Universidad Nacional Autónoma de México
Escuela Nacional Preparatoria
SECRETARÍA ACADÉMICA

Informática aplicada a la ciencia y la industria

SEXTO AÑO

Alejandro Villagómez Díaz
Arturo Hernández Sánchez
Israel Fernando Balderas Morales
Ivette Cruz Felipe
Josafath Benítez Rumbo
Milagros Pacheco Castañeda
Mónica Elizabeth González Chavarría
Norma Angélica Romero Badillo
Norma Gloria Covarrubias Rocha
Rebeca Guillermina Villegas Salas
Rebeca Rodríguez Ramírez
Rocío Velasco Bazán

Colegio de Informática
Clave: 1719
Plan: 96
Actualización curricular 2016

COLECCIÓN
Guías ~ Cuadernos
de trabajo académico

ESCUELA NACIONAL PREPARATORIA
COLEGIO DE INFORMÁTICA

ÁREA I. FÍSICO MATEMÁTICAS
Y DE LAS INGENIERÍAS

Grado: Sexto Clave: 1719 Plan: 1996

ACTUALIZACIÓN CURRICULAR 2018

INFORMÁTICA APLICADA A LA CIENCIA Y LA INDUSTRIA

Guía cuaderno de trabajo académico

Coordinación y revisión
Marcela Cuapio Campos

Autores:
Alejandro Villagómez Díaz
Arturo Hernández Sánchez
Israel Fernando Balderas Morales
Ivette Cruz Felipe
Josafath Benítez Rumbo
Milagros Pacheco Castañeda
Mónica Elizabeth González Chavarría
Norma Angélica Romero Badillo
Norma Gloria Covarrubias Rocha
Rebeca Guillermina Villegas Salas
Rebeca Rodríguez Ramírez
Rocío Velasco Bazán

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Universidad Nacional Autónoma de México
Escuela Nacional Preparatoria
Directora General: Biól. María Dolores Valle Martínez
Secretaria Académica: Dra. Virginia Hernández Ricárdez
Jefa del Departamento de Producción Editorial: Lic. María Elena Jurado Alonso

Primera edición, febrero 2019

Imagen de Portada:	Citlali Galván González Alejandro Villagómez Díaz
Diseño de portada:	Edgar Rafael Franco Rodríguez
Diseño editorial:	Citlali Galván González Marcela Cuapio Campos
Diseño gráfico:	Citlali Galván González
Revisón de estilo:	Rebeca Guillermina Villegas Salas Gloria Patricia Patlani Huerta

Impreso y hecho en México

PRESENTACIÓN

La Escuela Nacional Preparatoria, institución educativa con más de 150 años de experiencia formando jóvenes en el nivel medio superior, culmina en este ciclo escolar 2018-2019, la colección de **Guías de Estudio** correspondientes a los programas actualizados de nuestro Plan de Estudios vigente.

Después de varios años de trabajo, reflexión y discusión, se lograron dar dos grandes pasos: la actualización e implementación de los programas de estudios de bachillerato y la publicación de la nueva colección de **Guías de Estudio**.

Ciertamente, nuestra Escuela Nacional Preparatoria es una institución que no se detiene, que avanza con paso firme y constante hacia su excelencia académica, así como preocupada y ocupada por la formación integral, crítica y con valores de nuestros estudiantes, lo que siempre ha caracterizado a nuestra Universidad Nacional.

Aún nos falta más por hacer, por mejorarnos cada día, para que tanto nuestros jóvenes estudiantes como nuestros profesores seamos capaces de responder a esta sociedad en constante cambio y a la Universidad Nacional Autónoma de México, la Universidad de la Nación.

"POR MI RAZA HABLARÁ EL ESPÍRITU"
BIÓL. MARÍA DOLORES VALLE MARTÍNEZ
DIRECTORA GENERAL
ESCUELA NACIONAL PREPARATORIA

ÍNDICE

PÁG.

INTRODUCCIÓN

Este cuaderno guía de trabajo académico fue desarrollado con base en el programa de estudios de la asignatura de Informática Aplicada a la Ciencia y la Industria de la Escuela Nacional Preparatoria (ENP), y responde a las necesidades de promover el aprendizaje autónomo de los estudiantes, fortalecer con diversas actividades el proceso de enseñanza aprendizaje y servir como material guía tanto para trabajar en las asesorías permanentes de la asignatura, como para la preparación del examen extraordinario.

El material se sustenta en los procesos de aprendizaje acordes al enfoque educativo del plan de estudios de la ENP considerando sus cinco ejes transversales: lectura y escritura de textos para aprender a pensar, desarrollo de habilidades para la investigación y solución de problemas característicos del entorno actual, comprensión de textos en lenguas extranjeras, aprendizaje y construcción del conocimiento con tecnologías de la información y la comunicación y la formación en valores en congruencia con la coyuntura de los desafíos y transformaciones del mundo actual.

