



**GOBIERNO DE
MÉXICO**

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



DEPARTAMENTO ADSCRIPCIÓN

Sistemas y Computación

Programa y Modalidad

B.1 Elaboración de apuntes/

B.1.2 Elaboración de apuntes de asignatura: algoritmos y lenguajes de programación

Nombre del Docente:

M. en TE. Martha Leticia Martínez Mora

No dictamen:

No. SS-1-116/2024

Personas Revisoras:

M. en TE. Marelis Carillo Lara

M. en TE. Andrea Olimpia Nájera Ávila

M. en E. Carmen Fabiola Jiménez Govea

Fecha: 31 de julio 2024



GOBIERNO DE
MÉXICO

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



Contribución Académica.

Generar estrategias y recursos didácticos, contribuye e incrementa el acervo de materiales propios institucionales que pueden ser modelados atendiendo de manera puntual cada uno los contenidos temáticos diseñados en las asignaturas de los programas de estudio que se imparten en el Tecnológico Nacional de México. Es por ello que se hace necesario elaborar material que sirva de apoyo para los estudiantes que cursan la materia de algoritmos y lenguajes de programación que plasme estrategias y recursos para desarrollar y realizar el proceso de abstracción y generar el modelado de un problema que contribuyan y potencialicen la capacidad de deducir de manera lógica y plasmar una propuesta de solución ante un problema matemático, analítico y de índole sistémico y desarrollar pensamiento algorítmico en los estudiantes, por ello la importancia de elaborar materiales que sirvan de apoyo para fortalecer dicho pensamiento. La asignatura de Algoritmos y lenguajes de programación brinda a los estudiantes de ingeniería industrial la capacidad de resolver problemas mediante el uso de computadoras, competencia con alta demanda en la industria manufacturera principal entorno para los egresados del programa educativo por lo que al crear los apuntes de la asignatura como recurso didáctico, se genera la construcción de un material como herramienta didáctica de apoyo a los estudiantes para profundizar los conocimientos de los temas que aborda la asignatura además de generar componentes didácticos que permitirán al estudiante adquirir los conocimientos, habilidades y metodologías para resolver problemas utilizando hojas de cálculo y entornos de programación mediante procesos sistematizados y fortalecer y adquirir conocimientos previos para materias de su plan de estudios como simulación donde utilizan metodologías para el diseño de software.

Contenido

Contribución Académica.	2
Introducción.	6
1. Introducción a la computación y hoja de cálculo	8
1.1 Introducción a la computación.	8
1.2 Sistemas operativos.	10
1.3. Elementos de Excel.	14
1.4 Fórmulas y funciones.	29
1.5 Macros	46
1.6 Aplicaciones.	64
1.7 Ejercicios	72
1.8 Aprendizaje basado en casos	73
2. Desarrollo de lógica algorítmica.	75
2.1. Metodología para la solución de problemas.	75
2.2. Metodologías para el diseño de software	78
2.3. Elementos y reglas de los lenguajes algorítmicos.	81
2.3.1. Diagramas de flujo.	83
2.3.2. Pseudocódigo.	86
2.4. Implementación de algoritmos.	87
2.4.1. Secuenciales.	87
2.4.2. Selectivos.	93
2.4.3. Repetitivos.	100
2.5. Pruebas y depuración	106
2.6 Aprendizaje basado en problemas	115
3. Introducción a la programación de un lenguaje estructurado.	118
3.1 Introducción y estructura del entorno de un lenguaje de programación.	118
3.2 Estructura básica de un programa.	119
3.3 Tipos de datos.	120
3.4 Identificadores.	121
3.5 Almacenamiento, direccionamiento y representación en memoria.	123
3.6 Proposición de asignación.	124
3.7 Operadores, operandos y expresiones.	124
3.8 Prioridad de operadores, evaluación de expresiones	128



3.9 Elaboración de programas.....	129
3.10 Pruebas y depuración.....	134
3.11. Ejercicios	138
3.12. Aprendizaje basado en problemas	139
4. Estructuras selectivas y de repetición.....	141
4.1 Selectiva simple.....	142
4.2 Selectiva doble.....	144
4.3 Selectiva anidada	146
4.4 Selectiva múltiple.....	148
4.5 Repetir mientras	150
4.6 Repetir hasta	154
4.7 Repetir desde.....	156
4.8 Elaboración de programas.....	157
4.9 Ejercicios	163
4.10 Aprendizaje basado en problemas	164
4.11 Aprendizaje basado en casos	165
5. Arreglos y archivos.....	166
5.1 Arreglo unidimensionales	167
5.2 Arreglo bidimensionales y multidimensionales	173
5.3 Apertura, entrada-salida de datos, y cierre de archivos.....	182
5.4 Elaboración de programas.....	186
5.5 Aprendizaje basado en problemas	192
6. Funciones.....	193
6.1 Introducción	193
6.2 Funciones estándar	194
6.3 Entrada y salida de datos.....	196
6.4 Funciones definidas por el usuario	197
6.5 Pase por valor	203
6.6 Pase por referencia	204
6.7 Elaboración de programas.....	205
6.8. Ejercicios	208



GOBIERNO DE
MÉXICO

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO



6.9 Aprendizaje basado en problemas	210
Referencias	212



Introducción.

Actualmente la interacción que las personas tienen con las computadoras es muy alta ya sea utilizando o desarrollando programas, por lo que se hace necesario fortalecer en los estudiantes de nivel superior, habilidades que les permita usar de manera eficaz software de aplicación en todos los aspectos de su vida personal y profesional como distingo en su formación académica.

El propósito de estos apuntes, es que el lector sea capaz de utilizar diversos entornos y elaborar programas utilizando la computadora como una herramienta de apoyo para la solución de problemas y para aquellos que consideran que la asignatura es poco importante en su formación académica, al leer y desarrollar los ejercicios resueltos y propuestos en los entornos propuestos, logren comprender que programar se vuelve una tarea menos complicada, que la práctica es el elemento para desarrollar la habilidad y que la programación es aplicable en todas las áreas del conocimiento porque les permite desarrollar el pensamiento lógico de las cosas y por ende fortalece y estimula el proceso del razonamiento de distintos problemas que se presentan en su entorno.

Los temas que aborda este material se dividen en 6. El tema 1, describe el uso y aplicación que puede ofrecer el entorno Excel para la creación y manejo de hojas de cálculo, la gama de funciones que dicho software ofrece y como una macro permite optimizar tiempos en el procesamiento de datos al programar directamente sobre una hoja de cálculo para hacer más eficiente los procesos repetitivos, este tema aportará al lector los elementos básicos para entender y utilizar este software y será su interés y dedicación de manera personal para comprender lo poderosa que esta herramienta se puede convertir en su formación académica ya que puede abrirle las puertas al mundos laboral por ser un software de aplicación de alta demanda y uso, el tema es el inicio de lo que cada estudiante puede ser capaz de lograr con dicho entorno.



El Tema 2, aborda los tipos de metodologías que existen para elaborar un programa y para comenzar a programar. Hace hincapié que la parte más complicada es el desarrollo de un algoritmo, ya que exige del razonamiento y análisis profundo del problema a resolver porque se requiere de paciencia para entender cómo razona el ser humano y posteriormente analizar como lo haría la computadora. En este tema se describen y utilizan dos técnicas para representar algoritmos: el pseudocódigo y el diagrama de flujo, se explica a detalle su uso y aplicación y además se recomienda el uso del segundo entorno: **PseInt**, para el desarrollo de los ejercicios planteados, y para aquellos estudiantes que comienzan a programar, les facilitará el aprendizaje y con la práctica se logre incentivar su pensamiento algorítmico.

El tema 3 y 4 aborda el tercer entorno: **Dev C++**, para poder elaborar programas. Describe los conceptos básicos en el diseño de la programación estructurada, presenta una serie de ejemplos que llevan de la mano al lector para el desarrollo de programas utilizando las estructuras básicas de control. Posteriormente los temas 5 y 6, explican estructuras más complejas como son el uso de variables unidimensionales, bidimensionales y generación de archivos y por último la programación modular con el uso de funciones con y sin parámetros.

Los temas se organizaron de manera similar, se presenta una descripción de los conceptos necesarios para comprender el entorno, se plasman varios ejemplos de uso probados en el entorno correspondiente y culmina con ejercicios resueltos y propuestos para que el lector fortalezca su conocimiento con la lectura y práctica, intenta fomentar el autoaprendizaje, porque se explica paso a paso cada entorno ya que los ejemplos refuerzan la comprensión del tema y los ejercicios propician la práctica y aplicación progresiva del aprendizaje significativo y se espera que el uso de este material contribuya a la enseñanza de la asignatura de Algoritmos y lenguajes de programación de los estudiantes de Ingeniería Industrial y de todos aquellos que así lo requieran.



1. Introducción a la computación y hoja de cálculo.

1.1 Introducción a la computación.

Hoy en día es necesario reconocer que el rápido ritmo en el cambio de la tecnología abre la necesidad de impartir cursos sobre aplicaciones de software que permitan proveer capacitación técnica y curricular en la preparación académica del estudiante. El uso de la computadora se ha expandido a un sinnúmero de áreas como herramienta de trabajo para optimizar y hacer más eficientes los procesos repetitivos o de grandes volúmenes de datos, facilitando el análisis y procesamiento.



Figura 1.1. Tecnologías de la Información y las comunicaciones

Adquirir artículos en línea, enviar mensajes, realizar transferencias, registrar datos, el uso de correo electrónico son actividades que se realizan con el uso de dispositivos electrónicos y solo nos damos cuenta de su uso e importancia cuando dejan de funcionar (Figura 1.1).

Una computadora facilita procesos, permiten acceder a la información de manera rápida y se pueden utilizar un sinnúmero de servicios que sin ella no sería posible. Por lo anterior, se hace necesario estar familiarizado con los siguientes términos para introducirse al tema de la computación. (s/a, 2020).

Computadora, es un dispositivo capaz de desarrollar operaciones a velocidades miles de veces más rápido que una persona. (Figura 1.2).



Figura 1.2. Computadora



Este tipo de operaciones se procesa bajo el control de un *grupo de instrucciones* conocidos como **programas**, los cuales guían a la computadora mediante acciones ordenadas y específicas realizadas por *programadores*. (Miranda Chávez & Fuenlabrada Velázquez, 2015)

Se compone de varios dispositivos, tales como: teclado, monitor, discos de almacenamiento, mouse o ratón, unidades de procesamiento, conocidos como hardware (Figura 1.2). Los programas que se ejecutan en la computadora para realizar una tarea específica se le conoce como software. La programación estructurada, el mejoramiento paso a paso, el diseño orientado a objetos, la programación orientada a objetos, entre otras, son métodos para desarrollar software de calidad que permiten incluso optimizar costos. (Jimenez, 2018)

Los componentes de una computadora en general se pueden agrupar de la siguiente manera:

- Unidades de entrada. Son los dispositivos que permiten introducir la información a la computadora: teclado, scanner, micrófono, ratón, recepción por internet, esta información se pone a disposición de otras unidades o dispositivos para su procesamiento.
- Unidades de salida. Una vez procesada la información ésta debe estar disponible para quién lo requiera. La mayoría de la información se imprime en papel o se envía mediante medios como el internet
- Unidad de memoria. Son los dispositivos que permiten que la información se almacene y pueda estar disponible cuando se requiera.
- Unidad aritmética lógica. Es la responsable de realizar operaciones tales como: suma, multiplicación, resta, división, comparación entre otras.
- Unidad central de proceso. Conocida como CPU, es considerado el cerebro de la computadora ya que coordina y supervisa a los demás componentes



de la computadora. Actualmente las computadoras tienen varios procesadores por lo que realizan múltiples operaciones de manera simultánea.

- Unidad de almacenamiento secundario. Existen dispositivos que almacenan información de manera externa a la computadora y son necesarios para conservar información que no es necesario procesar directamente por la memoria interna y evitan la saturación de información.

Actualmente el costo de una computadora es accesible para ser adquirida por una persona o negocio, las computadoras personales permiten manejar requerimientos de cómputo de acuerdo con las necesidades del usuario, la información hoy en día se comparte de manera fácil mediante redes de computadoras o servidores de archivos, los cuales ofrecen grandes volúmenes de información que transita en milésimas de segundo por internet y disponible para quienes la requieren.

1.2 Sistemas operativos.

Los sistemas operativos son programas que proveen servicios que permiten a cada aplicación ejecutarse de forma segura y eficiente con otras aplicaciones (Deitel, 2012).

Las primeras computadoras solo realizaban una tarea o trabajo (monotarea). Los sistemas operativos actuales, han sido desarrollados para hacer las tareas de manera más sencilla. Compartir recursos es la solución que permitió aprovechar al máximo los componentes de una computadora y se conoce como multiprogramación, es decir, realizar muchas tareas de manera tan coordinada y rápida dentro de la computadora compartiendo recursos entre las tareas, que brinda al usuario el servicio que requiere con una respuesta casi inmediata. Los sistemas



operativos más populares de escritorio son: Linux, UNIX, OS x de Mac y Windows, los sistemas operativos móviles que se utilizan en teléfonos inteligentes son: Android, Apple iOS, Windows. A continuación, se describe brevemente algunos de ellos.

Windows. Sistema operativo propietario bajo el control de una compañía, es una interfaz gráfica creada sobre Dos, sistema operativo de la década de los 80, integra iconos, menús y ventanas utilizadas en SO como Apple Macintosh en esa época. Actualmente ofrece mejoras en la interfaz de usuario y es considerado el SO más utilizado.



Figura 1.3. Windows



Figura 1.4 Logo Sistema Operativo Linux

Linux. SO de código fuente abierto con lo cual se unen esfuerzos para mantener y mejorar el software a cambio de utilizarlo. Algunas fundaciones de la comunidad de código abierto son: Eclipse, quienes ayudan a los programadores de Java a desarrollar software de manera optimizada; la fundación Mozilla, creadores del navegador Firefox, la fundación de software Apache, para desarrollar software basado en WEB, entre otras. Facebook, es el software ejemplo de desarrollo con código abierto.



MacOS. Sistema operativo para computadoras Macintosh de Apple, está basado en Unix y se usa en computadoras Apple, es la versión de sistema operativo más reciente de Apple y desde 1984 ha sido el sistema operativo utilizado por esta marca de computadoras, ofrece al usuario una interfaz intuitiva, seguridad robusta y menor riesgo de virus.



Figura 1.5 Logo S.O. MacOS



Figura 1.6 Logo Sistema Operativo Android

Android. Sistema operativo diseñado para el uso en dispositivos móviles y teléfonos inteligentes, está basado en el kernel de Linux y en lenguaje Java. El grado de apertura para desarrollar aplicaciones con este sistema operativo es muy bueno para los programadores experimentados en Java.

Fue desarrollado por Android Inc. en 2005, en el 2010 se volvió uno de los sistemas operativos más utilizados y lo emplean teléfonos de marcas reconocidas como Samsung, motorola, entre otros. Con el uso del sistema operativo Android, los teléfonos inteligentes tienen la funcionalidad de un teléfono móvil, navegar y comunicarse por internet, consola de juegos, entre otras múltiples funcionalidades. Existen hoy en día alrededor de 200,000 aplicaciones en el Android Market de Google.

iOS. Ofrece una interfaz gráfica muy intuitiva, altos estándares de seguridad y privacidad, servicios de iCloud para almacenamiento



Figura 1.7 Logo Sistema Operativo iOS



En el uso de tecnología computacional, el **sistema operativo** es considerado el programa más importante de un dispositivo electrónico, ya que éste es la parte central y es la que ejecuta las órdenes para realizar las tareas o acciones que se requieren realizar con el dispositivo. La estructura de niveles que integran un sistema operativo fue desarrollada por Dijkstra y describe cada uno de sus elementos, los cuales se plasman a continuación:

- **Núcleo.** Su función es administrar todo el sistema y sincronizar todos los procesos.
- **Gestor de entradas y salidas.** Administra los dispositivos externos mediante el uso de controladores
- **Gestor de memoria.** Realiza el control y administración de la memoria principal y memoria virtual en almacenamiento y uso de la información
- **Sistema de archivos.** Se encarga de administrar los archivos del usuario mediante un directorio
- **Interprete de comandos.** Se encarga de leer y ejecutar las funciones del sistema que sean solicitadas

Ejercicio de aprendizaje.

Realiza una infografía de los usos que tiene hoy en día las tecnologías de la información y las comunicaciones y plasma las ventajas y desventajas de su uso en la industria, la educación y a nivel personal.



1.3. Elementos de Excel.

Antes de comenzar es necesario conocer que Excel es un software de aplicación que nos permite desarrollar una serie de aplicaciones sustentadas en hojas de cálculo. Ofrece una gama de herramientas que permite a quién lo utiliza organizar, analizar, administrar y compartir información con gran facilidad.

Para utilizar el entorno de Excel, se pueden seleccionar libros nuevos, existentes o compartidos. Para crear un libro nuevo es importante iniciar el entorno de Excel y seleccionar el comando nuevo. Si ya se está trabajando en el entorno entonces las teclas Ctrl+U son la forma más rápida o método simplificado.

Existen dos maneras de ingresar y utilizar el entorno de Excel. La primera es cuando se tiene el pago de licencia individual de office, es necesario buscar la aplicación en el navegador de inicio y acceder al entorno desde el archivo ejecutable. La segunda forma es si se cuenta con una cuenta educativa entonces abrir la aplicación de office e iniciar sesión anotando el correo institucional y la contraseña de acceso. (Figura 1.8).

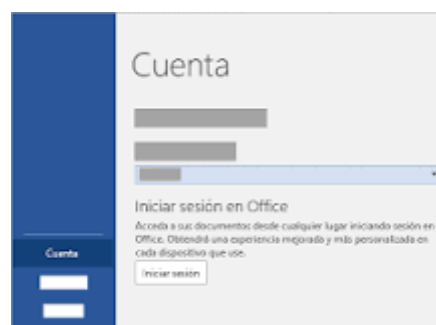


Figura 1.8. Acceder a Excel con licencia o con cuenta educativa



El inicio del navegador de Excel es importante ya que nos permite: Abrir un libro ya existente, crear uno nuevo, guardar datos, guardar una copia de un libro con otro nombre, compartirlo, exportar, entre otros. (Figura 1.9).

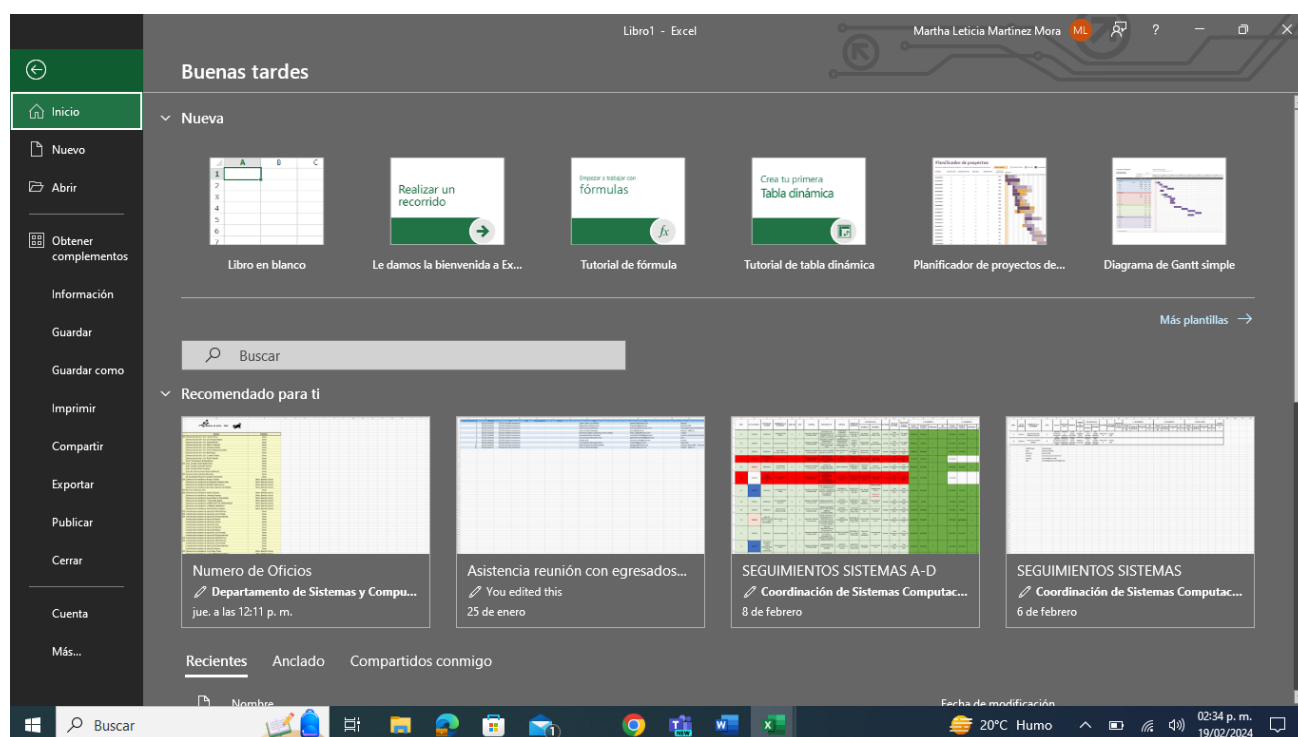


Figura 1.9. Entrono de inicio de Excel

Una vez que se activa una hoja de cálculo, Excel contiene una barra de opciones, que se puede observar en la parte superior de la ventana de inicio, la cual se organiza por comandos de tareas organizadas en función al tipo de trabajo que se va a realizar (Figura 1.10)

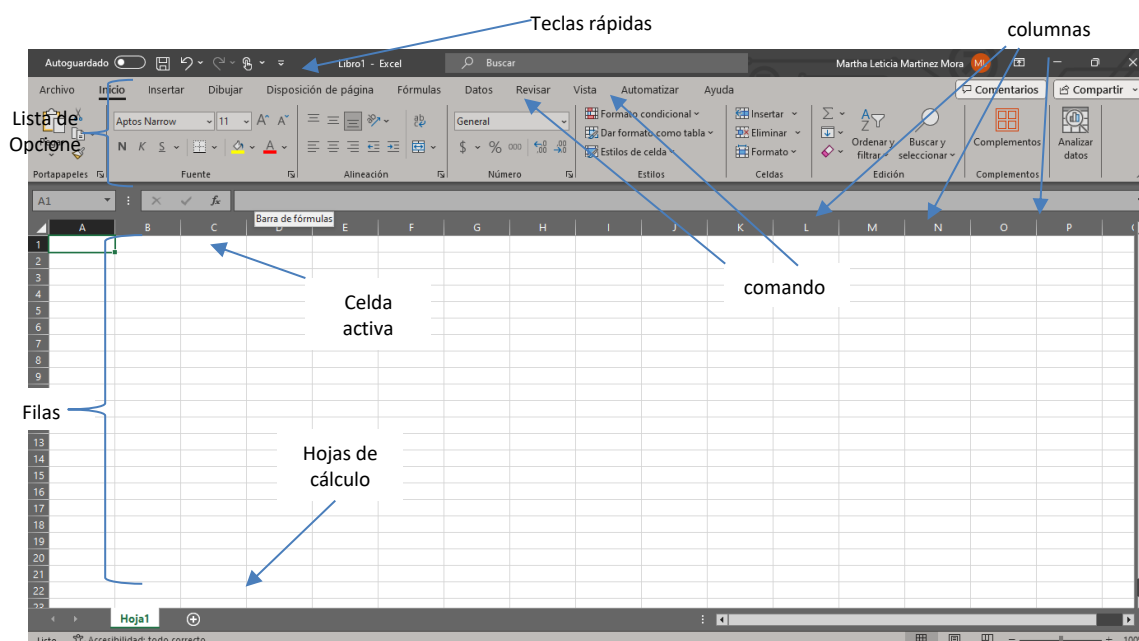


Figura 1.10. Hoja de cálculo de un libro de Excel

Una hoja de cálculo contiene miles de filas y columnas y puede desplazarse utilizando las teclas de dirección, las barras de desplazamiento, el ratón o las teclas de control. (Figura 1.11)



Figura 1.11 Teclas de control para navegar en el entorno

Al introducir datos en una hoja de cálculo, es necesario que considere que cada celda es un dato y que éstos pueden tener distinto tamaño, estilo y tipo. Por ejemplo, cuando capture datos de tipo texto más anchos que la celda, este se extenderá a

diferentes celdas, pero en cuanto presione ENTER o se mueva de celda, el texto excedente se ocultará o podrá ajustar el texto al ancho de una columna (en el tema de fórmulas se indica el proceso). También es importante notar que cuando se captura texto cuyos primeros caracteres coinciden con otra entrada de datos Excel introduce los valores siguientes de manera automática (Figura 1.12).

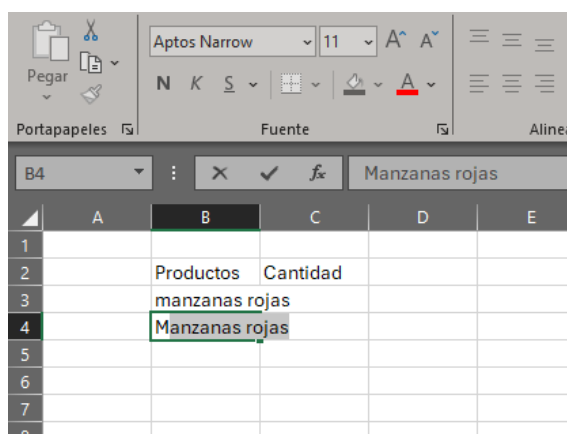


Figura 1.12 Texto desbordado y captura de valores automáticos

Para hacer cambios o corregir la información de una celda se debe realizar directamente en la celda dando doble clic o en la barra de fórmulas como se muestra en la figura 1.13

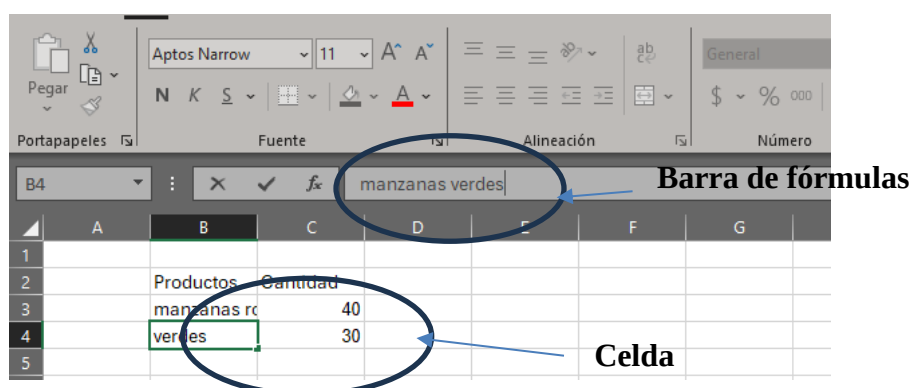


Figura 1.13. Actualizar y/o cambiar información en celda o barra de fórmulas



Cuando se tiene el modo edición el cursor se coloca en forma de barra y los demás comandos se inactivan (Figura 1.14 lado izquierdo). La tecla ENTER permitirá confirmar la acción de cambio, si por el contrario no se desea corregir la tecla ESC permite restablecer el texto antes de cambiarlo y la tecla rápida de Deshacer situada en la parte superior izquierda de la pantalla restablecerá el texto que se haya eliminado anteriormente (Figura 1.14 central). Hay que recordar que la función deshacer debe cuidarse ya que a diferencia de otros entornos no conserva muchos cambios (Figura 1.14 lado derecho).

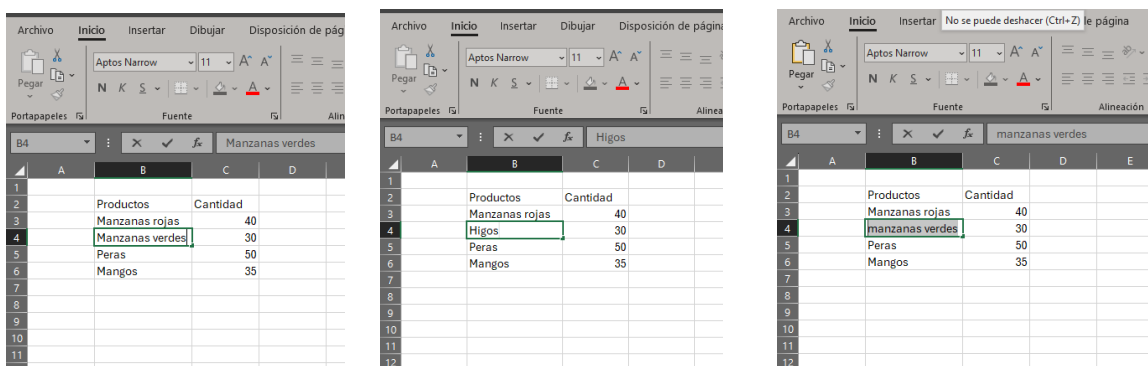


Figura 1.14. Seleccionar, Corregir o Quitar contenido de una celda

Copiar y agregar datos

Existen teclas rápidas para realizar la acción de copiar y pegar: Ctrl+C para copiar y Ctrl+V para pegar el último dato copiado o cortado. El portapapeles de Excel permite insertar elementos seleccionados en una nueva ubicación, para ello es necesario realizar las siguientes acciones:

1. Seleccionar la celda de donde se van a copiar los datos

2. Colocar el cursor en el comando copiar del portapapeles, observe que el borde de la celda seleccionada parpadea

3. Seleccionar la celda donde se colocarán los datos a copiar

Las anteriores acciones se muestran en la figura 1.15. Hay que recordar que cuando se copia un rango de celdas, se debe seleccionar un rango de igual tamaño donde se desee insertar.

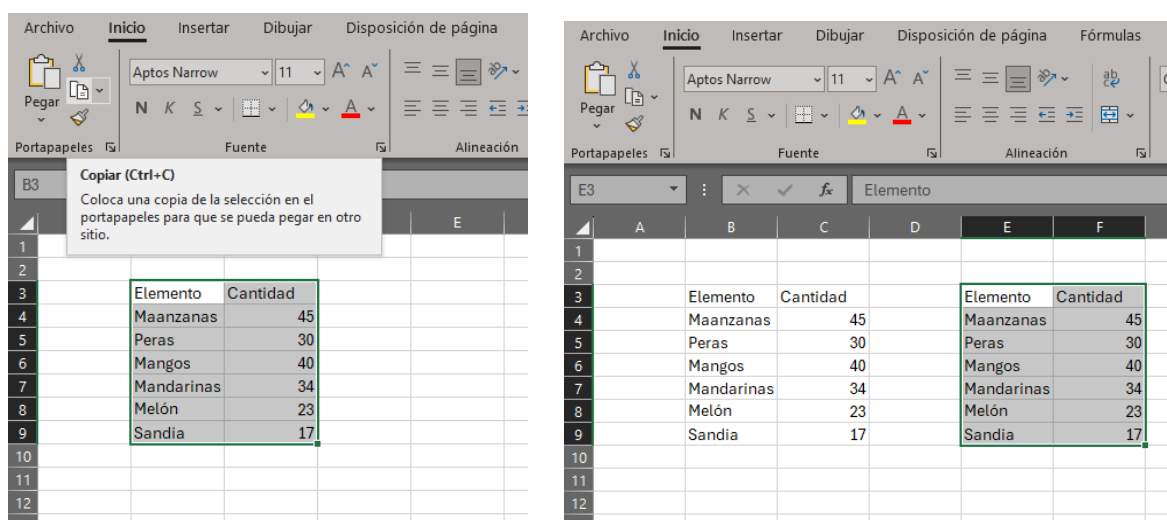


Figura 1.15. Copiar y pegar

La diferencia entre copiar y cortar es que los datos copiados permanecen en su ubicación original y los datos cortados solo permanecen en la celda destino cuando se usa el comando pegar, de lo contrario se eliminan de su ubicación destino.

Mientras siga parpadeando el borde de celdas seleccionadas en la acción copiar se puede pegar las veces que lo desee, para cerrar el proceso basta con realizar otra acción o presionar la tecla ESC.

Insertar celdas

Cuando se captura información en una hoja de cálculo se hace necesario en algunas ocasiones insertar celdas, para ello el comando insertar permite agregar celdas, filas, columnas entre otros. Para ello seleccione el comando Insertar con el botón derecho. El menú desplegable permite insertar celdas o filas o columnas y una vez seleccionada una actividad se realizará la acción seleccionada como se muestra en la figura 1.17.

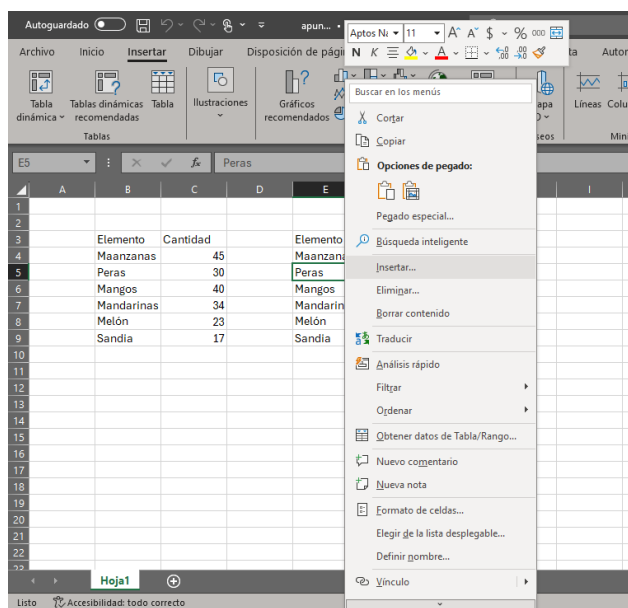


Figura 1.17. Menú de opciones con botón derecho

Insertar celdas puede ser desplazando los datos a la derecha o hacia abajo como se muestra en la figura 1.18.a y 1.18.b

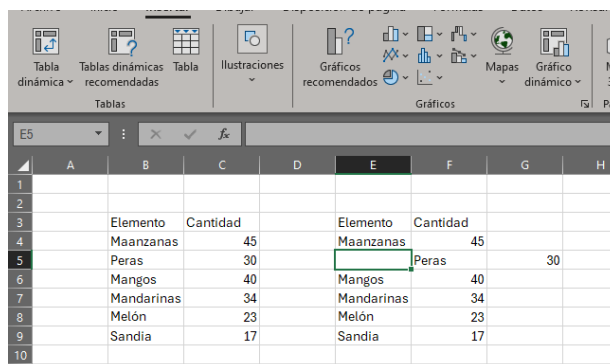
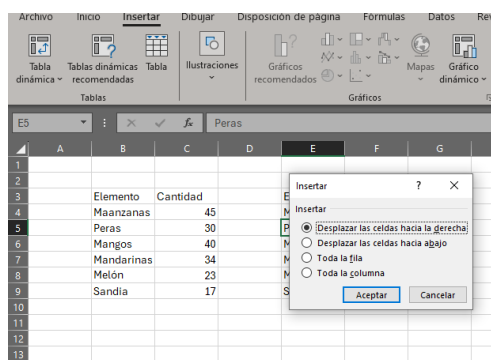


Figura 1. 18.a. Insertar celda a la derecha

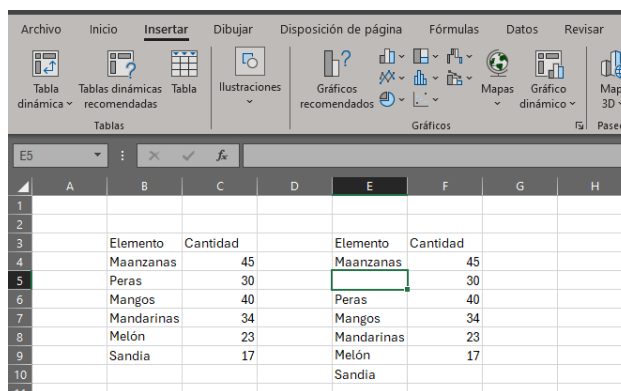
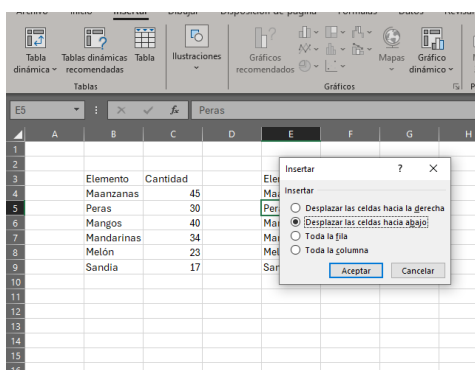


Figura 1.18.b. Insertar celda desplazando datos hacia abajo

Como puede observar el menú permite también insertar toda una fila o columna según lo que se desea insertar. Al seleccionar fila está se inserta en la parte inferior y para la columna se inserta a la izquierda de la celda seleccionada. (Figura 1.19)

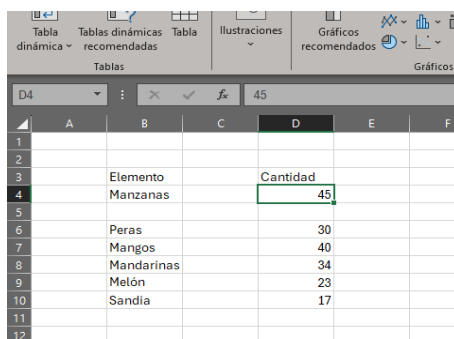


Figura 1.19 Insertar fila o columna a partir de una celda seleccionada

Otra forma de insertar filas y/o columnas es colocando el cursor en la barra correspondiente seleccionar una o varias y con el botón derecho del ratón seleccionar la forma y modo de insertar deseado, como se observa en la figura 1.20.

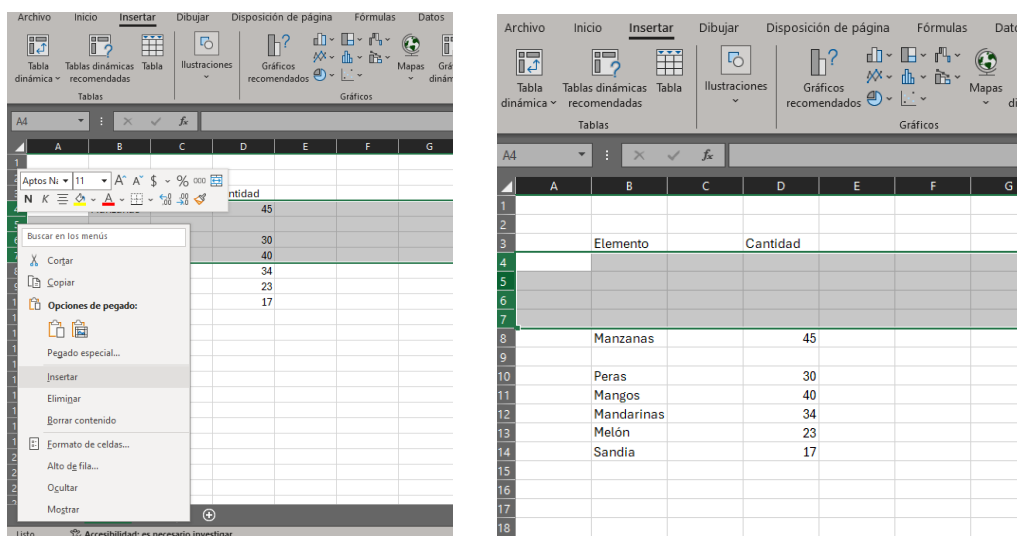


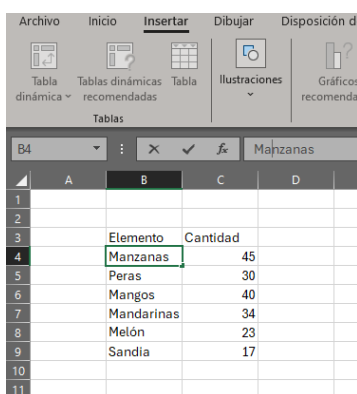
Figura 1.20 Insertar varias filas/columnas

Ejercicio de aprendizaje

1. Crea un libro y guárdalo con el nombre de Prácticas de Excel
2. Captura 10 artículos en una hoja de cálculo en la columna B, a partir de la fila 1
3. Inserta un rango de celdas y elige en qué dirección deseas insertarlas y menciona un ejemplo en el cuál esta acción aplica
4. Copia los datos en la misma hoja de cálculo, pero en una columna diferente
5. Anota tus observaciones de los posibles errores o aciertos cometidos durante el ejercicio

Formato de celdas

Alineación de datos. Es utilizado para colocar el contenido de una celda con una posición determinada. Cuando se insertan datos alfanuméricos de manera predeterminada Excel alinea a la izquierda la información, por el contrario, si lo que se introduce es un número el contenido se alinea a la derecha (Figura 1.21).

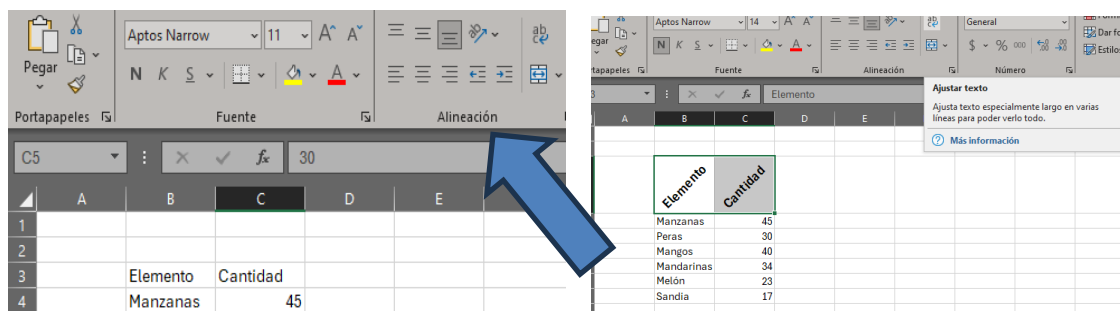


La imagen muestra una hoja de cálculo de Excel con la siguiente estructura de datos:

	A	B	C	D
1				
2				
3		Elemento	Cantidad	
4		Manzanas	45	
5		Peras	30	
6		Mangos	40	
7		Mandarinas	34	
8		Melón	23	
9		Sandia	17	
10				
11				

Figura 1.21. Alinear datos de manera predeterminada

Sin embargo, si se desea otro tipo de alineación basta con seleccionar las opciones que ofrece la barra de alineación y ajustar el contenido de una celda de acuerdo con las necesidades del usuario, como se muestra en la figura 1.22



La imagen muestra la barra de herramientas de Excel, específicamente la pestaña de Alineación. Se ha seleccionado la opción de 'Ajustar texto' (Wrap Text), lo que permite que el contenido de la celda se ajuste a la altura de la fila. La barra de fórmulas muestra el valor '30' en la celda C5. La siguiente tabla representa el contenido de la hoja de cálculo:

	A	B	C	D	E
1					
2					
3		Elemento	Cantidad		
4		Manzanas	45		

Figura 1.22 Alinear el contenido de una celda

La barra de alineación también ofrece combinar y centrar celdas, ajustar texto entre otras acciones. Ver figura 1.23

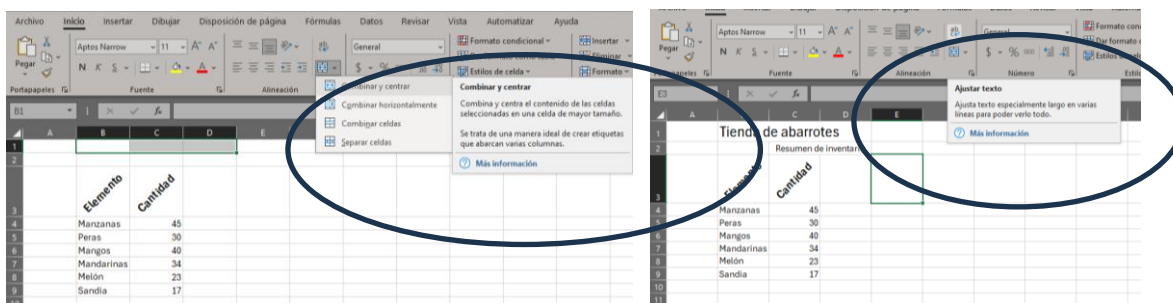


Figura 1.23. Combinar y centrar

Otra forma para aquellos que les gusta programar sería ingresar a la hoja de cálculo donde se captura información y seguir las siguientes instrucciones:

1. Posicionar el cursor en la parte inferior donde aparece el nombre de la hoja y oprimir el botón derecho del ratón, para seleccionar la pestaña: **ver código**. (Figura 1.23.a)

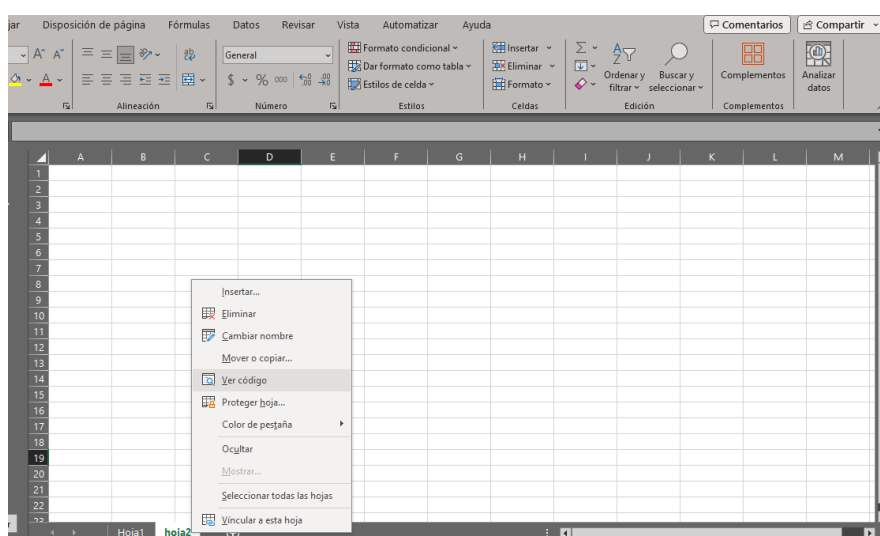


Figura 1.23.a Generando código



2. Seleccionar Worksheet (Figura 1.23.b)

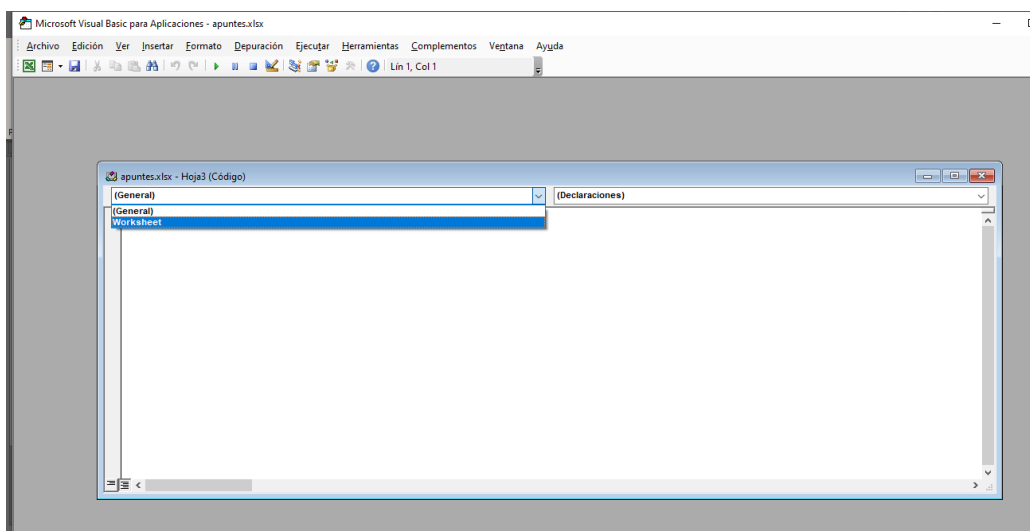


Figura 1.23.b. Editor Worksheet

3. Capturar la siguiente línea de código: ***cells.EntireColumn.autocomplete*** como se muestra en la figura 1.23.c y 1.23.d

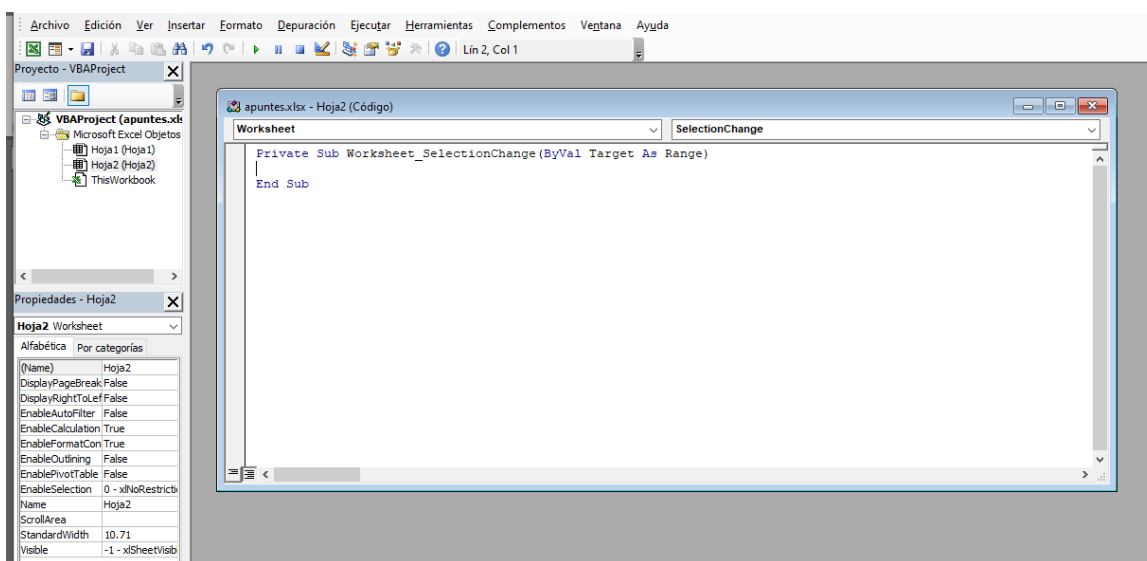


Figura 1.23.c. Edición de código para ajustar celdas al tamaño del texto

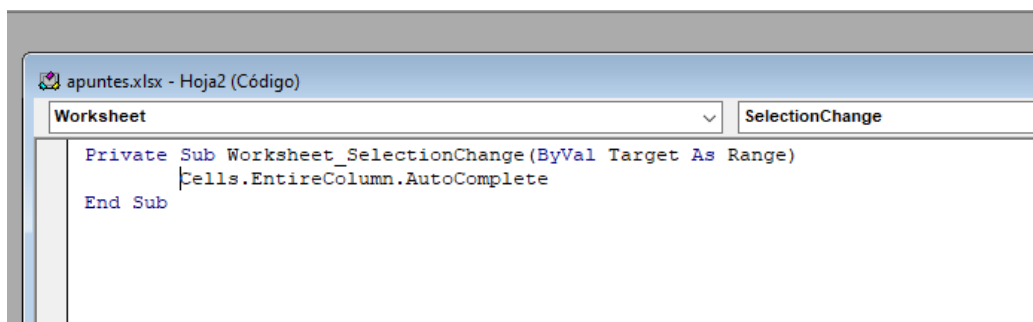


Figura 1.23.d Editor de código en worksheet

Al capturar datos en la columna, ésta se ajustará de manera automática al tamaño del texto capturado como se muestra en la figura 1.23.e

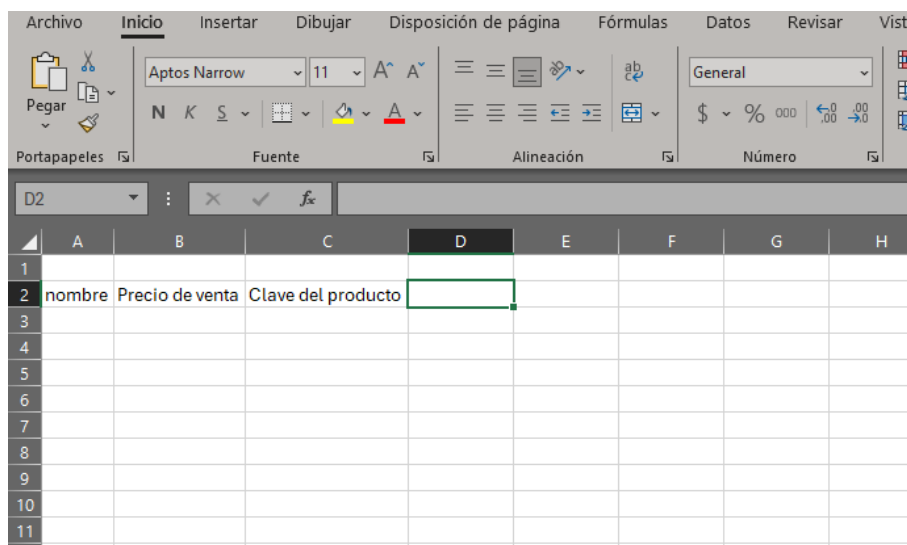


Figura 1.23.d. Ajustar contenido de celda de acuerdo con el texto capturado

Fuente

Una fuente, es el conjunto de características que puede tener un texto y con ello determinar el estilo o aspecto que se le puede brindar al contenido de una celda. Modificar el estilo, tamaño o tipo de la fuente solo seleccione el cuadro **Fuente** y elija la opción más conveniente (Figura 1.24).

