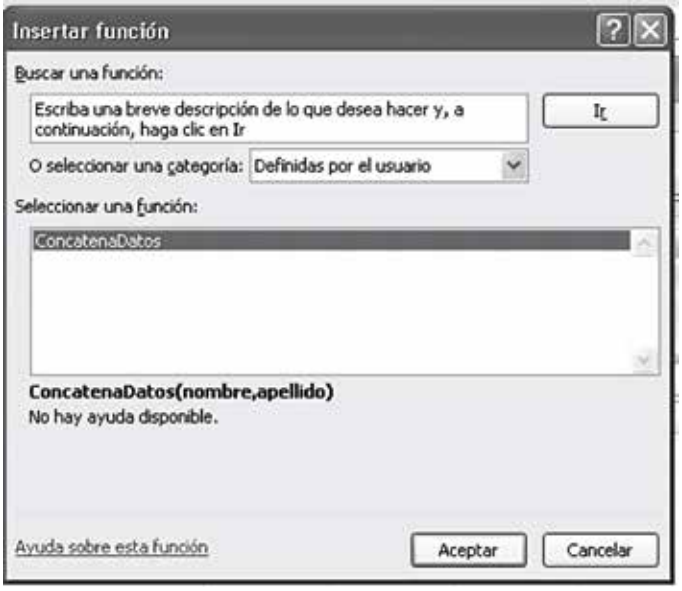
En conclusión en ambos casos aparecerá el siguiente diálogo, en el que deberemos escoger la categoría «Definidas por el usuario»:

Ilustración 6.15 Insertar función: funciones definidas por el usuario



Luego, aparecerá un listado con todas las funciones definidas por el usuario (escritas en los módulos):

Ilustración 6.16 Insertar función: funciones definidas por el usuario 2

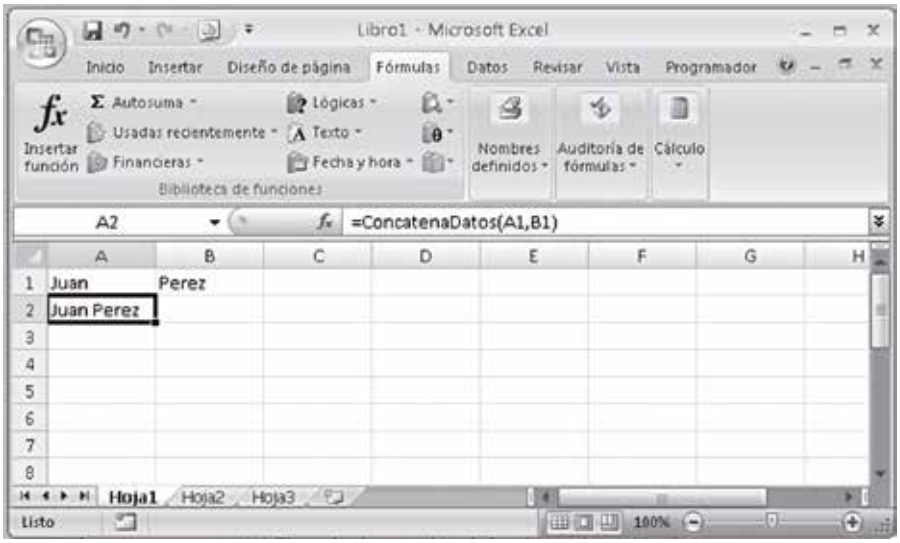


Una vez que hemos seleccionado la función que utilizaremos, se mostrará el siguiente diálogo, solicitando los valores o las referencias a los parámetros de la función. En este caso son dos los parámetros de la función: nombre y apellido:

Ilustración 6.17 Argumentos de función

Finalmente, el resultado de la función se posiciona en la celda correspondiente:

Ilustración 6.18 Resultado de función de usuario



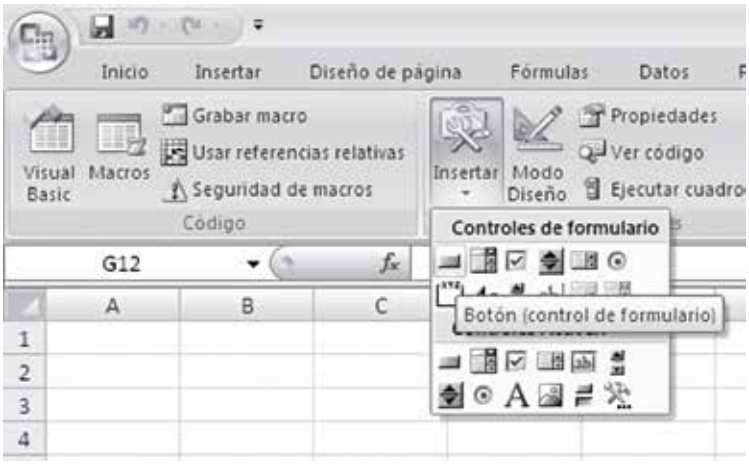
6.4. Ejemplo de creación de botón y asignación de macro

Se tiene el siguiente código en un módulo. Abra el editor de VBA (Alt+F11), inserte un módulo y copie el siguiente código:

```
Function CalcularPromedioFinal(ByVal PP As Single, _
    ByVal PL As Single, ByVal EP As Byte, ByVal EF As Byte) As Single
    Dim SumaNotas As Single
    SumaNotas = PP * 2 + PL * 2 + EP * 2 + EF * 4
    CalcularPromedioFinal = SumaNotas / 10
End Function

Sub Mostrar(ByVal PF As Single)
    Range("B5") = PF
End Sub

Sub LeerDatos(ByRef PP As Single, _
    ByRef PL As Single, ByRef EP As Byte, ByRef EF As Byte)
    PP = Range("B1")
    PL = Range("B2")
    EP = Range("B3")
    EF = Range("B4")
End Sub

Sub Principal()
    Dim PP As Single, PL As Single, EP As Byte
    Dim EF As Byte, PF As Single
    Call LeerDatos(PP,PL,EP,EF)
    PF = CalcularPromedioFinal(PP,PL,EP,EF)
    Call Mostrar(PF)
End Sub
```

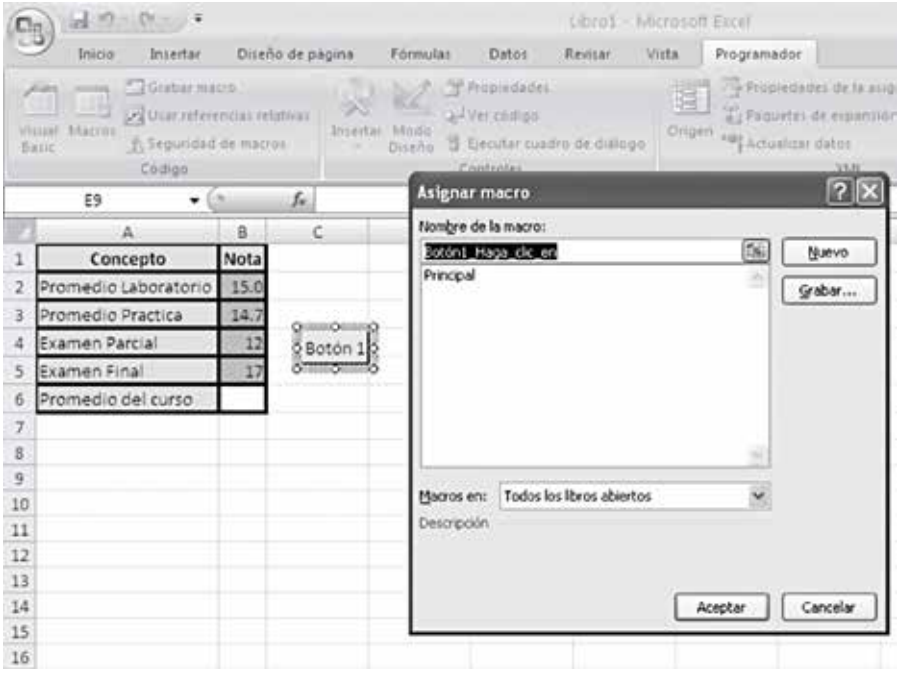
Luego, en la hoja de Excel inserte un botón:

Ilustración 6.19 Insertar botón



Seguidamente, asigne la macro al botón creado, «Principal»:

Ilustración 6.20 Asignar macro: ejemplo



Después de presionar el botón «aceptar», puede cambiar el nombre del nuevo botón. Finalmente, luego de presionarlo, obtendrá el resultado esperado:

Ilustración 6.21 Resultado botón: calcular

6.5. Errores de ejecución

6.5.1. «Desbordamiento»

Ilustración 6.22 «Desbordamiento»

Sucede cuando asignamos valores numéricos fuera del límite permitido. Por ejemplo, dada la siguiente línea de código:

*Producto=a*b*

Si definimos las tres variables, «producto», «a» y «b», de tipo byte, al multiplicar «a» por «b» obtendremos posiblemente un valor mayor al límite del byte. Por tanto, cuando tratemos de guardar dicho valor en una variable de este tipo se generará este error. Luego, para solucionarlo, tendremos que cambiar el tipo de dato de la variable «producto» a integer.