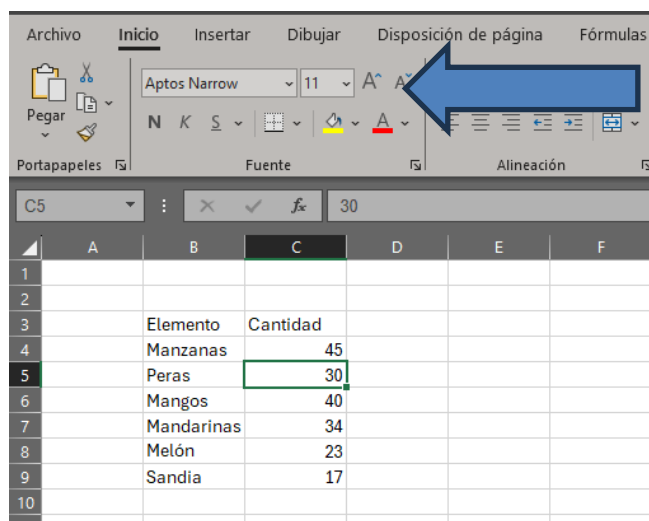


Figura 1.24. Elegir tipo de fuentes y tamaño

Adicionalmente, esta barra de Fuentes permite cambiar el color, el tipo de fuente, el estilo y el tamaño, como se observa en la figura 1.25

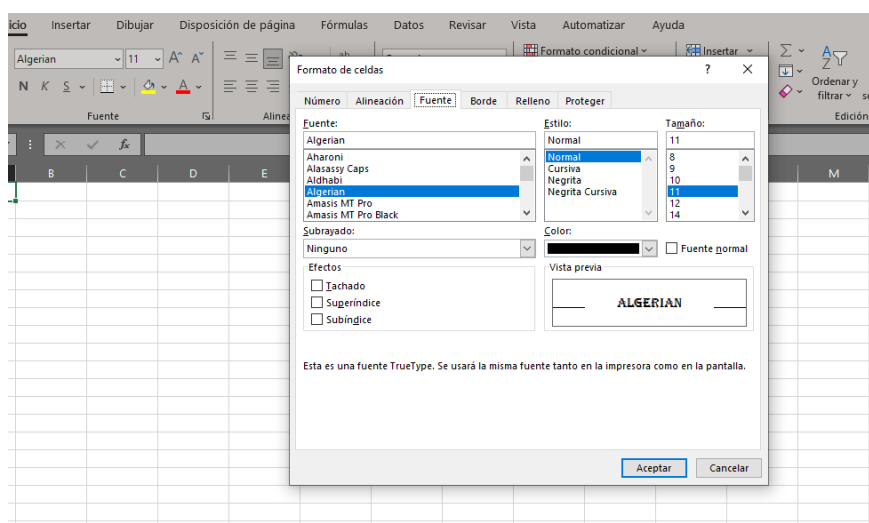


Figura 1.25. Estilos y tipos de fuente



Una **función** es un comando predefinido por el entorno que requiere de un argumento (valor o valores), realiza una operación y regresa un resultado.

Una **referencia** identifica a una celda o un rango de celdas de una determinada hoja de cálculo y le indica al entorno donde buscar los valores que se desean utilizar en una fórmula o en varias. Cabe destacar que también se pueden emplear celdas de otras hojas de cálculo del mismo libro o de otros libros. (Mauricio, Excel 2013, 2013)

Las **fórmulas** son argumentos que pueden contener referencias a celdas, referencia a rango de celdas, operadores y valores o constantes, trabajan con operadores que le indican al entorno la operación a realizar y se distinguen porque deben iniciar con el signo =

Sintaxis de uso:

= valor operador valor

Un ejemplo muy sencillo es sumar dos números. Observe que en la figura 1.26, de lado izquierdo, se introduce la fórmula iniciando con el símbolo =, y en la figura 1.26 de la derecha se refleja el resultado, el cual se produce con el simple hecho de oprimir la tecla <enter> al finalizar la fórmula.

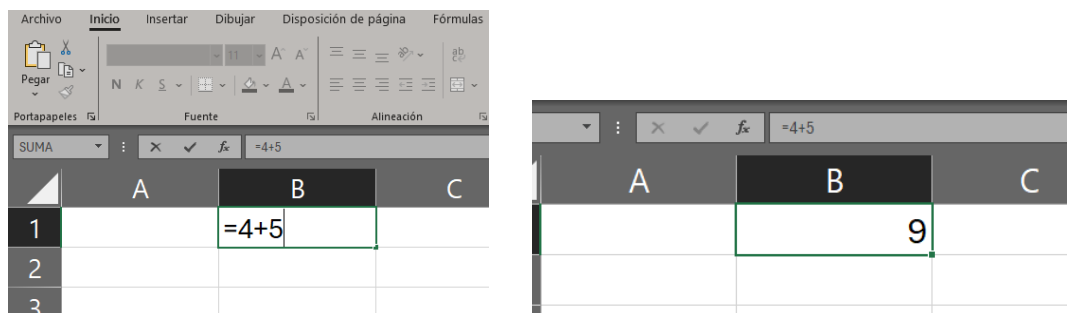


Figura 1.26. Uso de fórmulas en Excel

Si la fórmula utiliza referencias de celda, entonces en lugar del valor debe utilizar la celda donde se almacena dicho valor colocando el símbolo = y seleccionando cada celda y el tipo de operador que corresponda a la operación a realizar, como se muestra en la figura 1.27. Observe que en la barra de fórmulas se plasma la fórmula y en la celda el resultado correspondiente a ella.

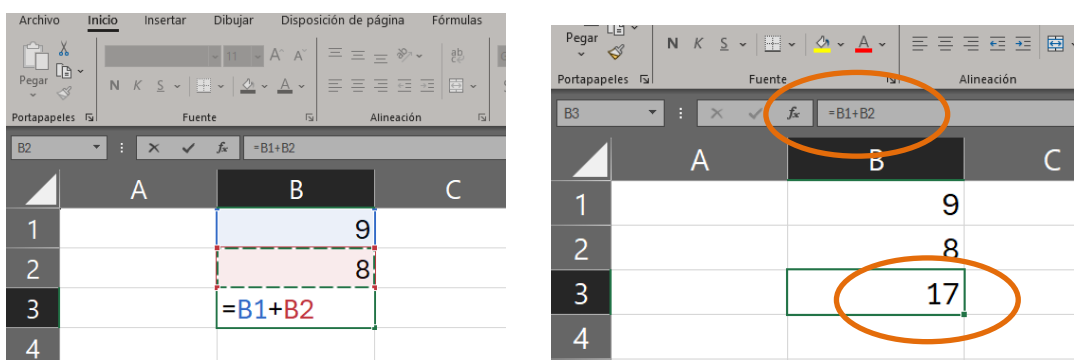


Figura 1.27 Uso de referencia de celda

Para editar una fórmula puede realizarse de dos formas: dando doble clic en la celda o colocar el cursor en la barra de fórmulas. Al utilizar referencias de celdas en lugar de valores tiene la ventaja de actualizar valores y en automático se actualiza la fórmula y permite que el entorno sea más dinámico y automatizado, por lo que se recomienda hacer uso de esta ventaja que ofrece el entorno.

Existen distintos tipos de referencias absolutas, relativas y mixtas:

- Referencias relativas. Al copiar una fórmula éstas se adaptan a los valores relativos a la celda en la que se copiaron
- Referencia absoluta. Indica al entorno que la fórmula al ser copiada en otra celda no debe cambiar a otras referencias, sino mantener su valor de la referencia seleccionada y para ello se utiliza el símbolo \$



- Referencia mixta. El uso de fórmulas con referencias absolutas de forma parcial, en este caso al copiar una referencia ésta se adapte a la columna y no a la fila o se actualice la fila y no la columna

Ejemplo de uso:

Se tienen una lista de productos y su precio unitario, se desea calcular el total sin IVA y con IVA de cada producto. En el caso del precio sin IVA se calcula multiplicando la columna de precio por la columna de cantidad para que el total de cada producto se calcule multiplicando la cantidad (C), por el precio (B), de cada uno de ellos tomando los valores correspondientes a las filas y columnas respectivas de cada producto, como se puede observar en la barra de fórmulas marcada en la figura 1.28. Esto se conoce como referencia relativa.

	A	B	C	D	E
1	Artículo	Cantidad	Precio	Total sin IVA	Total con IVA
2	Lápiz	300	\$ 5.00	\$ 1,500.00	
3	Bolígrafo	240	\$ 12.00	\$ 2,880.00	
4	Compás	180	\$ 35.00	\$ 6,300.00	
5	Libreta	70	\$ 65.00	\$ 4,550.00	
6	Block de notas	68	\$ 58.00	\$ 3,944.00	
7					

Figura 1.28 Referencia Relativa

Suponer que el IVA es un valor que se encuentra almacenado en alguna de las columnas o filas de la tabla y para calcular el IVA de cada producto se debe utilizar la referencia de celda del IVA, al copiar la fórmula de cada fila y columna correspondiente al producto, el resultado obtenido es incorrecto, como se observa en la figura 1.29, porque solo reconoce los valores de la celda donde ésta almacenado el valor del IVA.

Artículo	Cantidad	Precio	Total sin IVA	IVA %	IVA
Lápiz	300	\$ 5.00	\$ 1,500.00	240.00	0.16
Bolígrafo	240	\$ 12.00	\$ 2,880.00	-	-
Compás	180	\$ 35.00	\$ 6,300.00	-	-
Libreta	70	\$ 65.00	\$ 4,550.00	-	-
Block de notas	68	\$ 58.00	\$ 3,944.00	-	-

Figura 1.29 Uso de referencias relativas incorrectas

Para corregir este error, es necesario emplear referencia absoluta, para que el valor no cambie si es copiado a otras celdas. Para realizar esta acción es necesario colocar el símbolo \$ y la referencia de celda. Como se observa en la figura 1.30, si se utiliza referencia relativa como se describió anteriormente, las celdas posteriores no poseen valores y calcula el entorno valores erróneos. Por el contrario, si se realiza una referencia absoluta, el valor de la celda permanece fijo en las siguientes referencias calculadas.

Artículo	Cantidad	Precio	Total sin IVA	IVA %	IVA
Lápiz	300	\$ 5.00	\$ 1,500.00	240.00	0.16
Bolígrafo	240	\$ 12.00	\$ 2,880.00	460.80	-
Compás	180	\$ 35.00	\$ 6,300.00	1,008.00	-
Libreta	70	\$ 65.00	\$ 4,550.00	728.00	-
Block de notas	68	\$ 58.00	\$ 3,944.00	631.04	-

Figura 1.30. Referencias absolutas

Insertar funciones a las fórmulas

La principal ventaja de utilizar las funciones que utiliza el entorno de Excel es agilizar las operaciones, y simplificar la fórmula. Por ejemplo, el uso de la función SUMA, establece que debe utilizarse un rango de valores con los que se realizará la

operación matemática y su uso es el siguiente: = SUMA (rango de valores), Suma realiza la operación de sumar los valores almacenados en el rango de celdas contempladas. Para esta acción los pasos son:

1. Colocar el cursor en una celda vacía
2. Teclear **=SUMA(** y seleccionar el rango de celdas a sumar)
3. Oprimir la tecla ENTER

La figura 1.31 muestra del lado izquierdo el rango de valores sumados y el resultado obtenido, del lado izquierdo la barra de fórmulas lo que se muestra, es la función y sus argumentos.

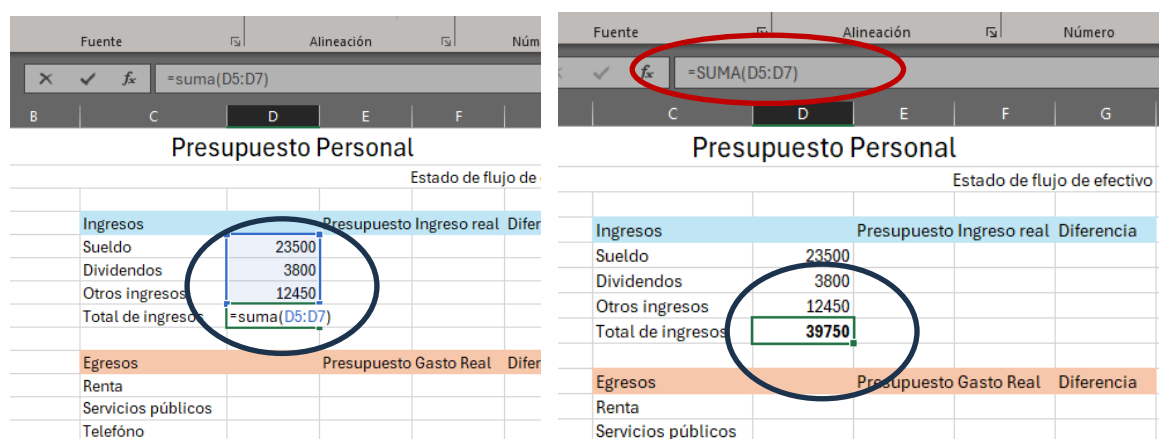


Figura 1.31. Uso de la función suma en Excel

Existen varias funciones en Excel y en general su uso establece dos reglas básicas:

1. Iniciar con el símbolo =
2. Respetar la sintaxis de uso que establece la función

Ejemplos de uso de la función **promedio y contar**. La primera función, promedio, obtiene la suma del conjunto de números y el resultado de dividir la sumatoria obtenida entre el total de números (ver figura 1.32 lado izquierdo). La segunda

función contar, como su nombre lo indica, cuenta la cantidad de números seleccionados y muestra el resultado obtenido (ver figura 1.32 lado derecho).

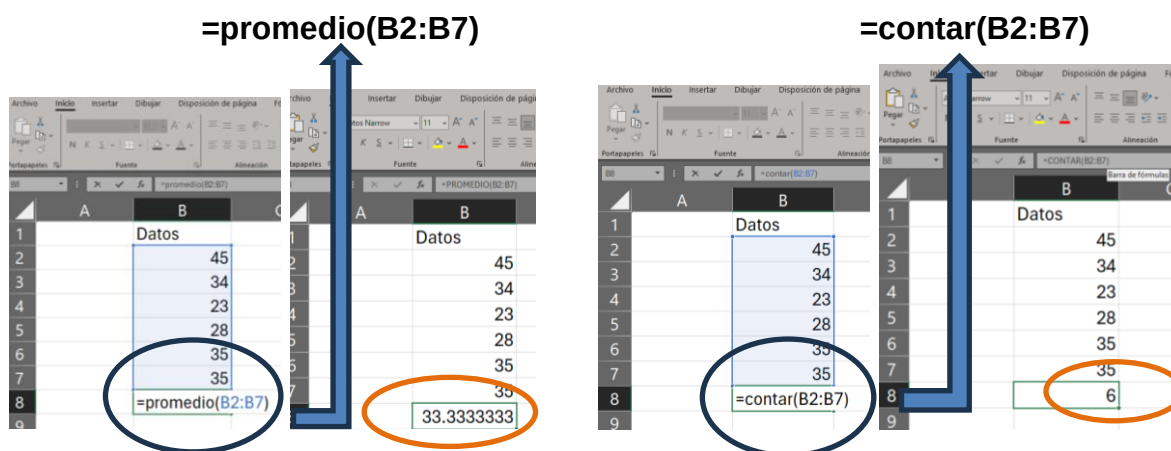


Figura 1.32 Ejemplos de fórmulas con funciones.

Es muy significativo conocer y utilizar la gama de funciones que ofrece Excel y su uso, este entorno cuenta con funciones trigonométricas, de texto, matemáticas, de base de datos, estadísticas, financieras y avanzadas, por lo que sería imposible mostrar todas y cada una de ellas en un solo documento ya que la gran variedad de fórmulas que el usuario del entorno puede crear, esta en función de sus necesidades laborales, personales o creativas, por ello es que a continuación, se muestra en la figura 1.33, un mapa conceptual de la clasificación de funciones y ejemplos de uso de algunas de ellas y aunque más adelante se muestran ejemplos seleccionados, será importante que el lector continúe explorando el alcance de ésta poderosa herramienta.

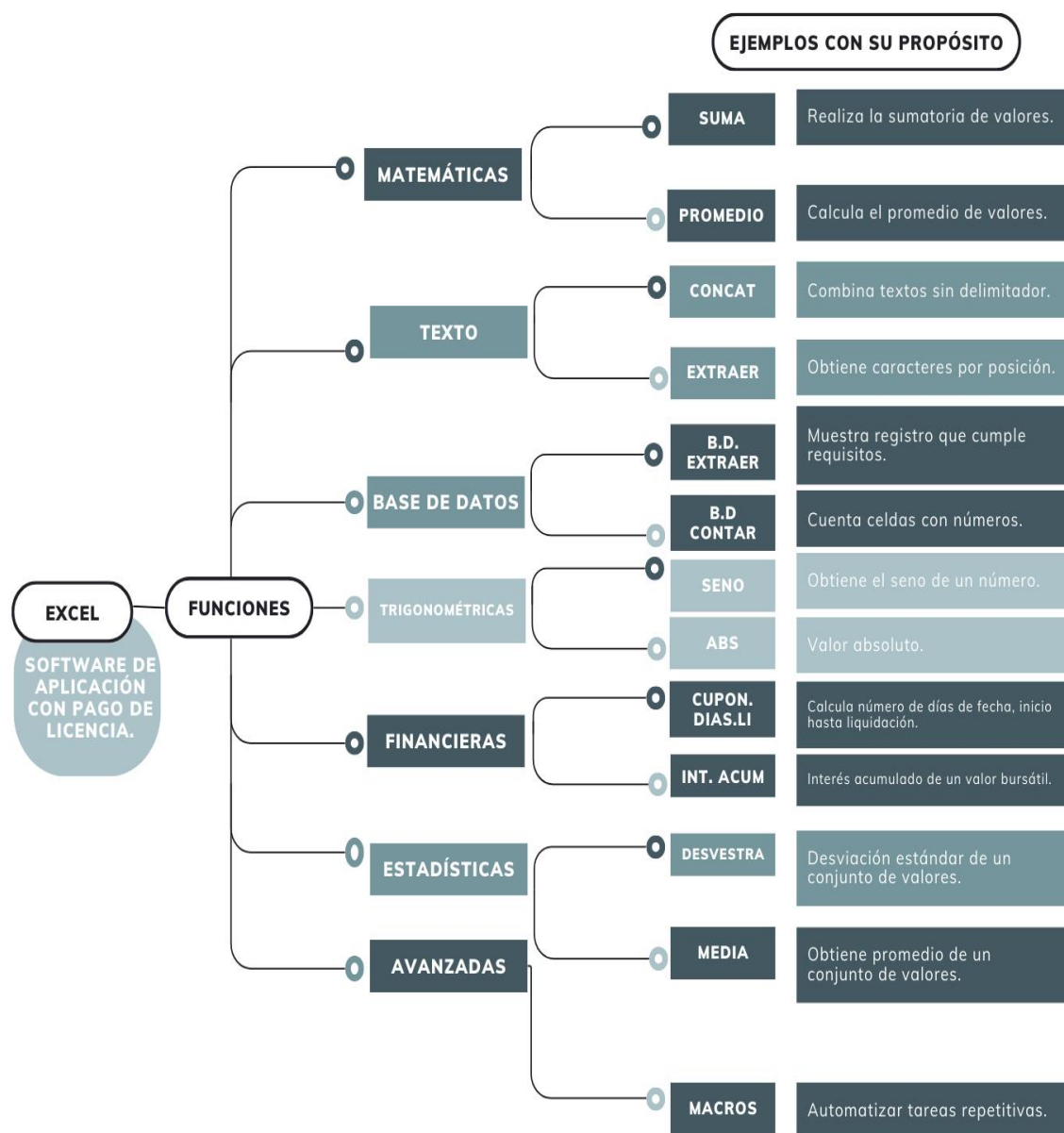


Figura 1.33. Clasificación de funciones de Excel

Fuente: Entorno de Excel 2013 (Microsoft, Microsoft, 2022)



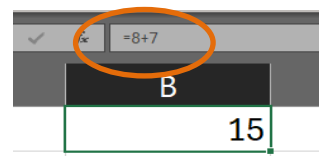
La tabla 1.3, muestra algunas funciones y describe la operación que realiza, su sintaxis de uso y un ejemplo con el resultado obtenido usando el entorno.

Tabla 1.3 ejemplo de funciones del entorno de Excel

Función	Descripción	Sintaxis	Ejemplo	Resultado
Contar.si	Cuenta el número de celdas de un rango determinado que cumplen con una condición establecida	contar.si(rango, Criterio) Donde: Rango: son las celdas para contar Criterio: es la condición que cumplir, la cual se coloca entre paréntesis.	Contar cuántos números son mayores a 400	
Promedio	Suma un número de celdas y divide el resultado entre el total de celdas	promedio(rango)	Obtener el promedio de una lista de 10 números	
Max	Obtiene de una lista de datos el máximo valor	Max(rango)	Obtener el valor máximo	
seno	Obtiene el seno de un ángulo determinado	Seno(x)	Obtener el seno de un determinado ángulo	



En resumen. Las fórmulas pueden contener, funciones, referencias a rangos de celdas, operadores y números, ejemplo de ello sería:

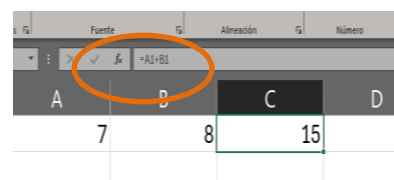


=A1+A2 Sumar Referencia a valores de celda

=B4+45 Sumar referencia a valor de celda y un valor constante

=23+56 Sumar valores constantes

=SUMA(C1:C5) en este ejemplo SUMA es una función predeterminada de Excel, que calcula los valores seleccionados y devuelve un resultado



Excel cuenta con más de 400 funciones que pueden ser utilizadas acorde a los fines que se desean obtener. En el entorno existen varias formas de realizar operaciones por ejemplo sumar valores

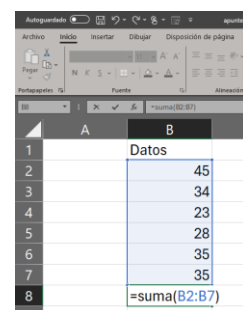
1. Usando el operador + y los valores a sumar:

Pasos:

Ingresar el símbolo = y a continuación los valores constantes a sumar.

Ejemplo:

=8+ 7 y el valor de la suma se refleja en la celda seleccionada



2. Sumar referencias de celdas, por ejemplo:

=A1+B1 y en este caso se realizará la suma de los valores almacenados en las celdas referidas

3. Cuando se requiere sumar una lista de valores de una misma columna o fila se utiliza la selección de celdas mismas que se clasificarán con el símbolo “ : ”

Sintaxis:

=SUMA (Argumento:Valores del rango de celdas a sumar)

=SUMA (B2:B7)



4, Si se desea sumar diferentes valores en celdas dispersas, listas y valores constantes o fijos, es necesario emplear el símbolo “ ; ”, para separar cada elemento a sumar

Sintaxis:

=SUMA (valor de una celda; Valores de celdas contiguas; Valor Fijo)

Uso:

=SUMA(B2:B7, 30)

Nota: Aunque es posible colocar números en la fórmula de suma, no es recomendable porque este valor puede cambiar, en estos casos se recomienda colocar dicho número en una celda independiente y cambiarlo cuando se haga necesario.

5. Sumar valores tomando en consideración una condición se emplea la siguiente fórmula:

Sintaxis:

=SUMAR.SI(rango de valores; "condición"

Uso:

=SUMAR.SI(B2:B7;">30") solo sumará los valores: 45, 34, 35 y 35, del conjunto de datos seleccionados

A	B	C
	Datos	
	45	
	34	
	23	
	28	
	35	
	35	
	149	

A	B	C
	Datos	
	45	
	34	
	23	
	28	
	35	
	35	
	350	

Ejercicio de aprendizaje

Capturar el siguiente conjunto de números (187, 456, 345,654, 45, 65, 34. 12, 34, 56, 34, 78, 32, 9, 55) en la columna desde B1 a B15 y sumar los valores utilizando el operador + y suma, almacenar resultados en la celda C1 con el operador y con la función suma en la celda C2. Explica cuál es la diferencia y anota tus conclusiones

Función Rellenar

En Excel se pueden realizar series de manera automática, en función de los valores que se desean generar. Para ello es necesario capturar los primeros valores de la serie, seleccionar las celdas y posicionar el cursor en la esquina inferior de las celdas como se muestra en la figura 1.34 de la izquierda y posteriormente debe arrastrar hacia abajo y esto realiza el rellenar celdas automáticamente, como se muestra en la figura 1.34 de la derecha.

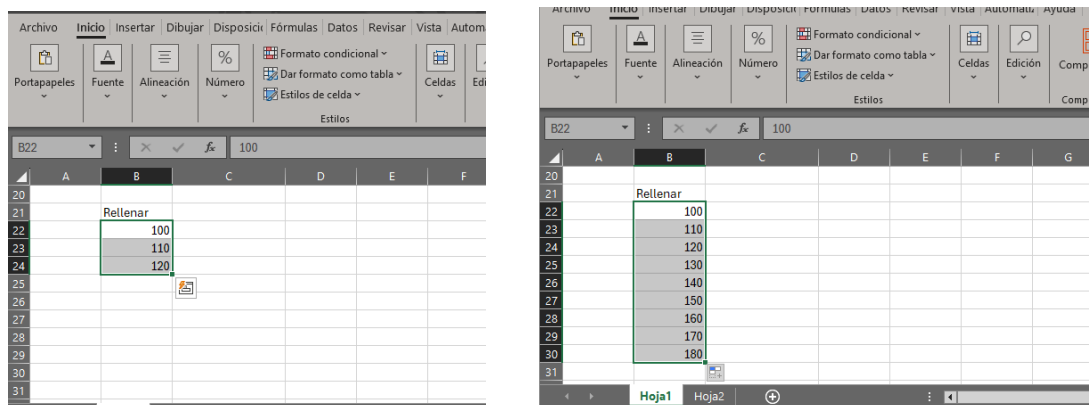


Figura 1.34. Función rellenar

Función Transponer

Es una función útil para invertir columnas por filas o viceversa. Para utilizarla es necesario tomar en cuenta que el número de filas y columnas deben ser equivalentes ya que la función trabaja con valores en forma de matriz. En el ejemplo siguiente se usará el cambio de 2 filas y 5 columnas a 5 filas y 2 columnas. Para ello siga los siguientes pasos:

1. Seleccionar las celdas correspondientes al número de filas y columnas que se desean transponer, en este caso se seleccionan 2 columnas y 5 filas. Ver figura 1.35.a

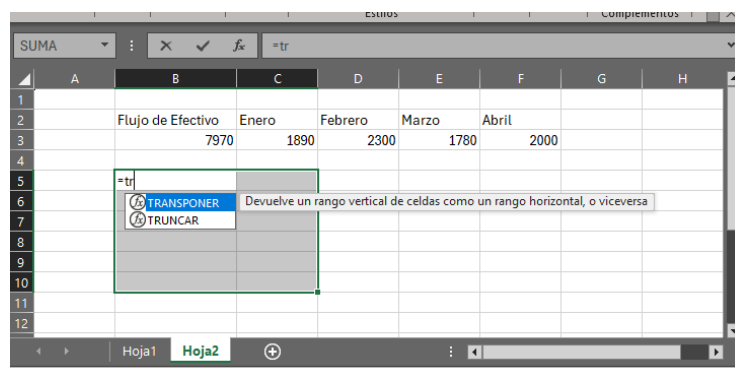


Figura 1. 35.a. Uso de la Función trasponer

2. En las celdas seleccionadas escribir **=transponer(** y seleccionar las celdas que se desean invertir) pero en lugar de oprimir la tecla <Enter>, debe presionar las teclas <Ctrl> <Mayus> <Enter> simultáneamente (Figura 1.35.b)

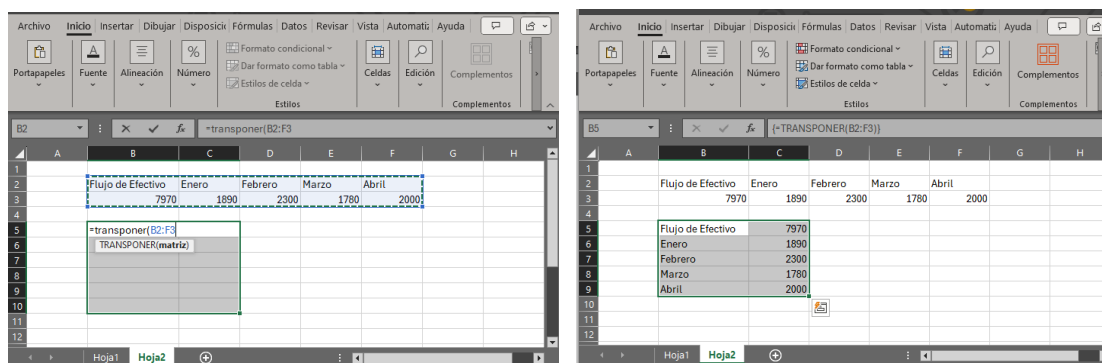


Figura 1. 35.b. Uso de la Función trasponer



Si se posiciona en cualquiera de las celdas transpuestas puede observar en la barra de fórmulas que la fórmula que aparece es (=transponer (B2:F3)) es la misma, esto se debe a que se trata de una fórmula matriz (Figura 1.35.c)

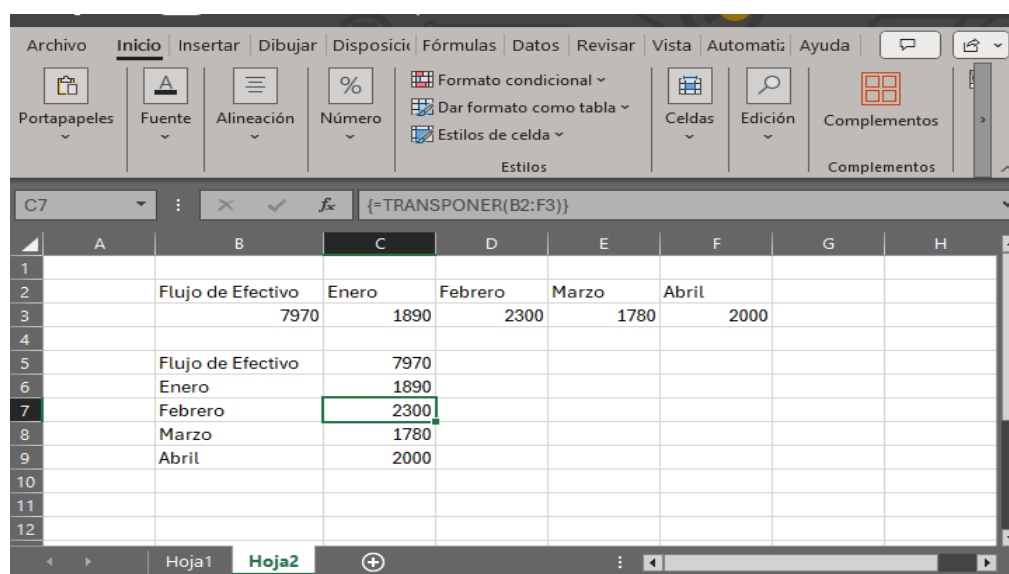


Figura 1. 35.c. Uso de la Función transponer

Una fórmula matriz, no puede ser corregida directamente, es decir, no se puede eliminar filas o columnas de la matriz, si fuera el caso, entonces hay que eliminar dicha matriz realizar los cambios y volver a crear la fórmula.

Ordenar y filtrar datos

Ordenar datos de manera alfabética, se realiza seleccionando los datos a ordenar y seleccionar de la barra de funciones la selección requerida, por ejemplo, ordenar de A a Z, de mayor a menor. En la figura 1.36 se observa la selección de datos a ordenar, el tipo de ordenamiento solicitado y al oprimir la tecla <Enter> el ordenamiento correspondiente.

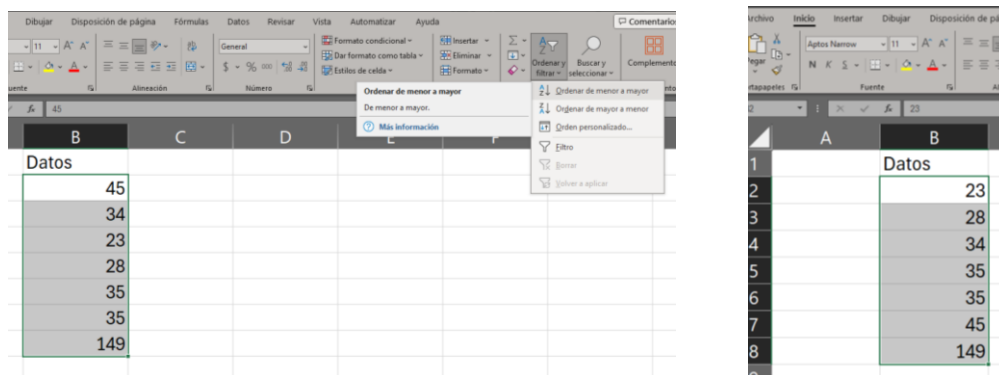


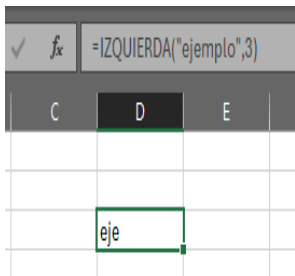
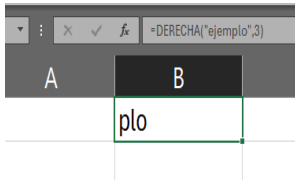
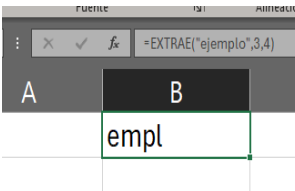
Figura 1.36. Ordenar datos

Las funciones más empleadas son las matemáticas, financieras y trigonométricas, sin embargo, existen funciones de texto muy útiles ya que permiten entre otras acciones: concatenar, hallar, extraer y contar caracteres de una celda. Por ejemplo, la función **Hallar**, busca una cadena de texto y devuelve el número de la posición inicial. Si utilizamos la siguiente acción: **=Hallar("medio"; "promedio"**, regresa un 4 porque la palabra medio inicia en la posición 4, en este ejemplo es importante tomar nota del alcance de la función, es decir, la función Hallar no distingue entre mayúsculas y minúsculas por lo que es una limitante cuando el texto las contiene, para ello se debe utilizar la función encontrar, que si lo hace. (Microsoft, 2022)

Algunas funciones de texto se muestran en la tabla 1.1, donde se plasma: el nombre de la función su descripción, sintaxis de uso, un ejemplo y el resultado obtenido utilizando el entorno de Excel. Cabe mencionar que solo se muestran las más utilizadas, pero para obtener la lista más completa puede visitar el sitio: <https://support.microsoft.com>



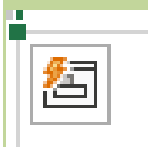
Tabla 1.1. Funciones de texto en Excel

Función	Sintaxis	Ejemplo	Resultado
Izquierda	Muestra el número de caracteres de la izquierda, marcados por la variable Numcar del texto ingresado =izquierda("texto",NumCar) Si se ingresa el texto almacenado en una celda, no se coloca entre comillas	=izquierda("Ejemplo",3) La función muestra los caracteres solicitados	
Derecha	Muestra el número de caracteres de la derecha, marcados por la variable Numcar del texto ingresado =derecha("texto", numcar)	=derecha("ejemplo",3)	
Extrae	Muestra el número de caracteres (numcar), contados de izquierda a derecha, del valor inicial (numinic) del texto ingresado =extrae("texto", numinic, numcar)	=extrae("ejemplo", 3,4) Extrae 4 caracteres a partir del 3o	
Mayúsc	Convierte en mayúsculas un texto	Mayusc(texto)	



Ejercicio de aprendizaje

1. Ingresa un conjunto de 10 números. Selecciona las celdas de los valores a sumar y busca en la esquina inferior derecha de la ventana de EXCEL, el símbolo mostrado y explica los comandos y/o funciones que ofrece



2. Para llevar la contabilidad de un negocio es importante registrar los ingresos, los gastos y los insumos para poder calcular las ganancias obtenidas por día, este reporte se conoce como flujo de efectivo y se puede realizar por día, por semana o por mes. Considere que los datos de la tabla y elabore en una hoja de cálculo que permita calcular el flujo de efectivo del mes indicado

Mes:Junio	Concepto	Semana 1	Semana 2	Semana 3	Semana 4
Ingresos	Ventas	22,000	25.690	14.500	28,450
	Ventas a crédito	12,000	11,000	0	0
	Efectivo de la semana anterior (Se toma el valor obtenido del flujo de efectivo anterior al solicitado)	6,400			
	Total ingresos (Sumatoria de ingresos por semana)				
Egresos o gastos	Sueldos	0	22,000	0	22,000
	Materia prima	4,500	4,500	4,500	4500
	Servicios	0	0	0	12000
	Administrativos	9500	9500	9500	9500
	Total egresos o gastos (sumatoria de los gastos)				
	Flujo de efectivo (Total de ingresos-total de egreso)				



1.5 Macros

Existen actividades o tareas que en el desarrollo de un proceso se realizan de manera repetitiva, una **macro**, es considerada un conjunto de instrucciones o comandos que pueden realizarse las veces que se requieran y con ello facilite y automatice los pasos para completar la acción o acciones requeridas. (Corona, 2020)

Una **macro** almacena una serie de comandos para ser ejecutados cuando se necesiten, una ventaja que se logra con su uso es agilizar procesos y el flujo de trabajo de una tarea repetitiva o que se realiza muchas veces. Una **macro** se puede ejecutar desde un botón de la barra de herramientas de acceso rápido o mediante una combinación de teclas, se recomienda la primera forma dado que el entorno maneja muchas teclas de control y puede generar confusión si alguna combinación se repite (Flores, 2013). Para crear una **macro** es necesario seguir los siguientes pasos: Los pasos básicos para generar una macro son:

1. En el menú de opciones de la parte superior buscar el comando Macro y seleccionar la acción de grabar
2. Colocar el nombre con el cuál se almacenarán los comandos
3. A partir de este momento los comandos o teclas que presione serán consideradas las actividades que la macro realizará y siempre y cuando sean introducidas con el teclado como dispositivo de entrada, ya que, si se seleccionan con el ratón, estas acciones no se grabarán.
4. Para finalizar la macro, hay que oprimir el comando de detener grabación
5. Para ejecutar la macro basta con ver la lista de macros y seleccionar la que se desea ejecutar y oprimir la tecla <Enter>



Ejemplo de uso

1. Para crear una **Macro**, es necesario contar con la pestaña de Desarrollador o programador según la versión de Excel que se tenga (Figura 1.37)

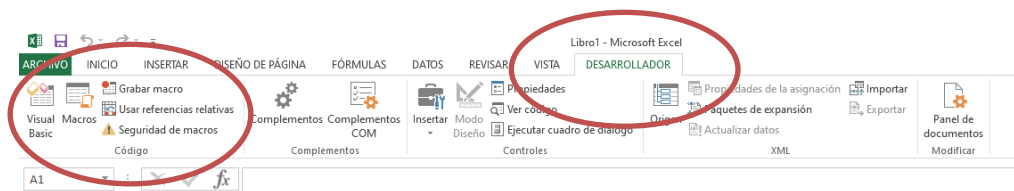


Figura 1.37. Comandos del Desarrollador

Si en la barra del menú superior no aparece el comando de Desarrollador, entonces es necesario habilitarla. Para ello se selecciona la opción de la barra de opciones del menú superior: **Archivo**, posteriormente **opciones**, personalizar barra de opciones y habilitar seleccionando el cuadro de la derecha donde figura la opción. (Figura 1.38).

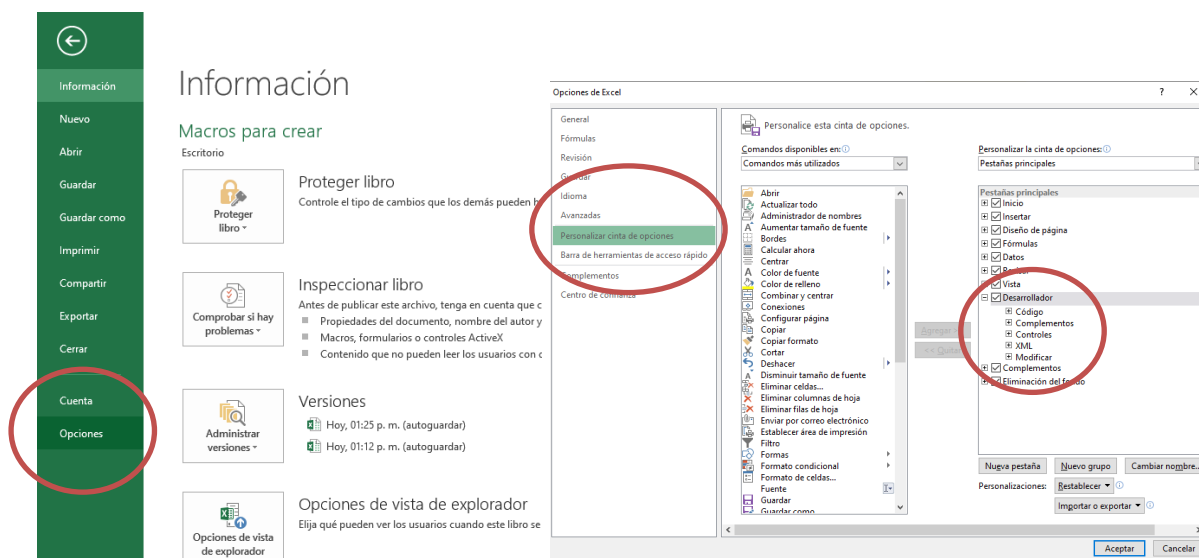


Figura 1.38. Habilitar la opción de Desarrollador

2. El libro de Excel debe estar habilitado para macros, para ello cuando se guarde debe seleccionarse la opción indicada (Figura 1.39)

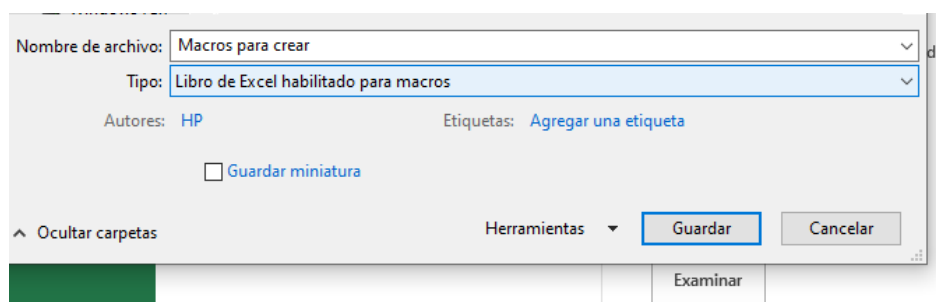
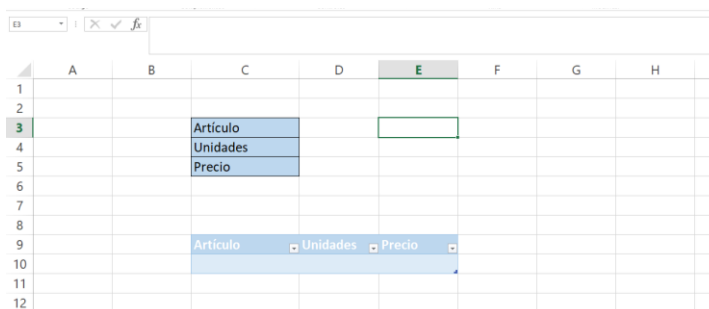


Figura 1.39. Guardar archivo habilitado para macros

3. Para crear la **Macro**, es necesario saber que instrucciones asignar en un proceso para realizar de manera automática dichas instrucciones. Un ejercicio para explicar cómo generar una Macro es el siguiente:

Un empleado de una tienda de abarrotes tiene que ingresar todos los días los artículos que va a vender, capturando el nombre del artículo, las unidades que se compraron y el precio de cada artículo, para ello tiene una lista de productos y aquellos productos que se ingresan por día deben almacenarse al inicio de la lista. Para realizar la colocar los encabezados: Artículo, Unidades y Precio en una fila para generar la lista y en una columna para realizar la captura, realizarlo como se muestra en la figura 1.40)



	A	B	C	D	E	F	G	H
1								
2								
3			Artículo					
4			Unidades					
5			Precio					
6								
7								
8								
9			Artículo	Unidades	Precio			
10								
11								
12								

Figura 1.40 Crear una Macro

La lista de productos debe ser una tabla por lo que selecciona la fila donde se almacenarán los datos, recuerde seleccionar encabezados antes de aceptar la acción (Figura 1.41).

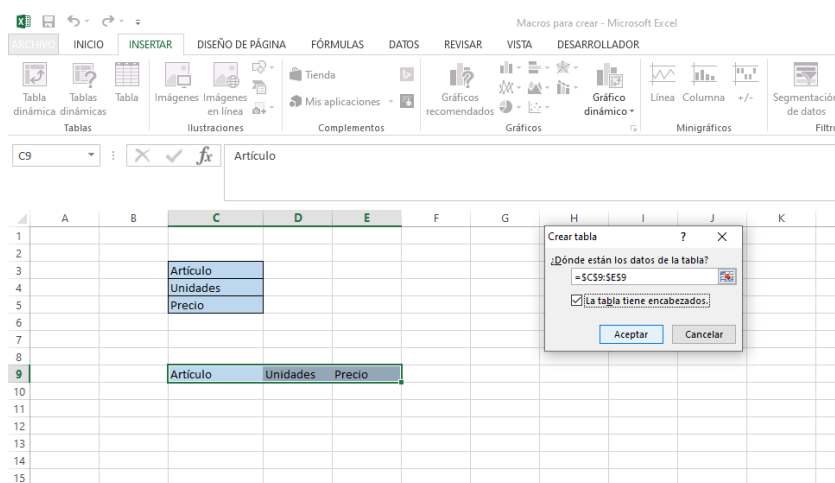


Figura 1.41. Crear tabla.

4. Para Grabar la Macro que permita ingresar uno a uno los artículos, es necesario llevar a cabo las siguientes acciones:

- a. Seleccionar el botón de grabar macro, colocar el nombre de la macro y oprimir la tecla <Enter> Observar que al grabar la macro nos permite agregar nombre, seleccionar algunas teclas rápidas y una descripción. Se recomienda no utilizar las teclas rápidas debido a que el entorno usa varias y puede generar confusión de uso (Figura 1.42).

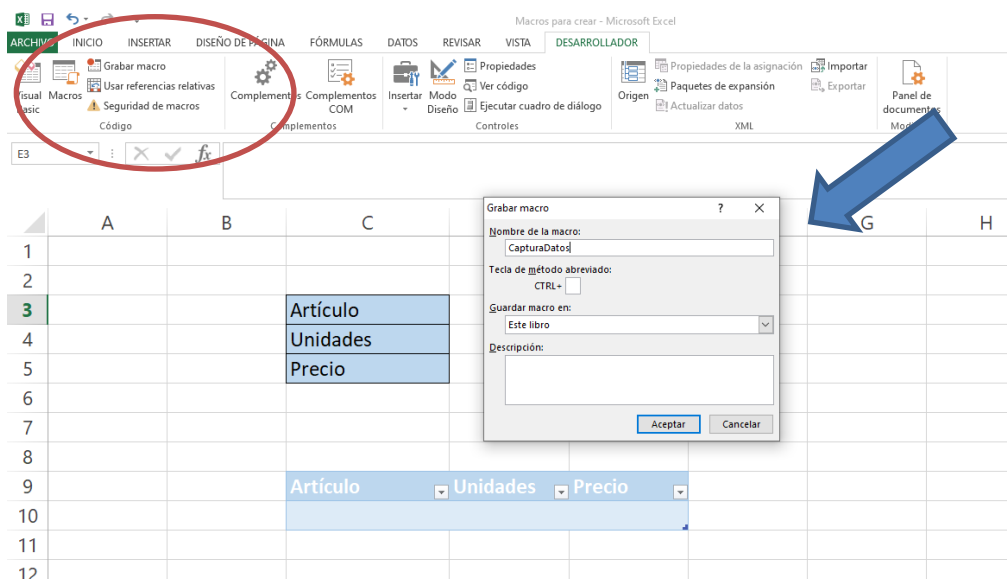


Figura 1.42. Colocar nombre a la Macro.

- b. El botón de grabar cambia a detener grabación y será en este modo que cada acción realizada se guardará y para finalizar debe oprimir el botón detener grabación (Figura 1.43).

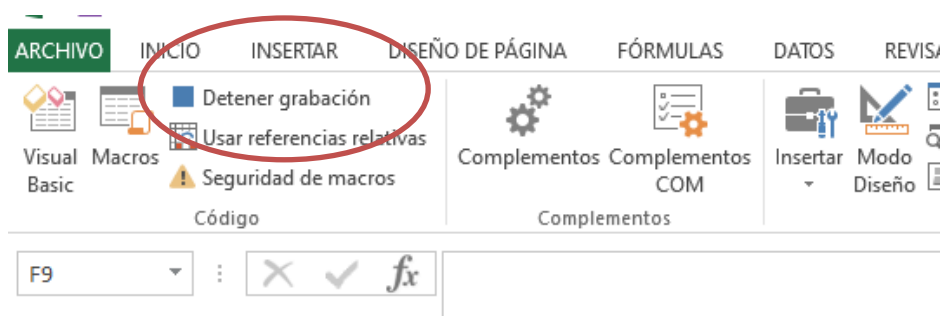


Figura 1.43. Grabando acciones de la Macro.

- c. Capturar instrucciones que debe contener la Macro para insertar un nuevo artículo al inicio de la lista:
1. Selecciona la fila donde se almacenará la lista e inserta una fila arriba
 2. Copiar la columna donde se capturan los datos y con pegado especial **transponer** en la fila de la lista de productos
 3. Terminar grabación oprime botón: **detener grabación**

- d. El siguiente paso es asignar la Macro. Para ello se puede ejecutar directamente con el comando ejecutar del menú de opciones de Macros, asignar un botón o colocar una forma y asignar a la forma la macro. (Ver figura 1.44).

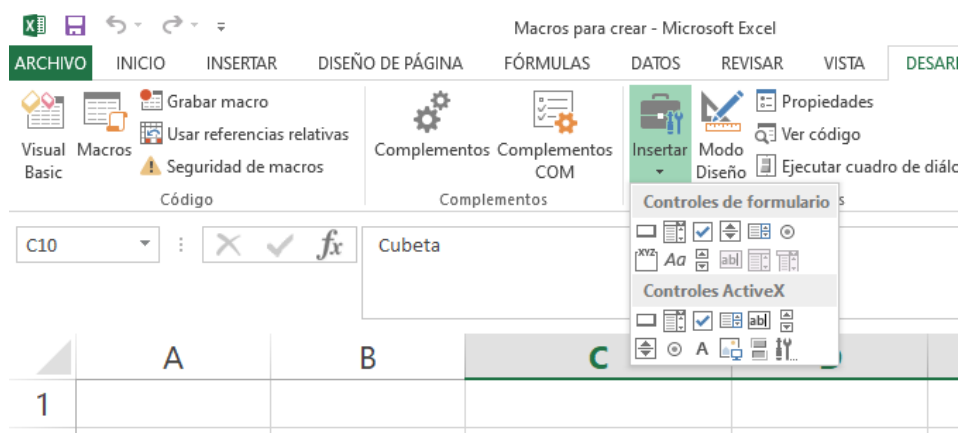


Figura 1.44. Insertar botón o control para la Macro.

Para esta macro se optará por elegir una forma, para ello: seleccionar de la barra de opciones **Insertar formas**, elegir una de ellas y capturar en la forma el texto **Guardar** y asignar a la forma la macro realizada, para ello es necesario hacer lo siguiente:

1. Insertar la forma en un espacio de la hoja de cálculo y asignar la macro oprimiendo el botón derecho (figura 1.45.a y seleccionar la macro (Figura 1.45.b).

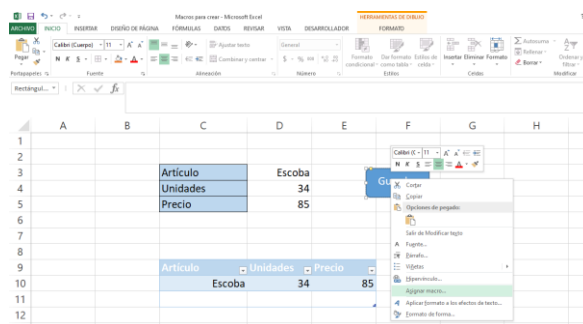


Figura 1.45.a Asignar la Macro a una forma

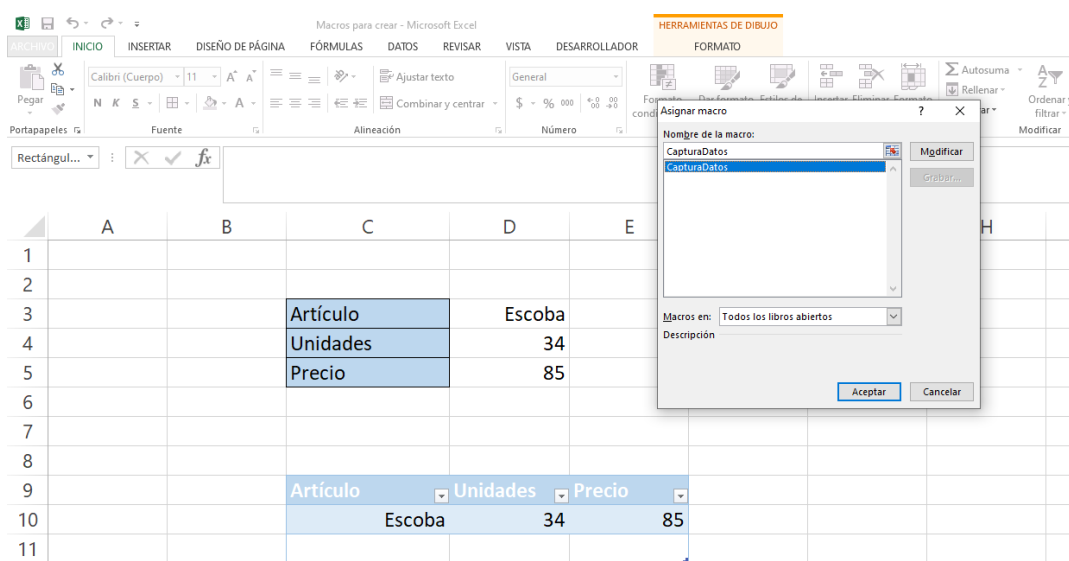
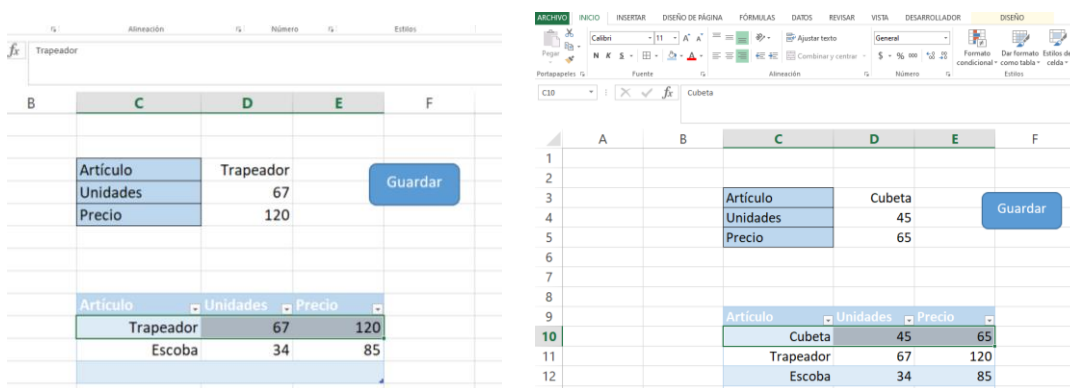


Figura 1.45.b, Seleccionar la Macro a asignar.

A partir de esta acción cada que se capture un nuevo artículo y se oprima el botón de guardar, el artículo se insertará en la lista de productos. (Figura 1.46).



Cuando se quiere agregar o cambiar una instrucción de la Macro existen dos opciones, generar una nueva macro (recomendado para personas que no programan) o realizar el cambio en el código (opción para programadores). Para

visualizar el código de una macro es necesario acceder al editor de Visual Basic (VB), para ello se selecciona la opción modificar Macro (Figura 1.47).

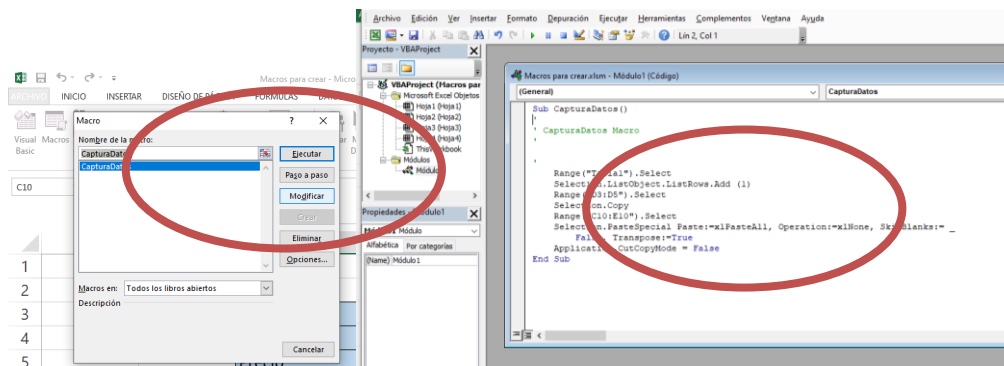


Figura 1.47. Modificar código

Para salir del editor de VB, simplemente se oprime el botón de Excel de la esquina superior izquierda (Figura 1.48).

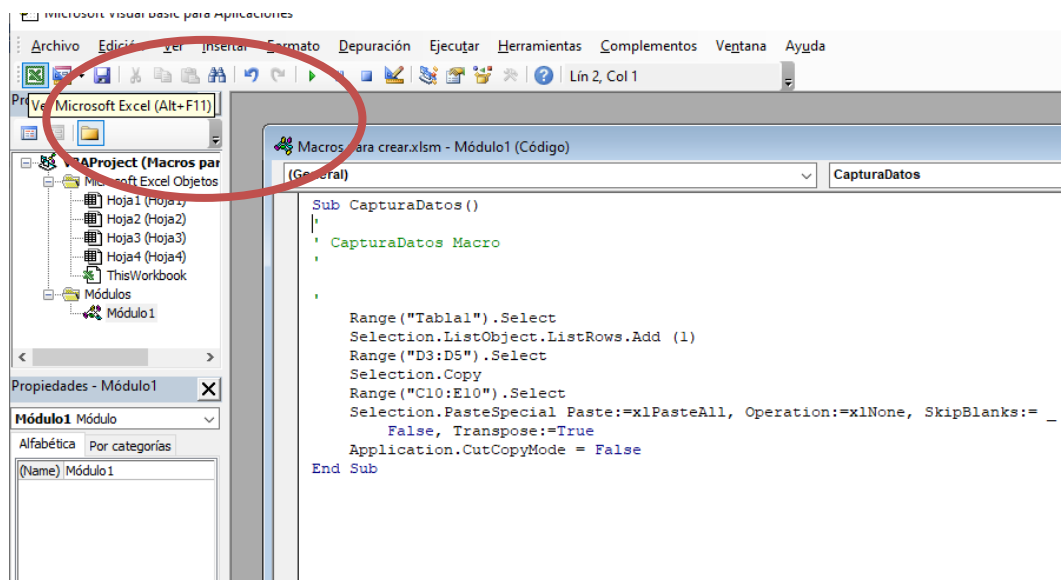


Figura 1.48. Salir del editor de VB.



Ejercicio de aprendizaje

Crear una lista de productos con los siguientes campos: código, producto, cantidad, precio, subtotal y fecha de compra. La lista se actualizará a partir de la captura de datos en un formulario. La lista de productos debe almacenarse en una hoja de cálculo y el formulario en otra hoja del mismo libro. Cada que se capture un nuevo producto el formulario debe tener los campos limpios y el campo de subtotal no debe calcularse multiplicando cantidad y precio de cada producto.

Alternativa de solución

Paso 1. Crear un libro y guardar archivo con la opción de habilitado para macros.

Paso 2. Crear el formulario en la hoja de cálculo no.1 del libro, con los campos solicitados y colocar 3 formas para utilizar las macros a crear: guardar, limpiar y fecha, sigue la propuesta de la figura 1.49

The screenshot shows the Microsoft Excel interface with the 'Nota de compra' form on Sheet1. The form is located in the range B4 to E16. It includes input fields for 'Código', 'Producto', 'Cantidad', 'Precio', 'subtotal', and 'Fecha de compra'. There are also buttons for 'Limpiar' and 'Grabar'. The ribbon at the top shows 'ARCHIVO', 'INICIO', 'INSERTAR', 'DISEÑO DE PÁGINA', 'FÓRMULAS', 'DATOS', 'REVISAR', 'VISTA', and 'DESARROLLADOR'. The 'DESARROLLADOR' ribbon is active, showing the 'Formato condicional' group.

Figura 1.49 Crear formulario

Paso 3. Asignar y validar datos a cada celda. El subtotal es una fórmula: =cantidad*precio y se recomienda validar el campo código para que no guarde campos vacíos. Ver figura 1.50

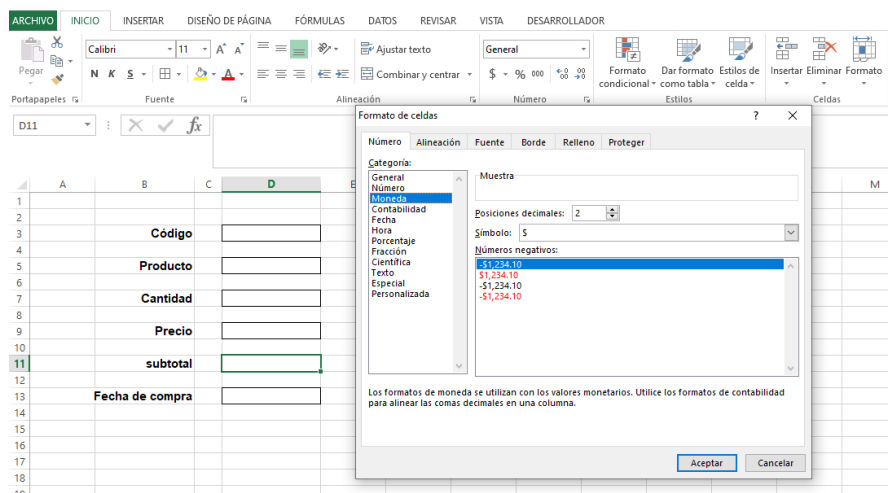


Figura 1.50 Validar y asignar datos al formulario

Paso 4. Colocar las formas donde posteriormente se asignarán las macros correspondientes (Figura 1.51)

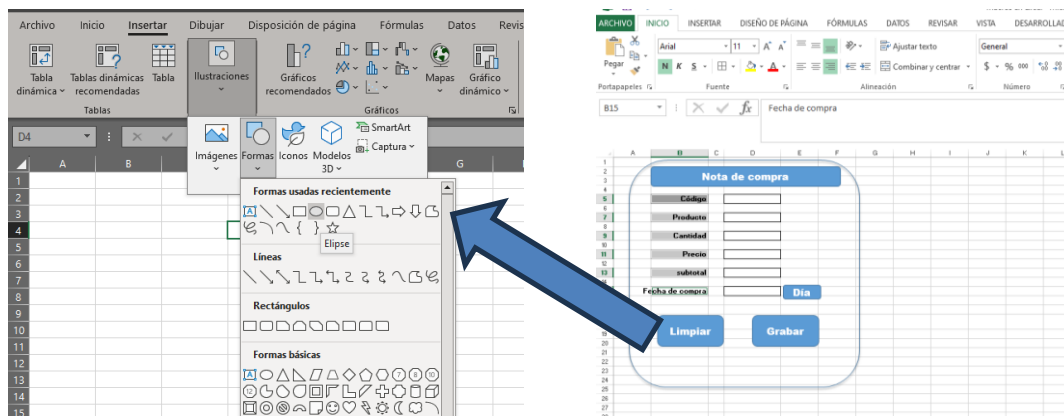


Figura 1.51. Formulario para captura de datos usando macros



Paso 5, Seleccionar la hoja no. 2 y copiar cada uno de los campos del formulario en forma de fila, utilizando el pegado especial y seleccionar la opción transponer como se muestra la figura 1.52

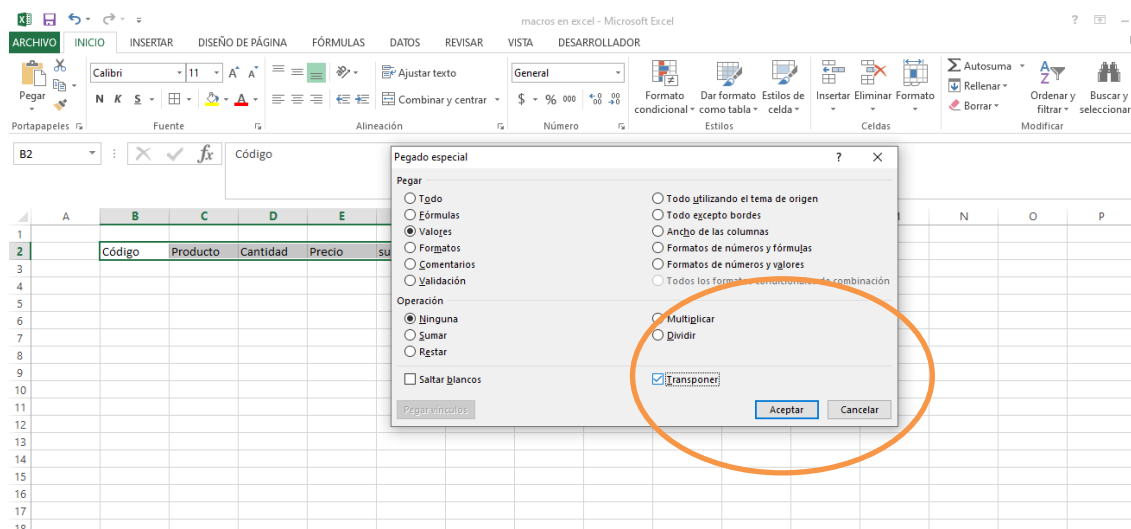


Figura 1.52. Generar lista de productos en hoja de cálculo

Paso 6. Dar formato de tabla a la lista de campos creada en el paso 5. (Figura 1.53)

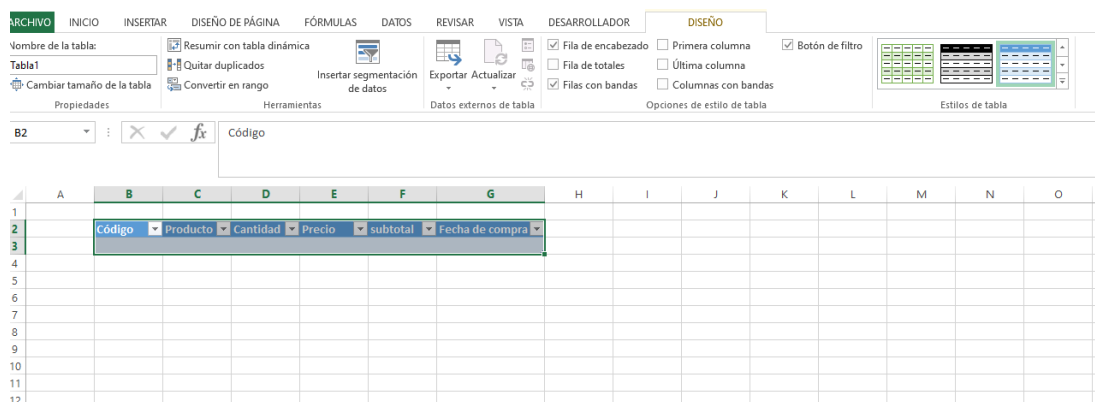


Figura 1.53. Generar formato de tabla



Paso 7. Iniciar la grabación de cada una de las macros colocando el nombre a la Macro. (Figura 1.54)

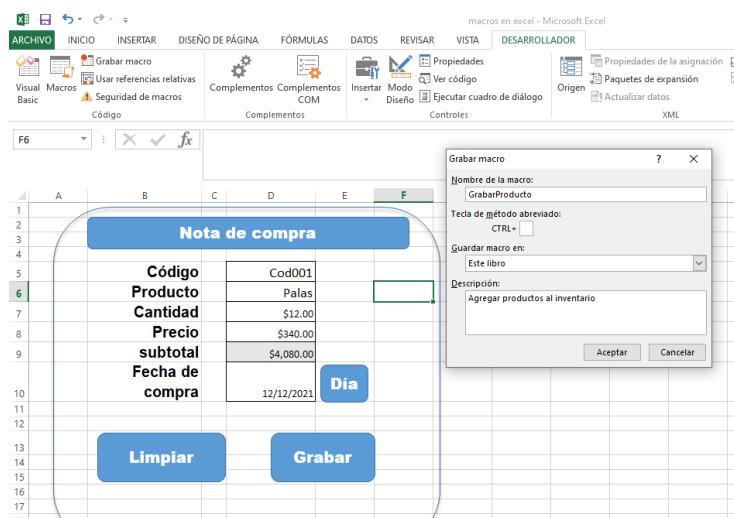


Figura 1.54. Crear macro **GuardarProducto**

Paso 8. Grabar las instrucciones de la macro **guardar** la cual debe integrar las instrucciones de: insertar una fila en la tabla de datos, copiar y pegar cada uno de los contenidos de las celdas donde se capturan los datos (Figura 1.55)

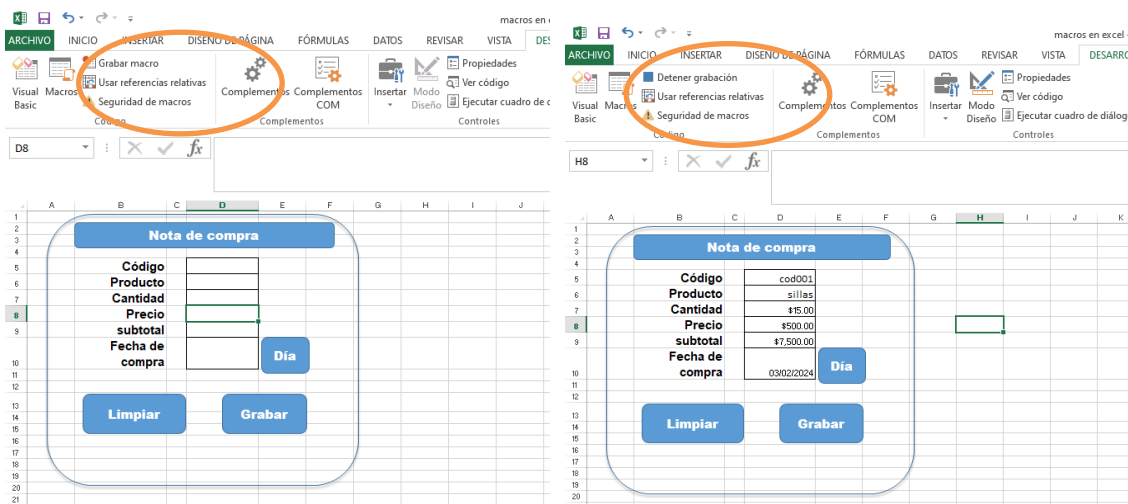


Figura 1.55 Grabar instrucciones en la Macro

Los formatos de los campos numéricos deben tener el mismo tipo, pero el de fecha debe validarse para que se ingrese un formato válido. Esta macro debe aceptar un formato específico para ello es necesario seleccionar el formato fecha y el tipo de valor que se desea. (Figura 1.56.a y 1.56.b).

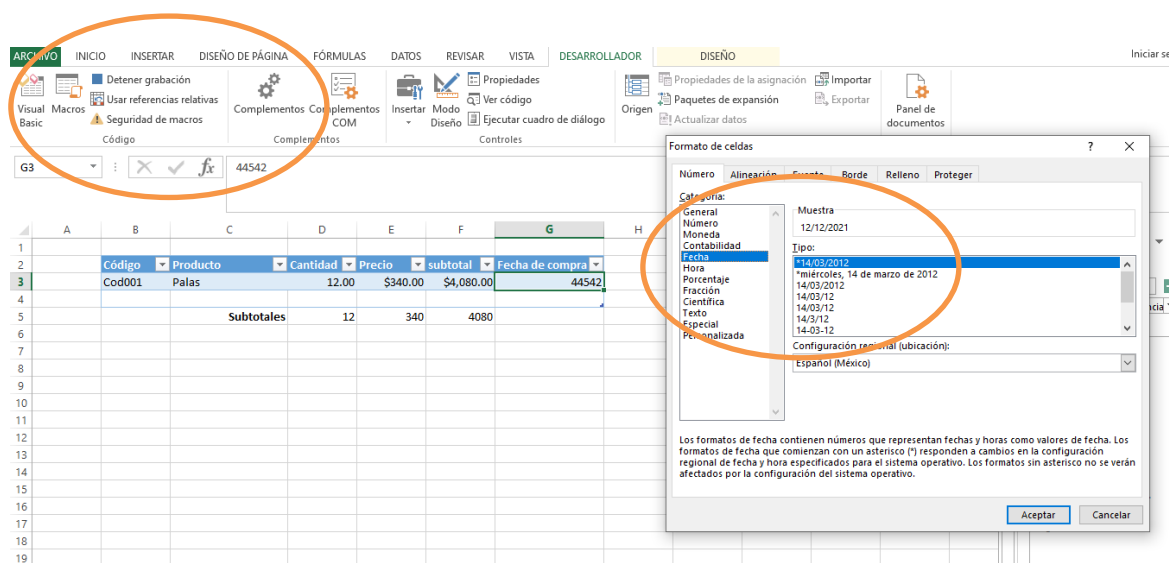


Figura 1. 56.a. Formato de Fecha.

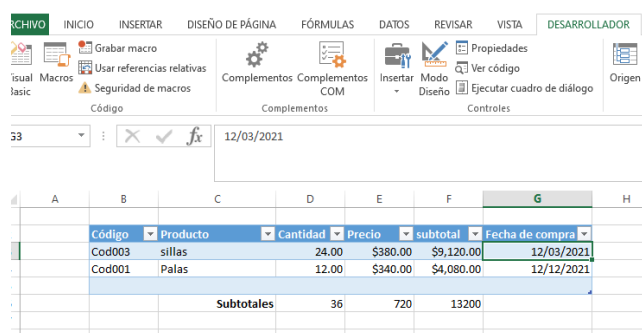


Figura 1.56.b. Macro para generar fecha en formato válido.

No olvidar asignar la Macro correspondiente al botón o forma donde se ejecutará (Figura 1.57).

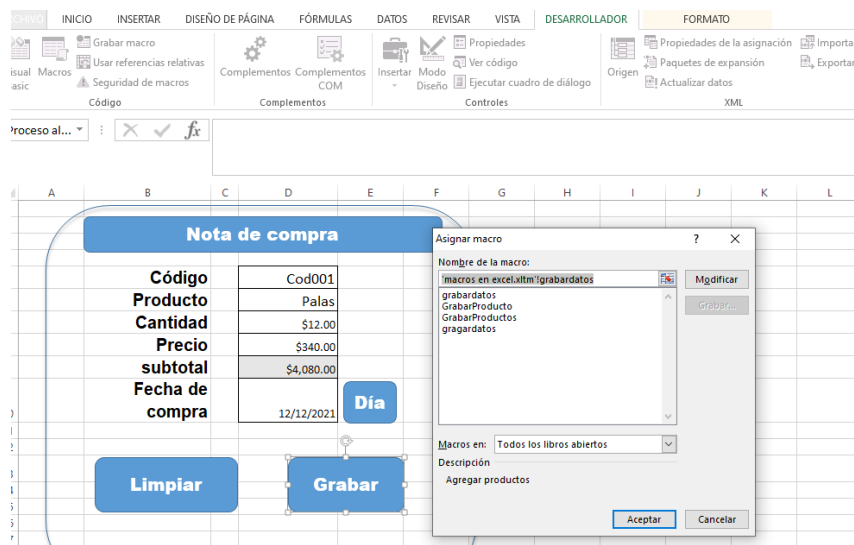


Figura 1.57 Asignar la Macro Guardar a la forma.

La Macro **Limpiar** simplemente borra los datos de captura del formulario cada que se capture un nuevo producto, las instrucciones al grabar la macro es seleccionar los campos de captura y oprimir la tecla <suprimir> del teclado (Figura 1.58).

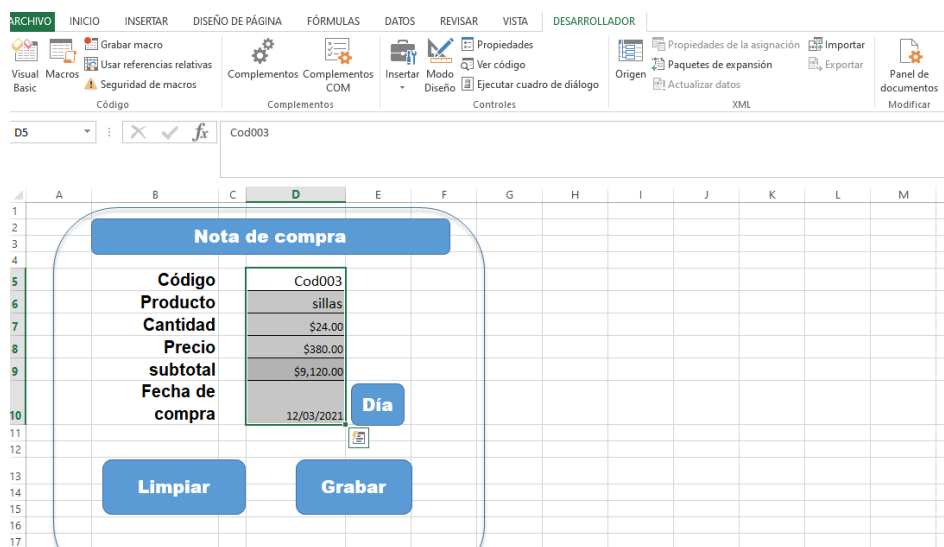


Figura 1.58. Grabar Macro **Limpiar**.



Asignar la macro al botón correspondiente. (Figura 1.59).

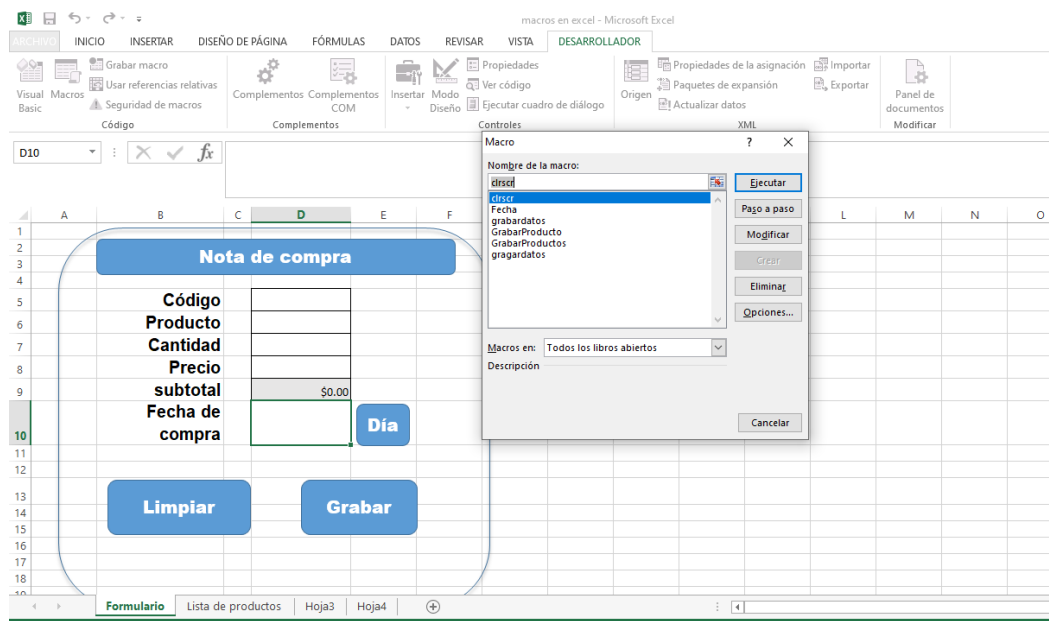


Figura 1.59 Asignar macro **Limpiar** al botón.

Al ejecutar las macros cada acción realizará las instrucciones al oprimir el botón correspondiente, capturar datos, agregar productos y validar datos ya que si se ingresa un valor inválido este mostrará el mensaje de no válido que se agregó al generar la macro (Figuras 1.60.a, 1.60.b, 1.60.c).

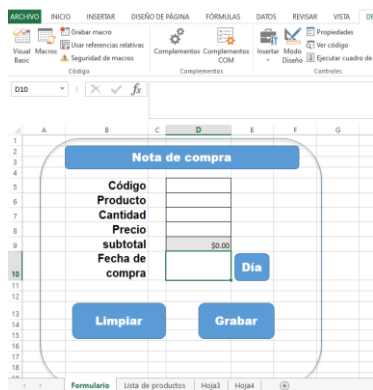
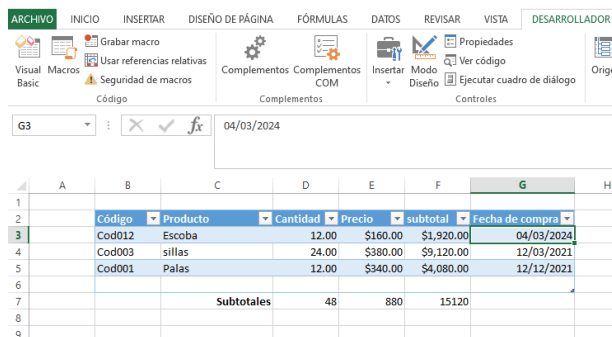


Figura 1. 60.a Captura



1.60.b. Agregar Productos

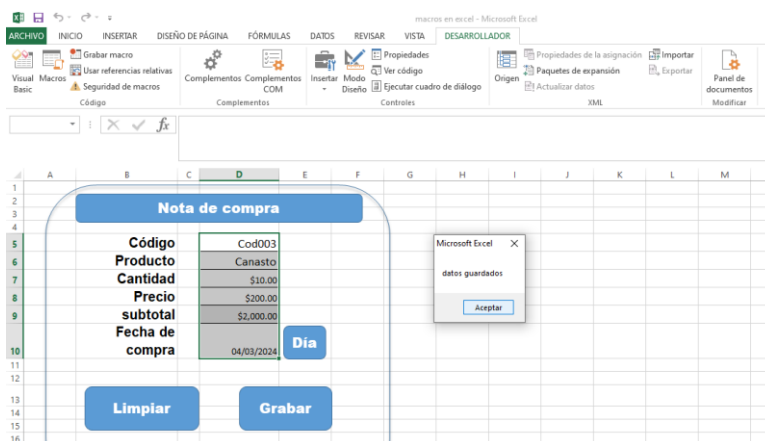


Figura 1.60.c Validar datos

Hay que recordar, que el código de cada macro se puede visualizar en el editor de VB, por si se quiere modificar alguna acción o para verificar cada una de las instrucciones que realiza la macro (Figura 1.61).

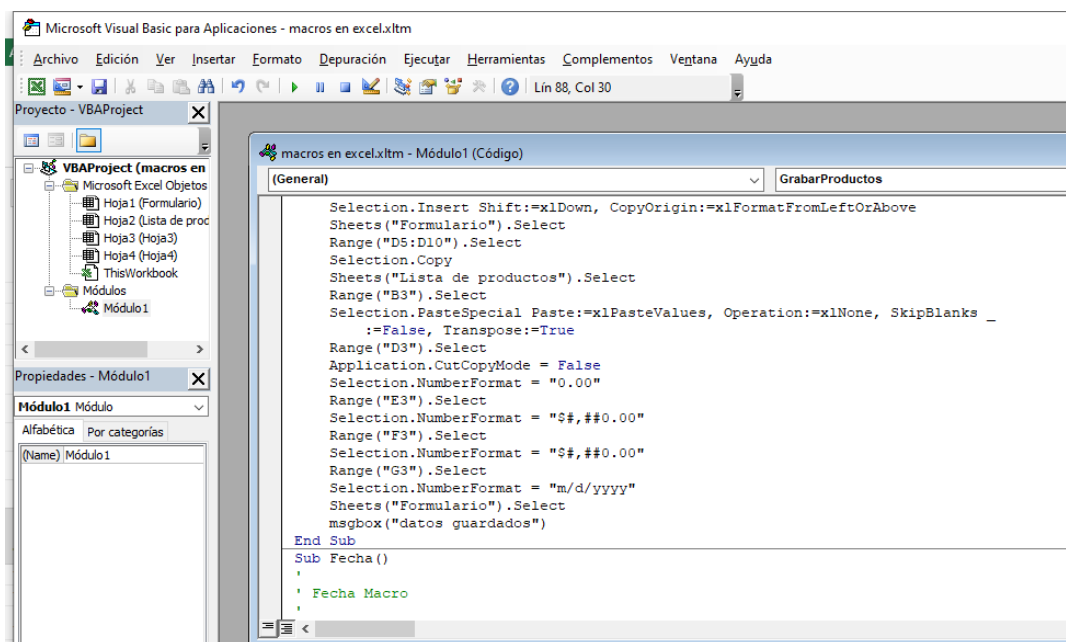


Figura 1.61 código de la macro **GrabarProductos**

En la visualización de código se puede integrar o llamar una macro o macros dentro de otra macro, se realiza seleccionando en el apartado de desarrollador o programador la acción de Visual Basic, se elige la macro a editar y en el código se inserta el nombre de la macro o macros a integrar con el mismo nombre con el que fue creada como se muestra en las figuras 1.62 y 1.63.

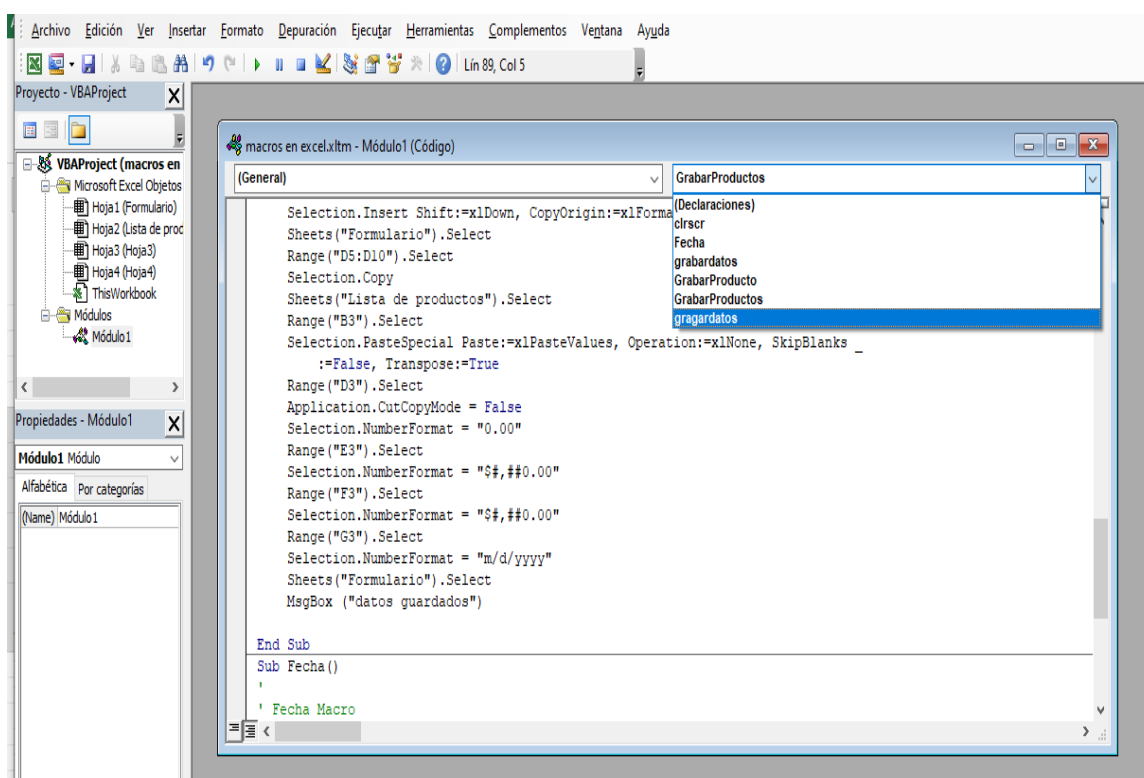


Figura 1.62. Integrar Macros en una Macro

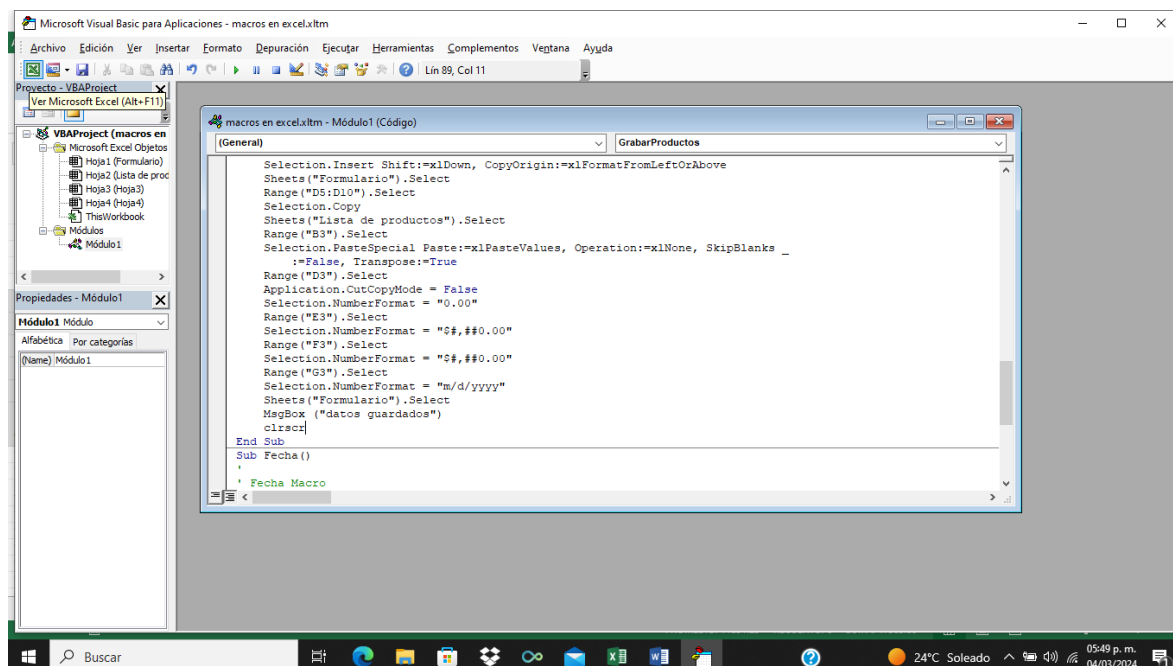


Figura 1.63. Llamar desde macro **grabarProductos** la macro limpiar datos **clrscr**

Ejercicio de aprendizaje

Generar un formulario personalizado para captura de datos de los residentes activos de una institución tomando la siguiente información: Número de control, Nombre del estudiante, carrera, semestre, nombre de la empresa, asesor interno, asesor externo, periodo de residencia. El formulario de captura debe figurar en la hoja de cálculo 1 y la base de datos debe almacenarse en la hoja de cálculo 2 del mismo libro.



1.6 Aplicaciones.

Hoy en día, existe un sinnúmero de necesidades y problemáticas a resolver en cada área o disciplina educativa, laboral, organizacional, etc. Excel es una herramienta que puede facilitar y agilizar algunos procesos donde el uso de hojas de cálculo simplifique y eficiente el tiempo de respuesta a estas necesidades y/o problemas detectados. En este apartado se describe el uso de varias aplicaciones donde los temas vistos se pueden aplicar en problemas aplicados al ámbito profesional.

Proyección de ventas. Suponer que el plan de mercadeo de una empresa establece que el margen neto a alcanzar debe ser de al menos 35,000 pesos. Se inicia con 800 unidades a un precio de venta de 50 pesos por unidad, descuento de 23% sobre ventas brutas y comisiones de venta del 14%. En este sentido se pretende calcular el total de unidades que se deben vender para alcanzar el margen neto propuesto para lograr el objetivo esperado.

Solución:

1. Capturar los datos de entrada en una hoja de cálculo tomando en consideración las siguientes fórmulas:

Ventas brutas = precio unitario*unidades

Descuentos = ventas brutas * 23%

Ventas netas= ventas brutas-descuentos

Comisión=ventas netas*14%

Margen neto=ventas netas-comisión

Tome en consideración que los datos fijos se capturen en una tabla y las fórmulas del proceso sean dependientes de la celda que se desea cambiar, para que la proyección de la variable con la que se desea trabajar sea correcta. (Figura 1.64)

Artículo	Ventas Trimestre	Precio unitario	Descuentos	Comisión
unidades	\$ 800.00	50	23%	14%
ventas brutas	\$ 40,000.00			
descuentos	\$ 9,200.00			
ventas netas	\$ 30,800.00			
comision	\$ 4,312.00			
margen neto	\$ 26,488.00			

Figura 1.64 Captura de datos y fórmulas del ejercicio

2. Realizar la proyección de ventas, Para ello es necesario utilizar la función **análisis** del menú de **datos** y el comando **buscar objetivo** (Figura 1.65.).

Artículo	Ventas Trimestre	Precio unitario	Descuentos	Comisión
unidades	800	50	23%	14%
ventas brutas	40000			
descuentos	9200			
ventas netas	30800			
comision	4312			
margen neto	26488			

Figura 1.65. Función Análisis de datos.

2. Seleccionar en la opción **definir celda**. la celda donde se encuentra la fórmula del margen neto, en nuestro caso la celda B7, en el **argumento con el valor**, introducir 35,000 que es el valor de ventas esperado y **cambiando la celda** será el número de unidades es decir la celda B2 para poder obtener el objetivo planteado en el ejercicio. En nuestro caso es la celda B7 y para finalizar y obtener resultados oprimir la tecla <Enter>. (Figura 1.66).

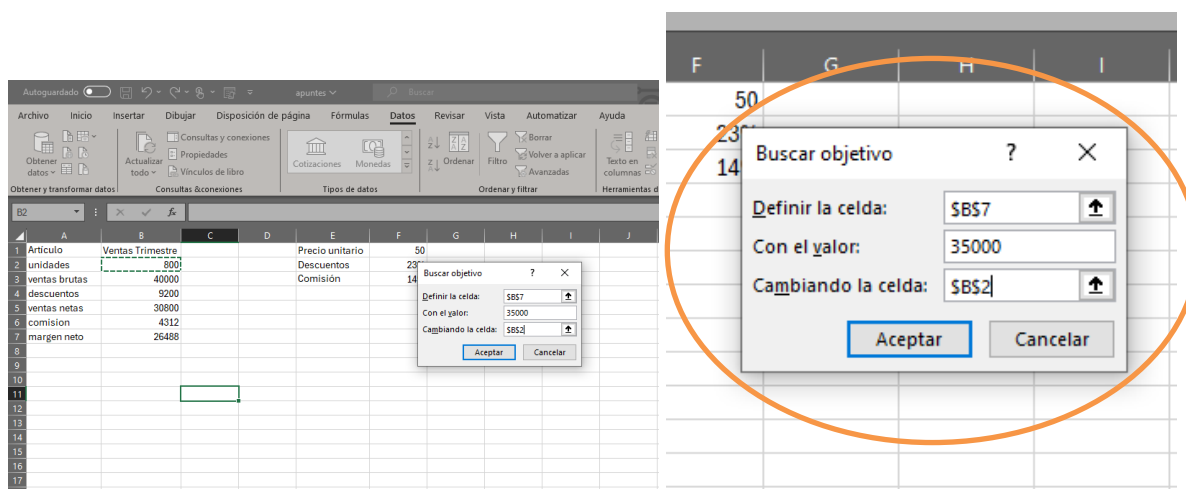


Figura 1.66. Función análisis Y si (hipótesis)

3. El uso de esta función modifica los datos de origen y sin cambiar precio, descuentos o comisión, muestra la cantidad proyectada de ventas que se deben efectuar para lograr el objetivo esperado. El resultado en este caso es una venta de 1057 unidades para lograrlo. (Figura 1.67).

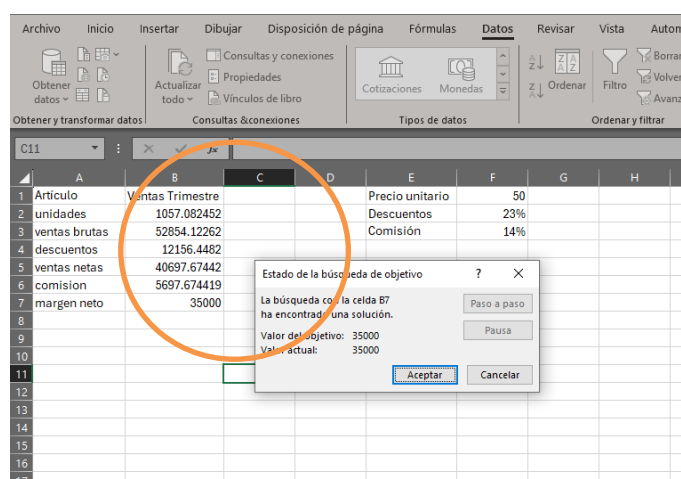


Figura 1.67. Proyección de ventas



Este tipo de análisis de datos permiten a las empresas, organizaciones o personas emprendedoras calcular el total de ventas cuando se fijan metas y objetivos bien definidos.

Ejercicio de aprendizaje

Una empresa desea colocar en el mercado llaveros con chip inteligente de búsqueda. El plan de mercadeo de la empresa establece que el margen neto a alcanzar debe ser de al menos 65,000 pesos. Se inicia con un precio unitario de 175 pesos por producto, descuento de 20% y comisiones de 16%. En este sentido se pretende calcular el total de unidades que se deben vender para alcanzar el margen neto propuesto para lograr el objetivo esperado. Para calcular el margen neto es necesario obtener los siguientes valores, aplicando las fórmulas siguientes:

Ventas brutas = precio unitario*unidades

Descuentos = ventas brutas * 20%

Ventas netas= ventas brutas-descuentos

Comisión=ventas netas*16%

Margen neto=ventas netas-comisión

Proyección financiera. Realizar la proyección mediante inversión y ahorro monetario se logra con funciones específicas que ofrece el entorno, un ejemplo de ello se realiza a continuación.