Por otra parte, como el intérprete de VBA maneja de una forma especial las variables, a veces este tipo de error se produce en casos como este:

*Producto=a*b*

Donde:

- «producto» es una variable de tipo long
- «a» es de tipo byte
- «b» es de tipo integer.

En este caso, el intérprete trata de almacenar temporalmente el resultado de la multiplicación de «a» y «b» en la variable «a» para, luego, asignar el resultado a la variable «producto» que es de tipo long. De este modo, para solucionar dicho error se tendrá que convertir temporalmente la variable «a» a un tipo de dato adecuado:

*Producto=CLng(a)*b*

6.5.2. «Error definido por la aplicación o el objeto»

Ilustración 6.23 «Error definido por la aplicación o el objeto»

Ocurre cuando tratamos de acceder a una celda inválida: por ejemplo, si buscamos Cells(0,3); como la fila 0 no existe, el error 1004 se produce.

6.5.3. «No coinciden los tipos»

Ilustración 6.24 «No coinciden los tipos»

Cuando intentamos asignar el tipo de dato equivocado en una variable. Por ejemplo, si tuviéramos una variable de tipo byte y en ella tratamos de guardar una cadena.

6.5.4. «División por cero»

Ilustración 6.25 «División por cero»

Cuando intentamos operar valores que valen 0. Esto sucede, por ejemplo, cuando no hemos asignado el valor a la variable o cuando se lee mal el dato.

6.6. Errores de compilación

6.6.1. «El tipo de argumento de ByRef no coincide»

Ilustración 6.26 «El tipo de argumento de ByRef no coincide»

Ocurre cuando no hemos declarado las variables o lo hemos hecho con el tipo de dato incorrecto, de tal manera que un subprograma espera un tipo de dato y recibe otro.

6.6.2. «Se esperaba una matriz»

Ilustración 6.27 «Se esperaba una matriz»

Cuando definimos una función o un procedimiento como si fuera una variable. Por ejemplo, la función «CalcularSuma» es declarada con dim dentro de algún subprograma.

6.6.3. «Se esperaba: identificador»

Ilustración 6.28 «Se esperaba: identificador»

Cuando llamamos inadecuadamente a una función o procedimiento. En este caso, debemos tener en cuenta que hay ciertas palabras reservadas del lenguaje que nos suelen llevar a este error; por ejemplo, «fix», que es una función definida en VBA.

6.6.4. «No se ha definido Sub o Function»

Ilustración 6.29 «No se ha definido Sub o Function»

Cuando hemos tipeado incorrectamente el nombre de un subprograma o se ha dejado de implementar alguno.

6.6.5. «…debe devolver un tipo Variant u Object»

Ilustración 6.30 «…debe devolver un tipo Variant u Object»

Cuando llamamos a una función al lado izquierdo de la asignación es necesario recordar que el valor devuelto por la función se guarda en una variable y no a la inversa.

Resolución de problemas usando **Visual Basic for Applications** *en Excel* presenta las herramientas y los métodos del lenguaje VBA en Excel y los emplea de manera didáctica mediante la propuesta y el desarrollo de ejercicios de aplicación. De esta manera, el libro permite a los estudiantes de cursos introductorios de informática y en general a usuarios avanzados de Excel aproximarse al proceso de resolución de problemas en ese programa de forma sencilla y directa. Sobre la base de operaciones matemáticas, se propone una metodología para iniciar tal proceso y llevarlo a cabo a través de seis fases: definición, análisis, diseño, codificación, depuración y documentación.

Asimismo, la autora desarrolla y propone ejercicios para que los estudiantes o usuarios se familiaricen con estos recursos informáticos y los puedan aprovechar en situaciones cotidianas.

ISBN: 978-612-4146-21-3