Un estudiante desea comprar una computadora cuyo costo es de 13,500 pesos. Cuenta con 2,000 y desea invertirlos en una cuenta de inversión que le brinda un 4% de interés anual, y aportar 600 pesos cada mes, para poder incrementar su fondo de inversión



En este caso se debe hacer uso de la función **NPER**, la cual permite calcular el número de periodos que se necesitan para lograr un determinado saldo en una inversión. Los pasos para obtener el dato son:

1. Ingresar los datos que se tienen en una hoja de cálculo como se muestra en la figura 1.68

	A	B	C
1	Importe del préstamo	13500	
2	Interés anual	4%	
3	Número de cuotas	=nper(B2/12,-B5,B6,B1)	
4			
5	Cuotas mensuales	600	
6	Pago inicial	2000	
7	Intereses pagados		

Figura 1.68. Ingreso de datos en hoja de cálculo

Una vez ejecutada la función donde se establece el saldo que se requiere, las cuotas mensuales que debe aportarse son de 25 meses, es decir, un poco más de dos años para poder comprarse la computadora y siempre y cuando no se tome en consideración la inflación. (Figura 1.69)

	A	B	C
1	Importe del préstamo	13500	
2	Interés anual	4%	
3	Número de cuotas	25.1698605	
4			
5	Cuotas mensuales	600	
6	Pago inicial	2000	
7			
8			

Figura1.69. Uso de la función NPER



Al utilizar formularios en Excel, se pueden crear lista de opciones en campos para que solo se introduzcan valores en función de la lista previamente capturada, o para validar datos con rango, entre otros. Por ejemplo, capturar un número telefónico sabiendo que en México se aceptan 10 dígitos, en este sentido se utiliza de la barra de opciones el menú de DATOS y el comando validación para colocar el formato que se necesita como se puede observar en la figura 1.70

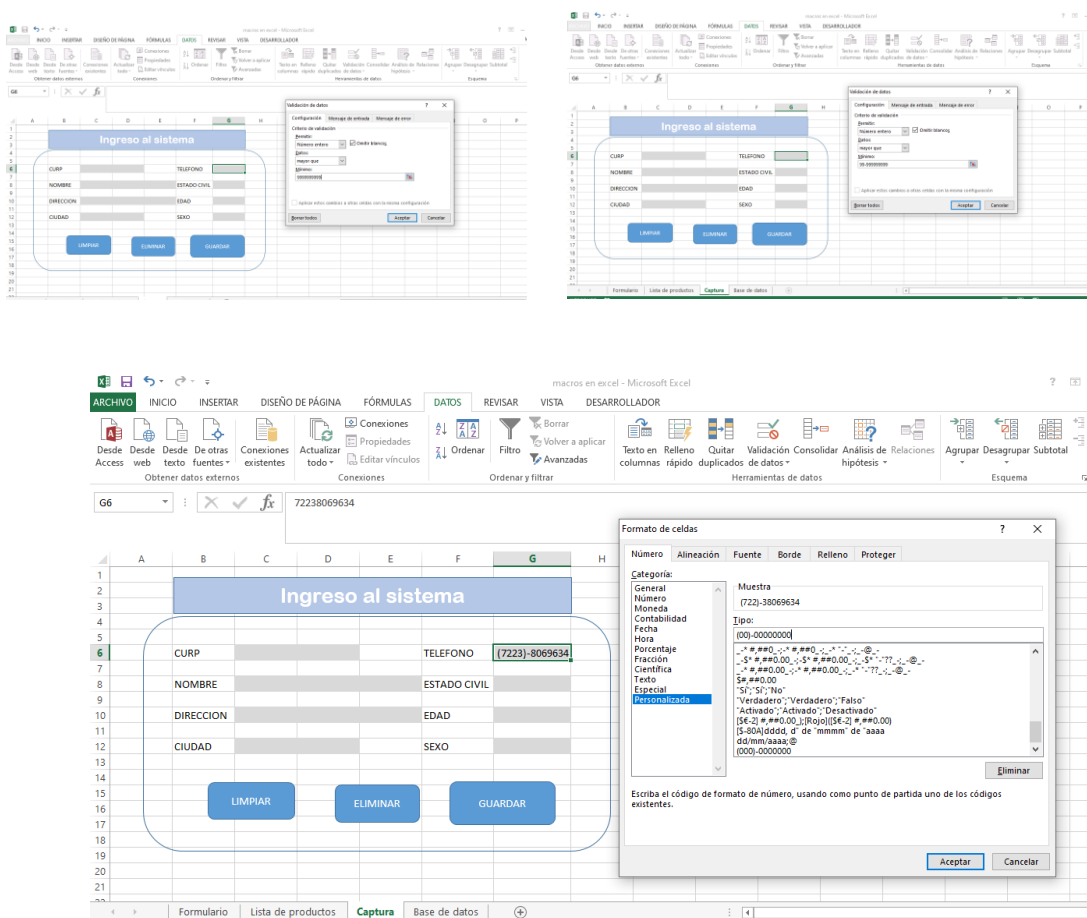


Figura 1.70 Validación de datos

Otro tipo de validación, por ejemplo, el campo Sexo debe aceptar masculino y/o femenino, o el caso de estado civil que corresponde a una lista, para que se despliegue una lista al seleccionar el campo correspondiente. Para lograr esta



validación, seleccionar validación de datos, lista y en origen capturar la lista de valores válidos, como se muestra en las figuras 1.71, 1.72, 1.73 y 1.74.

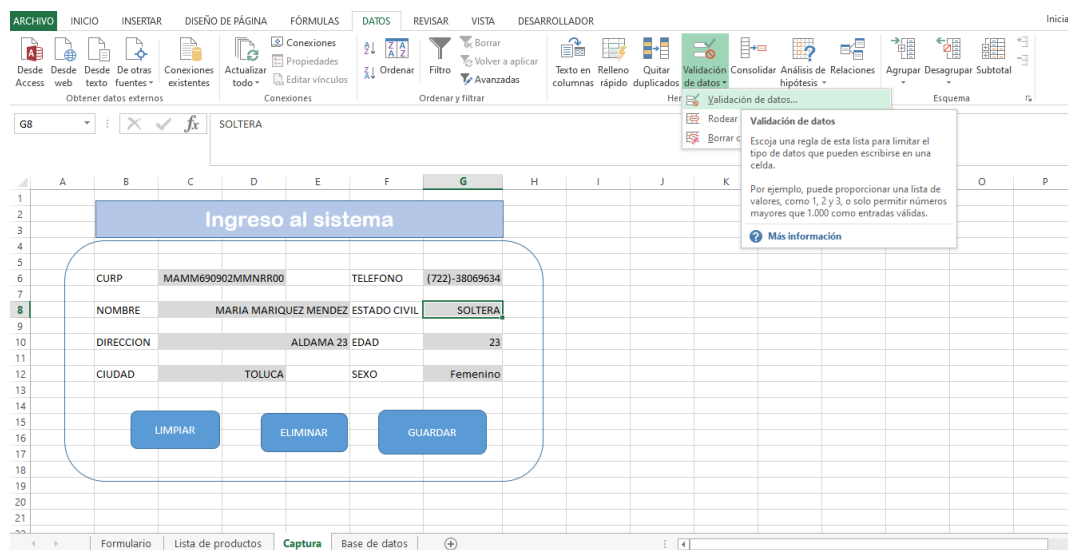


Figura 1.71 Validación de datos

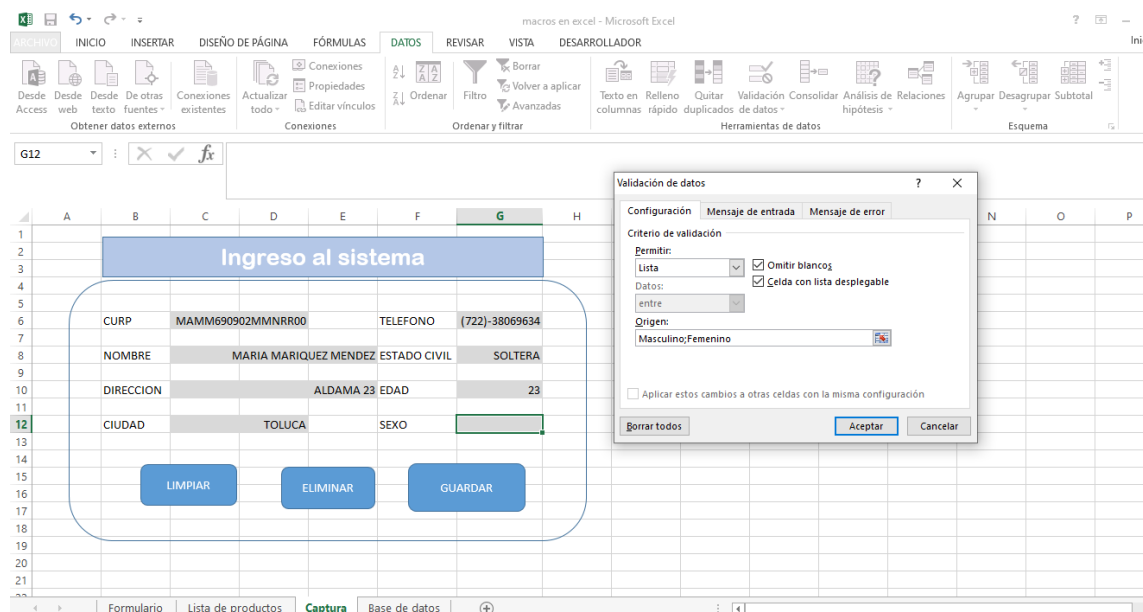


Figura 1.72. Validar datos mediante una lista preasignada



ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS REVISAR VISTA DESARROLLADOR Inicia

Desde Access Desde web Desde texto Desde otras fuentes Obtener datos externos Conexiones existentes Actualizar todo Conexiones Propiedades Editar vínculos Conexiones Ordenar Filtro Volver a aplicar Avanzadas Ordenar y filtrar Texto en columnas Relleno rápido Quitar duplicados Validación Consolidar Análisis de Relaciones hipótesis Agrupar Desagrupar Subtotal Esquema

G12 : X ✓ fx Femenino

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2		Ingreso al sistema														
3																
4																
5																
6		CURP	MAMM690902MMNRR00			TELEFONO	(722)-38069634									
7		NOMBRE	MARIA MARIQUEZ MENDEZ			ESTADO CIVIL	SOLTERA									
8		DIRECCION	ALDAMA 23			EDAD	23									
9																
10		CIUDAD	TOLUCA			SEXO	Femenino									
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																

Formulario Lista de productos Captura Base de datos

Figura 1.73 Desplegar valores en el campo

ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS REVISAR VISTA DESARROLLADOR macros en excel - Microsoft Excel ? - Iniciar

Desde Access Desde web Desde texto Desde otras fuentes Obtener datos externos Conexiones existentes Actualizar todo Conexiones Propiedades Editar vínculos Conexiones Ordenar Filtro Volver a aplicar Avanzadas Ordenar y filtrar Texto en columnas Relleno rápido Quitar duplicados Validación Consolidar Análisis de Relaciones hipótesis Agrupar Desagrupar Subtotal Esquema

G8 : X ✓ fx SOLTERO(A)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2		Ingreso al sistema														
3																
4																
5																
6		CURP	MAMM690902MMNRR00			TELEFONO	(722)-38069634									
7		NOMBRE	MARIA MARIQUEZ MENDEZ			ESTADO CIVIL	SOLTERO(A)									
8																
9		DIRECCION	ALDAMA 23			EDAD	23									
10																
11		CIUDAD	TOLUCA			SEXO	Femenino									
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																

Formulario Lista de productos Captura Base de datos

Figura 1.74. Lista de valores



1.7 Ejercicios¹

1. Generar de una lista de 20 nombres los correos electrónicos para gmail, tomando el ejemplo de la siguiente tabla

Nombre	Correo electrónico
Luis Martínez Juárez	luismartinezjuarez@gmail.com
Adán esparza Méndez	adanesparzamendez@gmail.com
Laura Mora Juárez	lauramorajuarea@gmail.com

2. Capturar una lista de calificaciones de las materias cursadas hasta el momento y usando la función SI, colocar acreditado (calificación ≥ 70) o no acreditado (calificación < 70)
3. Realiza el siguiente reporte usando las funciones más adecuadas

Presupuesto Personal				
Estado de flujo de efectivo				
Ingresos	Mes1	Mes2	Mes 3	Mes 4
Sueldo	\$18,600.00	\$18,600.00	\$18,600.00	\$18,600.00
Dividendos	\$3,800.00	\$1,200.00	\$1,200.00	\$800.00
Otros ingresos	\$1,200.00	\$500.00	\$90.00	\$0.00
Total de ingresos	\$23,600.00	\$20,300.00	\$19,890.00	\$19,400.00
Egresos	Mes 1	Mes 2	Mes 3	Mes 4
Renta	\$13,500.00	\$13,500.00	\$13,500.00	\$13,500.00
Servicios públicos	\$560.00	\$560.00	\$560.00	\$560.00
Teléfono	\$270.00	\$270.00	\$270.00	\$270.00
Reparaciones	\$123.00	\$123.00	\$123.00	\$123.00
Alimentos	\$350.00	\$350.00	\$350.00	\$350.00
Total de Egresos	\$14,803.00	\$14,803.00	\$14,803.00	\$14,803.00
Flujo de efectivo	\$8,797.00	\$5,497.00	\$5,087.00	\$4,597.00
Flujo de efectivo acu	\$8,797.00	\$14,294.00	\$19,381.00	\$23,978.00

4. Genera en una hoja de cálculo la retícula del plan de estudios siguiente, anotando en una celda en cada una de las asignaturas la calificación obtenida, con verde sombrear las asignaturas cursadas, con rojo las no

¹ En la carpeta de materiales de apoyo podrá consultar una alternativa de solución de algunos ejercicios de cada tema, pero la práctica puede permitirle crear nuevas alternativas por ello se recomienda realizar su propuesta y posteriormente comparar.



acreditadas y con azul aquellas que aún faltan por cursar. Colocar en la esquina inferior derecha de la retícula el total de créditos cursados, créditos que se están cursando y los créditos que faltan por cursar

- Realiza el formato de listas de asistencia para el control de los estudiantes y en otra hoja calcula g el reporte de asistencia mensual

LISTA DE ASISTENCIA																																			
Mes		Días hábiles																Asistencia media																	
CURSO	no	nombre del alumno	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	A	T
	1																																		
	2																																		
	3																																		
	4																																		
	5																																		
	6																																		
	7																																		
	8																																		
9																																			
CURSO	1																																		
	2																																		
	3																																		
	4																																		
	5																																		
	6																																		
	7																																		
	8																																		
	9																																		
Maestra/o																																			

1.8 Aprendizaje basado en casos

Mario es egresado del Instituto Tecnológico. Desde la prepa emprendió un negocio de diseño de muebles: escritorios, mesas y sillas, mismo que le permitió comprar maquinaria y contratar empleados. Actualmente atiende su negocio en un local que renta y tiene a su cargo 3 empleados: uno le ayuda con las ventas y 2 con la elaboración de muebles, cuenta con 3 máquinas que le permiten atender el número de pedidos. El negocio tiene buena reputación, sin embargo, la contabilidad no es su fuerte y le ha generado diversos conflictos con el control de ingresos, de gastos, insumos y ganancias. A ciencia cierta no tiene certeza de los insumos que se adquieren a granel y si éstos se utilizan completamente o existe robo por parte de los empleados en piezas que se adquieren y que no se controlan por inventario, por lo que considera necesario tener un control de manera automatizada de su empresa que le permita:

- Realizar la contabilidad de los ingresos, gastos e insumos
- Calcular sus ganancias por mes



3. Identificar el stock de mercancía para adquirir en tiempo con proveedores
4. Contar con una lista de proveedores, clientes
5. Llevar el control de pedidos
6. Generar el inventario de insumos y de muebles en existencia para evitar realizar muebles que se tienen en existencia

Realizar una propuesta que integre:

- a. Análisis de las actividades para establecer la problemática e identificar que procesos tienen fallas
- b. Analizar qué información debe almacenarse en una base de datos y generar las hojas de cálculo que contengan las tareas, operaciones y capturas necesarias como alternativas de solución
- c. Generar el tipo de reportes que eficiente la contabilidad del negocio
- d. Mencionar aspectos de mejora que debe tomar en cuenta Mario para mantener su negocio en el mercado.



2. Desarrollo de lógica algorítmica

2.1. Metodología para la solución de problemas.

Una metodología es la ciencia que aplica un método. Un método es un conjunto de instrucciones ordenadas utilizadas para obtener un resultado. Existen diversos tipos de metodologías, por ejemplo, la metodología de investigación, en este caso se describe la metodología de solución de problemas que se puede emplear para resolver mediante el uso de la computadora. (Miranda Chávez & Fuenlabrada Velázquez, 2015)

La resolución de los problemas utilizando una metodología, se realiza por etapas y depende de cada autor el número en las que se pueden clasificar. A continuación, se describen las fases más utilizadas en el desarrollo de algoritmos (Figura 2.1):

1. **Identificación del problema y análisis del problema.** Es la primera fase de la metodología en la solución de problemas, en esta etapa es importante el análisis de la información inicial, detectar los datos de entrada, es decir, la información requerida para obtener el resultado esperado, las fórmulas o acciones que son necesarios para obtener los resultados requeridos y los datos de salida, que consiste en la información resultado de un proceso o procesos.

2. **Alternativas de solución y selección de una de ellas.** La solución de un problema puede lograrse por distintas vías. En esta etapa es importante realizar el planteamiento de la mayor cantidad de alternativa sin exceder demasiado para evitar una mayor complejidad al momento de elegir una de ellas. Es necesario el análisis de cada alternativa generada y justificar adecuadamente aquella que se seleccione, tomando en consideración las características del problema, así como

de las características de solución, en función de la información y datos con los que se cuentan

4. **Elegir la técnica para realizar y formular el algoritmo.** En esta fase es importante seleccionar la forma en la que se formulará el algoritmo, por ejemplo: pseudocódigo, diagrama de flujo, diagrama NS, entre otros.

5. **Pruebas y depuración.** Al generar el pseudocódigo, es necesario ingresar datos y evaluar de manera manual o en un entorno, el correcto funcionamiento del programa realizado.

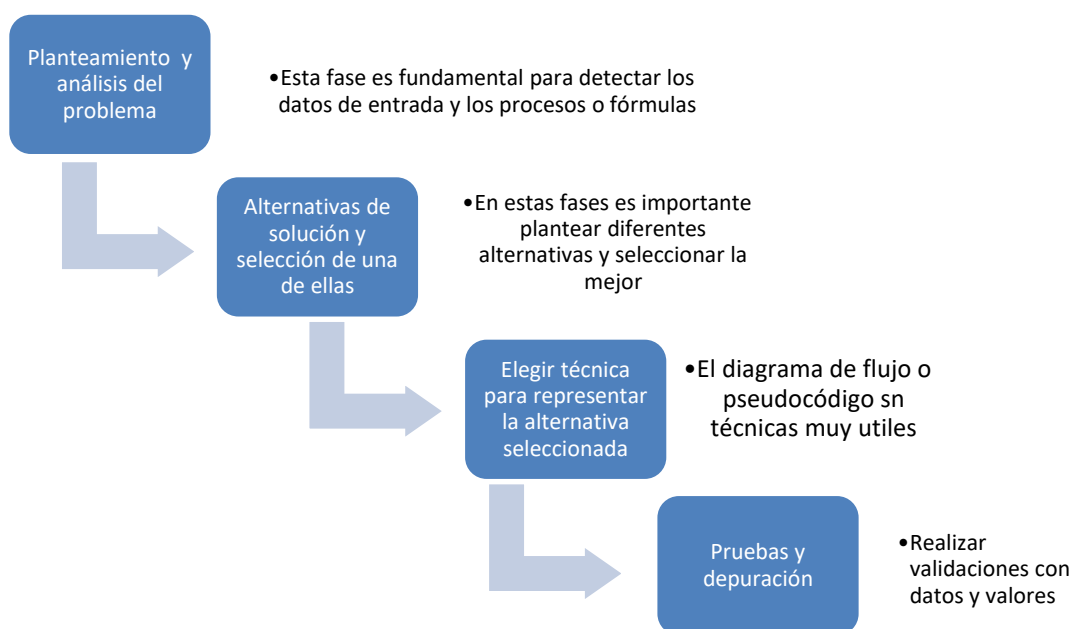


Figura 2.1. Etapas para el diseño de algoritmos



Ejemplo de uso:

Planteamiento del problema		
Cuántos libros se pueden comprar con 780 pesos, si dos libros cuestan \$195.00		
Identificación y análisis del problema		
Datos de entrada: Costo de 2 libros = 195 Dinero que se tiene = 780		Datos de salida: Total de libros que se pueden comprar con \$780.00
Alternativas de solución		
Alternativa 1 $780/195 = 4$ $4 \cdot 2 = 8$	Alternativa 2 $195/2 = 97.50$ $780/97.50 = 8$	Alternativa 3 $195+195+195+195=780$ $2+2+2+2=8$
Selección de alternativa.		
Seleccionar la alternativa más completa, sencilla o mejor, de acuerdo con las necesidades requeridas. En este caso se elige la alternativa 1 por ser la más sencilla y completa		
Desarrollo de la solución: (algoritmo) 1. inicio 2. Ingresar el precio de los 2 libros 3. Ingresar la cantidad de dinero		Datos de entrada: \$ 195.00 \$ 780.00
4. dividir la cantidad de dinero entre el precio de los libros 5. Multiplicar el resultado de la división por el número de libros que corresponde		Proceso: $PL=780/195$ $TL=PL \cdot 2$ Donde: PL y TL, son variables utilizadas para almacenar los valores correspondientes a pares de libros (PL) y total de libros (TL) el programador las define
6. Mostrar el número de libros que se pueden adquirir con la cantidad de dinero referida 7. Fin		Datos de salida: TL = 8
Prueba y validación		
Si 2 libros =195 y se tienen 780 pesos, entonces: $780/195 = 4$ Entonces Total de libros = $4 \cdot 2 = 8$ Se pueden adquirir 8 libros con 780 pesos porque: $195/2 = 97.5$ Y $97.5 \cdot 8 = 780$ lo que valida y comprueba la validez de la solución y del algoritmo		

2.2. Metodologías para el diseño de software

En el diseño de software existen varias metodologías de uso que permiten el aseguramiento de la calidad del desarrollo de software. A continuación, se describen las más utilizadas.

Metodología de diseño Top Down (desde arriba hacia abajo). Este tipo de metodología permite partir desde el proceso general y posteriormente desarrollar módulos por niveles de detalle. Este enfoque permite diseñar procesos claros y bien organizados, la característica de este modelo consiste en dividir el objetivo general en diversos módulos, las decisiones se toman en el nivel más alto y se transmiten a los demás participantes y la comunicación fluye por las vías establecidas. Comprende 3 fases: Planeación, Diseño e implementación. Es una metodología de diseño descendente, es decir, se debe establecer una serie de niveles de mayor a menor complejidad para dar solución al algoritmo.

Ejemplo de uso. Generar la nómina de empleados de la empresa Seasun S. de RL de CV

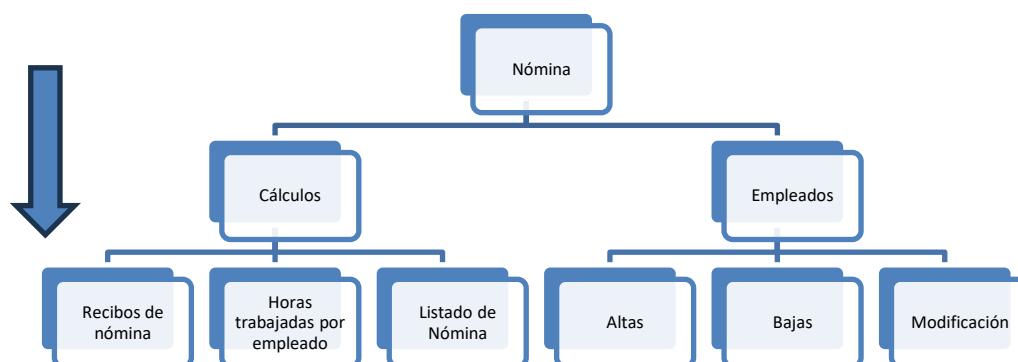


Figura 2.2. Modelo Top Down



Las ventajas que ofrece este modelo son: comunicación más clara, detección de problemas más fácil, implementación de soluciones más rápidas. Las desventajas: menos creatividad, poca injerencia en la toma de decisiones y en consecuencia falta de compromiso en el equipo

Metodología Bottom-up. Contrario al modelo Top Down, es una metodología donde los programadores de todos los niveles de la organización tienen voz y voto en la toma de decisiones, es decir, las ideas, decisiones y aportes se generan desde los empleados de nivel inferior y fluye hacia los niveles superiores. En el diseño descendente se modifica menos cuando se produce un cambio, porque únicamente la gerencia toma decisiones, en el enfoque ascendente por el contrario todos los empleados tienen voz en la toma de decisiones y la información fluye de abajo hacia arriba, por lo que se permite a los desarrolladores aportar ideas y realizar acciones de mejora. Fomenta el trabajo en equipo y es muy utilizada en áreas dónde se manejan presupuestos, ingresos, gastos, entre otros.

Las ventajas de este modelo son: decisiones más informadas, aumenta la creatividad en los equipos de trabajo, los altos directivos trabajan directamente con los miembros del equipo y se evitan puntos ciegos, se comparten opiniones entre los equipos de trabajo y se genera crítica constructiva que a menudo produce mejores resultados. Sin embargo, los procesos se vuelven más lentos cuando participa con todos los participantes a menos que se asignen líderes por equipo de trabajo, muchos empleados no externalan sus opiniones si no se tienen un nivel de confianza en el equipo y los proyectos se afectan cuando falta información de la gerencia relacionada con presupuestos, pronósticos entre otros.



Programación modular.

Consiste en dividir un problema complejo, en subproblemas o módulos más pequeños, con la finalidad de hacer más fácil el proceso de resolución al tratarlo por separado.

En este tipo de programación se espera que cada subprograma o módulo sea independiente entre sí y su ejecución este organizada desde un módulo raíz. Un software modular, es un sistema que se compone de distintos módulos cada uno con su propia funcionalidad y se pueden agregar o eliminar cuando se requiera y no afecta al sistema mismo. Con este modelo se logra un bajo costo en el diseño y construcción de software.

Un ejemplo de ello son los ERP, sistemas de planificación de recursos empresariales, por su complejidad se generan diversos módulos tales como: finanzas, ventas, recursos humanos, inventarios, entre otros, al utilizar la programación modular el mantenimiento y actualización es más sencilla y permiten intercambiar o modificar módulos con un impacto mínimo en el sistema. La figura 2.3, muestra un ejemplo de programación modular.



Figura 2.3 Programación modular



Actualmente la programación modular es eficaz en el diseño de software por su énfasis en la abstracción y encapsulamiento de código con módulos discretos y cohesivos. Su capacidad de mantenimiento, reúso de código y escalabilidad permite aplicaciones eficaces y adaptables por lo que se considera una piedra angular en el mundo de la ingeniería de software (AppMaster, 2020)

Programación estructurada.

Es una forma de programar que utiliza diferentes técnicas para realizar programas fáciles de escribir y validar, las técnicas que incluye son: el diseño descendente (top-down), estructura de datos, estructuras de control y programación modular (Corona Nakamura & Ancona Valdez, 2011)

En la programación estructurada las instrucciones tienen orden descendente, es decir de arriba abajo (top-down). Emplea estructuras de datos donde se almacenan de forma individual y por tanto pueden separarse (estructura de datos), posee estructuras que controlan el orden en las que se ejecutan las instrucciones (estructuras de control) y puede el problema dividirse en módulos o secciones para ser resuelto de una forma más sencilla y en forma paralela, es decir, en equipo. Este tipo de programación es la que detallará en las siguientes secciones de este material.

2.3. Elementos y reglas de los lenguajes algorítmicos.

Al elaborar un algoritmo debe cumplir con las siguientes reglas:

1. Debe tener un inicio y un final al momento de generar las instrucciones.



2. La secuencia lógica de las instrucciones que lo componen debe tener un orden secuencial y/o iterativo cuando así lo requiera el proceso.
3. La ejecución múltiple de las instrucciones, siempre deben reflejar el resultado esperado.

Si se desea formular e implementar un algoritmo es necesario utilizar una técnica de representación: Lenguaje natural, pseudocódigo, diagrama de flujo, o lenguaje de programación.

En la programación estructurada se debe seguir un orden y una estructura para respetar las reglas de un diseño descendente, estructurado y secuencial. A continuación, se plasma el orden en el que se recomienda escribir un algoritmo, empleando como base la estructura del entorno PseInt el cuál se utilizará y explicará en el tema 2.5.

Elementos y estructura de un algoritmo:

Algoritmo <nombre del programa>

Funciones y/o procedimientos (opcional)

Declaración de constantes

Declaración de variables

Cuerpo del programa

FinAlgoritmo

Reglas de un algoritmo

1. Son secuenciales. Las instrucciones se ejecutan una después de la otra y de arriba hacia abajo.



2. Precisos. Al resolver un problema debe ser objetiva la solución.
3. Ordenados. Deben leerse y ejecutarse las instrucciones de forma precisa.
4. Finitos. Existe un determinado número de pasos, es decir, deben tener un fin
5. Su estructura básica la integran:
 - a. **Entrada de datos.** Ingreso de datos para ser utilizados
 - b. **Proceso.** Conjunto de instrucciones a realizar una tarea específica
 - c. **Salida de datos.** Resultados que se obtienen después del procesamiento.

Un algoritmo se puede escribir y representar de diferentes formas: en lenguaje natural, diagrama de flujo, pseudocódigo y en lenguaje de programación.

2.3.1. Diagramas de flujo.

Los diagramas de flujo son representaciones que se realizan mediante el uso de figuras para la realización de algoritmos. El uso de símbolos y figuras se relacionan entre sí mediante flechas o conectores que indican el orden o secuencia de las instrucciones para lograr el resultado esperado en el proceso de resolución a un problema. Los aspectos generales y reglas en el uso de los diagramas de flujo son:

1. Todo diagrama debe tener inicio y fin.
2. No se definen o declaran variables.
3. Las líneas de flujo deben ser horizontal o vertical.
4. Las líneas de flujo no deben cruzarse.
5. El uso de conectores debe ser limitado o solo cuando sea muy necesario.
6. Todas las líneas de flujo deben tener conexión.
7. La lectura del diagrama de flujo debe ser de manera vertical de arriba a abajo u horizontal de izquierda a derecha.
8. No se permite el uso de términos de algún lenguaje de programación.



9. Cuando el diagrama ocupa más de una página debe numerarse e indicar con conectores el flujo de la información.
10. El símbolo de decisión es el único con más de una línea de flujo.
11. Respetar la simbología establecida.

La tabla 2.1, muestra los símbolos estándares más utilizados el instituto nacional de normalización estadounidense. (ANSI, *American National Standards Institute*).

Tabla 2.1 simbología de un diagrama de flujo

Símbolo	Uso	Símbolo	Uso
	Inicio/fin del algoritmo		Dirección o líneas de flujo de las acciones y/o procesos
	Lectura/Salida de datos		Conector
	Indica comparación entre valores		Disco magnético
	Proceso. Utilizado para operaciones aritmética-lógica y/o de asignación de valores en un espacio de memoria		Proceso predefinido

Ejemplo de uso

Obtener el promedio de 3 calificaciones. Los datos de entrada serán las 3 calificaciones que deberá ingresar desde teclado el usuario del programa, el proceso es aplicar la fórmula del promedio, sumando las tres calificaciones y

posteriormente dividir el resultado entre 3, y por último, los datos de salida será el promedio obtenido. La tabla 2.2, plasma los elementos del algoritmo y la figura 2.4, muestra el diagrama de flujo correspondiente.

Tabla 2.2 Elementos de un algoritmo

Datos de entrada	Proceso	Datos de salida
Solicitar al usuario 3 calificaciones y almacenar valores en 3 variables: X1, X2, X3	Suma= $X1+X2+X3$ Promedio=Suma/3	Imprimir Promedio

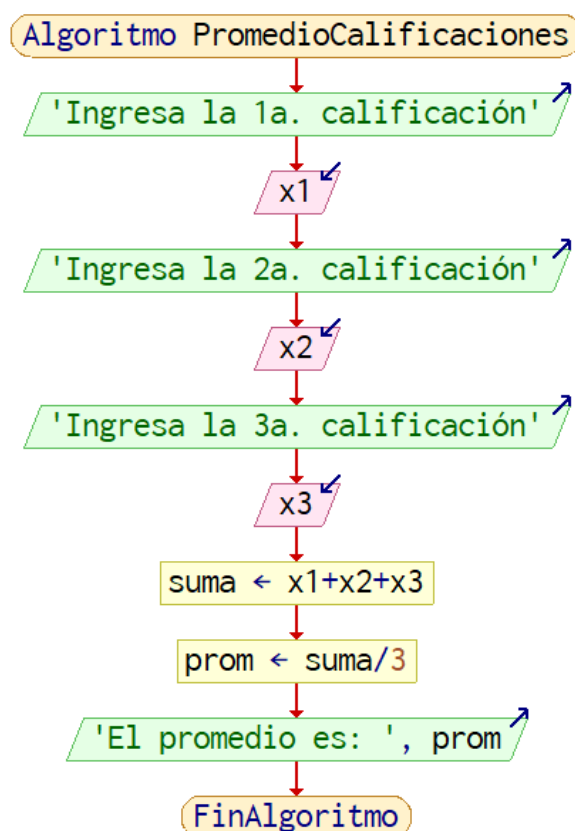


Figura 2.4. Diagrama de flujo para obtener promedio

Diagrama generado con el entorno PseInt



Ejercicio de aprendizaje

Un turista adquiere una cantidad determinada de los artículos que vende una artesana para llevar a sus familiares. Elaborar el diagrama de flujo que permita calcular el total de compra que debe pagar el turista a la artesana sabiendo que el precio de venta de cada artículo es el siguiente: muñeca \$145.00, balero \$60.00, juguete de madera \$180.00 e ingresando la cantidad de cada artículo adquirido. Imprimir el total de compra

2.3.2. Pseudocódigo.

El pseudocódigo es una técnica utilizada para la representación de algoritmos, emplea el uso de lenguaje natural, símbolos y términos empleados en la programación. El pseudocódigo es la técnica más sencilla de utilizar en el planteamiento de un problema ya que a diferencia del diagrama de flujo su elaboración es más parecida al uso del lenguaje humano. Las reglas de construcción de un pseudocódigo son:

1. Tener un inicio y un final
2. La secuencia de las instrucciones deben ser lógicas y secuenciales
3. Se permite el uso de símbolos matemáticos, operadores lógicos, operadores relacionales, etc.
4. Se permite el uso de términos similares a un lenguaje de programación, pero en su representación de lenguaje natural, por ejemplo: repetir, mientras, para, entre otras.
5. El uso de identificadores debe seguir las reglas de construcción.



2.4. Implementación de algoritmos.

El algoritmo, según Corona&Ancona, (2011), **“es un conjunto de pasos ordenados y finitos que permiten resolver un problema o tarea específica”** (Corona Nakamura & Ancona Valdez, 2011). Un algoritmo se utiliza para la formulación de instrucciones que permitan resolver problemas o tareas específicas mediante el uso una computadora, es el primer paso en la resolución de problemas, y aunque es independiente al lenguaje de programación y/o entorno de uso, es considerada una técnica que facilita al programador la comprensión de un problema. Existen 3 metodologías al implementar un algoritmo: de forma secuencial, selección o iterativa/repetitiva mismas que se explicaran a continuación.

2.4.1. Secuenciales.

Un algoritmo secuencial es aquel cuyas instrucciones como su nombre los dice se plasman en forma de secuencia, tiene un inicio, una o varias acciones o instrucciones y un final. En este tipo de estructura las instrucciones se ejecutan en orden cronológico o de aparición desde el inicio hasta el final de este, sin saltar o ignorar ninguna de las sentencias.

Ejemplo de uso:

Planteamiento del problema. Una tienda de abarrotes vende bolsos con un costo de 180 c/u. Imprimir el total de la compra realizada en función al número de artículos comprados y el desglose del IVA del 16%. Pasos en la formulación e implementación del algoritmo mediante la técnica seleccionada:



1. **Análisis del problema.** Detectar los datos de entrada (valores conocidos y valores que deben ingresar desde teclado) y el proceso (fórmulas requeridas):

Datos de entrada	Proceso
Valores conocidos: Producto: Bolso \$180 IVA: 16% Valores de entrada: cantidad comprada	Fórmulas requeridas: Total de compra= costo * cantidad IVA a pagar= Total de compra*0.16 Total a pagar= Total de compra + IVA a pagar

2. **Técnica para representar el algoritmo:** Pseudocódigo

3. **Codificación**

Algoritmo **Compra**

Leer Cantidad

TotalCompra= 180 * Cantidad

IVA= Total de compra*0.16

Total = TotalCompra + IVA a pagar

Imprimir "Bolso ", Cantidad

Imprimir "Total de compra: ", TotalCompra

Imprimir "IVA: ", IVA

Imprimir "Total a pagar: ", Total

FINAlgoritmo

4. **Prueba de escritorio.** Se debe realizar siguiendo paso a paso las instrucciones plasmadas como se muestra a continuación, del lado izquierdo se muestra el pseudocódigo y del lado derecho el o los valores que el programa ejecuta.



Código del programa

Algoritmo Compra

1. Leer Cantidad
2. $\text{TotalCompra} = 180 * \text{Cantidad}$
3. $\text{IVA} = \text{Total de compra} * 0.16$
4. $\text{Total} = \text{TotalCompra} + \text{IVA a pagar}$
5. Imprimir "Bolso ", Cantidad
6. Imprimir "Total de compra: ", TotalCompra
7. Imprimir "IVA: ", IVA
8. Imprimir "Total a pagar: ", Total

Prueba de escritorio

Valores

Cantidad= 4
 $\text{TotalCompra} = 180 * 4 = 720$
 $\text{IVA} = 720 * 0.16 = 115.20$
 $\text{Total} = 720 + 115.2 = 835.20$
 Bolso: 4
 Total de compra: 720
 IVA: 115.2
 Total: 835.20

FINAlgoritmo

Esta prueba y validación se puede realizar con diversos valores siguiendo el paso a paso de las instrucciones y de las variables y procesos que intervienen en cada una de ellas.

Ejercicios de aprendizaje. Realiza el diagrama de flujo y la prueba de escritorio de los siguientes pseudocódigos

1) Un cliente desea invertir su dinero en un banco y desea conocer cuánto ganará si el banco paga a razón de 2% de interés mensual.

Algoritmo inversión

Escribir "Ingresa el total a invertir"

Leer inversión

$\text{Ganancia} = \text{inversión} * 0.002$

Escribir "El interés mensual logrado es: ", Ganancia

FinAlgoritmo



2) Un trabajador percibe su sueldo base y un 10% extra por comisión de sus ventas, imprimir cuánto dinero obtendrá el trabajador, por concepto de comisiones de tres ventas que realizó durante un mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

Algoritmo sueldos

Escribir “Ingresa sueldo base, venta1, venta2, venta3”

Leer sueldo, venta1, venta2, venta3

*Comisiones= (venta1+venta2+venta3)*0.10*

totalPagar = sueldo + Comisiones

Escribir “El salario a pagar es”, totalPagar

FinAlgoritmo

3) Imprimir el total a pagar y la cantidad que se le descontó a un cliente, si la tienda realiza un 20% de descuento en el total de su compra.

Algoritmo Tiquet

Escribir “Ingresa el total de compra”

Leer compra

*Descuento= compra * 0.20*

Total = Compra – Descuento

Escribir “El total a pagar es: “ , Total

Escribir “Usted ahorro: “ , Descuento

FinAlgoritmo

4) Calcular la calificación final de un estudiante, tomando en consideración que la calificación se obtiene con los siguientes porcentajes

50% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

20% de la calificación de tareas.



Algoritmo Calificaciones

Escribir “ingresa las tres calificaciones parciales”

Leer a1, a2, a3

Escribir “ingresa calificación de examen final”

Leer EF

Escribir “Ingresa Calificación de las tareas

Leer T

$$prom = (a1 + a2 + a3)/3$$

$$V1 = prom * 0.50$$

$$V2 = EF * 0.30$$

$$V3 = T * 0.20$$

$$Final = V1 + V2 + V3$$

Escribir “ La calificación obtenida es: “ , Final

FinAlgoritmo

5) Calcular e imprimir el porcentaje de hombres y de mujeres hay en un grupo de estudiantes.

Algoritmo porcentaje

Escribir “Ingresa el total de hombres y el total de mujeres”

Leer TH, TM

$$Total = TH + TM$$

$$H = TH * 100 / Total$$

$$M = TM * 100 / Total$$

Escribir “El porcentaje de Hombres es: “ , H

Escribir “El porcentaje de mujeres es: “ , M

FinAlgoritmo



Aprendizaje basado en problemas

Siguiendo las fases: Análisis e identificación del problema, Alternativas de solución, Técnica de representación, codificación y pruebas de escritorio, resuelve con pseudocódigo y diagrama de flujo, cada uno de los siguientes problemas.

1. Calcular la masa de aire utilizando la siguiente fórmula:

$$\text{masa} = (\text{presión} * \text{volumen}) / (0.37 * (\text{temperatura} + 460))$$

2. Calcular el número de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la fórmula es:

$$\text{número de pulsaciones} = (220 - \text{edad})/10$$

3. Una PYMES cuenta con 3 áreas: Contabilidad, Recursos humanos y Servicio médico. El presupuesto anual se reparte conforme a la siguiente manera: Contabilidad 30%, recursos Humanos 45% y servicio médico 25%. Obtener la cantidad de dinero que recibirá cada área en el año, ingresando el monto presupuestal.
4. El dueño de una tienda de abarrotes adquiere un producto a un precio determinado. Calcular el precio de venta, si el vendedor desea obtener una ganancia del 45%.
5. Tres personas invierten una cantidad de dinero para generar una empresa. Obtener e imprimir el porcentaje que cada persona invirtió tomando en cuenta la cantidad total invertida.
6. Obtener el promedio general de un estudiante si curso 3 asignaturas, tomando en cuenta los porcentajes asignados en el examen y en las tareas de cada una de ellas, de acuerdo con la tabla de valores que se muestra a continuación:

Asignatura	Examen	Tarea
Algoritmos	80%	20%
Ética	90%	10%
Química	85%	15%

2.4.2. Selectivos.

Este tipo de estructura se controla mediante una condición o requisito. La condición puede ser verdadera o falsa y en función de la valoración se ejecutan las instrucciones de la opción correspondiente. En la figura 2.5, se muestra el pseudocódigo y diagrama de flujo de una estructura de selección

Algoritmo SELECCION

```
Si condición Entonces
    Escribir 'acciones Verdaderas'
SiNo
    Escribir 'acciones Falsas'
FinSi
```

FinAlgoritmo

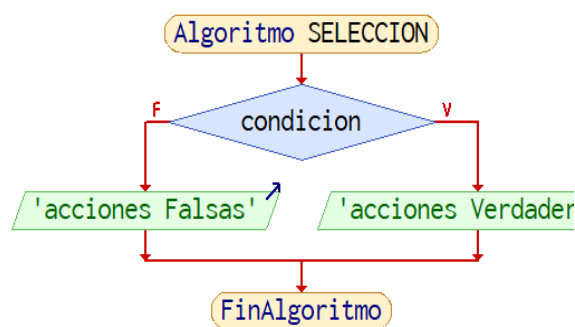


Figura 2.5 Estructura selectiva simple

Ejemplo de uso

Un cliente adquiere un producto en una tienda departamental. Si el precio del producto es mayor o igual a 2000 se le otorga un 20% de descuento sobre el precio del artículo

1. Identificación del problema y análisis de datos

Datos de entrada

Variable conocida: Si Condición $\text{Precio} \geq 2000$, Descuento otorgado 20%

Variable de entrada: Precio del producto

Proceso

Si $\text{precio} \geq 2000$ entonces $\text{descuento} = \text{precio} * 0.20$



Datos de salida

Total a pagar

2. Técnica de representación.

Pseudocódigo y diagrama de flujo

3. Codificación

Pseudocódigo

Algoritmo Venta

Escribir "Ingresa precio del producto: "

Leer precio

Si $\text{precio} \geq 2000$ entonces

$\text{Total} = \text{precio} - (\text{precio} * 0.2)$

Imprimir Total

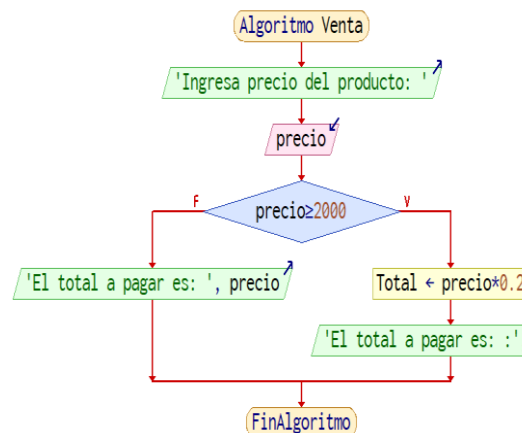
Sino

Imprimir precio

FinSi

FinAlgoritmo

Diagrama de flujo



4. Prueba de escritorio

Algoritmo Venta

Escribir "Ingresa precio del producto: "

Leer precio

Si $\text{precio} \geq 2000$ entonces

$\text{Total} = \text{precio} - (\text{precio} * 0.2)$

Imprimir Total

Sino

Imprimir precio

FinSi

FinAlgoritmo

Prueba de escritorio

Ingresa precio del producto

3500

Evalúa condición

Cumple: $\text{Total} = 3500 - 700 = 2800$

2800

Si se ingresa 1800 por ejemplo

La condición no se cumple e imprime 1800

En ambos casos se evalúa la condición



En este tipo de estructuras pueden existir una o más condiciones y se puede ejecutar solo la parte verdadera de la condición, es decir, la instrucción o valoración falsa es opcional.

Ejemplo de uso: Si un estudiante tiene más de 18 años y tiene un promedio mayor o igual a 85 se acepta como candidato a beca.

Codificación con ambas opciones:

Algoritmo PromedioCalificaciones

 Escribir "ingresar edad: "

 Leer edad

 Escribir "Ingresar promedio de calificaciones: "

 Leer promedio

 Si (edad ≥ 18) y (promedio ≥ 85) entonces

 Escribir "Seleccionado como candidato a obtener beca"

 Sino

 Escribir "No fuiste seleccionado"

FINAlgoritmo

Codificación donde solo se ejecuta la parte verdadera

Algoritmo PromedioCalificaciones

 Escribir "ingresar edad: "

 Leer edad

 Escribir "Ingresar promedio de calificaciones: "

 Leer promedio

 Si (edad ≥ 18) y (promedio ≥ 85) entonces

 Escribir "Seleccionado como candidato a obtener beca"

FINAlgoritmo

En el primer caso existe una salida de datos tanto en la parte verdadera como en la falsa de las condiciones, en el segundo caso solo se genera información de salida si la condición se cumple y en ambos casos no se afecta la información de salida.



El uso de la estructura selectiva puede emplearse de manera compuesta, es decir, al realizar la comparación de una condición o varias, las acciones para alguna de las alternativas: verdadera o falsa, será nuevamente otra estructura condicional, este tipo de estructura se conoce con el nombre de estructuras anidadas. Un ejemplo de uso se plasma a continuación:

Se desea seleccionar estudiantes para asistir a un congreso, se aceptarán estudiantes cuya edad sea mayor o igual a 18 años y menor o igual a 25, si el promedio obtenido en el semestre anterior es mayor o igual a 85 se le cubre el 50% del costo del congreso, en caso contrario solo se le hace un 10% de descuento. El costo del congreso es de \$3,500.00. Imprimir si el estudiante es seleccionado y el tipo de descuento que logró.

Algoritmo Selección

Escribir "Ingresa Edad"

Leer edad

Escribir "Ingresa Promedio"

Leer promedio

Si edad ≥ 18 y edad ≤ 25 entonces

Si promedio ≥ 85 entonces

Escribir "Seleccionado"

Pago= $3500 * 0.50$

Escribir "El pago del congreso es: ", Pago

sino

Escribir "Seleccionado"

Pago= $3500 * 0.10$

Escribir "El pago del congreso es: ", Pago

FinSi

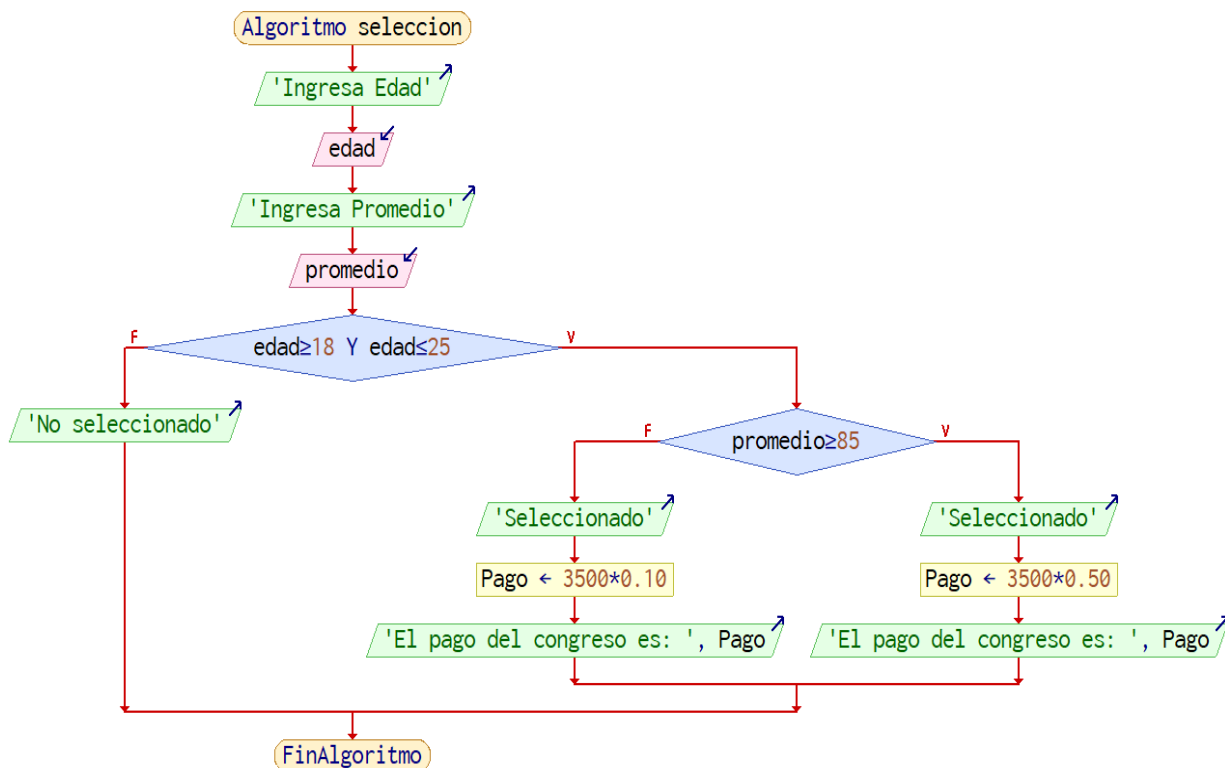
sino

Escribir "No seleccionado"

FinSi

FinAlgoritmo

Diagrama de flujo



Ejercicio de aprendizaje

Una trabajadora, desea saber cuánto dinero se genera por concepto de intereses sobre la cantidad que tiene invertida en el banco. La decisión de reinvertir los intereses dependerá de si el rendimiento que recibe es mayor a \$9,000, y en ese caso desea saber cuánto dinero tendrá finalmente en su cuenta. Realiza una alternativa de solución empleando estructuras condicionales y representando el algoritmo en pseudocódigo y diagrama de flujo



Estructura de selección múltiple.

Este tipo de estructura permite seleccionar una opción entre muchas que existen, dependiendo de la alternativa seleccionada. La estructura evalúa una expresión y ejecuta las instrucciones que correspondan. En el uso de pseudocódigo se utiliza con la palabra reservada **según**. La figura 2.6 Muestra la sintaxis de uso y el diagrama de flujo correspondiente.

Sintaxis de uso:

Según <opción>

<Opción 1>: Instrucciones

<Opción 2>: Instrucciones

...

<Opción n>: Instrucciones

De otro modo: Mensaje o instrucciones

FinSegun

Diagrama de flujo

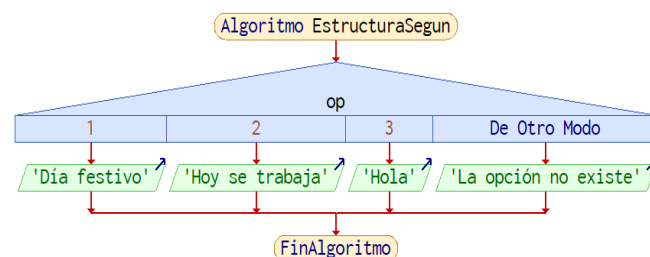


Figura 2.6. Estructura de selección múltiple.

En esta estructura en cada una de las opciones o alternativas se describen las instrucciones a realizar de acuerdo con la opción seleccionada, pero si se ingresa un valor que no coincida con las opciones entonces De otro modo, permite colocar un mensaje o instrucciones cuando esto sucede.

Ejemplo de uso

Carolina, estudiante universitaria es muy olvidadiza y necesita recordar que pagos debe realizar para no pagar intereses. Sus pagos de mes se distribuyen en función



al número de semanas que tiene un mes: en la semana 1 debe pagar luz e internet, la semana 2 pagará la colegiatura, la semana 3, realiza el pago de los insumos para preparar comida y la semana 4 el pago de gasolina. Realiza un planificador para que en función del número de semana ingresado le indique a Carolina que pagos debe realizar.

Alternativa de solución.

Algoritmo Pagos

Escribir “Ingresa el número de semana: “

Leer opción

Según (opción)

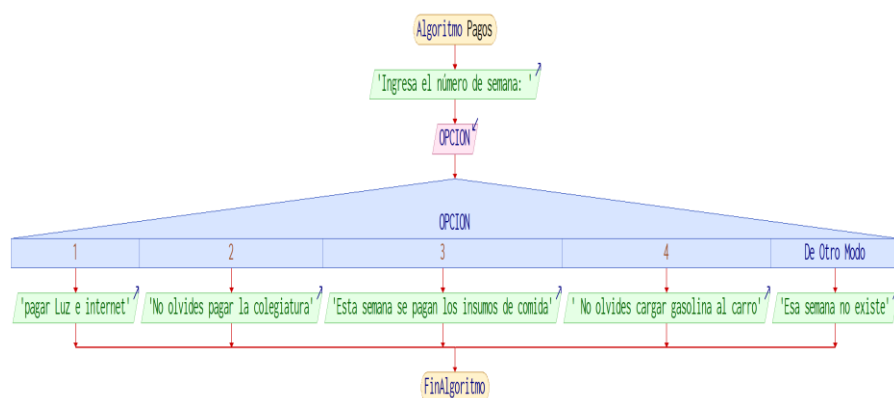
- 1: Escribir “pagar Luz e internet”
- 2: Escribir “No olvides pagar la colegiatura”
- 3: Escribir “Esta semana se pagan los insumos de comida”
- 4: Escribir “No olvides cargar gasolina al carro”

De otro modo: Escribir “Esa semana no existe”

FinSegun

FinAlgoritmo

Diagrama de flujo





Ejercicio de aprendizaje

Una trabajadora, desea calcular por semana cuánto dinero tiene una vez que realiza los pagos relacionados a la semana para saber cuánto dinero puede gastar en cine, compras o viajes esa semana. Realiza una alternativa de solución empleando estructura selectiva múltiple para calcular en función al ingreso de la semana que gastos tiene y cuánto dinero le queda para sus gustos adicionales, representando el algoritmo en pseudocódigo y diagrama de flujo

2.4.3. Repetitivos.

La repetición, es una instrucción o instrucciones que se ejecutan varias veces, siempre y cuando se cumpla una condición. Para que la repetición finalice debe existir dentro de la acción una modificación de la o las variables de control que interviene en la condición o condiciones. La condición puede estar predefinida o no dependiendo del tipo de repetición que se desee ejecutar en un programa. Las estructuras de control repetitivas o iterativas permiten que una o varias acciones se ejecuten una cantidad de veces de acuerdo con una condición ya sea simple o compuesta. En programación existen 3 tipos de estructuras iterativas: **para**, **mientras** y **repetir**, las cuáles se explican a continuación:

Estructura iterativa PARA.

Es una estructura que se utiliza para realizar una o varias instrucciones una cantidad conocida de veces, es decir, repite las instrucciones, un determinado número de

veces. (Figura 2.7). Es la estructura más sencilla de utilizar y se emplea con el uso de variables compuestas: arreglos y matrices (Ver tema 5).

Sintaxis de uso:

Para <variable>= <valor inicial> **hasta** <valor final> **con paso** <incremento>
hacer

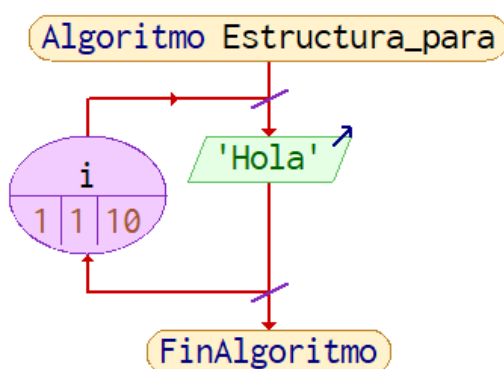


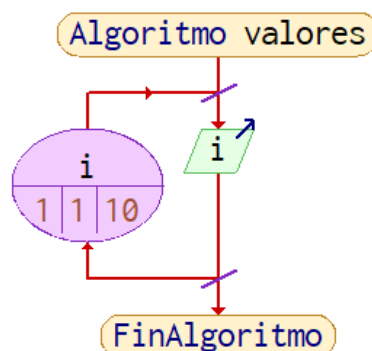
Figura 2.7 Diagrama de flujo de la estructura Para.

Ejemplo de uso. Imprimir los valores del 1 al 10

Algoritmo

PARA i=1 hasta i=10 con paso 1 hacer
 Imprimir i
FINAlgoritmo

Diagrama de flujo





Ejercicio de aprendizaje

Utilizando la estructura PARA. Imprimir los múltiplos de 5 encontrados en el rango de valores del 1 al 50. Genera el pseudocódigo y el diagrama de flujo.

Estructura iterativa REPETIR

Esta estructura se utiliza **cuando al menos una vez se desea ejecutar una acción(es)** y después, se evalúa una condición para volverse a ejecutar o no. La sintaxis y diagrama de flujo correspondientes se observan en la figura 2.8.

Sintaxis de uso:

Pseudocódigo

Repetir

<instrucciones>

Hasta que <condición>

Diagrama de flujo

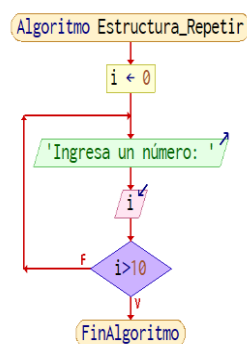


Figura 2.8 Estructura Repetir.

Ejemplo de uso

Imprimir el sueldo de uno o varios empleados



INICIO Algoritmo

REPETIR

Escribir "Ingresa Clave empleado:"

Leer clave

Escribir "Ingresa horas trabajadas:"

Leer HT

$\text{Sueldo} = \text{HT} * 120$

Imprimir Sueldo

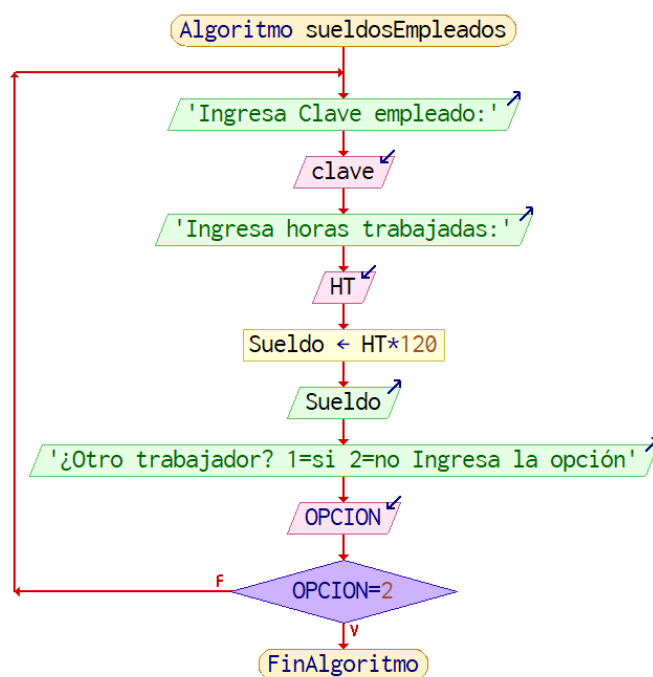
Escribir "¿Otro trabajador? 1=si 2=no Ingresa la opción"

Leer opción

HASTA opción=2

FIN Algoritmo

Diagrama de flujo





Ejercicio de aprendizaje

Leer un número entero positivo. El programa debe imprimir los números pares que se encuentran entre el 1 y el número leído

Estructura iterativa MIENTRAS.

Esta estructura se emplea cuando se desea evaluar una condición antes de realizar un proceso. Solo si se cumple la condición o condiciones, se ejecuta el proceso y la repetición termina cuando la condición no se cumpla, a diferencia de la estructura REPETIR, esta estructura se ejecuta 0 o más veces. La sintaxis y diagrama de flujo se muestra en la figura 2.9.

Sintaxis de uso

Pseudocódigo

```
Mientras <cond> hacer  
    <instrucciones>  
FinMientras
```

Diagrama de flujo

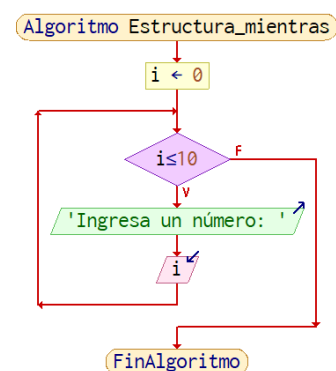


Figura 2.9 Estructura iterativa Mientras



Ejemplo de uso

Leer un conjunto de valores si es múltiplo de 4 imprimirlo, el programa debe finalizar cuando se ingresen 5 múltiplos

Algoritmo Múltiplos

V=0

Mientras (V<=5) hacer

Escribir "ingresa un número: "

Leer N

Si (N mod 4 = 0) entonces

Imprimir N, "Es múltiplo de 4"

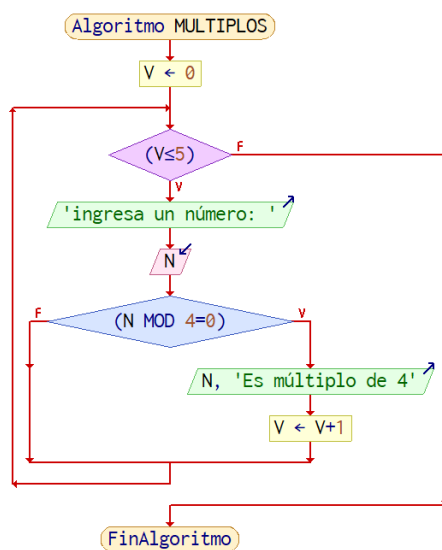
V=V+1

FinSi

FinMientras

FinAlgoritmo

Diagrama de flujo





Ejercicio de aprendizaje

Leer un conjunto de números y cuándo se ingresen 5 números que sean pares terminar la captura de números

2.5. Pruebas y depuración.

La importancia de automatizar procesos radica en la veracidad y validez de los datos que proporciona el programa en los resultados que se obtienen. Al realizar un programa es necesario realizar la prueba y depuración de estos para poder garantizar su veracidad y validez. Para llevar a cabo esta fase se utilizan 2 métodos: Prueba de escritorio y verificación paso a paso en un entorno de programación.

1. Prueba de escritorio. Es una técnica que permite que el programador verifique la validez y veracidad de su programa antes de capturar en el entorno, es una técnica que permite detectar con oportunidad algún tipo de error de lógica o de secuencia cometido. La prueba de escritorio consiste en seguir paso a paso un pseudocódigo o diagrama de flujo una vez que se ha realizado con diferentes datos de entrada, es necesario realizar varias pruebas para tener la certeza de que el pseudocódigo o diagrama de flujo funciona correctamente.

A continuación, se plasma un ejemplo donde de lado izquierdo se muestra el pseudocódigo y de lado derecho la prueba de escritorio que consiste en seguir paso a paso cada instrucción del programa y asignado valores a las variables o ejecutando la instrucción se verifique su calidad. Cuando se detecta un error se corrige o depura el programa para corregirlo y nuevamente se vuelve a verificar las veces que sea necesario.



Algoritmo Venta

Escribir "Ingresa precio del producto: "

Leer precio

Si $\text{precio} \geq 2000$ entonces

$\text{Total} = \text{precio} - (\text{precio} * 0.2)$

Imprimir Total

Sino

Imprimir precio

FinSi

FinAlgoritmo

Prueba de escritorio

Ingresa precio del producto

3500

Evalúa condición

Cumple: $\text{Total} = 3500 - 700 = 2800$

imprime 2800

Si se ingresa 1800 la condición no se cumple e imprime 1800

En ambos casos se evalúa la condición

Cuando un pseudocódigo contiene varias variables se aconseja generar una tabla con las variables que utiliza el programa y se vaya registrando el valor que se obtiene siguiendo el paso a paso de las instrucciones que contiene dicho algoritmo. Por ejemplo:

Pseudocódigo

Prueba de escritorio

Suponer un precio 350

Algoritmo ejemplo

1. Escribir "ingresa precio de venta 1

2. Leer precio

3. $X = \text{precio} * 1.25$

4. $\text{Ganancia} = X + \text{precio}$

5. Escribir "Precio de venta", precio

6. Escribir "Ganancia lograda: ", Ganancia

FinAlgoritmo

Línea código	Precio	X	Ganancia
2	350		
3		437.50	
4			87.50

Imprime 437.50

Imprime 87.50

El entorno de PseInt como herramienta

PSeInt, es un software que desarrollo Pablo Novara y lanzó en diciembre del 2004, para aquellas personas principiantes en la programación y que es de mucha utilidad



para todos los que desean iniciar por ser un entorno muy amigable y de fácil uso e instalación. Este entorno permite entre muchas otras acciones capturar, compilar y ejecutar programas en pseudocódigo, (Prueba y depuración). Para fines de usar esta herramienta, a continuación, se describen los pasos para instalar y hacer uso de PSeInt. (SourceForge, 2024)

Paso 1. Ingresar al siguiente enlace: <https://pseint.sourceforge.net/> (Figura 2.11)

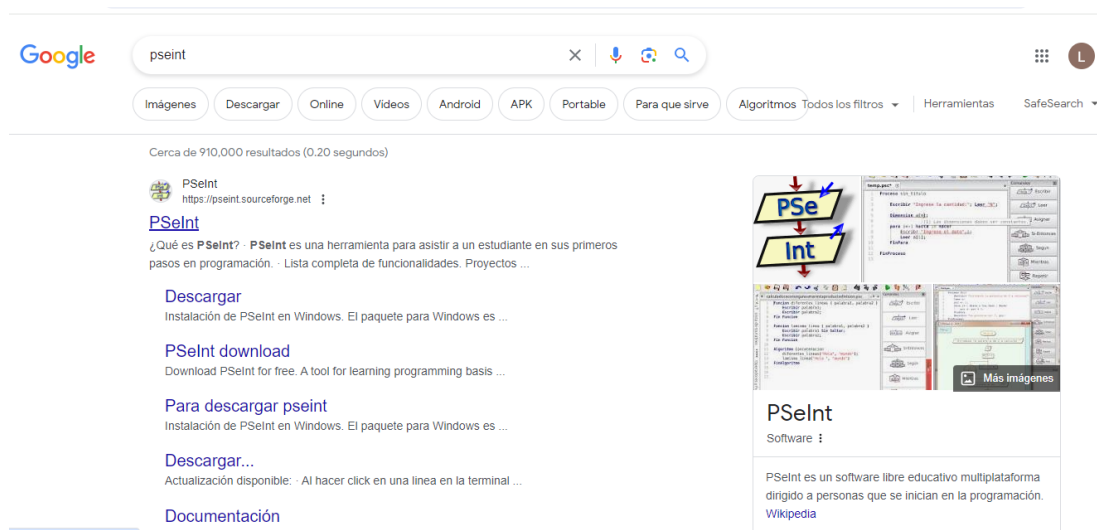


Figura 2.11 Entorno PSeInt

Paso 2. Elegir el tipo de equipo de computadora o medio de instalación (Figura 2.12)



GOBIERNO DE
MÉXICO

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

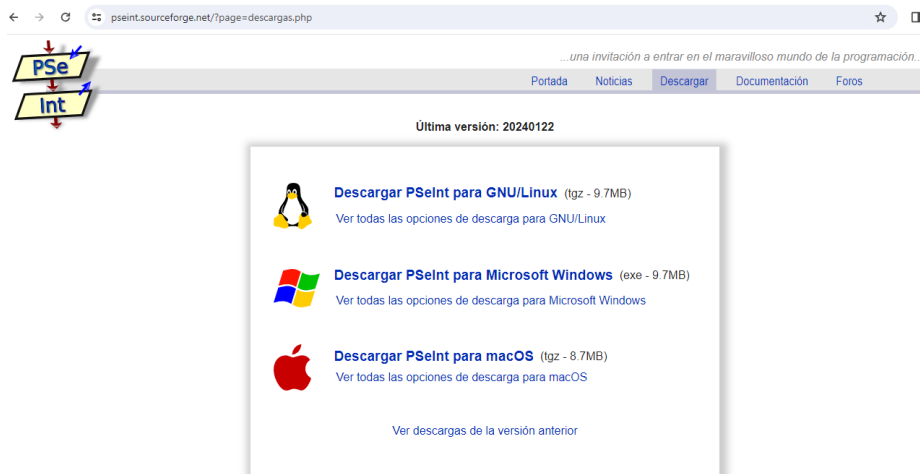


Figura 2.12 Elegir opción según Sistema operativo

Paso 3. Descargar el archivo ejecutable (Figura 2.13)

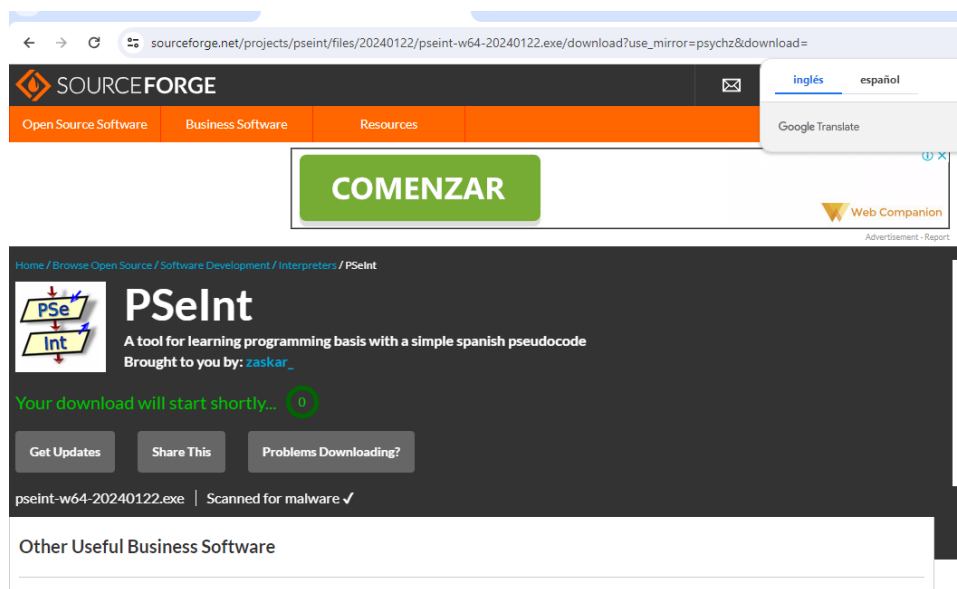


Figura 2.13 Descargar Entorno PSeInt

Paso 4. En el entorno de escritorio de su equipo, buscar en la carpeta de descargas el archivo y ejecutar el archivo como administrador permitiendo la ejecución cuando el sistema le solicite el permiso de instalar (Figura 2.14)

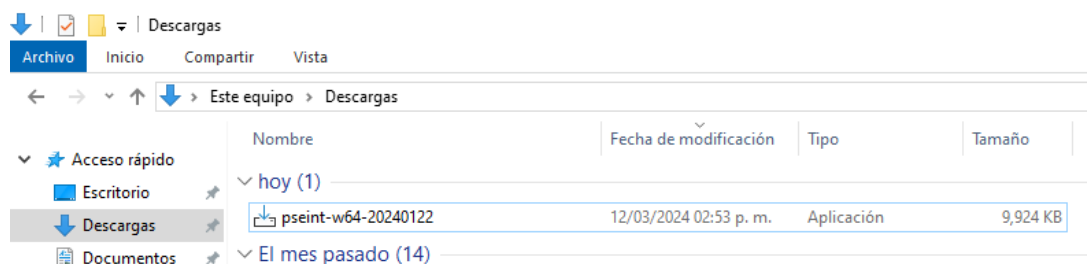


Figura 2.14 Ejecutar entorno PSeInt

Paso 5. Iniciar la descarga en el equipo o medio seleccionado tomando en consideración las peticiones que el entorno le irá solicitando. Se recomienda solo oprimir opciones de aceptar o siguiente según sea el caso y no cambiar directorios (Figura 2.15)

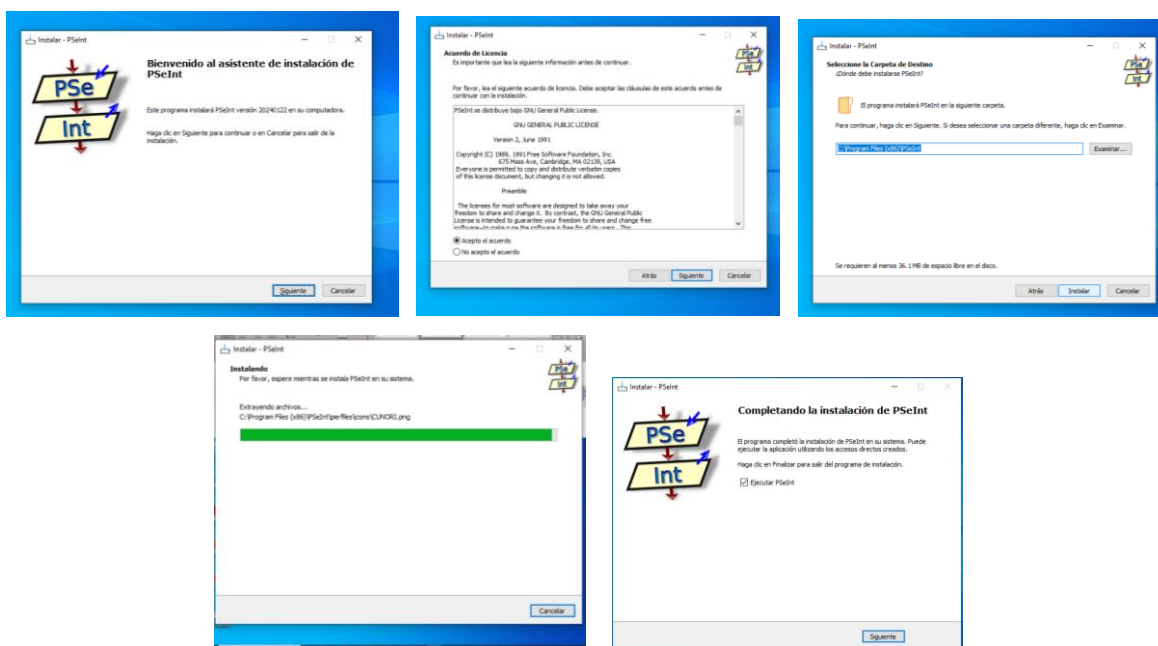


Figura 2.15 Pantallas durante la instalación

Paso 6. Una vez completada la instalación debe elegir un perfil. Se recomienda elegir el perfil flexible (Figura 2.16).

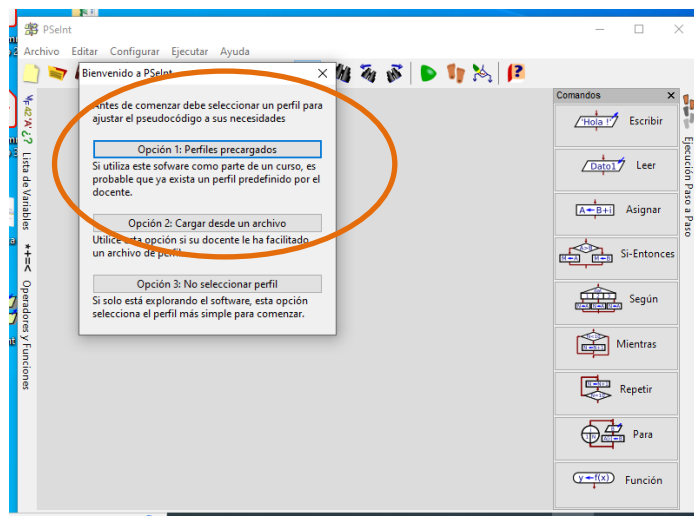


Figura 2.16 Perfil flexible entorno PSeInt

Paso 7. El entorno está listo para ser utilizado y si puede observar es un ambiente amigable que en su barra de opciones superior le muestra los comandos y acciones que puede generar: abrir archivos, editar, compilar, ejecutar, entre otras. (Figura 2.17)

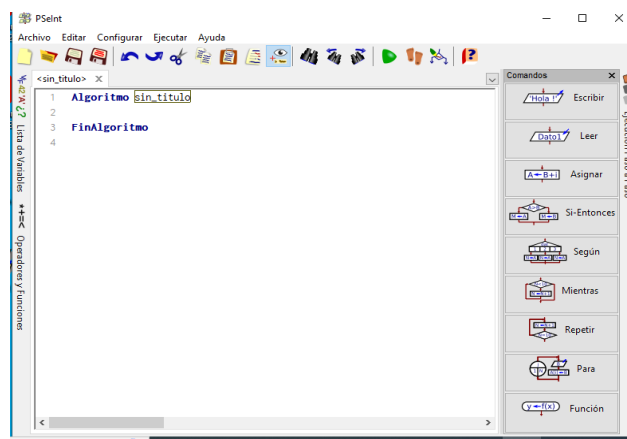


Figura 2.17 Editor y entorno de PSeInt

El entorno de PSeInt contiene varias ventanas. La ventana lateral derecha muestra los comandos y estructuras para construir el pseudocódigo, mismo que puede ocultarse o visualizarse cada que se oprima <Enter> en la barra donde figuran los encabezados; la ventana lateral izquierda, muestra los operadores y las funciones que son usadas y permitidas por el programa. En la parte superior la barra de opciones, la ventana central es el editor del entorno y la parte inferior muestra la depuración y compilación del programa² (Figura 2.18).

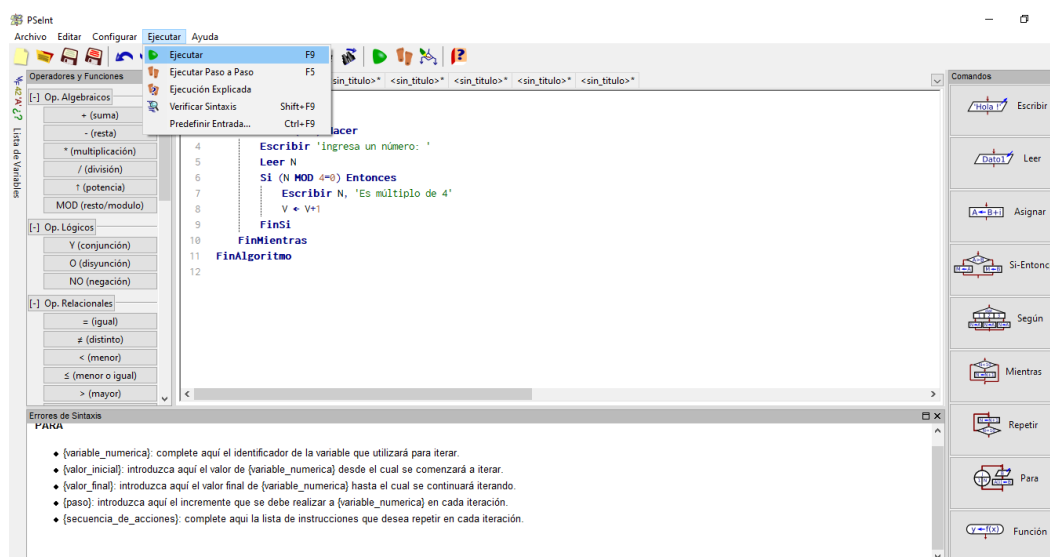


Figura 2.18 Entorno PSeInt

Uso del entorno PSeInt.

Paso 1. Una vez que se realice la alternativa de solución de un problema dado, capturar en el entorno y guardar el archivo con un nombre. Si el programa tiene

² También puede consultar el video de PseInt que se incluye en los materiales de apoyo

errores de sintaxis el mismo editor los marcará y será hasta que este escrito con la sintaxis correcta que permitirá su ejecución. (Figura 2.19)

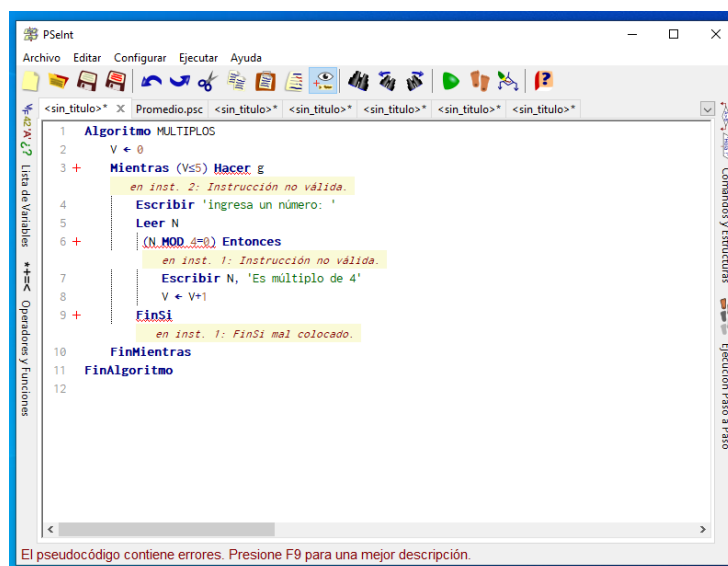


Figura 2.19 Entorno PSeInt

Paso 2. Una vez que el programa no contiene errores es importante guardar el archivo para que si ocurre un error de ejecución no se pierda el archivo. Recuerde que solo se trabaja con memoria principal y por ello la importancia de su almacenamiento correspondiente. (Figura 2.20)

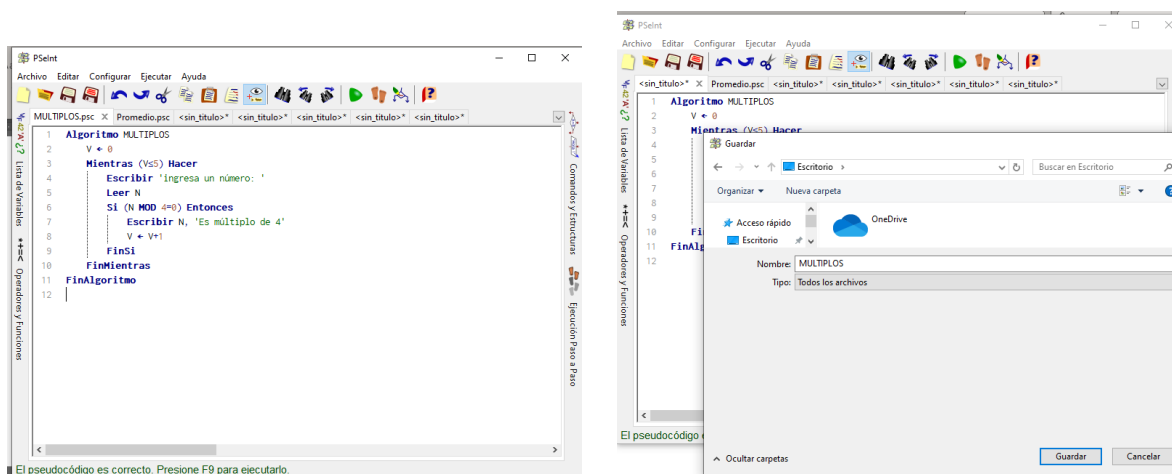


Figura 2.20 Depurar programa y guardar archivo

Paso 3. Procede entonces ejecutar el programa (prueba), para verificar la veracidad de sus resultados. Para este paso el entorno ofrece ejecutar paso a paso cada instrucción si se oprime el símbolo de pies, como se observa en la figura 2.21. Si, por el contrario, se oprime el símbolo con el triángulo verde, el programa se ejecuta completamente (Figura 2.22)

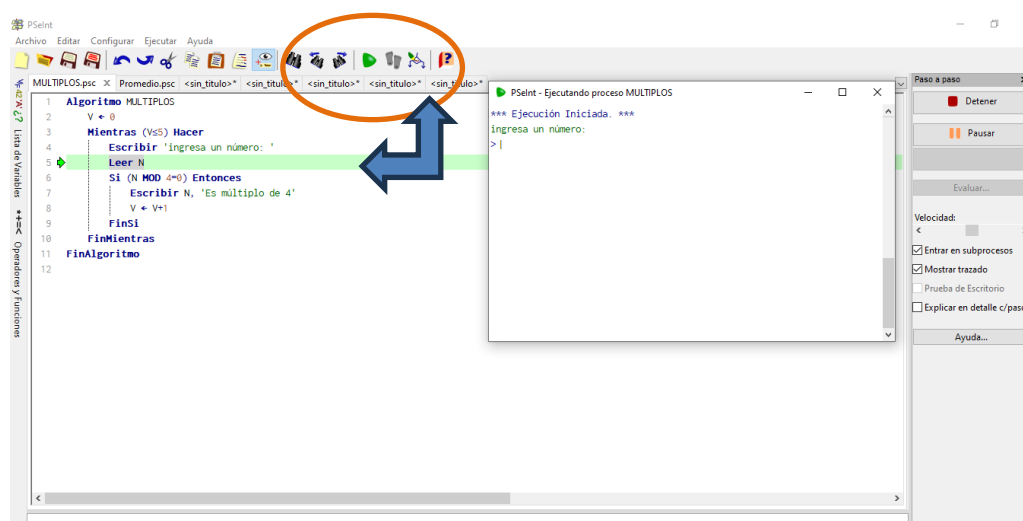


Figura 2.21 Ejecutar programa paso a paso

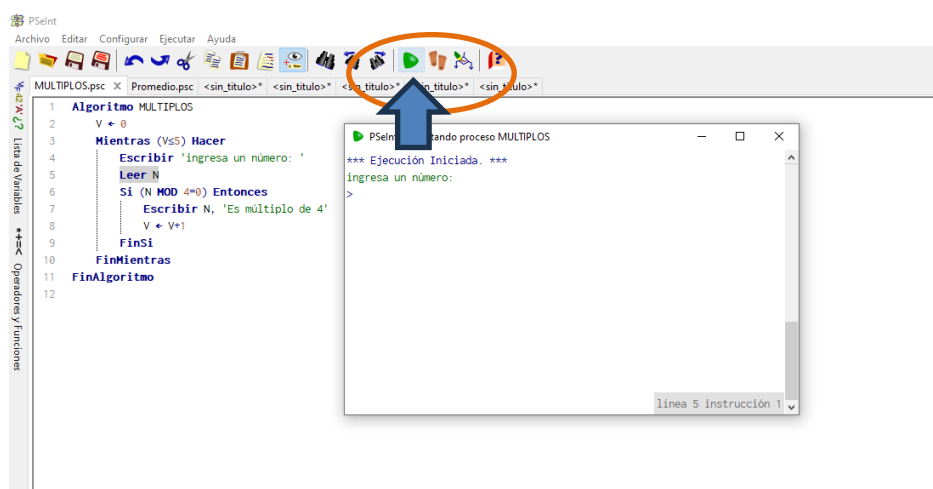


Figura 2.22 Ejecutar programa

La ejecución se muestra en la figura 2.23, donde se pueden validar los resultados con diferentes valores y diferentes ejecuciones del programa. Observe que el programa acepta introducir varios números, pero cuando se ingresan 3 números múltiplos de 4, el programa termina su ejecución como lo establece la condición.

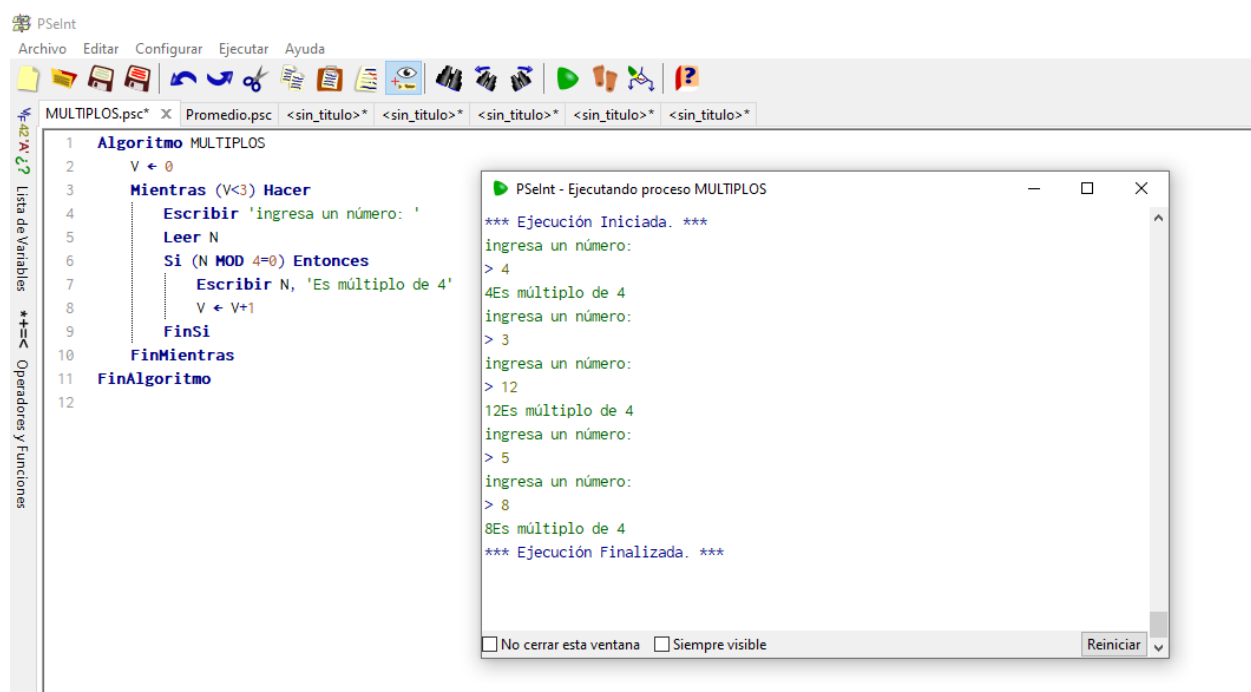


Figura 2.23 Depuración y prueba de un programa en PSeInt

2.6 Aprendizaje basado en problemas

1. Un estudiante desea conocer cuánto dinero se genera por concepto de intereses sobre la cantidad que tiene invertida en el banco. Si el interés excede a \$7000, lo vuelve a invertir y en ese caso desea saber cuánto dinero obtiene



2. Determinar si un alumno aprueba a reprueba un curso, sabiendo que aprobará si su promedio de tres calificaciones es mayor o igual a 70; reprueba en caso contrario.
3. En un almacén se hace un 20% de descuento a los clientes cuya compra supere los \$1000 ¿Cuál será la cantidad que pague una persona por su compra?
4. Una empresa quiere hacer una compra de varias piezas de la misma clase a una fábrica de refacciones. La empresa, dependiendo del monto total de la compra, decidirá qué hacer para pagar al fabricante. Si el monto total de la compra excede de \$500 000 la empresa tendrá la capacidad de invertir de su propio dinero un 55% del monto de la compra, pedir prestado al banco un 30% y el resto lo pagará solicitando un crédito al fabricante. Si el monto total de la compra no excede de \$500 000 la empresa tendrá capacidad de invertir de su propio dinero un 70% y el restante 30% lo pagará solicitando crédito al fabricante. El fabricante cobra por concepto de intereses un 20% sobre la cantidad que se le pague a crédito.
5. Una persona enferma, que pesa 70 kg, se encuentra en reposo y desea saber cuántas calorías consume su cuerpo durante todo el tiempo que realice una misma actividad. Las actividades que tiene permitido realizar son únicamente dormir o estar sentado en reposo. Los datos que tiene son que estando dormido consume 1.08 calorías por minuto y estando sentado en reposo consume 1.66 calorías por minuto.
6. Hacer un algoritmo que imprima el nombre de un artículo, clave, precio original y su precio con descuento. El descuento lo hace en base a la clave, si la clave es 01 el descuento es del 10% y si la clave es 02 el descuento es del 20% (solo existen dos claves).
7. Un obrero necesita calcular su salario semanal, el cual se obtiene de la siguiente manera: Si trabaja 40 horas o menos se le paga un salario de \$16 por



hora, si trabaja más de 40 horas se le paga un salario de \$16 por cada una de las primeras 40 horas y un salario de \$20 por cada hora extra.

8. Leer dos números y los imprima en forma ascendente
9. Una empresa quiere hacer una compra de varias piezas de la misma clase a una fábrica de refacciones. Dependiendo del monto total de la compra, decidirá qué hacer para pagar al fabricante. Si el monto total de la compra excede de \$500 000, el 55% será a crédito, el 30% lo paga en efectivo y el resto firma un pagaré. Si el monto total de la compra no excede de \$500 000, 50% lo paga a crédito y 50% en efectivo. El fabricante cobra por concepto de intereses un 20% sobre la cantidad que se le pague a crédito.
10. Elevar un número a una potencia usando una estructura iterativa.



3. Introducción a la programación de un lenguaje estructurado.

3.1 Introducción y estructura del entorno de un lenguaje de programación.

Uno de los lenguajes más conocidos de programación estructurada es el lenguaje C. Para utilizar el entorno es necesario comprender los siguientes conceptos básicos:

Palabra reservada. Son palabras que utiliza el entorno para realizar una tarea específica, no puede utilizarse como un identificador. main, printf, scanf, while, do, if son algunos ejemplos de palabras reservadas que utiliza Dev C++. La tabla 3.1, muestra las palabras reservadas más utilizadas

Tabla 3.1 Palabras reservadas.

Palabras Reservadas				
main	Scanf	Gets	int	float
If	Do	clrscr	void	return
else	While	switch	for	char
abs()	Default	puts	double	define
case	Typedef	break	include	cls

Librerías. Son bibliotecas que proporciona el compilador de Dev C++, en las cuales se incluyen funciones que serán utilizadas por el programador. Para ser librería el archivo debe tener extensión **.h**. Las librerías más utilizadas son:

stdio.h, integra funciones de rutinas de entrada/salida

conio.h, integra funciones de consola y puertos de entrada y salida

math.h, se compone de funciones matemáticas



3.2 Estructura básica de un programa.

La estructura básica de un programa fuente realizado en el entorno de Dev C++, es la siguiente:

1. *Incluir librerías*
2. *Funciones (opcional)*
3. *Declaración de variables globales y constantes (opcional)*
4. *Programa principal*
 {
 Declaración de variables locales
 Cuerpo del programa
 }
5. *Declaración de funciones (opcional)*
 {
 Instrucciones;
 }

A continuación, se describe cada una de las partes.

1. **Librerías.** Para utilizar una librería es necesario utilizar la siguiente sintaxis:

```
# include <nombre de la librería>
```

Ejemplo:

```
#include <stdio.h>
```

2. **Funciones.** Cuando el programa se divide en módulos en este apartado se coloca la función o solamente la declaración de la función y ésta a su vez puede o no regresar un valor. En la unidad 5 se detalla este apartado.

3. **Declaración de constantes y variables globales.** Al ser elementos que almacenan datos en memoria, si se declaran en esta parte se podrán utilizar en todo el programa.



4. **Programa principal.** Indica donde inicia el programa y se emplea la palabra reservada **main()**, y los elementos que lo integran deben ir entre dos llaves, una después de la palabra reservada { y la otra al terminar las instrucciones }. Todo programa en Dev C++ debe al menos tener un programa principal.

Ejemplo de uso

```
# include <conio.h>
# include <stdio.h>
main()
{
    printf("Programa ejemplo");
}
```

3.3 Tipos de datos.

Un **dato** es considerado un objeto que contiene información y son necesarios para que un programa funcione o trabaje. Un tipo de dato se limita por un conjunto de valores o rango y establece el conjunto de datos que puede almacenar. Por ejemplo, un tipo de dato entero almacenará únicamente valores numéricos enteros positivos o negativos y el rango permitido lo establecerá el entorno de uso. En Dev C existen varios tipos de datos los cuales se muestran en la tabla 3.2.

Los datos empleados en un entorno de programación se consideran simples o compuestos. Un dato simple no se puede separar por ser único, los datos compuestos se componen de varios datos. En Dev C++, el único dato compuesto es el tipo cadena el cuál se detalla su uso en el tema 5.



Tabla 3.2. Tipos de datos en Dev C++

Tipo de dato	Rango	Dev C++	Descripción
Entero	-32,768 a +32,767	int	Son números que no tienen decimales y pueden ser positivos o negativos
Real o de punto flotante	3.4e-38 a 3.4e+38	float	Son números que contienen punto decimal y pueden ser positivos o negativos
Flotante de doble precisión	1.7e-308 a 1.7e+308	double	Para cantidades muy grandes soporta valores de notación científica de 308
Carácter	-128 a 127	char	Representa los caracteres ASCII. Contiene las letras mayúsculas, minúsculas y los dígitos, pero se almacenan como números. Por ejemplo, 'b' es 98 y 'B' es 66 en código ASCII

3.4 Identificadores.

Un identificador, es un grupo de caracteres alfanuméricos los cuales se utilizan para brindar un nombre a una variable, programa, constante, función, procedimiento, etc. Existen reglas de uso para definir un identificador las cuáles todo programador debe seguir:

1. Iniciar con una letra
2. No debe tener espacios en blanco
3. No se puede utilizar una palabra reservada
4. No pueden existir dos identificadores iguales para diferentes elementos
5. No puede usar caracteres especiales, solo se permite el guion abajo
6. No utilizar nombres muy largos, es mejor el uso de abreviaturas
7. En Dev C las mayúsculas y minúsculas poseen diferentes valores por lo que A y a son dos identificadores distintos



Aunque un programador tiene libertad para definir identificadores, se recomienda que el nombre se relacione con la información que almacenará, por ejemplo: **sueldototal**, refleja plenamente el tipo de información que dicho identificador almacena. La tabla 3.3, muestra algunos ejemplos de identificadores válidos y no válidos.

Tabla 3.3 Identificadores válidos y no válidos

Identificadores válidos		Identificadores no válidos	
suma	A	3ms	main
TotAlum	salario	Valor#	¿precio?
area	x3	a-b	año
prom	A_B	sueldo total	3*mas

Identificador Variable. Dato cuyo valor puede cambiar durante la ejecución de un programa. Representa un valor almacenado en memoria que puede modificarse al llamado de esta en una línea de código en el programa al momento de su ejecución. El programador puede formular el nombre de la variable, respetando las reglas de uso de un identificador: iniciar con una letra, no incluir caracteres especiales, sin espacios, no sea palabra reservada y se recomienda una extensión de caracteres no mayor a 16.

Identificador constante. Es un dato que no cambia durante la ejecución de un programa. El ejemplo más claro es el valor de $\pi = 3.1416$, mismo que permanece fijo y no cambia.



3.5 Almacenamiento, direccionamiento y representación en memoria.

Una variable es un dato cuyo valor puede cambiar durante la ejecución de un programa, representa un valor almacenado en memoria y puede cambiar, conservarse y usarse cuantas veces sea necesario durante la ejecución del programa.

Todas las variables que utilice un programa deben declararse y asociarse a un tipo de dato antes de su uso. Sintaxis para poder declarar una variable:

Tipo de dato variable;

Tipo de dato Lista de variables;

Ejemplo:

int suma;

float x1, x2, promedio; // cuando se tienen variables del mismo tipo pueden definirse
separadas por una coma

Al declarar una variable también se le puede asignar un valor de inicio:

int suma=0;

Al declarar una variable, el compilador establece y reserva un espacio de memoria al cual le asigna un nombre y un número determinado de bytes dependiendo del tipo de dato y permitirá almacenar información cada que se haga un llamado a la variable definida. El almacenamiento es temporal mientras se ejecuta el programa.



3.6 Proposición de asignación.

Durante la elaboración de un programa se utilizan expresiones las cuales se forman uniendo variables, constantes y otras expresiones con operadores aritméticos, lógicos o relacionales. Las expresiones se clasifican en aritméticas y lógicas. Una expresión aritmética es aquella que integra fórmulas matemáticas. Al formular una expresión matemática es importante agrupar las operaciones que indiquen la correcta evaluación de la expresión.

Ejemplo:

Ecuación	Expresión aritmética
$x = \frac{a + b + c}{2a}$	$x = (a + b + c) / (2 * a)$
$N = \frac{3x + 2y}{a}$	$N = (3 * x + 2 * y) / a$

Una expresión lógica utiliza operadores relacionales y solo puede tener dos valores: verdadero o falso.

Ejemplo:

(a > b)
(x >= 3 && x <= 10)
(calificación >= 100)

3.7 Operadores, operandos y expresiones.

Para asociar valores o datos en una expresión y evaluar un resultado, es necesario el uso de un conector llamado operador. Existen múltiples operaciones matemáticas y aritméticas por lo que hay diferentes tipos de operadores para realizarlas:

operadores aritméticos, relacional, lógicos y de asignación, los cuales se describen a continuación:

- **Operadores aritméticos.** Se utilizan con datos de tipo entero o real para realizar sumas, restas, multiplicaciones, divisiones entre otras operaciones. La tabla 3.4, muestra su sintaxis y uso en Dev C++

Tabla 3.4. Operadores aritméticos

Operador	Significado	Ejemplo	Resultado
+	Suma	5 + 4	9
*	Multiplicación	8 * 4	32
/	División	18 / 6	12
%	Residuo	37 % 3	7
^	Potencia	2 ^ 3	8

Dev C++ cuenta con 2 operadores propios:

++ incremento

-- decremento

Si la variable se coloca al inicio se usa el valor actual de la variable y posteriormente se incrementa o decrementa según sea el caso. Si la variable se coloca al final, esta se incrementa y se utiliza el nuevo valor (Tabla 3.5). Este tipo de operadores solo se utiliza en variable no aplica en constantes y expresiones.

Tabla 3.5 Operadores usados en el entorno Dev C++

Operador	Significado	Ejemplo	Resultado
++	incremento	i++	Usa el valor de i y posteriormente incrementa en 1 la variable
		++i	Primero incrementa el valor de i en 1 y luego usa su valor
--	Decremento	i--	Usa el valor de i y posteriormente decrementa en 1 la variable
		--i	Decrementa en 1 el valor de i y usa su valor actual

- **Operadores Relacionales.** Se usan para evaluar condiciones y su resultado es de tipo lógico, es decir, verdadero o falso. La tabla 3.6, muestra los operadores usados en Dev C++

Tabla 3.6. Operadores Relacionales

Operador	Significado	Ejemplo	Resultado
>	Mayor que	8 > 9	Falso
<	Menor que	8 < 9	Verdadero
=	Igual	8 = 9	Falso
>=	Mayor igual	10 >= 9	Verdadero
<=	Menor igual	10 <= 9	Falso
!=	Diferente	10 != 10	Falso

- **Operadores lógicos.** Se usan en expresiones lógicas simples o combinarse en expresiones más complejas. Los operadores son: o, y negación. Para su valoración es necesario aplicar las tablas de verdad de cada uno de los operadores donde 0 =Falso y 1= verdadero (ver tabla). Para el caso del operador lógico y las dos condicionales deben ser verdaderas para que el resultado sea verdadero. En el operador o, con una de las condiciones que sea verdadera el resultado será verdadero y solo cuando las expresiones sean falsas el resultado es falso. (tabla 3.7)

Tabla 3.7. Tabla de verdad de los operadores lógicos

A	B	A && B	A B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

En la tabla 3.8, se muestra su sintaxis y uso en Dev C++

Tabla 3.8. Operadores Lógicos

Símbolo	Significado	Ejemplo	Resultado
&&	y	(4>2) && (6<8)	Verdadero
	o	(6>3) (5>8)	Verdadero
!	no	!(8<5)	Verdadero



- **Operadores de asignación.** Se utiliza para transferir a una variable el resultado derivado de una expresión. El símbolo utilizado es: =, empleando la siguiente sintaxis:

Variable = expresión

En la tabla 3.9, se muestran algunos ejemplos de uso.

Tabla 3.9. Ejemplos de asignación

Ejemplo	Descripción
A=8	Transfiere el valor de 8 a la variable A
Num=A	Transfiere el valor de A a la variable Num
S=8+7	Realiza la suma y almacena el resultado en la variable S
R= A%2	Divide el valor de A entre 2 y almacena el residuo en R

En Dev C++, existen otros operadores de asignación propios del entorno que permiten optimizar o simplificar expresiones, estos operadores se describen a continuación:

Tabla 3.10. Operadores incremento y decremento

Operador	Sintaxis de uso	Expresión completa
+=	B += 10	B = B +10
-=	B -= 10	B = B - 10
*=	B *= 10	B = B * 10
/=	B /= 10	B = B/10
%=	B %= 10	B = %=10

- **Operador condicional.** En dev C++ existe un operador que sustituye la estructura condicional if..else, “?:”. Su sintaxis es:
<expresión booleana> ? <instrucción verdadera> : <instrucción falsa>

Ejemplo de uso:



$X = (\text{opción} > 3) ? 8 : 10;$

En este ejemplo se establece que la variable X toma el valor de 8 si el dato almacenado en la variable opción es mayor a 3 en caso contrario, transfiere a X, el valor de 10. Este tipo de operador simplifica líneas de código ya que utilizando la estructura condicional el equivalente en código sería:

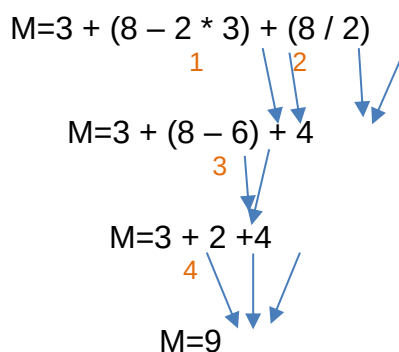
```
if (x>3)
    X=8;
else
    X=10;
```

3.8 Prioridad de operadores, evaluación de expresiones

Las operaciones que se ejecutan en un programa tienen un orden o prioridad, dicho orden se realiza de la siguiente manera:

1. Los paréntesis tienen la prioridad más alta. Toda expresión agrupada en paréntesis se evalúa en primera instancia.
2. La potencia, multiplicación, división y residuo tienen la misma prioridad y se evalúan de izquierda a derecha
3. La suma y resta se efectúan al final de derecha a izquierda

Ejemplo de uso:





3.9 Elaboración de programas.

Para generar un programa en Dev C++, es necesario instalar el entorno en una computadora, siguiendo los siguientes pasos:

1. Buscar en el navegador el entorno de Dev C++ o acceder a la siguiente liga:

<https://sourceforge.net/projects/orwelldvcpp/> (figura 3.1)

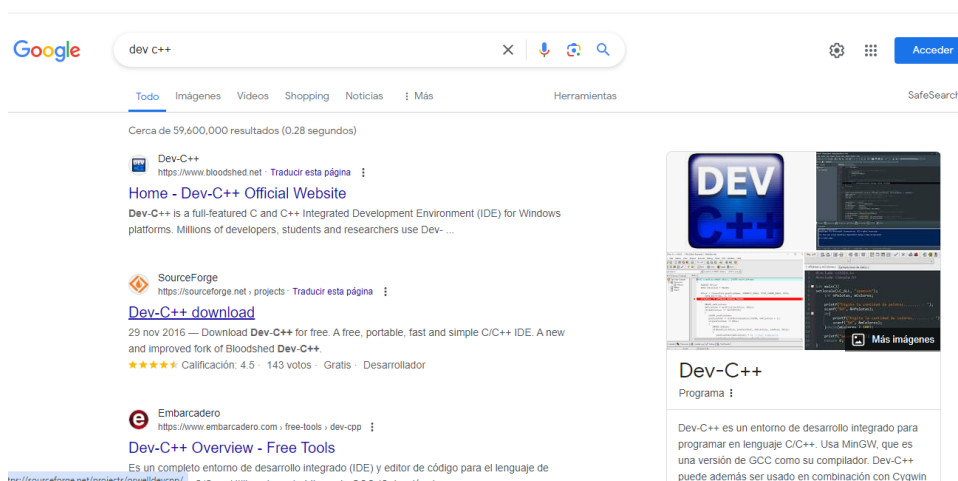


Figura 3.1 entorno de Dev C++

2. Descargar el software para instalar el entorno (Figura 3.2)

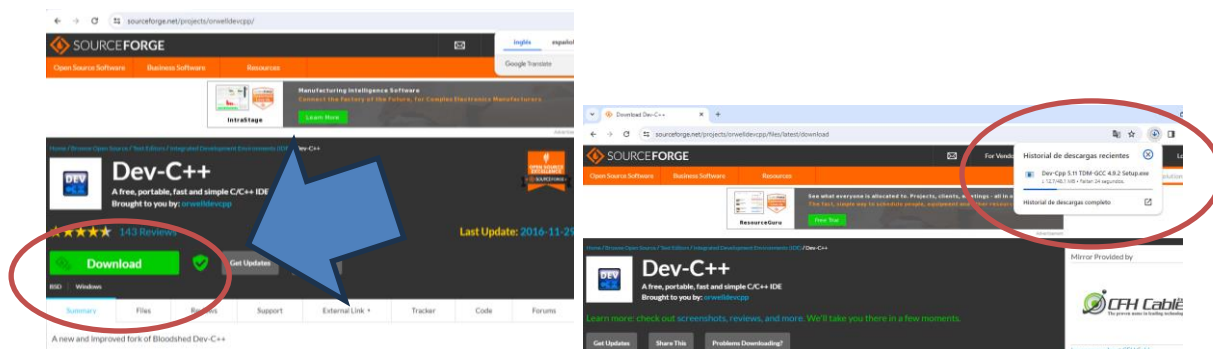


Figura 3.2. Descargar entorno de Dev C++

3. En la carpeta de descargas ejecutar el entorno como administrador y aceptar la instalación (Figura 3.3)

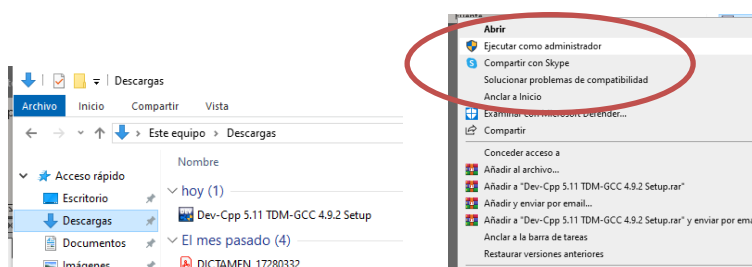


Figura 3.3. Instalar el entorno de Dev C++

4. Aceptar las condiciones para que se inicie con la descarga de archivos y preparar el entorno en el equipo seleccionado (Figura 3.4).

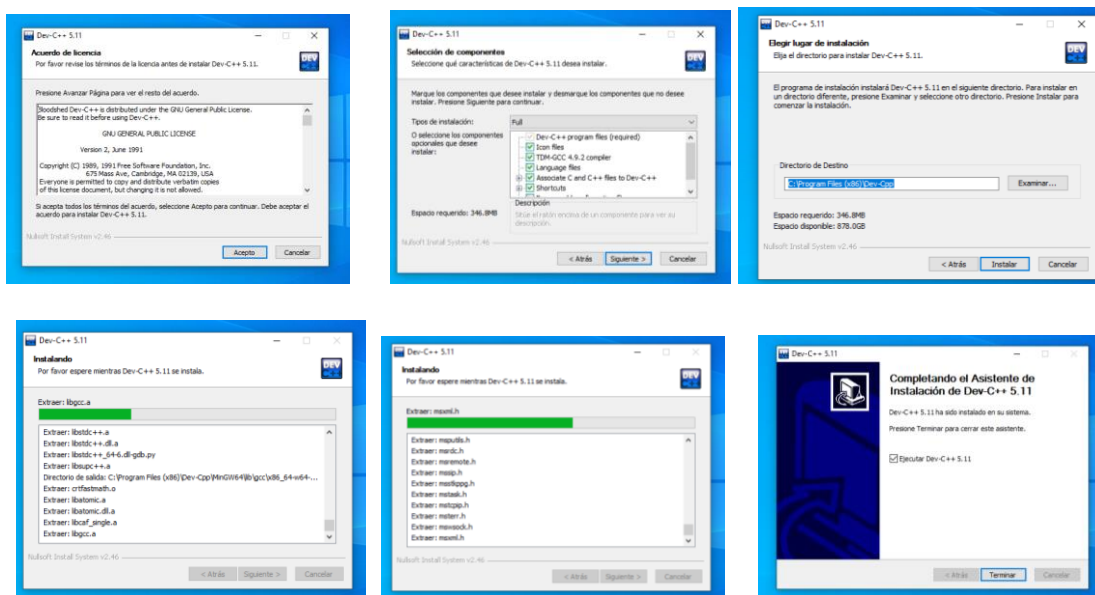


Figura 3.4. Instalando entorno de Dev C++

5. Es necesario elegir el idioma, el tipo de consola para finalizar adecuadamente la instalación, se recomienda el idioma español (latino américa) y la consola que sugiere el entorno. (Figura 3.5).

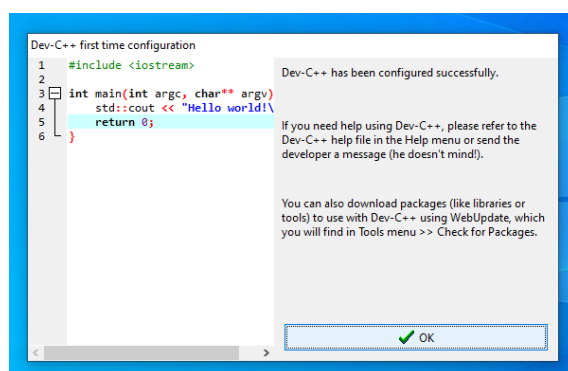
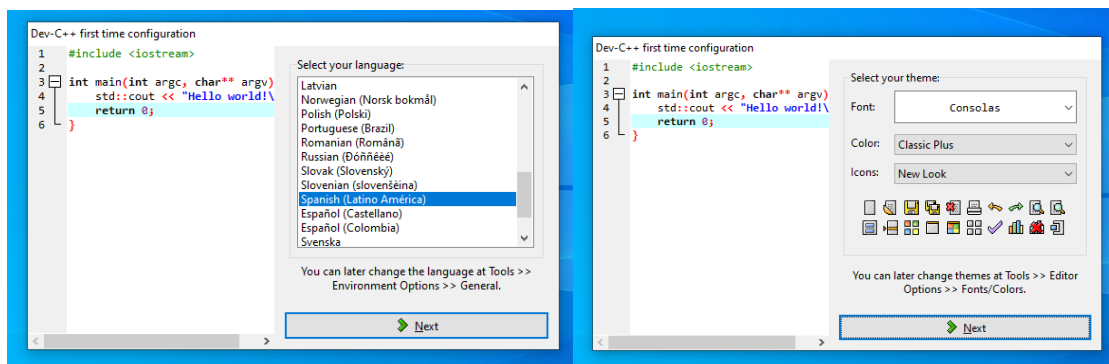


Figura 3.5. Entorno de Dev C++

6. El entorno muestra la barra de opciones y comandos que ofrece en tono gris (Figura 3.6.a) y será hasta que se elija un nuevo lienzo (Figura 3.6.b), que se podrá realizar un programa fuente o proyecto (Figura 3.6.c)

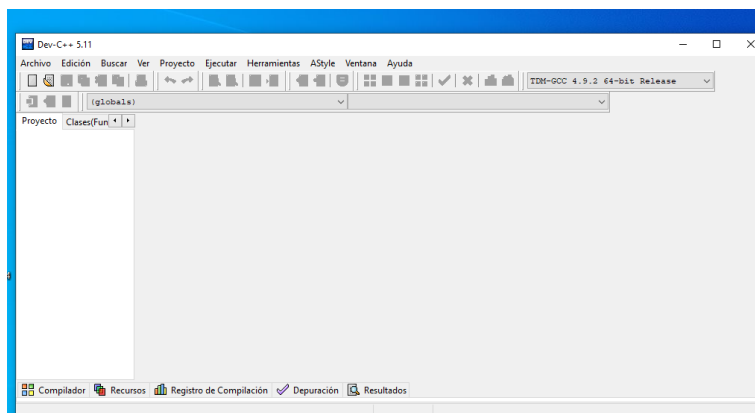


Figura 3.6.a. Entorno inhabilitado de Dev C++

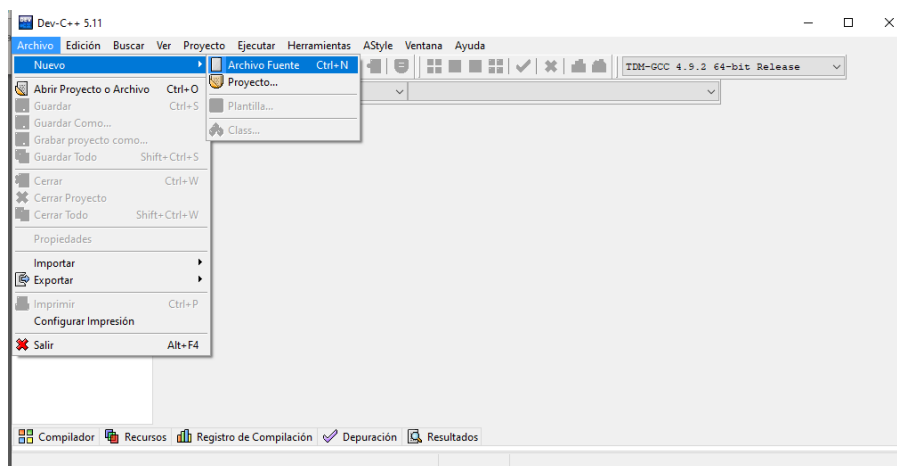


Figura 3.6.b. Crear un archivo en Dev C++

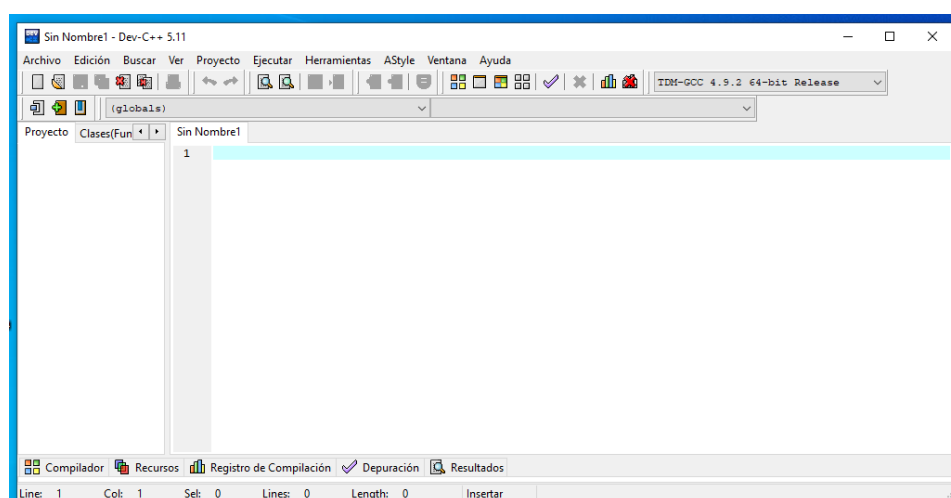
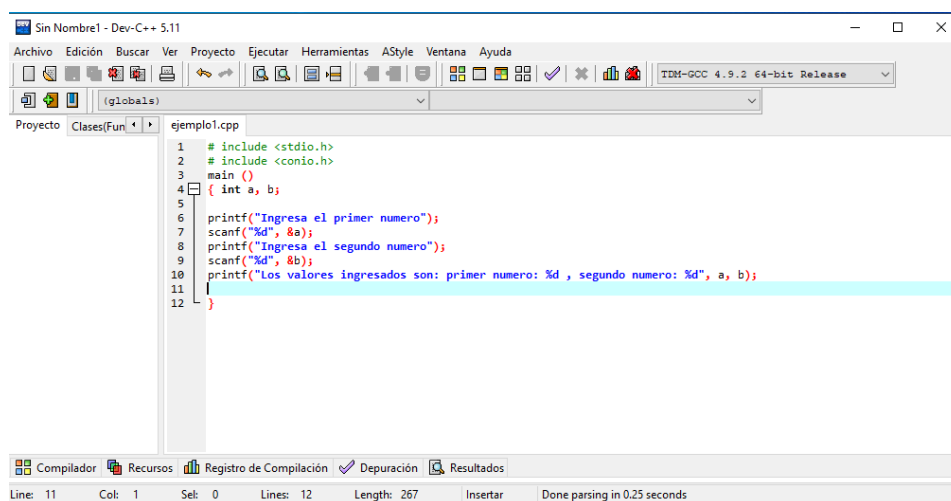


Figura 3.6.c. Editor de Dev C++

Pasos para crear un programa en el entorno.

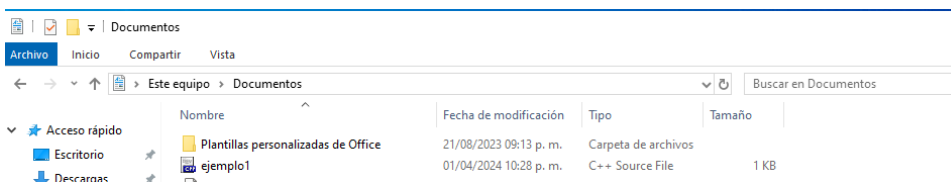
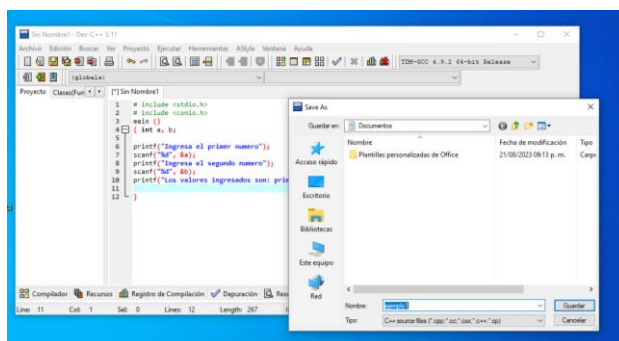
1. Capturar en el entorno las líneas de código necesarias respetando la estructura básica de un programa en Dev C++ descritas anteriormente (Figura 3.7.a).

2. Guardar el archivo asignando un nombre al archivo (Figura 3.7.b). Este programa se conoce con el nombre de programa fuente.
3. Posteriormente compilar y ejecutar el programa como se describe en el siguiente apartado.



```
1 #include <stdio.h>
2 #include <conio.h>
3 main ()
4 { int a, b;
5
6 printf("Ingresa el primer numero");
7 scanf("%d", &a);
8 printf("Ingresa el segundo numero");
9 scanf("%d", &b);
10 printf("Los valores ingresados son: primer numero: %d , segundo numero: %d", a, b);
11
12 }
```

Figura 3.7.a Programa fuente en Dev C++



	Nombre	Fecha de modificación	Tipo	Tamaño
Plantillas personalizadas de Office		21/08/2023 09:13 p. m.	Carpeta de archivos	
ejemplo1		01/04/2024 10:28 p. m.	C++ Source File	1 KB

Figura 3.7.b Guardar archivo

3.10 Pruebas y depuración.

Una vez que se ha creado un programa fuente, los siguientes pasos implican compilar y ejecutar el programa para validar los resultados en la resolución del programa. La descripción de cada paso se detalla a continuación.

Compilar un programa fuente. Para poder ejecutar un programa y observar los resultados en la computadora es necesario compilar el programa fuente creado y guardado previamente, con la finalidad de detectar los errores sintácticos cometidos en la captura, es decir, aquellos errores de escritura, o que falten a las reglas previas establecidas por el entorno. (Figura 3.8).

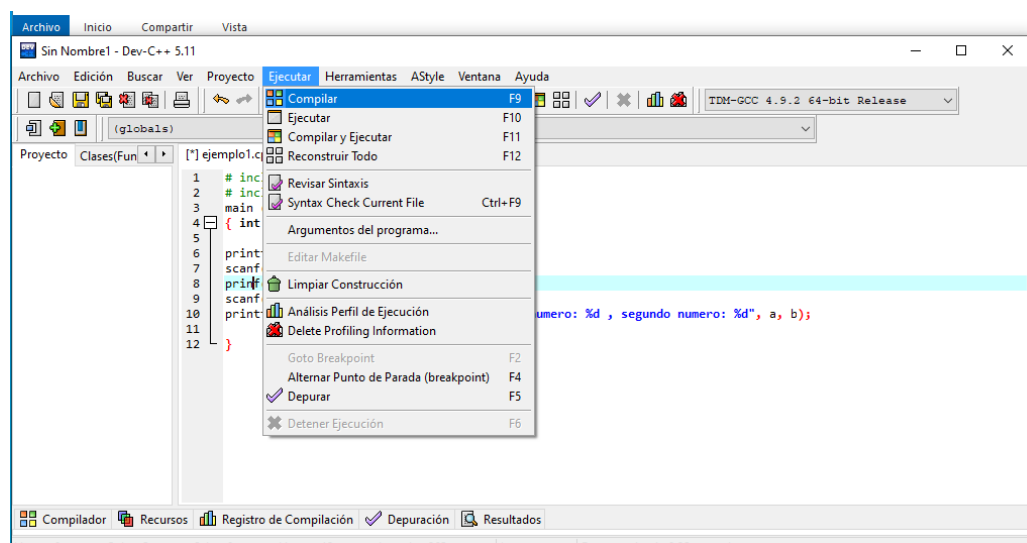


Figura 3.8 Compilar programa fuente

Si el programa tiene errores estos se marcan uno a uno de manera secuencial y deben ser corregidos hasta lograr 0 errores (Figura 3.9)

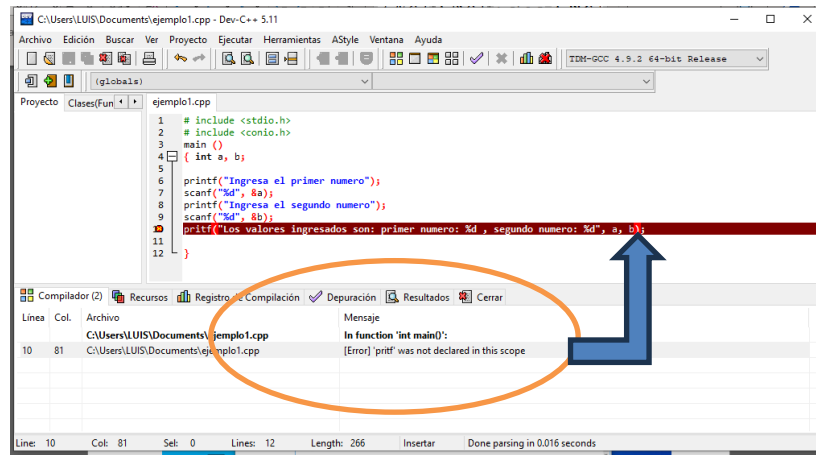
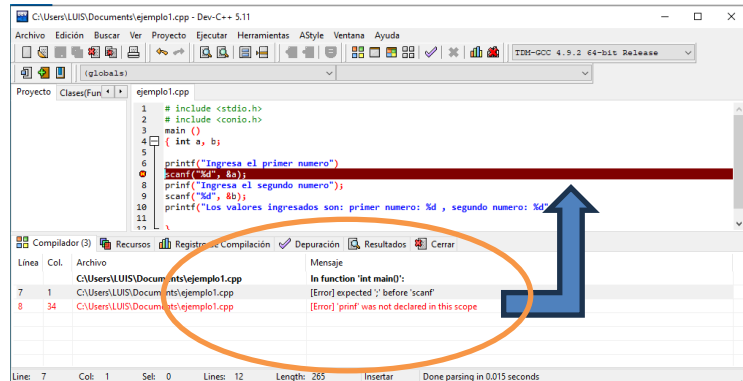


Figura 3.9 Errores de sintaxis detectados

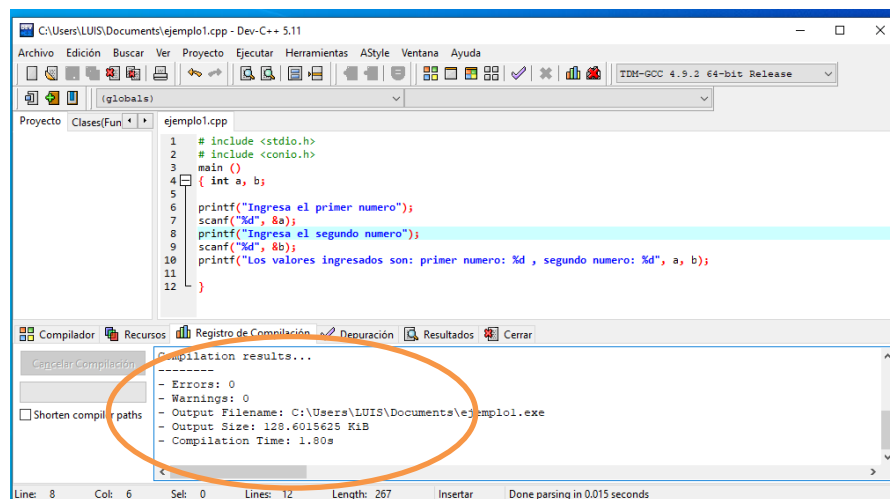


Figura 3.10 Programa compilado sin errores

Una vez que el programa ya no tiene errores se ejecuta y el entorno genera un programa ejecutable con extensión .exe (Figura 3.10 y 3.11).

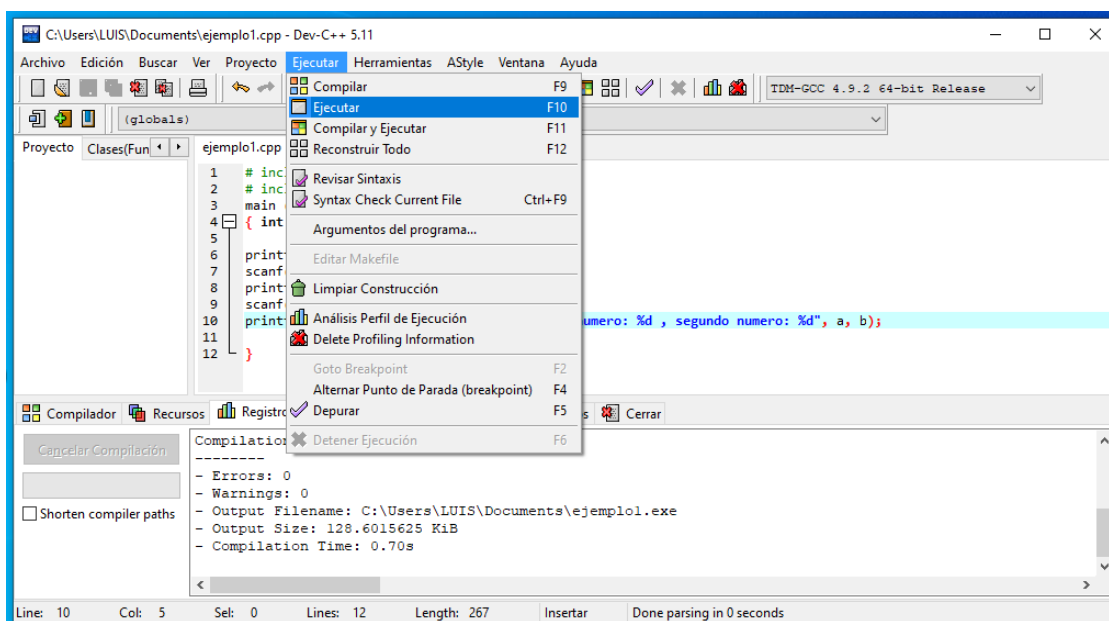


Figura 3.11. Ejecutar un programa fuente sin errores

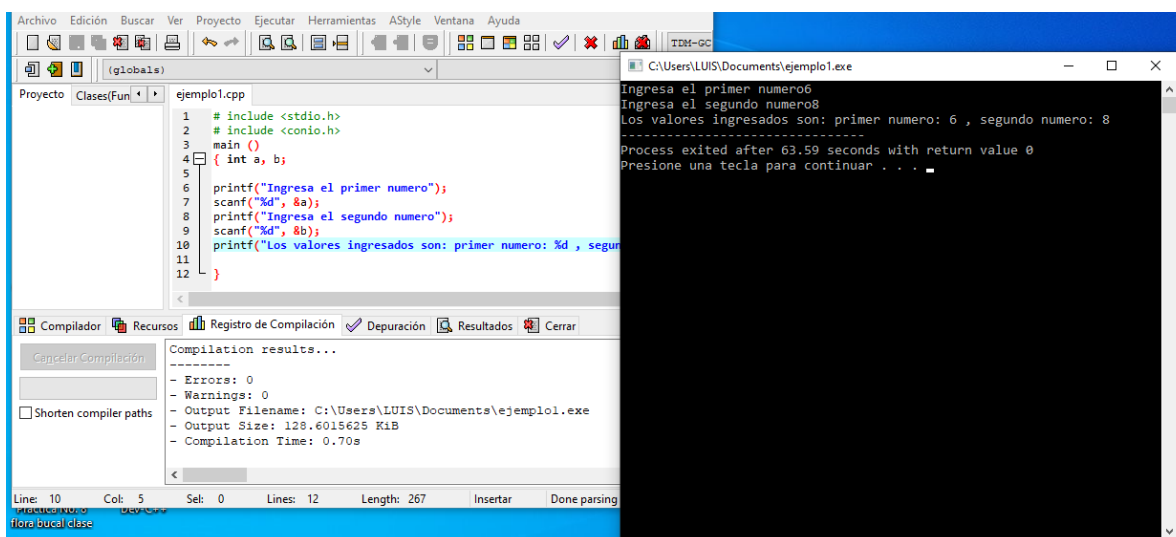


Figura 3.12. Ejecución de un programa en Dev C++

En la depuración o compilación de un programa se pueden detectar 2 tipos de errores:

- Sintácticos. Este tipo de error es fácil de detectar ya que el mismo compilador los detecta y establece el error cometido, los más comunes son: errores de escritura, errores de sintaxis, utilizar identificadores no válidos, no declarar variables, entre otros. En la figura 3.13, se muestra un error de sintaxis y se puede observar que una instrucción anterior al cierre del programa principal no termina con punto y como como lo establecen las reglas sintácticas.

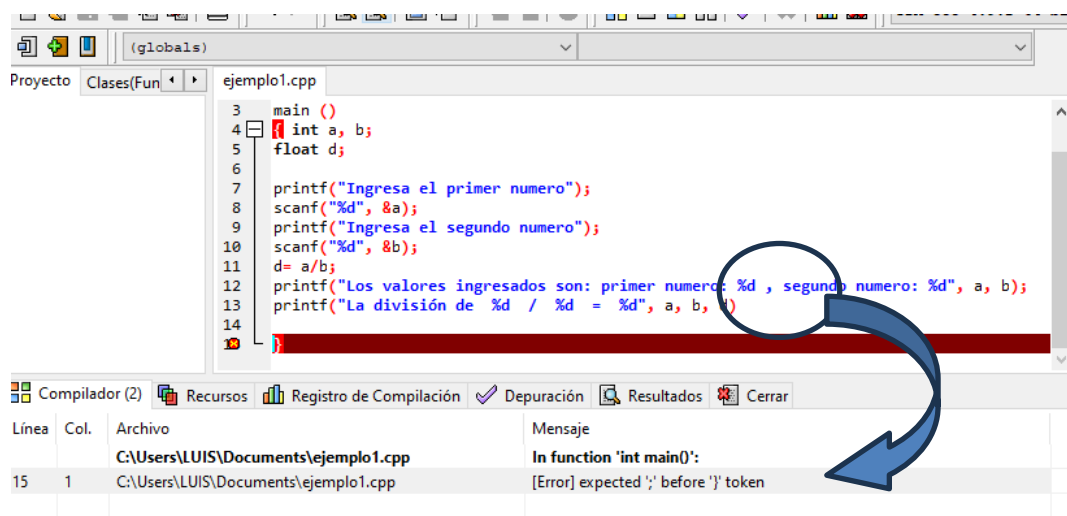


Figura 3.13. Error de sintaxis de un programa en Dev C++

- Lógicos. Son errores un poco más difíciles de detectar ya que corresponden a la validación y veracidad de los resultados obtenidos y se generan por no cumplir con reglas aritméticas, de secuencia, de procesos, entre otros. Un ejemplo de ello es cuando una división no puede efectuarse si el divisor es valor cero o nulo. En la figura 3.14, de la izquierda se observa que el compilador no detecta errores de sintaxis y por lo tanto se puede ejecutar.

Al comprobar los resultados con diferentes valores al ingresar números que respetan la regla aritmética de la división el resultado obtenido es válido, pero si se introduce el valor 0 en el divisor el resultado es incorrecto, este tipo de error se detecta hasta la ejecución del programa, por lo tanto, la importancia de depurar, validar y comprobar un programa es fundamental para la creación de programas confiables, válidos y veraces.

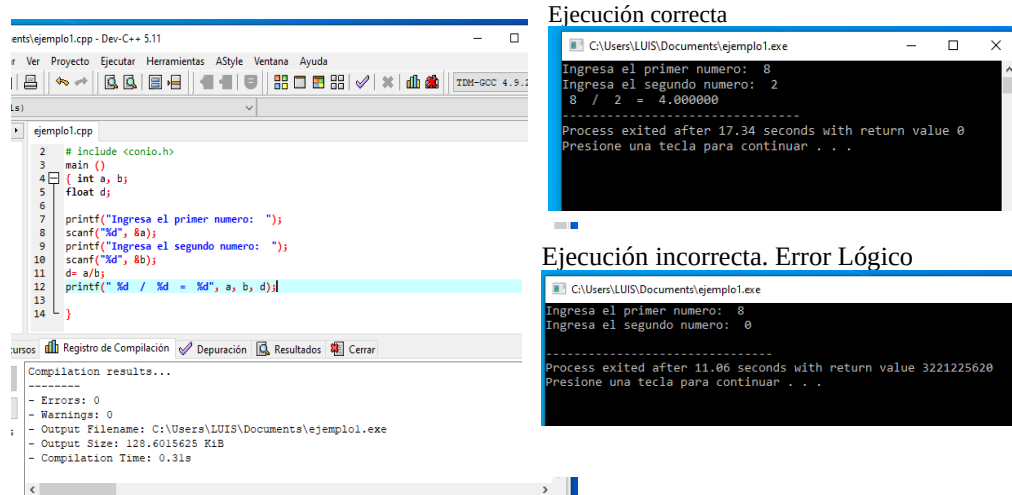


Figura 3.15. Prueba y depuración lógica

3.11. Ejercicios

Realiza el programa en Dev C++ de los siguientes ejercicios:

1. Calcular e imprimir el área de un triángulo, un cuadrado y un círculo
2. Calcular e imprimir el total de ingresos de una persona que obtiene los siguientes conceptos: ingreso por sueldo, ingreso por ventas e ingreso de beca educativa
3. Leer 2 números; si son iguales que los multiplique, si el primero es mayor que el segundo que los reste y si no que eleve el primero a la potencia del segundo
4. Tres estudiantes juntan su dinero para comprar una bicicleta. Sumar las cantidades e imprimir que porcentaje aportó cada uno de ellos del total obtenido



5. Un trabajador recibe su el reparto de utilidades anual por parte de la empresa. Para calcular dicha cantidad se debe tomar en cuenta su salario mensual y el porcentaje adicional que le corresponde en función a su antigüedad, en la empresa de acuerdo con la siguiente tabla.

Tiempo	Utilidad
Menos de 1 año.	5%
De 1 a 2 años.	9%
Más de 2 años y menos de 5 años.	10%
5 años o más y menos de 10 años.	15%
10 años o más.	20%

3.12. Aprendizaje basado en problemas

Integrar equipos de trabajo. La distribución de problemas y la determinación de integrantes lo realiza el profesor o los estudiantes pueden tener la libertad de hacerlo.

El reto es lograr que cada estudiante analice y genere alternativas de solución para el problema asignado y con su equipo elijan una alternativa para modelar en Dev C++ la propuesta de solución seleccionada y al final cada equipo realice su presentación justificando las razones de su decisión. Los elementos que debe integrar el documento generado son:

- Análisis del problema
- Alternativas de solución sustentadas en la investigación teórica y de campo que se utilizó
- Propuesta de solución utilizando el entorno Dev C++
- Resultados y conclusiones



1. Una estudiante quiere vender galletas y muffins para poder tener un ingreso económico extra y seguir estudiando, por lo que debe realizar un estudio de mercado que le permita calcular el total de establecimientos que hay en su colonia, cuantas familias viven cerca o acuden a los establecimientos y que cantidad de familias atiende en promedio cada establecimiento.
2. Si Luis vende a 25 pesos una docena de galletas y a 40 pesos una docena de muffins y puede producir 12 docenas de galletas y 5 docenas de muffins por día. ¿Cuáles serían sus ventas diarias esperadas? ¿Cuál es la venta total por mes si trabaja 6 días a la semana y un mes tiene 4 semanas? El programa debe desplegar producto, precio por docena, cantidad elaborada por docena, subtotal por producto, total por semana de cada producto y el total del mes
3. Un estudiante recibe su beca académica cada mes. Desea adquirir productos de la canasta básica gastando un 60% del ingreso recibido para poder cubrir otros gastos con el 40% restante. ¿Qué cantidad debe destinar para productos de la canasta básica y por qué?
4. ¿Cómo calcular la altura de un edificio?
5. Miguel realiza un estudio de mercado para vender galletas y se da cuenta que en su colonia existen 100 familias, 1 panadería, 1 papelería y 3 tiendas de abarrotes. En otra colonia cercana existen 400 familias, 3 pastelerías, 4 panaderías y 7 tiendas de abarrotes. Tomando en cuenta que Miguel entra al mercado con su establecimiento, realiza un programa en Dev C++ que calcule el total de establecimientos que hay por colonia y que cantidad de familias atiende en promedio cada establecimiento. Imprimir los resultados obtenidos



4. Estructuras selectivas y de repetición

En muchas ocasiones al plantear alguna propuesta de solución, se tiene la necesidad de ejecutar instrucciones dependientes de una determinada condición, para ello existen las estructuras selectivas, mismas que se pueden desarrollar de manera simple, doble o anidada. Así también en programas de cómputo se deben ejecutar un determinado número de veces una misma instrucción o instrucciones para lograr el cumplimiento de un proceso y en estos casos es necesario el uso de estructuras de repetición. Para poder emplear este tipo de estructuras es importante comprender los siguientes términos: condición, sentencias y bloque de sentencias.

Una **condición**, es una expresión lógica que, al ser evaluada, expresa dos escenarios, el verdadero y el falso.

Ejemplo de condiciones:

$N > 4$

$5 > 3$

$A + B < 6$

$A \leq B$

Tomando los siguientes valores $N=4$, $A=2$ y $B=1$ al evaluar las condiciones anteriores se obtiene:

$N > 4$

Verdadero

$5 < 3$

Falso

$A + B < 6$

Verdadero

$A \leq B$

Falso

Sentencia, es una instrucción que se ejecuta en un programa y un **bloque de sentencias**, son un grupo de instrucciones que deben ser ejecutadas por bloques dependiendo de una condición o iteración.



Cada entorno de programación emplea sus propios delimitadores para agrupar el grupo de sentencias o instrucciones. En el caso de Dev C++, se utilizan las llaves {}.

Ejemplo de sentencias o instrucciones en lenguaje Dev C++:

```
A=A+3;  
printf("Este es un ejemplo");  
scanf("%f",&s);
```

Ejemplo de un bloque de sentencias:

```
{  
    printf("Ingresa un número: ");  
    scanf("%d",&a);  
    s=s+a;  
}
```

Cada una de las estructuras tiene sus propias reglas y formas de uso, por lo que a continuación, se explican cada una de las estructuras que se pueden utilizar en el entorno de programación Dev C++ que es el entorno que se describe en este material.

4.1 Selectiva simple

Es considerada una estructura que ejecuta una sentencia o conjunto de sentencias, dependiendo del resultado que genere la evaluación de una o más condiciones. La estructura selectiva simple, nos permite ejecutar una sentencia o un bloque de sentencias cuando se cumple la condición.



Sintaxis de uso:

Sentencia
if (condición)
 instrucción;

Bloque de sentencias
if (condición)
{
 instrucción 1;
 instrucción 2;
 instrucción n;
}

El uso de operadores lógicos y relacionales es fundamental en este tipo de estructura, es importante hacer notar la diferencia al comparar dos elementos, el operador de igual debe utilizarse doble para que el compilador realice la comparación en lugar de la asignación.

Uso correcto del operador de igualdad:

Correcto	Incorrecto
if (a==b)	if (a=b)
Comparar	Asignar

Ejemplo 1. Se Ingresa una calificación y si es mayor o igual a 70 y menor o igual a 100, imprimir el mensaje de aprobado

Alternativa de solución:

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int calificación;
    printf("ingresa la calificación obtenida: ");
    scanf("%d",&calificación);
    if (calificación>=70 && calificación<=100)
        printf("Aprobado"); //Nota: es solo una sentencia por lo que no requiere llaves
} //main principal
```



Ejemplo 2. Leer 2 números y si el primero es mayor al segundo obtener la suma, la resta y la multiplicación de ambos valores e imprimir resultados obtenidos.

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int a,b, S, M, R;
    printf("ingresa el primer número: ");
    scanf("%d",&a);
    printf("ingresa el segundo número: ");
    scanf("%d",&b);
    if (a>b)
    { S=a+b;          //Nota: Bloque de sentencias, por lo que requiere llaves
      R=a-b;
      M= a*b;
      printf("Resultados: suma: %d, resta: %d , multiplicación: %d, ", S, R, M);
    }
}
```

Observe que la estructura selectiva simple solo ejecuta la sentencia en caso de evaluar como verdadera la condición, en el caso de incumplir el programa no ejecuta nada, simplemente termina.

Ejercicio de aprendizaje

Un estudiante puede obtener una beca si su promedio es mayor a 85. Leer la calificación e imprimir el mensaje de "becado" si cumple la condición.

4.2 Selectiva doble

Es una estructura que permite evaluar una condición y ejecutar sentencias si la condición es verdadera y otras sentencias si la condición es falsa, cuando las sentencias son diferentes para cada caso.

Sintaxis:

```
if (condición)
    Sentencia verdadera;
else
    Sentencia falsa;
```



Ejemplo 1. Leer un número y evaluar si es múltiplo de 3 o no lo es.

```
#include <stdio.h>
#include <conio.h>
main ()
{
    int num;
    printf("ingresa un número: ");
    scanf("%d",&num);
    if (num%3==0)
        printf("El %d, es múltiplo de 3", num);
    else
        printf("El %d, no es múltiplo de 3", num);
}
```

Ejemplo 2. Realizar el 15% de descuento en las compras mayores a 2000 y un 10% cuando sea igual o menor a 2000

```
#include <stdio.h>
#include <conio.h>
main ()
{
    float c, desc, total;
    printf("Ingresa el total de compra: ");
    scanf("%f",&c);
    if (c>2000)
    {
        desc= c*0.15;
        total=c-desc;
        printf("El descuento otorgado es: %f", desc);
        printf("El total a pagar con el 15% de descuento es %f, ",
total);
    }
    else
    {
        desc= c*0.10;
        total=c-desc;
        printf("El descuento otorgado es: %f", desc);
        printf("El total a pagar con el 10% de descuento es %f, ", total);
    }
}
```



Ejercicio de aprendizaje

Calcular el precio de venta de un producto tomando en consideración los siguientes valores: Si el precio de producción es menor que 150 agregar un 15% de ganancia para determinar el precio de venta y si es mayor o igual a 150, agregar un 12%. Imprimir el precio de venta obtenido

4.3 Selectiva anidada

Otra forma de uso de la estructura selectiva es utilizar una estructura dentro de otra, este concepto se conoce como anidar, esto significa que en lugar de una sentencia o bloque de sentencias lo que ejecuta la acción es otra estructura. El modelo anidado se puede generar en la valoración verdadera o en la falsa o en ambas según se requiera para lograr la solución deseada. La sintaxis de uso es la siguiente:

Estructura selectiva anidada

if (condición)	if (condición)
if (condición)	sentencia verdadera;
Sentencia verdadera;	else
else	if (condición)
Sentencia falsa;	sentencia verdadera;
else	else
Sentencia falsa;	sentencia falsa;

Ejemplo de uso:

Determinar qué tipo de dato corresponde a la tecla que oprime el usuario: letra mayúscula, letra minúscula, número o valor especial.



```
#include <stdio.h>
#include <conio.h>
main()
{
    char tecla;
    printf("oprime una tecla: ");
    scanf("%c",&tecla);
    if (tecla>=48 && tecla<=57)
        printf("el valor de la tecla corresponde a un número");
    else
        if (tecla>=65 && tecla <=90)
            printf("el valor de la tecla corresponde a una letra mayúscula");
        else
            if (tecla>=97 && tecla <=102)
                printf("el valor de la tecla corresponde a una letra minúscula");
            else
                printf(("El valor corresponde a una tecla con carácter especial"));
}
```

Observar que las 4 posibles opciones que deben valorarse utilizan estructuras selectivas dobles anidadas y los valores que cada condición integra corresponde al código ASCII y al rango en los cuáles dichos valores se encuentran, por ejemplo, los dígitos se localizan entre el rango de valores entre el 48 = 0 y 57=9, y así sucesivamente.

Ejercicio de aprendizaje

Luís tiene un negocio de dulces de cajeta que él elabora. Decide otorgarle un 15% de descuento a los clientes que le compren 300 dulces o más, un 10% de descuento a los que adquieran entre 100 y 300 dulces y los que adquieran menos de 100 solo un 5%. Cada dulce tiene un costo de \$15 pesos. Cuál es el precio con descuento al que le va a vender a cada cliente y cuánto debe pagar el cliente del total de su compra sin descuento y cuánto con descuento



4.4 Selectiva múltiple

Es una estructura que permite seleccionar como su nombre los dice, una opción entre varias. Este tipo de estructura evalúa una expresión que pueden tomar n opciones, pero solo ejecutar una. Switch, es la palabra reservada que define a la estructura selectiva múltiple. La expresión empleada solo acepta valores numéricos o de tipo carácter. La sintaxis de uso es la siguiente:

```
switch (expresión)
{
    case expresión_1: sentencias 1:
        break;
    case expresión_2: sentencias 2:
        break;
    case expresión_3: sentencias 3:
        break;
    .
    .
    case expresión_n: sentencias n:
        break;
    default: mensaje de error;
}
```

La estructura utiliza en cada una de las opciones las palabras reservadas **case** y **break**, las cuales permiten al compilador establecer con **case** el bloque de sentencias a ejecutar de acuerdo con la opción elegida y con **break**, la terminación de dicha opción, sin esta palabra se genera un error de ejecución o ambigüedad. Por otro lado, la palabra reservada **default** permite al programador considerar algún mensaje o sentencia cuándo el usuario del programa oprime un valor u opción que no está contemplada en la lista de opciones y como es la última instrucción no requiere de la palabra **break**.

Ejemplo de uso:

Elegir una marca de bloqueadores solares para conocer el precio y si tiene descuento



```
#include <conio.h>
#include <stdio.h>
main()
{
    int valor;
    printf("Bloqueadores solares precio y promoción del producto")
    printf(" 1. Nivea SUN");
    printf("2. La Roche ");
    printf("3. ISDIN fotoprotector ");
    printf("4. Eucerin ");
    printf("Ingresa la opción deseada: ");
    switch (valor)
    {
        case 1: printf("protector solar corporal hidratante");
                printf(" Frasco de 125ml ");
                printf("Precio: $132.00 ");
                printf(" Descuento del 10% del 10 al 15 de mayo");
                break;
        case 2: printf("La roche Posay Anthelios toque seco");
                printf(" Protector solar facial para piel factor 50+, 50ml");
                printf("Precio: $352.00 ");
                printf(" Descuento del 20% del 10 al 15 de mayo");
                break;
        case 3: printf("Fotoprotector fusión wáter magic");
                printf(" SPF 50 protector solar OIL, 120ml ");
                printf("Precio: $400.00 ");
                printf(" sin descuento");
                break;
        case 4: printf("Protector solar facial efecto mate");
                printf(" toque seco OI Control 50ml ");
                printf("Precio: $300.00 ");
                printf(" Descuento del 15% del 10 al 15 de mayo");
                break;
        default: printf("Esa opción no existe");
    }
}
```

Observar que el programa solo permite la elección de una alternativa y una vez que la ejecuta el programa termina, si se quisiera ejecutar más de una vez entonces se hace necesario emplear estructuras iterativas las cuales se explican más adelante.



Cabe mencionar que la estructura switch no acepta rangos, por lo que si se desea analizar opciones basadas en rangos debe utilizarse la estructura if anidada.

Ejercicio de aprendizaje

Utilizando la estructura switch, realizar un menú de opciones que permita una vez leídos 2 números, ejecutar operaciones básicas de una calculadora, si elige la opción 1: suma los números, la opción 2: multiplica los números, la opción 3: realiza la resta de los números y la opción 4 divide el primer número entre el segundo número.

Reta tu mente y realiza el mismo ejercicio con la estructura if y discute en grupo la diferencia entre las estructuras selectivas empleadas

4.5 Repetir mientras

El uso de una computadora primordialmente es para realizar operaciones repetitivas. La repetición o iteración es un segmento de un programa que se ejecuta un determinado número de veces mientras una condición se cumpla y en algún momento la condición debe tener un fin para evitar iteraciones o ciclos infinitos. En Dev C++ existen 3 estructuras iterativas, pero para comprender su uso es necesario conocer los siguientes términos:

- **Contadores y acumuladores.** Son palabras que se utiliza mucho cuando se realiza un proceso varias veces. Un contador es una variable que se utiliza cuando se desea llevar la cuenta del número de ejecuciones ya sea en incremento o decremento y se distingue dado que el valor de incremento o



decremento es fijo. Un acumulador permite que una variable acumule valores en sí misma durante la ejecución de un proceso repetitivo y los valores empleados son variables. Son variables de tipo numérico en el caso de los contadores de tipo entero y en el caso de los acumuladores pueden ser números enteros y flotantes. Las características de este tipo de variables son:

1. Deben inicializarse antes de empezar el proceso iterativo con un valor de inicio según corresponda.
2. El nombre de la variable debe aparecer en ambos lados de la asignación, el operador requerido y el valor del incremento o decremento.
3. No debe ser una palabra reservada

Ejemplos de contadores de incremento:

$c=c+1$ $n=n+4$ $s=s+2$ $i++$

Ejemplos de contadores de decremento:

$c=c-1$ $n=n-4$ $s=s-1$ $i--$

Ejemplos de acumuladores de incremento y decremento:

$\text{Suma} = \text{Suma} + A$ $R = R - \text{Número}$

- **Centinela y/o bandera.** Son variables que permiten controlar ciclos o iteraciones. Una variable tipo **centinela** se emplea inicializando la variable con un valor antes del ciclo y dentro del proceso se actualiza o cambia para que finalice el proceso. Ejemplo de ello es cuando se realiza la pregunta ¿dese continuar S/N? fuera del proceso se inicializa la variable con S y dentro



del ciclo se actualiza cuando el proceso ya no se debe ejecutar. Una variable tipo bandera toma el valor de 0 y 1 antes del ciclo y dentro del ciclo para iniciar y terminas las iteraciones.

Una estructura iterativa se le conoce como repetir mientras y se utiliza con la palabra reservada **while**. Este tipo de estructura permite evaluar una condición al inicio y si su valor es verdadero se realizará el proceso; dentro del ciclo debe existir la variable tipo centinela o bandera que permita la terminación o continuación del ciclo. Las características de esta estructura son:

1. La condición se evalúa antes del ciclo y puede ser de tipo numérica, lógica o relacional
2. En esta estructura si la condición es falsa de inicio, nunca se realiza el ciclo, es decir, la iteración se logra 0 o más veces
3. Cuando la condición se verdadera el ciclo se ejecutará de manera indefinida por lo que debe existir una variable que modifique la condición
4. Las sentencias de la iteración deben ir entre llaves {}

Sintaxis:

```
while (condición)
{
    Sentencia 1;
    Sentencia 2;
    .
    .
    Sentencia n;
}
```

Ejemplo 1. Mostrar los múltiplos de 4 que se encuentren entre el rango de 1 a 200



```
#include <stdio.h>
#include <conio.h>
main()
{
    int num=4; //se declara variable e inicializa con valor seleccionado
    while (num<=200)
    {
        printf("\t %d", num); //el carácter \t realiza un tabulador
        num=num+4
    }
}
```

Ejemplo 2. Leer una lista de números y sumar los números pares y multiplicar los impares hasta que la suma sea mayor a 50 y la multiplicación mayor a 200

```
#include <stdio.h>
#include <conio.h>
main()
{
    int s=0,p=1, n;
    while (s<=50 || p<=200)
    {
        printf("ingresa un número: ");
        scanf("%d",&n);
        if (n%2==0)
            s=s+n;
        else
            p=p*n;
    }
    printf(" La suma resultante es: %d\n", s);
    printf("La multiplicación de impares es: %d\n", p);
}
```

Ejercicio de aprendizaje

Del ejercicio 2, agrega las instrucciones necesarias para que cuente cuántos números impares ingresó, cuántos pares y cuál fue el total de números ingresados. Imprime valores obtenidos



4.6 Repetir hasta

Es una estructura iterativa que ejecuta un proceso y posteriormente evalúa una condición, si la condición es verdadera repite el proceso, pero en cuanto sea falsa el proceso termina. A diferencia de la estructura **while**, el ciclo se puede ejecutar 1 o más veces. Las características de esta estructura son:

1. Utiliza las palabras reservadas **do .. while**
2. El ciclo se ejecuta al menos una vez
3. Debe existir una condición dentro del ciclo para que este se ejecute o termine
4. El proceso o sentencia por repetir debe ir entre llaves {}
5. La palabra **do** no lleva punto y coma
6. La palabra reservada **while** lleva punto y coma después de la condición

Sintaxis:

```
do
{
    Sentencia 1;
    Sentencia 2;
    .
    .
    Sentencia n;
}
while (condición);
```



Ejemplo 1. Sumar los números del 1 al 20 e imprimir la sumatoria obtenida

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i=1, suma=0;
    do
    {
        suma = suma + i; //acumulador
        i++; //contador
    }
    while (i<=20);
    printf("La sumatoria obtenida es: %d", suma);
}
```

Ejemplo 2. Sumar un conjunto de números hasta que se ingrese un cero

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n, suma=0;
    do
    {
        Printf("ingresa un número: ");
        Scanf("%d",&n);
        suma = suma + n;
    }
    while (n!=0);
    printf("La sumatoria obtenida es: %d", suma);
}
```

Ejercicio de aprendizaje

Ingresa varios números y realizar la suma de aquellos que sean múltiplos de 5, el programa finaliza cuando se ingresen 5 múltiplos. Imprimir el total de números ingresados y la suma de los múltiplos.



4.7 Repetir desde

La estructura desde (for), es una estructura muy sencilla porque solo requiere del número de iteraciones a realizar para poder utilizarla. Su uso se recomienda con tipos de datos compuestos como los arreglos y matrices (tema 5). La estructura se representa con la palabra reservada **for** y sus características son:

1. Utiliza la palabra reservada **for** seguida de paréntesis y diferentes argumentos
2. En el paréntesis se requiere de 3 componentes: valor inicial, condición y contador, separados por punto y coma.
3. Se recomienda cuando se conoce el número de iteraciones a realizar
4. Solo lleva llaves {} cuando se ejecutan más de una sentencia en el ciclo
5. La estructura no lleva punto y coma

Sintaxis:

for (variable=valor inicial; variable<=valor final; incremento o decremento de la variable)

Ejemplo 1. Calcular la media de 10 valores e imprimir el resultado obtenido

```
# include <conio.h>
# include <stdio.h>
main()
{
    int i;
    float media, suma=0, valor;
    for (i=1; i<=10; i++)
    {
        printf("ingresa el valor %d :", i);
        scanf("%d",&valor);
        suma=suma+valor;
    }
    media=suma/10;
    printf("La media de los valores ingresados es: %.2f , media);
}
```



Ejemplo 2. Leer una lista de 10 números y sumar aquellos que sean múltiplos de 7 y contar los que no lo son. Imprimir resultados

```
# include <conio.h>
# include <stdio.h>
main()
{
    int i, num, contador=0, suma=0;
    for (i=1; i<=10; i++)
    {
        printf("ingresa el %d número: ", i);
        scanf("%d",&num);
        if (num%7==0)
            suma=suma+num;
        else
            contador++;
    }
    printf("La suma de múltiplos es: %.d , suma);
    printf("Los números que no fueron múltiplos es: %.d , contador);
}
```

Ejercicio de aprendizaje

Ingresar un número desde teclado e imprimir la serie del 1 hasta el número leído

4.8 Elaboración de programas

La importancia de utilizar un entorno de programación es que permite modelar alternativas de solución a problemas planteados en el entorno utilizando: las estructuras, librerías, funciones, comandos y todo aquello que el entorno ofrece. Tres aspectos deben tomarse en cuenta al momento de generar un programa en el entorno seleccionado:



1. Respetar la sintaxis y reglas semánticas y gramaticales del entorno seleccionado
2. Realizar el planteamiento de las posibles alternativas de solución derivado del análisis y planteamiento del problema
3. Capturar el programa en el entorno seleccionado y asegurarse mediante la compilación y ejecución repetida del programa, de la veracidad de sus resultados

A continuación, se presenta un menú de opciones que integran una serie de programas elaborados en el entorno Dev C++ utilizando las estructuras vistas anteriormente y mostrando al lector, el código del programa, así como la compilación y ejecución de este con la finalidad de describir los posibles escenarios que el programador puede enfrentar al utilizar dicho entorno.

Ejemplo. Elaborar un menú de opciones iterativo, considerando los siguientes ejercicios en cada opción y empleando las estructuras indicadas. Usar estructura selectiva múltiple e iterativa en el menú.

1. Leer un número e imprimir si es Par/Impar (Estructura condicional doble)
2. Calcular el promedio de 5 calificaciones (estructura iterativa for)
3. Calcular el valor de x (Estructura condicional y validación de errores lógicos)

$$x = \frac{2ab+3c}{b-c}$$

4. Obtener el promedio de un estudiante solicitando su calificación final de preparatoria y la calificación obtenida en su examen de admisión (estructura Selectiva doble)



5. Obtener las raíces utilizando la fórmula general de una Ecuación cuadrática (Estructura selectiva y validación lógica)
6. Finalizar menú

Código del programa completo

```
#include <conio.h>
#include <stdio.h>
#include <math.h>

main()
{
    int a, op, suma, num, i;
    float prom, z, b, c, x, cal_p, cal_e, promedio, t, y, u, x1, x2, d;
    do
    {
        printf("1. Número Par/Impar\n");
        printf("2. Promedio de 5 números\n");
        printf("3. Valor de X\n");
        printf("4. Promedio admisión\n");
        printf("5. Ecuación cuadrática\n");
        printf("6. Fin del programa\n");
        printf("Ingrese una opción por favor: ");
        scanf("%d", &op);
        switch (op)
        {
            case 1:
                printf("Ingresa un numero: ");
                scanf("%d", &a);
                if (a % 2 == 0)
                    printf("el numero %d, es par\n", a);
                else
                    printf("el numero %d, es impar\n", a);
                break;
            case 2: suma = 0;
                for (i = 1; i <= 5; i++)
                {
                    printf("Ingresar la %d calificacion: ", i);
                    scanf("%d", &num);
                    suma += num;
                }
                .... .continúa en la siguiente página
```



.....Continuación

```
prom = (float)suma / 5;
printf("El promedio obtenido es: %.2f\n", prom);
break;

case 3:
    printf("Ingresa el valor de a: ");
    scanf("%f", &a);
    printf("Ingresa el valor de b: ");
    scanf("%f", &b);
    printf("Ingresa el valor de c: ");
    scanf("%f", &c);
    if (b - c != 0)
    {
        x = (2 * a * b + 3 * c) / (b - c);
        printf("El resultado es: %.2f\n", x);
    }
    else
        printf("Error/división entre cero");
    break;

case 4:
    printf("Promedio de la preparatoria?: ");
    scanf("%f", &cal_p);
    printf("Calificación del examen de admisión?: ");
    scanf("%f", &cal_e);
    promedio = (cal_p + cal_e) / 2;
    printf("El promedio es: %f", promedio);
    printf("Por lo tanto: ");
    if (promedio >= 85)
        printf("El estudiante es admitido.\n");
    else
        printf("El estudiante no es admitido\n");
    break;

case 5:
    printf("Ingresa el valor de a: ");
    scanf("%f", &t);
    printf("Ingresa el valor de b: ");
    scanf("%f", &y);
    printf("Ingresa el valor de c: ");
    scanf("%f", &u);
```

..... continúa en la siguiente página



.... continuación

```

        if (t != 0)
        {
            d = pow(y, 2) - (4 * t * u);
            if (d > 0)
            {
                x1 = (-y + sqrt(d)) / (2 * t);
                x2 = (-y - sqrt(d)) / (2 * t);
                printf("Las raíces son: %f, %f\n", x1, x2);
            }
            else if (d == 0)
            {
                x1 = -b / (2 * t);
                printf("Raíces repetidas= %f\n", x1);
            }
            else
            {
                x1 = -b / (2 * t);
                x2 = sqrt(fabs(d)) / (2 * t);
                printf("Raíces complejas: %f+-%f\n", x1, x2);
            }
        }
        else
            printf("No es ecuación cuadrática\n");
        break;

    case 6:
        printf("Fin del programa\n");
        break;

    default:
        printf("Opción errónea\n");
    }

} while (op != 6);
}

```

Compilación. Hay que recordar, que siempre se debe guardar antes de compilar un programa, para evitar la pérdida de la información y si no se ha realizado el entorno lo solicitará al momento de compilar el programa. Existen dos formas para compilar:



1. Desde el menú de barra de opciones del entorno. Seleccionar la opción **compilar** una vez que se guardó el programa fuente (Figura 4.1 Lado izquierdo)
2. Con la función F9 del teclado de su computadora. (Figura 4.1 Lado derecho)

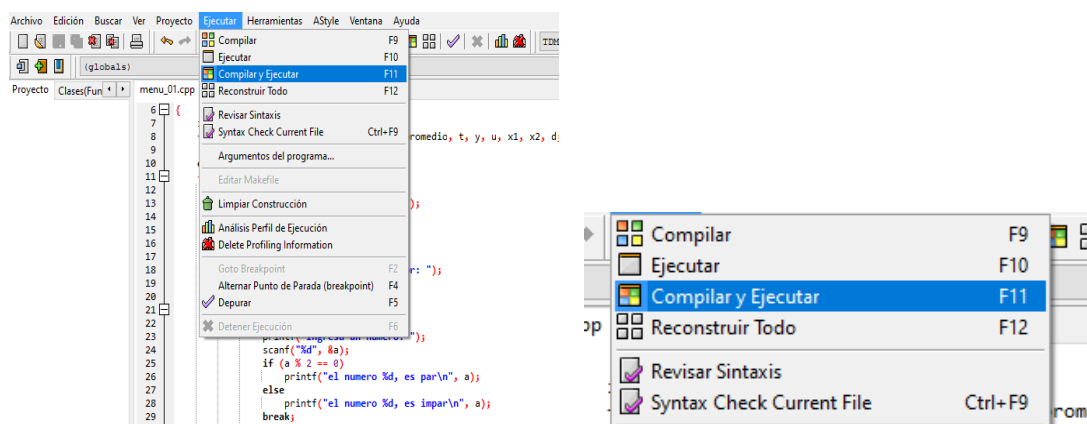


Figura 4.1 Formas de compilación de un programa fuente

Si el código fuente no presenta errores lo indicará la ventana de la parte inferior de la pantalla como se observa en la figura 4. 2

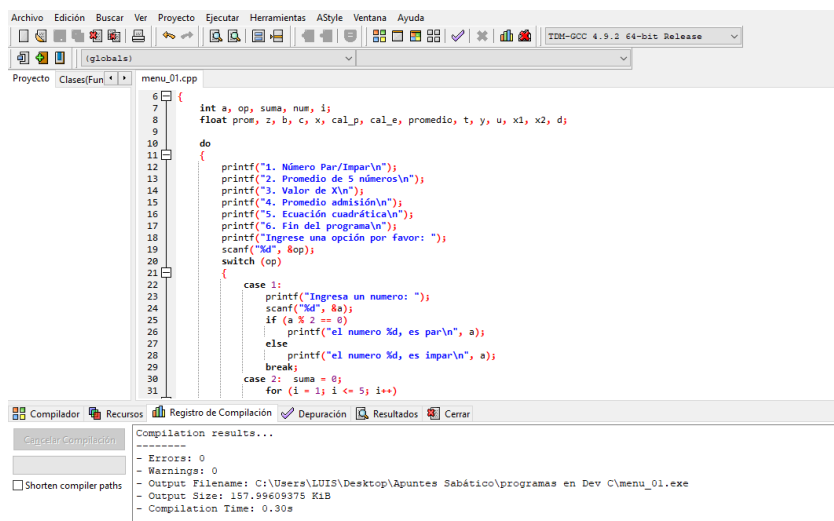


Figura 4.2 Compilación sin errores



Ejecución. Una vez que el programa no presente errores de sintaxis, es momento de ejecutar el programa varias veces con diferentes valores para validar la veracidad de los resultados obtenidos, recordar que la validez de los datos se ofrece cuando los resultados sean los esperados y no se genere ningún tipo de error lógico durante la ejecución. La figura 4.3 muestra unos fragmentos de la corrida del menú presentado.

```

6  {
7      int a, op, suma, num, i;
8      float prom, z, b, c, x, cal_p, cal_e, promedio, t, y, u;
9
10     do
11     {
12         printf("1. Número Par/Impar\n");
13         printf("2. Promedio de 5 números\n");
14         printf("3. Valor de X\n");
15         printf("4. Promedio admisiKn\n");
16         printf("5. EcuaciKn cuadrática\n");
17         printf("6. Fin del programa\n");
18         printf("Ingrese una opción por favor: ");
19         scanf("%d", &op);
20         switch (op)
21         {
22             case 1:
23                 printf("Ingresa un numero: ");
24                 scanf("%d", &a);
25                 if (a % 2 == 0)
26                     printf("el numero %d, es par\n", a);
27                 else
28                     printf("el numero %d, es impar\n", a);
29                 break;
30             case 2:
31                 suma = 0;
32                 for (i = 1; i <= 5; i++)

```

Output window content:

```

1. Número Par/Impar
2. Promedio de 5 n-meros
3. Valor de X
4. Promedio admisiKn
5. EcuaciKn cuadrática
6. Fin del programa
Ingrese una opción por favor: 1
Ingresa un numero: 5
el numero 5, es impar
1. Número Par/Impar
2. Promedio de 5 n-meros
3. Valor de X
4. Promedio admisiKn
5. EcuaciKn cuadrática
6. Fin del programa
Ingrese una opción por favor: 2
Ingresa la 1 calificación: 80
Ingresa la 2 calificación: 85
Ingresa la 3 calificación: 90
Ingresa la 4 calificación: 75
Ingresa la 5 calificación: 80
El promedio obtenido es: 82.00
1. Número Par/Impar
2. Promedio de 5 n-meros
3. Valor de X
4. Promedio admisiKn
5. EcuaciKn cuadrática
6. Fin del programa
Ingrese una opción por favor: _

```

Figura 4.3 Ejecución del programa

4.9 Ejercicios

Elabora los siguientes programas, captura en el entorno de Dev C++ y compila y ejecuta. Anota tus comentarios y comenta el tipo de estructura empleada y la razón de uso.

1. Imprimir los números pares del 1 al 300
2. Encontrar el mayor de una lista de 10 números y que se imprima dicho valor



3. Calcular las 5 primeras potencias de un número introducido desde teclado
4. Leer un número entero positivo e imprimir la serie del 1 hasta el número leído, excepto aquellos que sean múltiplos de 3.
5. Calcular la siguiente sumatoria para un valor de n introducido desde teclado

$$f(x) = \sum_{n=1}^{\infty} (1/n)$$

4.10 Aprendizaje basado en problemas

1. María debe dar a conocer a una lista de aspirantes a beca, si cumplen con los requisitos de la convocatoria para ello debe analizar los siguientes datos: el promedio de preparatoria debe ser mayor o igual a 85, el ingreso económico debe ser menor a 3,000 pesos mensuales y debe estar inscrito en alguna escuela de educación superior. Realiza un programa que permita a María validar requisitos y dar a conocer al aspirante si fue seleccionado o no y cuáles fueron los requisitos que incumplió.
2. Para José es importante contar con un programa que permita a su hijo practicar las tablas de multiplicar del 1 al 10. Realiza dicho programa.
3. En un salón de clases se desea seleccionar a los líderes de cada equipo a conformar, para ello es importante clasificar a los estudiantes cuya edad cumplida sea par y menor o igual a 30, para conocer cuántos equipos se deben formar. Realiza un programa que resuelva el problema planteado.
4. Obtener el promedio de llenado de refrescos de un conjunto de 50 datos aleatorios, para determinar si una empresa refresquera cumple con la norma de llenado de sus refrescos.
5. Descarga el programa fuente menú_01 de la carpeta de programas y realiza las siguientes modificaciones.



- a) El menú de opciones debe finalizar cuando se introduzca un cero
- b) El promedio de calificaciones del ejercicio 2 debe ser en función al número de valores que el usuario desee promediar
- c) El ejercicio 1, debe leer una lista de 10 números y sumar los pares y contar los impares.

4.11 Aprendizaje basado en casos

El flujo de efectivo es una herramienta fundamental para cualquier empresa porque representa los movimientos de dinero que ingresan y salen en un periodo determinado ya sea mensual o anual (Oliver, 2021). Se determina usando la siguiente fórmula: $\text{flujo de efectivo} = \text{ingresos} - \text{egresos}$, donde ingresos es todo el dinero efectivo que ingresa por ventas, comisiones, intereses, banco, entre otros y los egresos son todos los gastos que la empresa realiza tales como: pago de servicios, sueldos, insumos por mencionar algunos.

Ernesto como gerente de una empresa, desea contabilizar los ingresos y egresos de cada mes, para conocer el estatus que mantiene la organización con respecto al mercado, para ello solicita a las diferentes áreas: las ventas del mes, el dinero que ingresa por transferencia y los intereses generados en el mes y el estado de cuenta del dinero del banco, así también revisa los gastos de gasolina, de luz, de internet y de sueldos del mes analizado para determinar las ganancias o pérdidas según resulte. Desarrolla un programa que permita Ernesto ingresar los datos solicitados, para calcular las ganancias realizando la resta de ingresos menos egresos. Si el resultado de la operación es negativo, indica que la empresa está perdiendo en lugar de ganar, por lo que el programa debe mandar un mensaje de alerta avisando que hay pérdidas. Si por el contrario el resultado es positivo, entonces el mensaje deberá ser indicando cuál es la ganancia obtenida en el periodo. Recuerda que: ingreso es todo aquel dinero que ingresa por ventas de productos, servicios o intereses. Los Egresos por el contrario es todo aquello que se paga o genera gasto, como sueldos, pago de luz, renta, etc.



5. Arreglos y archivos

En Dev C++ y otros entornos de programación, existe un tipo de dato estructurado que permite almacenar en una sola variable un conjunto limitado de valores del mismo tipo, este tipo de variables se les conoce como arreglo. Sus características principales son:

1. Es una lista que contiene un número finito de datos del mismo tipo
2. El almacenamiento de dato es de forma contigua
3. La variable se identifica con un solo nombre y cada elemento que almacena posee una posición de memoria
4. Utiliza corchetes [] para establecer el tamaño y posiciones del arreglo
5. Para extraer o almacenar un dato en la variable estructurada se debe emplear un índice o subíndice
6. El valor de la primera posición o índice debe ser 0
7. El programador define el tipo de datos y el tamaño del arreglo antes de utilizarse

Se debe tomar en cuenta que este tipo de variable ocupa al igual que las variables simples un espacio de memoria, si se excede el número de datos a almacenar con respecto al tamaño definido del arreglo, será hasta que se ejecute el programa que el error de desborde de memoria se presente y no al momento de la compilación. La sintaxis de uso es:

<tipo de dato> <Nombre del arreglo> [tamaño]

Ejemplo:

int vector[10] es un arreglo unidimensional llamado **vector**, que permite almacenar 10 valores de tipo entero.



Existen 3 tipos de arreglo:

- Arreglo unidimensional
- Arreglo bidimensional
- Arreglo multidimensional

Por sus características y formas de uso, se describen de manera específica en los siguientes apartados.

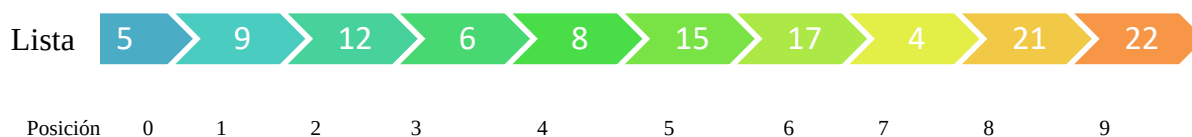
5.1 Arreglo unidimensionales

Al conjunto de datos almacenados de manera contigua en forma de lista y de un mismo tipo se le conoce con el nombre de arreglo unidimensional o simplemente arreglo. Para almacenar o extraer un dato requiere de un índice o posición para realizar la acción requerida. El índice representa la posición en la que el dato o elemento se encuentra almacenado o se requiere almacenar.

Ejemplo:

int lista[10]

En este caso lista es considerada una variable tipo arreglo unidimensional que permite almacenar hasta 10 datos de tipo entero. En una lista de datos, para conocer la posición de un valor es necesario hacer referencia a la variable declarada como arreglo y dentro de corchetes colocar el índice requerido, en la figura 5.1, se muestra gráficamente como se representa un arreglo unidimensional y cómo se visualizan los elementos por posición. Tomando el ejemplo que **Lista** es el nombre de una variable tipo arreglo con 10 posiciones y que permite almacenar datos de tipo entero, entonces el valor 5 se encuentra almacenado en **Lista [0]** y el valor 15 en **Lista [5]**.



Si se realiza un cambio de dato y se asigna en Lista [7]=45 la actualización será:

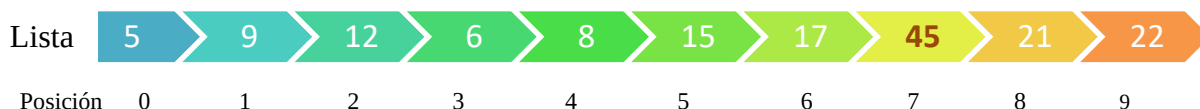


Figura 5.1 Esquema gráfico de una variable unidimensional

La sintaxis de uso para emplear este tipo de datos en Dev C+, es:

<tipo de dato> <nombre de la variable> [tamaño]

Donde:

Tipo de dato, se refiere a los datos que se desean almacenar en el arreglo y pueden ser: int, float, char. Si se desea almacenar una cadena de caracteres es necesario utilizar la función *gets* en lugar de *scanf*, esta acción se explica en el tema 6 para detallarlo de manera específica.

Nombre de la variable, es el identificador que representa el arreglo unidimensional y debe construirse respetando las reglas de un identificador.

Tamaño, es la cantidad máxima de elementos que contiene el arreglo y debe ir entre corchetes []

Para el almacenamiento de valores en un arreglo es necesario emplear estructuras iterativas, como la variable es de tamaño conocido o limitado, se recomienda emplear la estructura **for** pero no es restrictivo el uso de otras estructuras iterativas.



A continuación, se muestra la parte de un programa que permite almacenar 10 valores en un arreglo

```
#include <conio.h>
#include <stdio.h>
main ()
{
    int A[10]; //se declara variable tipo arreglo tamaño 10
    //para almacenar 10 valores es necesario usar una estructura iterativa que permita repetir el proceso de lectura
    for(i=0; i<10; i++)
    // el ciclo inicia con posición 0 y la condición debe considerar 10 valores máximo
    {
        printf(" ingresa el %d valor", i+1);
        scanf("%d", &A[i]); // se ingresan los valores en el arreglo A
    }
}
```

En el anterior código solo se almacenan datos de tipo entero en el arreglo A, para poder visualizar datos es necesario nuevamente emplear un ciclo para extraer la información como se puede observar en el siguiente programa, se almacena e imprimen los datos de un arreglo A.

```
#include <conio.h>
#include <stdio.h>
main ()
{
    int A[10];
    //para almacenar 10 valores en el arreglo A
    for(i=0; i<10; i++)
    {
        printf(" ingresa el %d valor", i+1);
        scanf("%d", &A[i]); // se ingresan los valores en el arreglo A
    }
    //imprimir los valores almacenados previamente
    for(i=0; i<10; i++)
        printf(" %d \t", A[i]); // se imprimen los valores en el arreglo A y se agrega un tabulador \t para separar los datos almacenados ya que de lo contrario se imprimen de manera continua sin separación
}
```



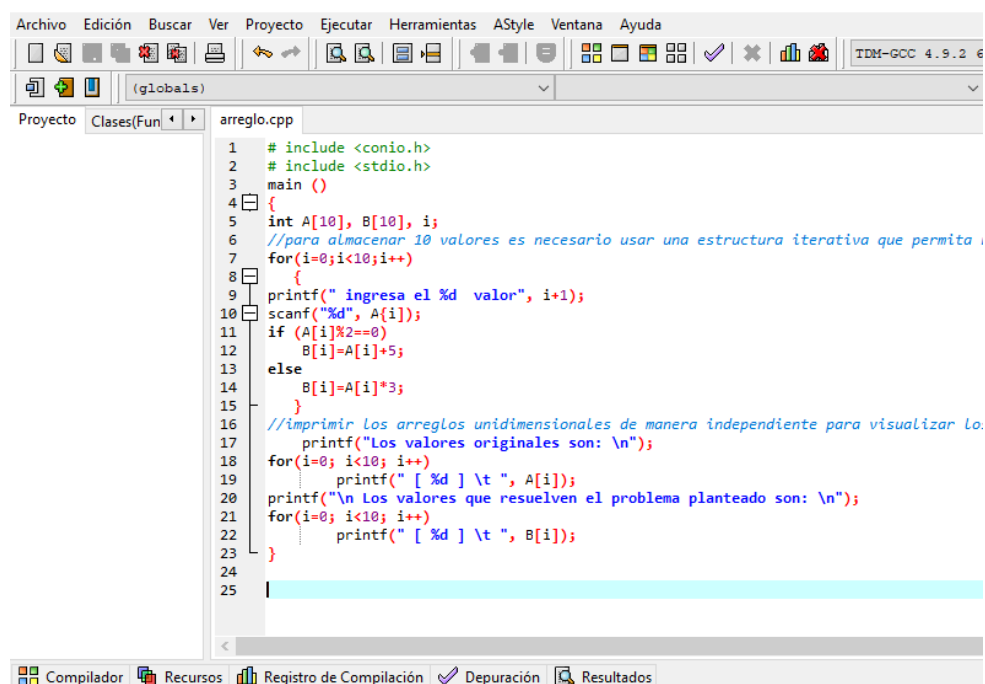
Observe que el primer ciclo requiere de {} por ejecutar un bloque de sentencias y al momento de imprimir los datos solo se ejecuta una sentencia por lo que no son necesarias. Este código lo puede capturar y ejecutar en el entorno DevC++ para visualizar su funcionamiento como parte de su aprendizaje.

Ejemplo de uso:

Almacenar en un arreglo unidimensional A, 10 elementos de tipo entero. En un arreglo B, realizar la siguiente acción: si el número almacenado es par, sumarle 5 y si es impar multiplicarlo por 3. Imprimir los arreglos A y B

```
# include <conio.h>
# include <stdio.h>
main ()
{
    int A[10], B[10], i;
    //para almacenar 10 valores es necesario usar una estructura iterativa que permita
    //repetir el proceso de lectura y se aprovecha el ciclo para realizar el proceso
    //solicitado de verificar si el número ingresado es par e impar
    for(i=0;i<10;i++)
    {
        printf(" ingresa el %d valor", i+1);
        scanf("%d", &A[i]);
        if (A[i]%2==0)
            B[i]=A[i]+5;
        else
            B[i]=A[i]*3;
    }
    //imprimir los arreglos unidimensionales de manera independiente para
    //visualizar los elementos ingresados y los procesados
    printf("Los valores originales son: \n");
    for(i=0; i<10; i++)
        printf(" [ %d ] \t ", A[i]);
    printf("\n Los valores que resuelven el problema planteado son: \n");
    for(i=0; i<10; i++)
        printf(" [ %d ] \t ", B[i]);
}
```

La figura 5.3, muestra el código fuente del programa anterior capturado en el entorno. En las líneas de código se puede observar la gama de colores que utiliza para diferenciar las librerías con color verde, las estructuras y palabras reservadas con negritas y sintaxis requerida con rojo, en azul se observan los textos introducidos por el programador y que son parte de los mensajes de salida cuando se ejecute el programa.



```
1  #include <conio.h>
2  #include <stdio.h>
3  main ()
4  {
5  int A[10], B[10], i;
6  //para almacenar 10 valores es necesario usar una estructura iterativa que permita
7  for(i=0;i<10;i++)
8  {
9  printf(" ingresa el %d valor", i+1);
10 scanf("%d", A[i]);
11 if (A[i]%2==0)
12 B[i]=A[i]+5;
13 else
14 B[i]=A[i]*3;
15 }
16 //imprimir los arreglos unidimensionales de manera independiente para visualizar lo
17 printf("Los valores originales son: \n");
18 for(i=0; i<10; i++)
19 printf(" [ %d ] \t ", A[i]);
20 printf("\n Los valores que resuelven el problema planteado son: \n");
21 for(i=0; i<10; i++)
22 printf(" [ %d ] \t ", B[i]);
23 }
24
25
```

Figura 5.3 Código fuente

La figura 5.4, muestra al lector la pantalla una vez que se compila el programa y una vez que este no refleje errores de compilación podrá ser ejecutado con el comando ejecutar o la función F11 y podrá realizarse la prueba de ejecución tantas veces sea necesario para verificar la veracidad de los resultados obtenidos.

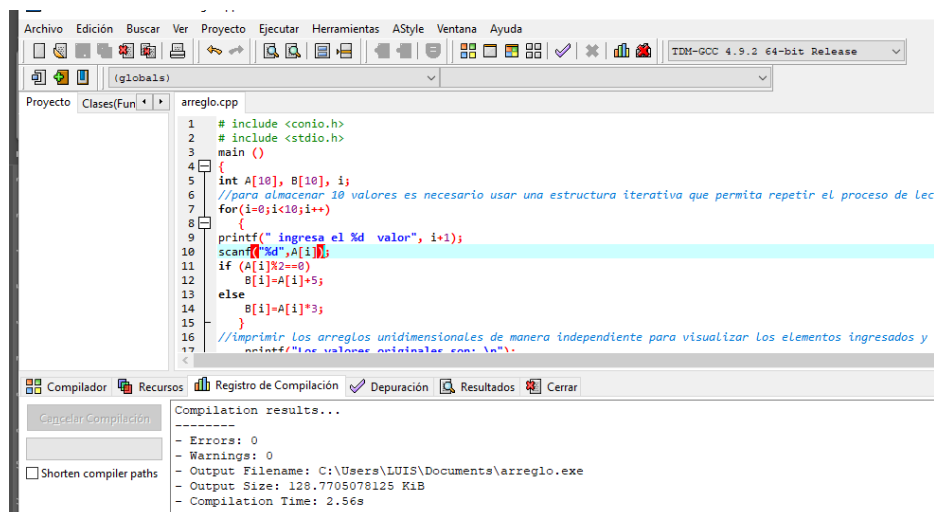


Figura 5.4 Compilación del programa fuente

La figura 5.5, muestra una corrida del programa y como se puede observar cumple con lo solicitado por el ejercicio ya que muestra los datos originales introducidos por el usuario y posteriormente muestra la lista de números tomando en consideración lo solicitado cuando es un número par le suma 5 al número introducido, por ejemplo cuando ingresa el 4 imprime 9, con el 8 imprime 13 y lo mismo con los valores impares al multiplicarlos por 3, en el caso del valor 5 imprime 15 y así sucesivamente con los demás valores por lo que la alternativa de solución presentada se considera válida.

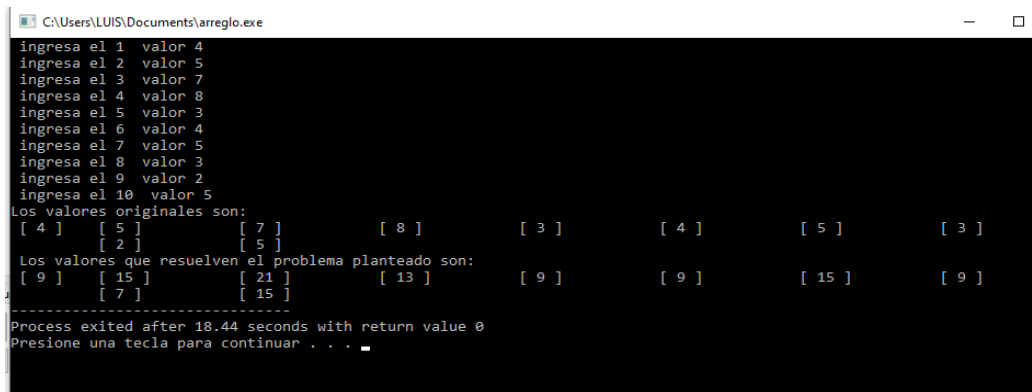


Figura 5.5 Resultados obtenidos al ejecutar el programa fuente



Ejercicio de aprendizaje

Realiza la lectura de 10 valores y almacena en un arreglo Lista1, en un arreglo lista2, almacena un valor 1, si el número leído es múltiplo de 3 y un valor 0 si no lo es. Imprimir ambos arreglos

5.2 Arreglo bidimensionales y multidimensionales

Los arreglos bidimensionales o también conocidos como matriz o tabla, es un tipo de dato que permite almacenar un conjunto de elementos de manera continua. Requieren para su uso de un tipo de dato, un nombre de variable y dos índices, el primer índice corresponde a las filas y el segundo a las columnas que la tabla o matriz permitirá almacenar. La multiplicación de filas por columnas determinará el tamaño de la matriz y por tanto la cantidad de elementos a almacenar, por ejemplo, un arreglo bidimensional de 3 filas por 4 columnas podrá almacenar un total de 24 elementos.

La figura 5.6, muestra el esquema de un arreglo bidimensional de 3 filas y 4 columnas. En cada celda se muestra el nombre de la variable y las posiciones que maneja de almacenamiento. Observe que la primera posición sigue siendo a partir de la posición 0.

M [0][0]	M [0][1]	M [0][2]	M [0][3]
M [1][0]	M [1][0]	M [1][0]	M [1][0]
M [2][0]	M [2][0]	M [2][0]	M [2][0]

Figura 5.6 Arreglo bidimensional M [3][4]

Para declarar un arreglo bidimensional se utiliza la siguiente sintaxis:

<tipo de dato> <Nombre de la variable> [Total_filas] [total_columna]

Donde:

- Tipo de dato, establece que información será almacenada: int, float o char
- Nombre de la variable, es el identificador con el cual se identifica la variable
- Total, de filas y/o columnas, establece el tamaño de la matriz y debe ir entre [] cada uno de los índices requeridos

Por ejemplo, una matriz o arreglo bidimensional de 3 filas y 3 columnas que almacene datos de tipo entero se debe declarar con los elementos mostrados en la figura 5.7

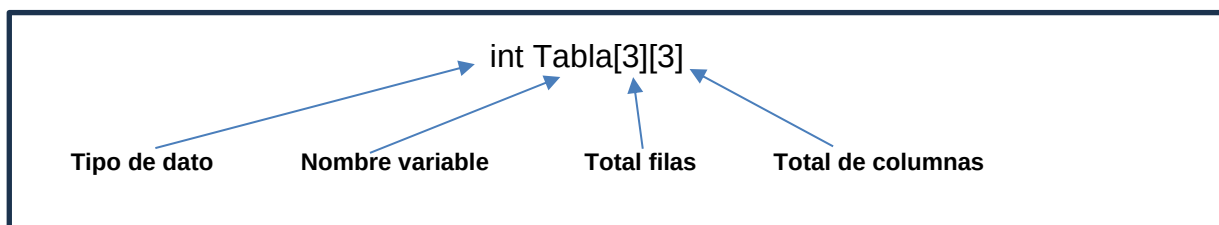


Figura 5.7. Elementos de un arreglo bidimensional

Al igual que un arreglo unidimensional, este tipo de estructura requiere de estructuras iterativas para el control de los índices que exige el almacenamiento de datos, para ello se requiere de dos ciclos anidados para poder controlar cada uno de los índices, el siguiente código muestra cómo se debe llevar a cabo la declaración de una variable bidimensional, el almacenamiento de datos en dicha variable y por último la extracción de los datos almacenados. Observe que es necesario del manejo de 2 índices para el control de los ciclos. A continuación, se muestra el programa y se plasma los comentarios que permite comprender a el lector, la acción de cada proceso, así como una ejecución parcial del programa descrito.

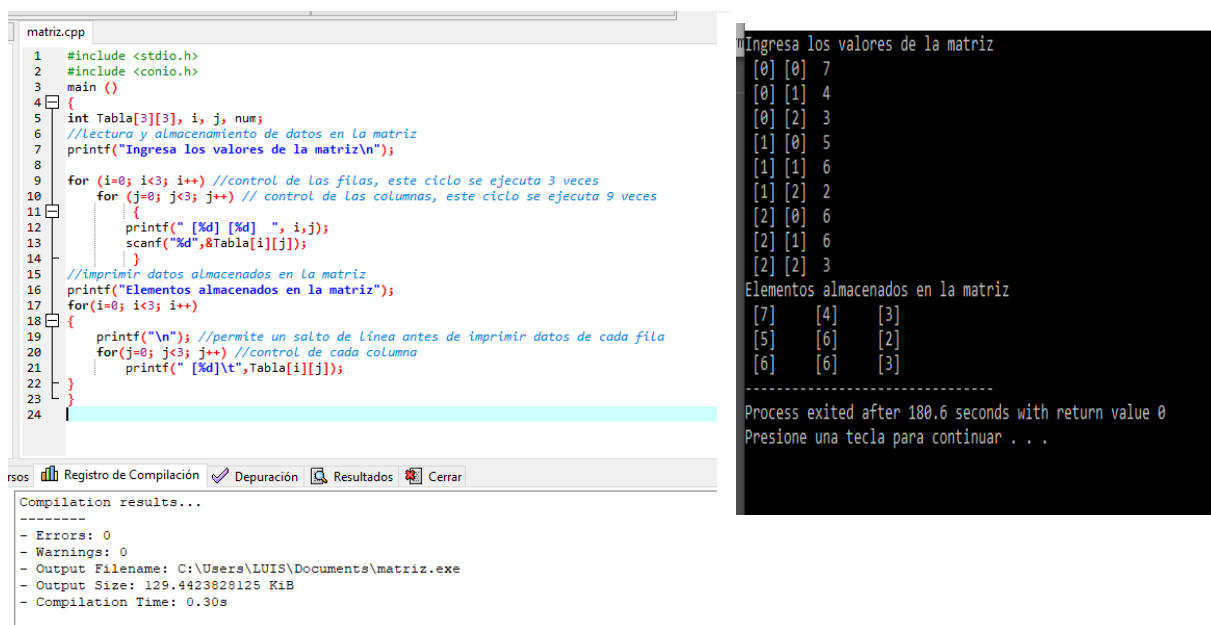


```
#include <stdio.h>
#include <conio.h>
main ()
{
    int Tabla[3][3], i, j, num;
    //lectura y almacenamiento de datos en la matriz
    printf("Ingresa los valores de la matriz\n");

    for (i=0; i<3; i++) //control de las filas, este ciclo se ejecuta 3 veces
        for (j=0; j<3; j++) // control de las columnas, se ejecuta 9 veces
        {
            printf(" [%d] [%d] ", i,j);
            scanf("%d",&Tabla[i][j]);
        }
    //imprimir datos almacenados en la matriz
    printf("Elementos almacenados en la matriz");
    for(i=0; i<3; i++)
    {
        printf("\n"); //permite un salto de línea antes de imprimir cada fila
        for(j=0; j<3; j++) //control de cada columna
            printf(" [%d] \t", Tabla[i][j]);
    }
}
```

```
Ingresa los valores de la matriz
[0] [0] 7
[0] [1] 4
[0] [2] 3
[1] [0] 5
[1] [1] 6
[1] [2] 2
[2] [0] 6
[2] [1] 6
[2] [2] _
```

La figura 5.8, se observa el programa en el entorno una vez compilado y del lado derecho la ejecución del programa y el resultado obtenido.



```
matriz.cpp
1  #include <stdio.h>
2  #include <conio.h>
3  main ()
4  {
5      int Tabla[3][3], i, j, num;
6      //Lectura y almacenamiento de datos en la matriz
7      printf("Ingresa los valores de la matriz\n");
8
9      for (i=0; i<3; i++) //control de las filas, este ciclo se ejecuta 3 veces
10         for (j=0; j<3; j++) // control de las columnas, este ciclo se ejecuta 9 veces
11         {
12             printf(" [%d] [%d] ", i, j);
13             scanf("%d",&Tabla[i][j]);
14         }
15     //Imprimir datos almacenados en la matriz
16     printf("Elementos almacenados en la matriz");
17     for(i=0; i<3; i++)
18     {
19         printf("\n"); //permite un salto de línea antes de imprimir datos de cada fila
20         for(j=0; j<3; j++) //control de cada columna
21             printf(" [%d]\t", Tabla[i][j]);
22     }
23 }
24
```

Ingresa los valores de la matriz

```
[0] [0] 7
[0] [1] 4
[0] [2] 3
[1] [0] 5
[1] [1] 6
[1] [2] 2
[2] [0] 6
[2] [1] 6
[2] [2] 3
```

Elementos almacenados en la matriz

```
[7] [4] [3]
[5] [6] [2]
[6] [6] [3]
```

Process exited after 180.6 seconds with return value 0
Presione una tecla para continuar . . .

rsos Registro de Compilación Depuración Resultados Cerrar

Compilation results...

```
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\LUIS\Documents\matriz.exe
- Output Size: 129.4423828125 KiB
- Compilation Time: 0.30s
```

Figura 5.8 Código fuente para almacenar e imprimir datos de una matriz

Ejemplo de uso

Leer el tamaño de una matriz cuyo valor máximo del tamaño sea de 10x10, leer los valores acordes al tamaño definido y almacenarlos en la matriz A1, en una segunda matriz A2, calcular el cuadrado de cada valor e Imprimir ambas matrices.

Para resolver este ejercicio es necesario observar que el tamaño de la matriz lo define el programador, por tanto, se recomienda declarar las matrices después de conocer el valor del tamaño para optimizar espacio de memoria y antes de ser utilizadas las variables para evitar errores de ejecución. El siguiente código refleja una alternativa de solución.

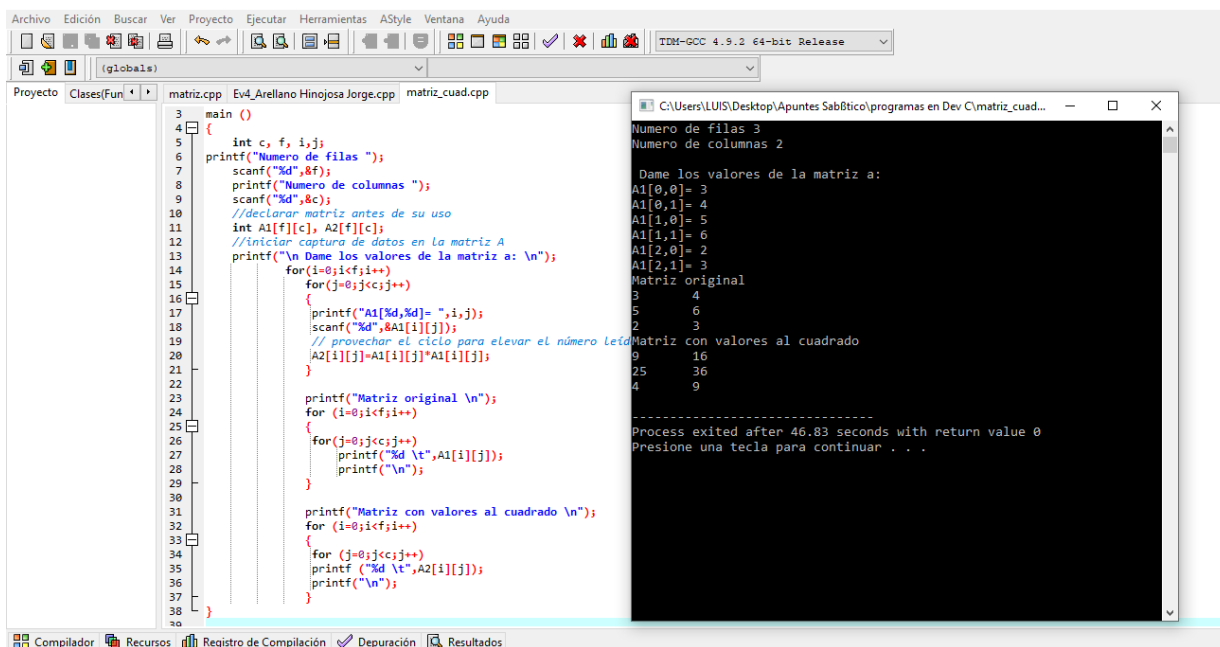


```
#include <stdio.h>
#include <conio.h>
main ()
{
    int c, f, i, j;
    printf("Numero de filas ");
    scanf("%d",&f);
    printf("Numero de columnas ");
    scanf("%d",&c);
    //declarar matriz antes de su uso
    int A1[f][c], A2[f][c];
    //iniciar captura de datos en la matriz A
    printf("\n Dame los valores de la matriz a: \n");
    for(i=0;i<f;i++)
        for(j=0;j<c;j++)
        {
            printf("A1[%d,%d]= ",i,j);
            scanf("%d",&A1[i][j]);
            // provechar el ciclo para elevar el número leído al cuadrado
            A2[i][j]=A1[i][j]*A1[i][j];
        }

    printf("Matriz original \n");
    for (i=0;i<f;i++)
    {
        for(j=0;j<c;j++)
            printf("%d \t",A1[i][j]);
        printf("\n");
    }

    printf("Matriz con valores al cuadrado \n");
    for (i=0;i<f;i++)
    {
        for (j=0;j<c;j++)
            printf ("%d \t", A2[i][j]);
        printf("\n");
    }
}
```

La figura 5.9, muestra del lado izquierdo el código fuente y del lado izquierdo y la ejecución del programa que resuelve el ejercicio planteado en el entorno de Dev C++



```
3 main ()
4 {
5     int c, f, i, j;
6     printf("Numero de filas ");
7     scanf("%d", &f);
8     printf("Numero de columnas ");
9     scanf("%d", &c);
10    //declarar matriz antes de su uso
11    int A1[f][c], A2[f][c];
12    //iniciar captura de datos en la matriz A
13    printf("\n Dame los valores de la matriz a: \n");
14    for(i=0; i<f; i++)
15        for(j=0; j<c; j++)
16        {
17            printf("A1[%d,%d]= ", i, j);
18            scanf("%d", &A1[i][j]);
19            // aprovechar el ciclo para elevar el número leído al cuadrado
20            A2[i][j] = A1[i][j] * A1[i][j];
21        }
22
23    printf("Matriz original \n");
24    for (i=0; i<f; i++)
25    {
26        for(j=0; j<c; j++)
27            printf("%d\t", A1[i][j]);
28        printf("\n");
29    }
30
31    printf("Matriz con valores al cuadrado \n");
32    for (i=0; i<f; i++)
33    {
34        for (j=0; j<c; j++)
35            printf("%d\t", A2[i][j]);
36        printf("\n");
37    }
38 }
39 }
```

Numero de filas 3
Numero de columnas 2
Dame los valores de la matriz a:
A1[0,0]= 3
A1[0,1]= 4
A1[1,0]= 5
A1[1,1]= 6
A1[2,0]= 2
A1[2,1]= 3
Matriz original
3 4
5 6
2 3
Matriz con valores al cuadrado
9 16
25 36
4 9

Process exited after 46.83 seconds with return value 0
Presione una tecla para continuar . . .

Figura 5.9 Uso de matrices

Ejercicio de aprendizaje

Realiza la lectura de valores que se almacenen en una matriz de 3x3 y almacena en una segunda matriz la raíz cuadrada de aquellos elementos que sean impares, para los elementos pares colocar un cero. Imprimir matriz original y matriz con el proceso solicitado. No olvides integrar la librería de uso para funciones matemáticas

Estructura Multidimensional



Un arreglo **multidimensional** es considerado un dato estructurado definido por el usuario ya que permite almacenar datos de diferentes tipos en una sola variable, es una estructura que la terminología de las bases de datos se les conoce como registros y a su vez si la variable contempla el almacenamiento de varios registros se considera un archivo de registros. Existe software de aplicación especializado para la generación de bases de datos, en Dev C, se puede generar una variable multidimensional básica utilizando la palabra reservada **struct**, la cual permite generar una plantilla con la que posteriormente se puede emplear en una variable compuesta o multidimensional básica. La forma más simple de definir una estructura y declarar una variable del tipo de la estructura definida es realizando los siguientes pasos:

1. Definir la estructura con la siguiente sintaxis:

```
struct <nombre de estructura>
{
    <Tipo de dato1> < campo1>
    <Tipo de dato2> < campo2>
    ...
    <Tipo de dato n> < campo n> };
```

2. Declarar la variable del tipo estructura definida

```
<Nombre de estructura> <Nombre de la variable>
```

Si se define una estructura para almacenar los datos de nombre, edad y promedio de 10 estudiantes en una sola variable, se define la estructura y posteriormente la variable sin olvidar que la estructura debe plasmarse antes del menú principal y lleva punto y coma “;” al final de la estructura como se muestra a continuación:

```
struct Datos
{
    char nombre[20];
    int edad;
    float prom;
};
```



```
main ()
```

```
{
```

```
    Datos Lista[10];
```

Ejemplo de uso.

Se desea almacenar en un arreglo el nombre y edad de los 5 trabajadores de una empresa de dulces en una misma variable e imprimir la lista de trabajadores.

```
# include <stdlib.h> //uso de librería para variables tipo cadena
```

```
# include <stdio.h>
```

```
# include <conio.h>
```

```
struct datos //establecer el nombre de la estructura y sus elementos
```

```
{ //elementos de la estructura
```

```
    char nombre[30]; //se define un arreglo de caracteres
```

```
    int edad;
```

```
};
```

```
typedef datos registro_emp; //definir el tipo de dato con la estructura establecida
```

```
main ()
```

```
{
```

```
    registro_emp Lista[5]; //definir la variable con el tipo de dato definido
```

```
    int i;
```

```
    printf("Captura de registros\n");
```

```
    for (i=0; i<5; i++)
```

```
    {
```

```
        printf(" ingresa el nombre del trabajador:\t ");
```

```
        fflush (stdin);
```

```
        gets(Lista[i].nombre); //para leer una cadena de caracteres
```

```
        printf("Ingresa su edad: \t");
```

```
        scanf("%d",&Lista[i].edad);
```

```
    }
```

```
    printf("Los empleados son: \n");
```

```
    for (i=0; i<5; i++)
```

```
    {
```

```
        printf("Nombre del trabajador \t %s",Lista[i].nombre);
```

```
        printf("\t Edad:\t  %d", Lista[i].edad);
```

```
        printf("\n");
```

```
    }
```

```
}
```

La figura 5.10 muestra la corrida del programa, donde se puede observar que Lista es una variable tipo arreglo unidimensional que a su vez almacena una estructura definida por el programador que contiene 2 campos: nombre y edad y ambos son de diferente tipo, es una estructura pequeña pero la cantidad de elementos los define el programador en función a la alternativa de solución que permita resolver el problema planteado.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <conio.h>
4 struct datos
5 {
6     char nombre[30];
7     int edad;
8 };
9 typedef struct datos registro_emp;
10 main ()
11 {
12     registro_emp Lista[5];
13     int i;
14     printf("Captura de registros\n");
15     for (i=0; i<5; i++)
16     {
17         printf("ingresa el nombre del trabajador:\t ");
18         fflush (stdin);
19         gets(Lista[i].nombre);
20         printf("ingresa su edad: \t");
21         scanf("%d",&Lista[i].edad);
22     }
23     printf("Los empleados son: \n");
24     for (i=0; i<5; i++)
25     {
26         printf("Nombre del trabajador \t %s",Lista[i].nombre);
27         printf("\t Edad:\t %d", Lista[i].edad);
28         printf("\n");
29     }
30 }
31
```

```
Captura de registros
ingresa el nombre del trabajador:    Luis Hernandez
ingresa su edad:    45
ingresa el nombre del trabajador:    Maria Perez
ingresa su edad:    34
ingresa el nombre del trabajador:    Raul Lara
ingresa su edad:    23
ingresa el nombre del trabajador:    Laura Juarez
ingresa su edad:    56
ingresa el nombre del trabajador:    Martha Perez
ingresa su edad:    34
Los empleados son:
Nombre del trabajador    Luis Hernandez    Edad:    45
Nombre del trabajador    Maria Perez    Edad:    34
Nombre del trabajador    Raul Lara    Edad:    23
Nombre del trabajador    Laura Juarez    Edad:    56
Nombre del trabajador    Martha Perez    Edad:    34
-----
Process exited after 49.72 seconds with return value 0
Presione una tecla para continuar . . .
```

Figura 5.10 Arreglos multidimensionales o estructuras compuestas

Ejercicio de aprendizaje

Tomando como base el ejercicio anterior, capturar en un arreglo, el nombre del estudiante, el nombre de la materia cursada, 3 calificaciones y el promedio de estas. Imprimir la lista final con los datos obtenidos



5.3 Apertura, entrada-salida de datos, y cierre de archivos

Un archivo es una estructura compuesta que permite el almacenamiento en memoria, de información tipo texto o de manera binaria. La creación y manejo de archivos requiere del uso de una librería llamada ***fstream*** que permite crear una instancia ***fich*** para leer información de un archivo, con esta función se abre un archivo para lectura. Por lo que cuando se quiera leer o escribir en un archivo es necesario utilizar esta función, para poder trabajar con archivos, con ***ifstream***, permite la lectura de datos; ***ofstream***, permite la escritura de datos y con ***fstream*** se emplea cuando se requiere leer y escribir en el archivo, en el mismo programa.

La librería ***iostream***, permite organizar el código en grupos lógicos, así también, evitan conflictos de nombres que pueden generarse cuando el programa utiliza bibliotecas. ***Stdlib***, es una librería estándar que contiene prototipos para gestionar memoria dinámica, control de procesos, entre otras funciones. El uso de archivos más completo requiere de un modelo orientado a objetos, en la programación estructurada el manejo de archivos solamente es de tipo texto o con estructuras definidas por el usuario. Para definir estructuras basadas en objetos y clases se requiere un modelo diferente de programación, así como el entorno C++ que soporta la programación Orientada a Objetos.

A continuación, se plasma un programa que permite crear un archivo cuya estructura se define por el usuario tipo texto, el cual se considera como una secuencia de caracteres organizadas en una línea y terminadas por un carácter establecido. En la figura 5.11, se muestra la compilación y ejecución del código fuente generado en el entorno Dev C++. El programa presenta una estructura definida por el programador considerando algunos campos de captura como nombre, edad, apellido, teléfono, entre otros. Utiliza a diferencia de ***printf*** y ***scanf*** las



funciones *cin*, *cout* y *gets* porque al ser un almacenamiento en memoria secundaria éstas funciones permiten al compilador distinguir el tipo de carácter de inicio y término de una línea de caracteres³.

#include <fstream>	cout << "0.Salir" <<	fprintf(fichero, "%d ",
#include <stdio.h>	endl;	persona.Edad);
#include <iostream>		puts("\nNombre:");
#include <cstdlib>	cout << "\nIngrese una	gets(persona.Nombre);
	opcion: ";	fprintf(fichero, "%s ",
using namespace std;	cin >> opcion;	persona.Nombre);
		puts("Apellido: ");
int main()	switch (opcion) {	gets(persona.Ape);
{	case 1:	fprintf(fichero, "%s ",
int opcion;		persona.Ape);
bool repetir = true;	struct datos	puts("Telefono: ");
	{	gets(persona.Telf);
FILE *fichero;	char Nombre[21];	fprintf(fichero, "%s ",
int i;	int Edad;	persona.Telf);
	char Ape[10];	puts("Cedula: ");
fichero =	char Telf[20];	gets(persona.Cedu);
fopen("Registro.txt",	char Cedu[20];	fprintf(fichero, "%s ",
"wt");		persona.Cedu);
if (fichero == NULL)	char producto[21];	fclose(fichero);
printf("No se pudo abrir	char codi[21];	}
el archivo.\n");	char cant[21];	fichero =
		fopen("Registro.txt",
do {	}persona;	"rt");
system("cls");		if (fichero == NULL)
	{	printf("No se pudo abrir
	int Edad;	el archivo.\n");
cout << "\n\nMenu de	do{	else
Registro" << endl;	puts("Edad:");	{
cout << "1.Registrar	scanf("%d",	while (!feof(fichero))
Cliente" << endl;	&persona.Edad);	
cout << "2.Registrar	getchar();	{
Producto" << endl;	}while(persona.Edad<	fgets(persona.Nombre,
	18);	21, fichero);

³ Por la extensión del programa se muestra a 3 columnas y el lector puede acceder al código en la carpeta de materiales de apoyo.



```
puts(persona.Nombre);
fscanf(fichero, "%d",
&persona.Edad);
getchar();
printf("%d",
persona.Edad);
fgets(persona.Ape, 10,
fichero);
puts(persona.Ape);
fgets(persona.Telf, 20,
fichero);
puts(persona.Telf);
fgets(persona.Telf, 20,
fichero);
puts(persona.Telf);
}
fclose(fichero);
}
```

```
system("pause>nul");
break;
```

```
case 2:
{
puts("\nRegistrar
Producto");
gets(persona.producto
);
fprintf(fichero, "%s ",
persona.producto);
```

```
puts("\nNombre del
Producto: ");
gets(persona.producto
);
fprintf(fichero, "%s ",
persona.producto);
puts("\nRegistrar
Codigo de Producto:
");
gets(persona.codi);
fprintf(fichero, "%s ",
persona.codi);
puts("\nRegistrar
Cantidad de Producto:
");
gets(persona.cant);
fprintf(fichero, "%s ",
persona.cant);
```

```
fclose(fichero);
}
fichero =
fopen("Registro.txt",
"rt");
if (fichero == NULL)
printf("No se pudo abrir
el archivo.\n");
else
{
while (!feof(fichero))
```

```
{
fgets(persona.producto
, 21, fichero);
puts(persona.producto
);
fgets(persona.codi, 21,
fichero);
puts(persona.codi);
fgets(persona.cant, 21,
fichero);
puts(persona.cant);
}
fclose(fichero);
}

system("pause>nul");
break;

case 0:
repetir = false;
break;
} while (repetir);

return 0;
}
```

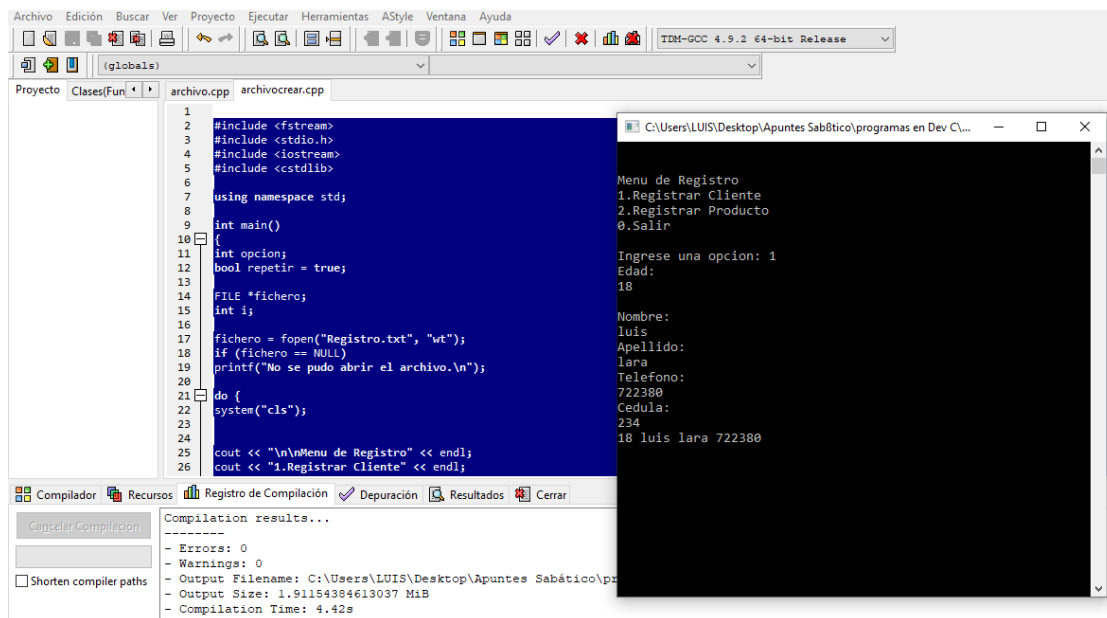


Figura 5.11 Compilación y ejecución de un archivo

La figura 5.12, muestra el archivo generado en memoria secundaria y el almacenamiento de la información procesada.

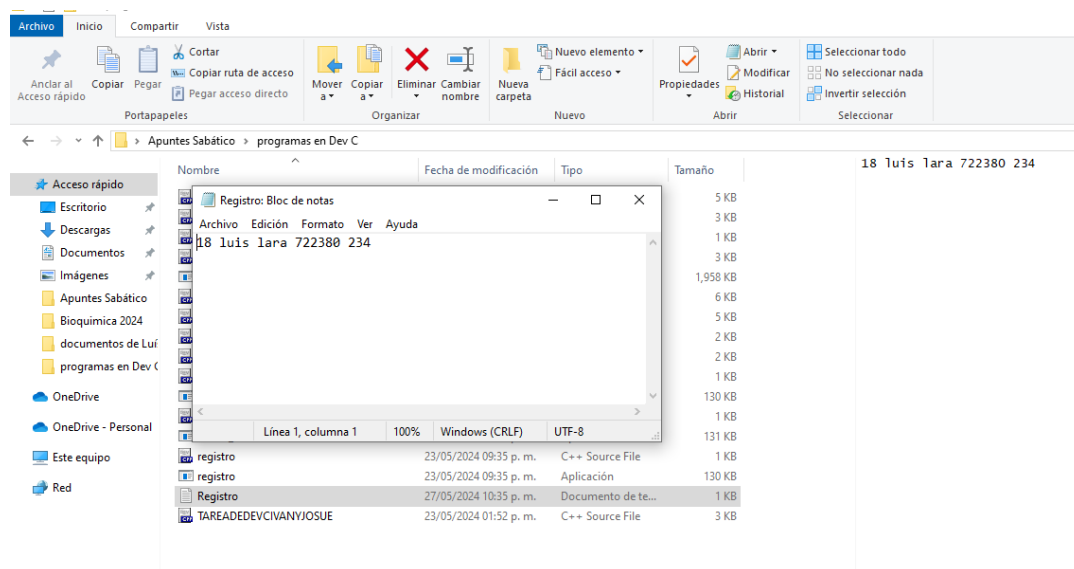


Figura 5.11 Almacenamiento de un archivo de texto



5.4 Elaboración de programas

A continuación, se plasma un conjunto de programas los cuales utilizan las estructuras compuestas: unidimensionales y bidimensionales, así como las diferentes estructuras iterativas y de selección descritas anteriormente, para que el lector continúe practicando y fortaleciendo su aprendizaje en el tema, capturando los programas en el entorno con la finalidad de que sean compilados y ejecutados y pueda comprender su funcionamiento.

Programa 1. Capturar en un arreglo A, 5 calificaciones obtenidas por un estudiante. En un arreglo B, considere un 0 si la calificación es menor z 70 y respetar la calificación si es mayor o igual a 70. Imprimir ambos arreglos.

Alternativa de solución

```
#include <conio.h>
#include <stdio.h>
main()
{
    int i;
    int a[5],b[5];
    for(i=0;i<5;i++)
    {
        printf("Ingrese la calificacion %d: ",i+1);
        scanf("%d",&a[i]);
        b[i]=(a[i]<70)?0:a[i];
    }
    printf("\nCalificaciones ingresadas en el arreglo a:\n");
    for(i=0;i<5;i++)
    {
        printf("%d\t",a[i]);
    }
    printf("\nCalificaciones ingresadas en el arreglo b:\n");
    for(i=0;i<5;i++)
    {
        printf("%d\t",b[i]); } }
```



Programa 2. Capturar 10 números, Sumar los números pares y contar los números impares e imprimir los resultados solicitados

```
#include <conio.h>
#include <stdio.h>
main()
{
    int i,num, sumapares=0, contadorimpares=0;
    printf("Ingrese 10 numeros:\n");
    for(i=0;i<10;i++)
    {
        printf("Numero %d: \n",i+1);
        scanf("%d",&num);
        if(num%2==0)
        {
            //Es par, agregar a la suma
            sumapares+=num;
        }
        else
        {
            //Es impar, incrementar contador
            contadorimpares++;
        }
    }
    // Salida de resultados
    printf("La suma de los numeros pares es:%d\n",sumapares);
    printf("La cantidad de numeros impares es:%d\n",contadorimpares);
}
```

Programa 3. Realizar un menú de opciones con los siguientes ejercicios:

- Almacenar 5 números en un arreglo, pero si el número ingresado es múltiplo de 5 se almacena un cero. Imprimir el arreglo
- Leer un número, e imprimir los múltiplos de 6, del 1 hasta el valor ingresado
- Otorgar un descuento en la inscripción a un estudiante de acuerdo con su promedio. Si su promedio es mayor o igual que 90 se le otorga el 25% de



descuento y si es mayor o igual a 80 y menor que 90, el descuento será del 10%.

d) Fin del menú

Alternativa de solución

```
#include <stdio.h>
#include <conio.h>
int main() {
    int opcion;
    do {
        printf("Menú de opciones:\n");
        printf("1. Calcular múltiplos de 5\n");
        printf("2. Imprimir múltiplo 6 \n");
        printf("3. Monto de Inscripción\n");
        printf("0. Salir\n");
        printf("Ingresa el numero de la opción deseada:\n");

        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                int numeros[5];
                for (int i = 0; i < 5; ++i) {
                    printf("Ingresa el numero %d: ", i + 1);
                    scanf("%d", &numeros[i]);

                    if (numeros[i] % 5 == 0) {
                        numeros[i] = 0;
                    }
                }
                printf("Arreglo final: ");
                for (int i = 0; i < 5; ++i)
                    printf("%d ", numeros[i]);
                break;
            case 2:
                int numero;
                printf("Ingresa un numero: ");
                scanf("%d", &numero);
                printf("Múltiplos de 6 hasta %d:\n", numero);
```



```
for (int i = 1; i <= numero; i++) {
    if (i % 6 == 0)
        printf("%d\n", i);
    break;
case 3:
    int costoInscripcion = 2750;
    // Datos del estudiante
    float promedio;
    // Solicitar el promedio al usuario
    printf("Ingresa el promedio del estudiante: ");
    scanf("%f", &promedio);
    // Calcular el descuento y el monto a pagar
    float descuento = 0;
    float totalAPagar = 0;

    if (promedio > 90)
        descuento = 0.25; // Descuento del 25%
    else if (promedio >= 80)
        descuento = 0.10; // Descuento del 10%
    totalAPagar = costoInscripcion - (costoInscripcion * descuento);
    // Imprimir el total a pagar y el descuento aplicado
    printf("Total a pagar: %.2f\n", totalAPagar);
    printf("Descuento aplicado: %.2f%%\n", descuento * 100);
    break;
case 4:
    printf("Saliendo del programa...\n");
    break;
default:
    printf("Opcion no valida. Ingrese una opcion valida.\n");
} //switch
} while (opcion != 4);
} //main
```

Programa 4. Realice un menú de opciones con los siguientes ejercicios:

1. Obtener el promedio de n calificaciones usando arreglos
2. Realizar la suma de matrices de $n \times n$
3. Leer valores en una matriz de $n \times m$ y en otra matriz almacenar la raíz cuadrada de cada valor de la matriz original. Imprimir ambas matrices



4. Fin del programa

Alternativa de solución

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
#include <String.h>
#include <iomanip>
#include <math.h>
main()
{
    int op;
    int mat,i2;
    float cal[10], prom=0;
    int alumn[45], total_a,i;
    int i4,j4,n4,m4, x5;
    float a4[10][10], b[10][10], c4[10][10];
    int i3,j,f,c;
    float x[20][20], y[20][20];
    do
    {
        printf("Ingrese la opcion que desea utilizar \n");
        printf( "1: Promedio de una lista no mayor a 10 materias\n");
        printf( "2. Suma de matrices nxn \n");
        printf("3. Raiz de una matriz \n");
        printf("4. Finalizar \n");
        printf("Ingrese la opción deseada: ");
        scanf ("%d",&op);
        switch (op) {
            case 1:
                printf ("¿ Cuantas materias llevas (máximo 10)? \n");
                scanf("%d",&mat);
                for (i2=0; i2<mat;i2++){
                    printf ("Calificacion de la %d materia: ",i2+1);
                    scanf("%f",&cal[i2]);
                    prom=prom+cal[i2];
                }
                cal[mat]=prom/mat;
                printf("El promedio es %4.2f \n",cal[mat]);
                break;
```



case 2:

```
printf("Número de filas");
scanf("%d",&n4);
printf ("Numero de columnas");
scanf ("%d",&m4);
printf("Dame los valores de la matriz a \n");
for (i4=0; i4<n4; i4++)
    for (j4=0; j4<m4; j4++)
    {
        printf("a(%d,%d)= ",i4,j4);
        scanf("%f",&a4[i4][j4]);
    }
printf("Dame los valores de la matriz b \n");
for (i4=0;i4<n4;i4++)
for (j4=0;j4<m4;j4++)
{
    printf("b(%d,%d)=",i4,j4);
    scanf("%f",&b[i4][j4]);
}
for (i4=0;i4<n4;i4++)
for(j4=0;j4<m4;j4++)
c4[i4][j4]=a4[i4][j4]+b[i4][j4];
printf("Matriz Suma: \n");
for (i4=0; i4<n4; i4++)
for (j4=0; j4<m4; j4++)
printf("%f \n", c4[i4][j4]);
break;
```

case 3:

```
printf("Numero de filas ");
scanf("%d",&f);
printf("Numero de columnas ");
scanf("%d",&c);
printf("\n Dame los valores de la matriz a: \n");
for(i3=0;i3<f;i3++)
    for(j=0;j<c;j++)
    {
        printf("a(%d,%d)= ",i3,j);
        scanf("%f",&x[i3][j]);
        y[i3][j]=sqrt (x[i3][j]);
    }
}
```



```
printf("Matriz original \n");
for (i3=0;i3<f;i3++)
{
    for(j=0;j<c;j++)
        printf("%5.2f ",x[i3][j]);
    printf("\n");
}

printf("Matriz Raiz cuadrada \n");
for (i3=0;i3<f;i3++)
{
    for (j=0;j<c;j++)
        printf ("%5.2f",y[i3][j]);
    printf("\n");
}

break;
case 4:
    printf("Gracias por su atención");
    break;
default:
    printf("Opción no válida");
}
} while (op!=4); }
```

5.5 Aprendizaje basado en problemas

Revisa el código del programa 4 y realiza lo siguiente:

1. Analiza el programa y modifica el código que pueda optimizar líneas sin alterar el resultado.
2. Valida el ingreso de datos para que no exceda el tamaño de la matriz y solo acepte valores del 1 al 5, en los ejercicios 2 y 3 del programa 4
3. Revisa el tiempo de ejecución del programa 4 original y del que tú propones con mejoramiento de código y menciona que cambios significativos existieron.



6. Funciones

6.1 Introducción

En la programación se presentan problemas complejos y una de las formas de resolverlos es dividiendo en partes dicho problema y plantear alternativas de solución para cada una de las partes, las cuales se conocen con el nombre de módulos y de esta manera es más sencillo programar y modificar cada uno de los módulos que analizar y programar el programa completo. En el tema 1, se habló del diseño descendente, el cual hace referencia a dividir un problema en módulos o subprogramas desde el problema como eje central y los subprogramas o módulos en los que se puede dividir para hacer más sencilla el análisis y la programación.

En el entorno Dev C++, un módulo o subprograma se le denomina función. Existen funciones predefinidas las cuales se agrupan en librerías y se encuentran disponibles en la biblioteca estándar del entorno y permiten ejecutar diversas acciones que el programador utiliza.

Una función, es un conjunto de sentencias que realiza una tarea específica y puede recibir o no valores para su ejecución, para poder emplearlas es necesario declararlas previamente. Las funciones poder colocarse antes o después del programa principal (main), respetando algunos requisitos que establece el entorno, como se explica más adelante. Las funciones se dividen en: con pase de parámetros y sin pase de parámetros. En los siguientes subtemas se detalla el uso e importancia de las funciones en Dev C++

Las ventajas de dividir un programa en funciones o módulos son:

- Simplificar el diseño de las alternativas de solución
- La depuración y/o validación del código se puede realizar en cada función



- La función puede ser parte de una librería o biblioteca y ser utilizada en otros programas
- Cada función se puede elaborar de manera individual facilitando el trabajo en equipo

En Dev C++, se pueden diseñar diferentes tipos de funciones, a continuación, se describen de manera específica para poder comprender su diseño, uso y aplicación.

6.2 Funciones estándar

La mayoría de los lenguajes permiten el uso de bibliotecas con funciones predefinidas o estándar que pueden ser utilizadas en casi todos los entornos, por ejemplo: raíz cuadrada de un número, el seno, coseno, tangente de un ángulo, el valor absoluto, entre otras muchas funciones estándar que se emplean para resolver problemas ya sea matemáticos, aritméticos o trigonométricos por mencionar algunos. La tabla 6.1, muestra una lista de algunas funciones estándar matemáticas que se almacenan en la librería `math.h` y pueden ser utilizadas en cualquier programa siempre y cuando se incluya la librería y se respete la sintaxis de uso en el entorno.

Tabla 6.1 Algunas funciones estándar de la librería `math.h`

Función	Descripción	Ejemplo de uso	Resultado Obtenido
<code>exp(x)</code>	e^x exponente de x	<code>exp(1)</code>	2.7183
<code>sin(x)</code>	Seno de un ángulo x	<code>sen(45)</code>	0.70710
<code>sqrt(x)</code>	Raíz cuadrada del x	<code>sqrt(25)</code>	5.0
<code>pow(x,y)</code>	Elevar x al valor de y	<code>pow(2,4)</code>	16.0
<code>log(x)</code>	Logaritmo natural de x	<code>log(10)</code>	2.3026



Ejemplo de uso:

Generar 100 números aleatorios e imprimirlos

Para resolver el ejercicio es necesario incluir la librería `stdlib.h` ya que en ella se encuentra la función `rand`, que nos permite generar números aleatorios. A continuación, se plasma el programa fuente que permite resolver el ejemplo y la figura 6.1, muestra el código y la ejecución en el entorno Dev C++

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h> //se incluye la librería para hacer uso de la función rand
main()
{
    int num_al,i;
    printf("Generando números aleatorios\n");
    rand();
    for(i=1; i<100; i++)
    {
        num_al=rand()%10;
        printf("%d ",num_al);
    }
}
```

```
Proyecto Clases(Fun... aleatorio.cpp
1 #include <conio.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 main()
5 {
6     int num_al,i;
7     printf("Generando números aleatorios\n");
8     rand();
9     for(i=1; i<100; i++)
10    {
11        num_al=rand()%10;
12        printf("%d ",num_al);
13    }
14 }
```

Generando n-meros aleatorios

7 4 0 9 4 8 8 2 4 5 5 1 7 1 1 5 2 7 6 1 4 2 3 2

2 1 6 8 5 7 6 1 8 9 2 7 9 5 4 3 1 2 3 3 4 1 1 3

8 7 4 2 7 7 9 3 1 9 8 6 5 0 2 8 6 0 2 4 8 6 5 0

9 0 0 6 1 3 8 9 3 4 4 6 0 6 6 1 8 4 9 6 3 7 8 8

2 9 1

Process exited after 0.07387 seconds with return value 0

Presione una tecla para continuar . . .

Figura 6.1 Uso de función estándar `rand()`



Ejercicio de aprendizaje

Realiza un menú de opciones que permita dados 2 números, realizar las siguientes opciones:

1. Elevar el primer número a la potencia del segundo
2. Multiplicar los números y con la multiplicación obtenida generar números aleatorios
3. Obtener el área de un triángulo tomando el primer número como la base y el segundo como la altura
4. Finalizar el menú

6.3 Entrada y salida de datos

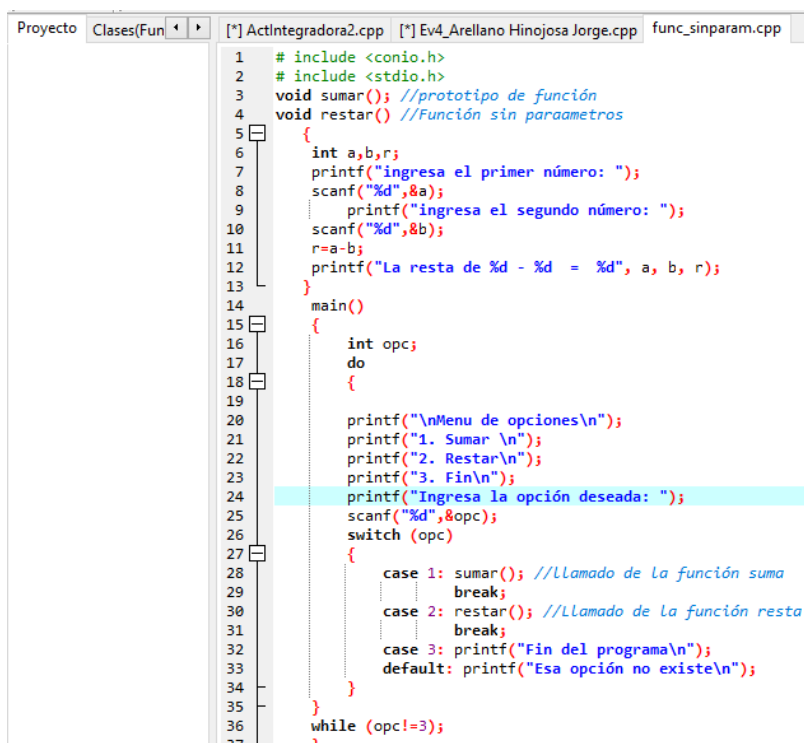
A la primera línea de una función se le conoce con el nombre de prototipo y la ventaja de utilizarla es porque se pueden colocar las funciones en cualquier lugar y en cualquier orden. El uso de funciones permite la entrada y salida (E/S) de datos y dependiendo de cada E/S, es el nombre que recibe cada una de ellas.

Las funciones que no reciben datos de entrada y solo realizan el proceso solicitado y emiten salida de datos, se les conoce con el nombre de **función sin parámetros**.

Las funciones que reciben datos de entrada y generan un dato de salida se les conoce como **funciones con paso de parámetros** y son las más utilizadas. Cada una de estas funciones tienen características importantes de atender por lo que en los siguientes apartados se explica cada una de ellas y se le proporciona al lector ejercicios de uso para su comprensión y aplicación.

6.4 Funciones definidas por el usuario

La base principal de un programa son las funciones que usa, por ejemplo `main()`, es la función principal de un programa fuente y para iniciar la ejecución debe existir y esta función a su vez hace llamar a otras desde el contenido ubicado entre `{}`. Cuando se ejecuta, el código, el compilador siempre inicia con la ejecución de la primera instrucción o sentencia después de `main()` y continúa de manera secuencial hasta el cierre de su contenido. La figura 6.3, muestra un ejemplo de ello.



```
1  #include <conio.h>
2  #include <stdio.h>
3  void sumar(); //prototipo de función
4  void restar() //Función sin parametros
5  {
6      int a,b,r;
7      printf("ingresa el primer número: ");
8      scanf("%d",&a);
9      printf("ingresa el segundo número: ");
10     scanf("%d",&b);
11     r=a-b;
12     printf("La resta de %d - %d = %d", a, b, r);
13 }
14 main()
15 {
16     int opc;
17     do
18     {
19
20         printf("\nMenu de opciones\n");
21         printf("1. Sumar \n");
22         printf("2. Restar\n");
23         printf("3. Fin\n");
24         printf("Ingresa la opción deseada: ");
25         scanf("%d",&opc);
26         switch (opc)
27         {
28             case 1: sumar(); //llamado de la función suma
29                     break;
30             case 2: restar(); //Llamado de la función resta
31                     break;
32             case 3: printf("Fin del programa\n");
33                     default: printf("Esa opción no existe\n");
34         }
35     }
36     while (opc!=3);
37 }
```

Figura 6.3 Uso de funciones

El factor principal de la función `main()`, es coordinar a otras funciones utilizando el proceso de llamar o invocar a la función, en este sentido, el programador no solamente puede utilizar funciones predefinidas, también puede generar sus propias funciones solo se requiere seguir los siguientes pasos:



1. Declarar la función previamente
2. Llamar la función dentro del main(), con el nombre que se generó seguida de paréntesis⁴
3. Ubicar la función en el programa. Una función puede ir antes o después del main(), solo se debe tomar en cuenta que cuando se coloque el código de la función después del main(), entonces, debe declararse el prototipo de la función antes del main() para que sea reconocida la función.

Como ya se había mencionado anteriormente, existen dos tipos de funciones: **con parámetros y sin parámetros**, en ambos casos es necesario nombrar o identificar la función seguida de paréntesis, los cuales estarán vacíos en su interior si no requieren de parámetros y tendrán información cuando sea con parámetros, como se muestra a continuación.

Función sin parámetros:

```
void nombre_función ()  
{  
    Declarar variables  
    Sentencia o bloque de sentencias de la función  
}
```

Función con parámetros

```
<tipo de dato> nombre_función (parámetros)  
{  
    Sentencia o bloque de sentencias de la función  
}
```

⁴ Revisar tema 6.5 y 6.6 para validar la estructura con parámetros y sin parámetros de una función



Donde:

void, es un tipo de dato nulo o vacío, y **parámetros**, son los valores que puede recibir una función y si tienen al inicio un tipo de dato, entonces la función regresa un valor de ese tipo.

Las **funciones sin parámetros** son aquellas que no reciben valores, solo realizan un proceso específico y emite los resultados obtenidos dentro de la función. A continuación, se muestran las dos formas en las que se puede declarar y definir las funciones. Cuando se coloca la función antes de la función principal (main), y cuando solo se declara el prototipo de la función y las sentencias de la función se plasman después del main.

Sintaxis de uso cuando la función se declara antes del main():

```
<incluir librerías>
void nombre_función()
{
    Declarar variables;
    Bloque de sentencias de la función;
}
main()
{
    Declarar variables;
    Sentencias;
    Nombre_función();
    Sentencias;
}
```

Sintaxis de uso cuando se declara solo el prototipo de la función antes del main() y el cuerpo de la función después del main();



```
<incluir librerías>
void nombre_función();
main()
{
    Declarar variables;
    Sentencias;
    Nombre_función();
    Sentencias;
}
void nombre_función()
{
    Declarar variables;
    Bloque de sentencias de la función;
}
```

Ejemplo de uso

Leer 2 números y mediante funciones sin parámetros, obtener la suma y la resta de ellos.

En este ejemplo se emplearan las dos formas de uso al declarar las funciones, observe en el código y note que la función sumar(), solo se declara al inicio el prototipo y al final la función de manera completa y por otro lado la función resta() se declara de manera completa antes del main() y el llamado de las funciones se realiza en el orden de aparición. La figura 6.x, muestra parte del código y la ejecución del programa en el entorno.

```
# include <conio.h>
# include <stdio.h>
void sumar(); //prototipo de función
void restar() //Función sin parámetros
{
    int a,b,r;
    printf("ingresa el primer número: ");
    scanf("%d",&a);
    printf("ingresa el segundo número: ");
    scanf("%d",&b);
    r=a-b;
```



```
printf("La resta de %d - %d = %d", a, b, r);
}

main()
{
    int opc;
    do
    {
        printf("\nMenu de opciones\n");
        printf("1. Sumar \n");
        printf("2. Restar\n");
        printf("3. Fin\n");
        printf("Ingresa la opción deseada: ");
        scanf("%d",&opc);
        switch (opc)
        {
            case 1: sumar(); //llamado de la función suma
                    break;
            case 2: restar(); //Llamado de la función resta
                    break;
            case 3: printf("Fin del programa\n");
            default: printf("Esa opción no existe\n");
        }
    }
    while (opc!=3);
}

void sumar()
{
    int a,b,s;
    printf("ingresa el primer número: ");
    scanf("%d",&a);
    printf("ingresa el segundo número: ");
    scanf("%d",&b);
    s=a+b;
    printf("La suma de %d + %d = %d", a, b, s);
}
```

La figura 6.3, muestra parte del código y la ejecución de algunas opciones del menú realizado.



```
1 #include <conio.h>
2 #include <stdio.h>
3 void sumar(); //prototipo de función
4 void restar() //función sin parametros
5 {
6     int a,b,r;
7     printf("ingresa el primer número: ");
8     scanf("%d",&a);
9     printf("ingresa el segundo número: ");
10    scanf("%d",&b);
11    r=a+b;
12    printf("La resta de %d - %d = %d", a, b, r);
13 }
14 main()
15 {
16     int opc;
17     do
18     {
19         printf("\nMenu de opciones\n");
20         printf("1. Sumar\n");
21         printf("2. Restar\n");
22         printf("3. Fin\n");
23         printf("Ingresa la opción deseada: ");
24         scanf("%d",&opc);
25         switch (opc)
26         {
27             case 1: sumar(); //Llamado de la función suma
28                 break;
29             case 2: restar(); //Llamado de la función resta
30                 break;
31             case 3: printf("Fin del programa\n");
32                 default: printf("Esa opción no existe\n");
33         }
34     } while (opc!=3);
35 }
```

```
Menu de opciones
1. Sumar
2. Restar
3. Fin
Ingresa la opción deseada: 1
ingresa el primer n-mero: 6
ingresa el segundo n-mero: 7
La suma de 6 + 7 = 13
Menu de opciones
1. Sumar
2. Restar
3. Fin
Ingresa la opción deseada: 2
ingresa el primer n-mero: 45
ingresa el segundo n-mero: 34
La resta de 45 - 34 = 11
Menu de opciones
1. Sumar
2. Restar
3. Fin
Ingresa la opción deseada: 3
Fin del programa
Esa opción no existe
Process exited after 33.77 seconds with return value 0
Presione una tecla para continuar . . .
```

Figura 6.3 Funciones sin parámetros

Ejercicio de aprendizaje

Realiza un menú de opciones iterativo utilizando funciones sin parámetros para cada opción, con las siguientes acciones:

1. Calcular el área de un triángulo
2. Elevar un número a una potencia
3. Finalizar

Las funciones con pase de parámetros se describen a continuación de manera específica, en los subtemas 6.5 y 6.6 para su mejor comprensión.



6.5 Pase por valor

Este tipo de funciones se emplean con mucha frecuencia en la programación, porque pueden recibir uno o más valores los cuales se les conoce como parámetros, y regresan un valor de tipo entero, real o carácter. Es importante comprender que en la función solo se realizan las operaciones o cálculos matemáticos requeridos a partir del valor o valores que recibe, por lo que al regresar un valor de salida es importante definir el tipo de valor que debe retornar.

El paso de parámetros en una función por valor se realiza mediante el uso de variables temporales, mientras se ejecuta la función y tiene la limitante de solo resolver el problema tomando en consideración los valores proporcionados como se puede observar en el siguiente ejercicio:

```
#include <conio.h>
#include <stdio.h>
// función sumar
int sumar(int x, int y)
{   return (x+y); }
int multiplicar(int x, int y)
{   return(x*y); }
main()
{   int op, x, y;
    printf("Elige una opción\n");
    printf("1. Sumar dos números\n");
    printf("2. Multiplicar dos números\n");
    scanf("%d",&op);
    switch (op)
    {
        case 1: printf(" \nla suma es %d", sumar(6,8));
                break;
        case 2: printf("\n la multiplicación es %d", multiplicar(6,8));
                break;
        default: printf("\nesa opción no es válida");
    }
}
```



6.6 Pase por referencia

En el caso anterior, los parámetros se envían por valor, sin embargo, cuando se requiere modificar el valor de una variable se debe proporcionar en lugar del valor de la variable, su dirección a este tipo de parámetros se les conoce como parámetros por referencia o por variable.

```
#include <conio.h>
#include <stdio.h>
// función sumar
int sumar(int x, int y)
{
    return (x+y);
}
int multiplicar(int x, int y)
{
    return(x*y);
}
main()
{
    int op, x, y;
    printf("Elige una opción\n");
    printf("1. Sumar dos números\n");
    printf("2. Multiplicar dos números\n");
    scanf("%d",&op);
    printf("\nIngresa el primer número: ");
    scanf("%d",&x);
    printf("\ningresa el segundo número: ");
    scanf("%d",&y);
    switch (op)
    {
        case 1: printf(" \nla suma de %d +%d = %d", x,y,sumar(x,y));
                break;
        case 2: printf("\n la multiplicación de %d * %d = %d", x,y,
                multiplicar(x,y));
                break;
        default: printf("\nesa opción no es válida");
    }
}
```



Ejercicio de aprendizaje

Realiza un menú de opciones usando funciones sin parámetros y funciones con parámetros, el menú debe finalizar cuando se ingrese un 0

Menú de opciones

1. Promedio de 10 números
2. Factorial de un número
3. Calcular el área de un círculo

6.7 Elaboración de programas

El siguiente programa muestra al lector algunas alternativas de solución al problema planteado utilizando funciones con parámetros y algunas recomendaciones de uso, dichos programas se integran en un menú de opciones para que se note el uso de prototipos de función, función con parámetros y función sin parámetros. La figura 6.4 plasma una parte del código en el entorno y la ejecución de algunas opciones del menú desarrollado. El programa completo lo puede descargar de la carpeta de materiales de apoyo.

Ejercicio 1. Calcular el área de las siguientes figuras: círculo, cuadrado y un rectángulo.

Alternativa de solución

```
# include <conio.h>
```

```
# include <stdio.h>
```

```
# define p 3.1416 //valor constante de pi
```



```
float AreaCirculo (int radio); // prototipo de función
int areacuadrado (int lado); // prototipo de función
int arearectangulo (int base, int altura)
{
    int arearec;
    arearec=base*altura;
    return arearec;
}
main()
{
    int opc, a, b, c;
    float d, x, y, z;
    do
    {
        printf("\nElige una opción: \n");
        printf("1. Área de un círculo\n");
        printf("2. Área de un cuadrado\n");
        printf("3. Área de un rectángulo\n");
        printf("4. finalizar\n");
        printf("ingresa opcion: ");
        scanf("%d",&opc);
        switch(opc)
        {
            case 1: printf("Ingresa el radio: \n");
                    scanf("%d",&a);
                    d=AreaCirculo(a);
                    printf("El área del círculo es: %f",d);
                    break;
```



```
case 2: printf("Ingrese el lado: \n");
        scanf("%d",&a);
        b=areacuadrado(a);
        printf("El área del cuadrado es: %d",b);
        break;

case 3: printf("Ingrese la base: \n");
        scanf("%d",&a);
        printf("Ingrese la altura: \n");
        scanf("%d",&b);
        c=arearectangulo(a,b);
        printf("El área del círculo es: %d",c);
        break;

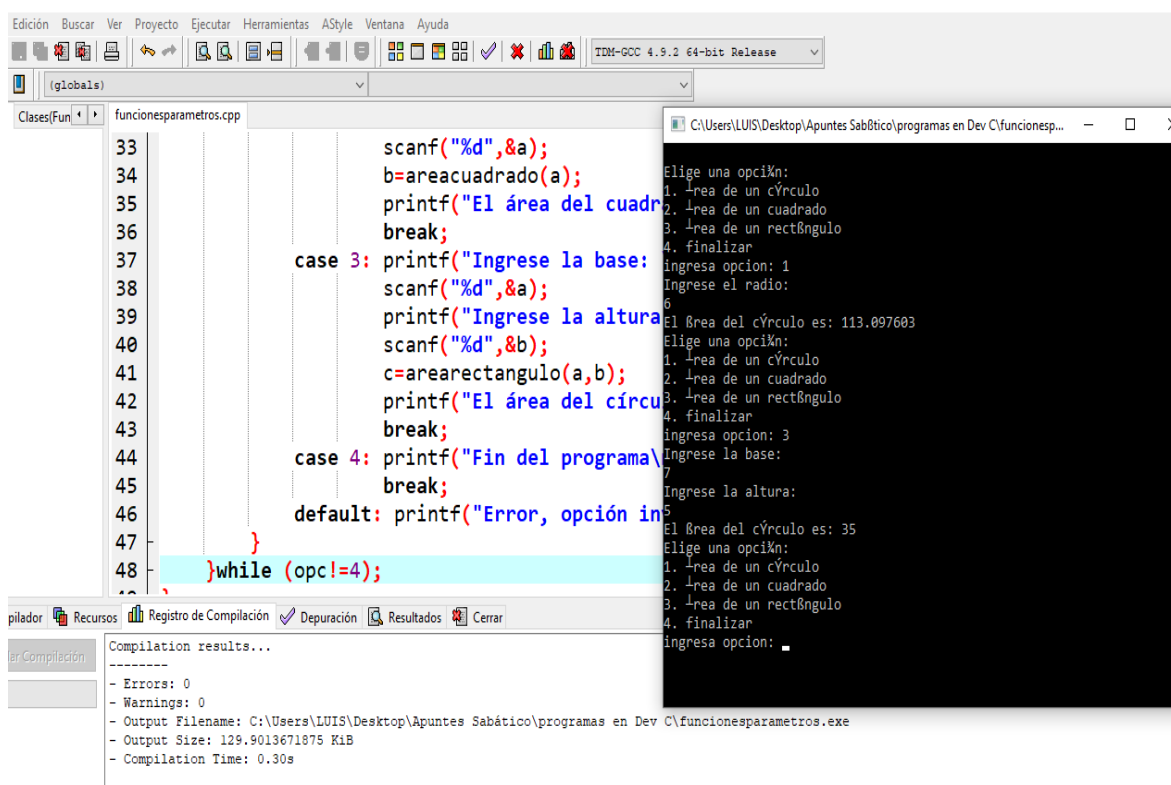
case 4: printf("Fin del programa\n");
        break;

default: printf("Error, opción invalida\n");

    }
}while (opc!=4);
}

float AreaCirculo( int radio)
{
    float areacir;
    areacir=p*radio*radio;
    return areacir;
}

int areacuadrado(int lado)
{
    int areacua;
    areacua=lado*lado;
    return areacua;
}
```



```
33 scanf("%d",&a);
34 b=areacuadrado(a);
35 printf("El área del cuadrado es: %d",b);
36 break;
37 case 3: printf("Ingrese la base: ");
38 scanf("%d",&a);
39 printf("Ingrese la altura: ");
40 scanf("%d",&b);
41 c=arearectangulo(a,b);
42 printf("El área del rectángulo es: %d",c);
43 break;
44 case 4: printf("Fin del programa\n");
45 break;
46 default: printf("Error, opción incorrecta\n");
47 }
48 }while (opc!=4);
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\LUIS\Desktop\Apuntes Sabático\programas en Dev C\funcionesparametros.exe
- Output Size: 129.9013671875 KiB
- Compilation Time: 0.30s

Elige una opción:
1. Área de un círculo
2. Área de un cuadrado
3. Área de un rectángulo
4. finalizar
Ingresa opción: 1
Ingresa el radio: 6
El área del círculo es: 113.097603
Elige una opción:
1. Área de un círculo
2. Área de un cuadrado
3. Área de un rectángulo
4. finalizar
Ingresa opción: 3
Ingresa la base: 7
Ingresa la altura: 5
El área del rectángulo es: 35
Elige una opción:
1. Área de un círculo
2. Área de un cuadrado
3. Área de un rectángulo
4. finalizar
Ingresa opción: 4

Figura 6.4. Funciones con parámetros

6.8. Ejercicios

Completar los siguientes programas, capturar en el entorno de Dev C++, compilar y ejecutar y anotar sus comentarios con relación a ¿qué hace el programa y cómo lo hace?



Ejercicio 1

```
# include <conio.h>
# _____ <stdio.h>
float funcion2(float m, _____);
_____ funcion3(_____);
{int i, z=1;
for(i=1;i<=x;i++)
    z=z*i;
    printf("Resultado : %d\n", z);
}
float funcion1(float a, float b)
{
    float z;
    z=funcion2(a,b)/4;
    _____(z);
}
main()
{
    float x, y=3; int z=2;
    for (x=2;x<3;x=x+2)
    {
        printf("Resultado: %f\n",
funcion1(x,y));
        funcion3(z);
    }
}
float funcion2(float m, _____)
{
    if (m>n) return(n+n);
    else return(m*n);
}
```

Ejercicio 2

```
# include <conio.h>
# include <stdio.h>
int valor(int lista[], _____)
{
    int i, suma=0;
    for(i=0; i<x;i++)
        suma+=lista[i];
    return(suma);
}
int suma(int x, int y=3)
{
    _____ x+y;
}
main ()
{
    int v[6]={8,11,4,15,10},s;
    s=valor(v,suma(3));
    _____("El resultado es: \n%d",s);
}
```

Ejercicio 3.

```
# include <conio.h>
# include <stdio.h>
_____ fac(int n)
{
    float f;
    if (n==0)
        f=1;
    else
        f=n*fac(n-1);
    _____ f;
}
main()
{
    float valor;
    int n;
    _____f("Damen un número: ");
    scanf("%d",&n);
    valor=fac(n);
    printf("El factorial de %d
es: %7.2f\n",n, _____);
}
```



Ejercicio 4. Sumar 2 matrices cuyo tamaño máximo sea de 5x5 elementos y almacenar el resultado en una tercera matriz utilizando funciones con paso de parámetros

Ejercicio 5. Realizar un programa utilizando funciones que realice lo siguiente:

- Imprimir el mayor de 3 números
- Calcular el promedio de 5 números almacenados en un arreglo
- Ordenar el arreglo utilizando un método de ordenamiento

6.9 Aprendizaje basado en problemas

1. Una tienda de abarrotes tiene 2 sucursales y desea conocer el inventario de productos que tiene en total de ambas sucursales por mes para lo cual solicita a cada una de ellas le envíen un archivo con la siguiente estructura:

Producto/mes	Enero	Febrero	Marzo	Abril
Escoba				
Trapeador				
Cubeta				
Jerga				
Sacudidor				

Realice el programa que sume ambas matrices e imprima el resultado obtenido utilizando funciones

2. Javier tiene almacenado en dos arreglos el total de productos que ofrece al público, pero desea concentrar el total de cada casilla de ambos arreglos en un tercero. Realiza el programa que imprima los arreglos originales y la suma de ambos utilizando funciones



3. Una empresa elabora cajas en forma de cilindro de diferentes tamaños y para no generar mucha merma(desperdicio), del cartón corrugado que utiliza debe proporcionar el área del círculo, el área de un aro y el volumen del cilindro. Elabora un programa que calcule estos valores utilizando funciones con parámetros.
4. Para determinar la temperatura en diferentes países es necesario realizar un programa que convierta:
 - a. Grados centígrados a grados Fahrenheit
 - b. Grados Fahrenheit a grados centígrados
 - c. Grados centígrados a grados kelvin
 - d. Grados kelvin a grados centígrados

Realizar dicho programa usando un menú de opciones y/o funciones con pase de parámetros

5. Realiza la función que calcule la hipotenusa de un triángulo rectángulo, que le permita a un arquitecto determinar la longitud de la pendiente de un registro hidráulico



Referencias

AppMaster. (2020). Obtenido de <https://appmaster.io/es/glossary/programacion-modular>

Corona Nakamura, M. A., & Ancona Valdez, M. (2011). *Diseño de algoritmos y su codificación en lenguaje C*. México: Mc. Graw Hill.

Corona, F. (2020). *Macros Excel*. México: Alfaomega.

Deitel, P. (2012). *Como programar en Java, 9a. edición*. México: Pearson.

Jimenez. (2018). *Desarrollo WEB*. Obtenido de <https://desarrolloweb.com/faq/programacion-estructurada-modular>

Mauricio, F. (2013). *Excel 2013*. México: Alfaomega.

Mauricio, F. (2013). *Macros. arrancar con todo el poder*. México: Alfaomega.

MEDIAactive. (2012). *Aprender fórmulas y funciones*. Barcelona, España: Alfaomega .

Microsoft. (2022). Obtenido de <https://support.microsoft.com/es-es/office/hallar-y-hallar-b-funciones-hallar-y-hallar-b-9ab04538-0e55-4719-a72e-b6f54513b495>

Microsoft. (2022). *Microsoft*. Obtenido de <http://www.microsoft.com>

Miranda Chávez, E. M., & Fuenlabrada Velázquez, S. (2015). *Manejo de técnicas de programación*. México: Pearson.

Oliver, C. M. (2021). *Contabilidad Básica*. Bogotá: Pirámide.

s/a. (2020). *Universidad Latina*. Obtenido de <https://www.ulatina.ac.cr/articulos/que-son-las-tic-y-para-que-sirven>

SourceForge. (2024). *PseInt*. Obtenido de <https://pseint.sourceforge.net/>