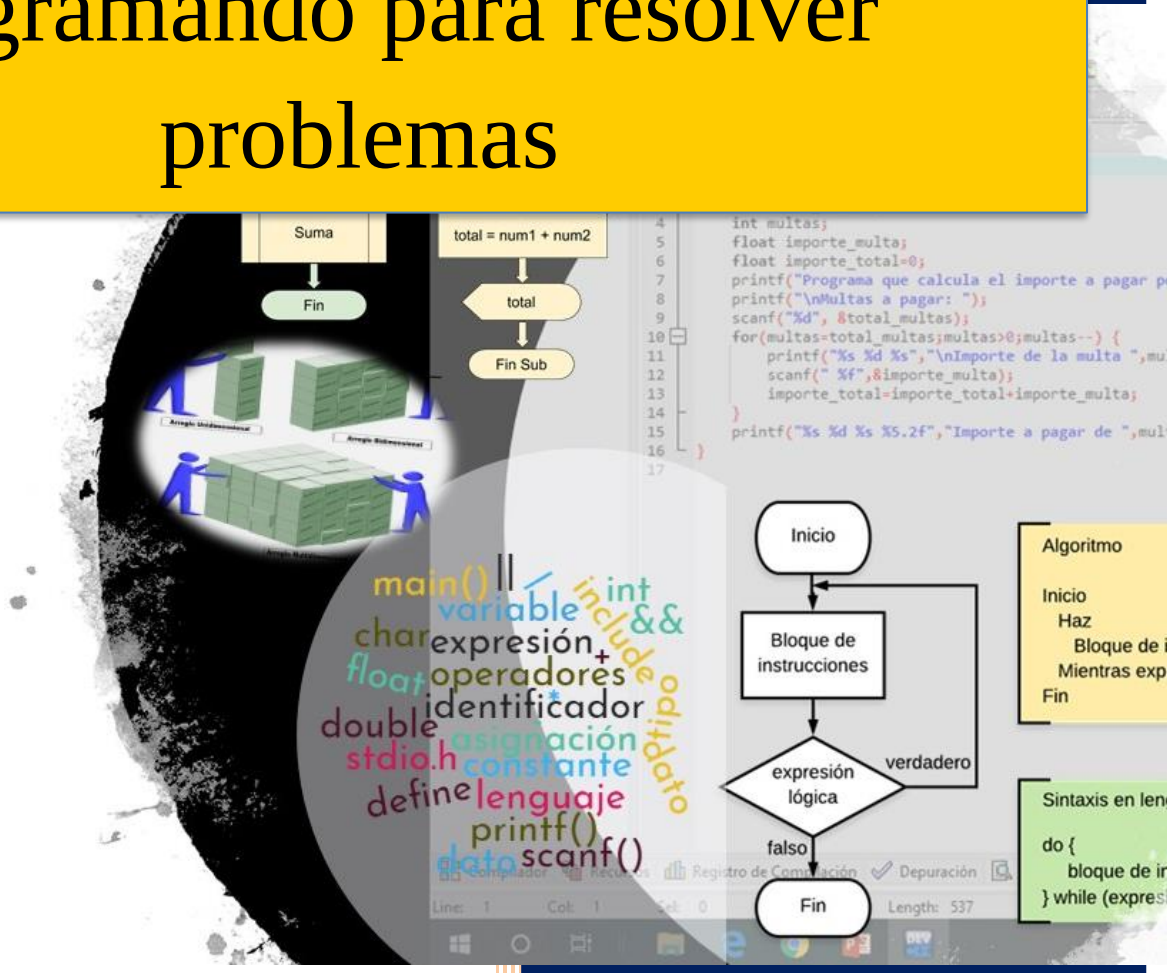
El cuaderno aborda los tópicos necesarios para que el estudiante desarrolle habilidades de investigación y manejo avanzado de herramientas digitales para la automatización y control de procesos relacionados con la ciencia y la industria, la programación de sistemas de cómputo y la adquisición, análisis e interpretación de datos mediante el uso responsable y ético de la información.

Estos temas se desarrollan a lo largo de tres unidades. La primera introduce al estudiante a la programación de computadoras. En la segunda se estudian conceptos básicos de estadística para calcular, analizar, interpretar datos y presentar resultados empleando una hoja de cálculo. Finalmente, la tercera unidad está dedicada a explicar la construcción y programación de prototipos automatizados para resolver problemas de la vida cotidiana relacionados con la ciencia y la industria.

El cuaderno guía de trabajo académico está estructurado con los siguientes apartados: conjunto de contenidos y actividades de aprendizaje para adquirir los conocimientos conceptuales, procedimentales y actitudinales, una sección con las respuestas a todos los ejercicios planteados, una auto evaluación con preguntas modelo para la preparación de exámenes extraordinarios, un glosario que incluye los términos fundamentales de la disciplina y por último, una lista de referencias bibliográficas que ayudarán al alumno a profundizar, si así lo desea, en el estudio de los temas que se presenta en esta obra.

Unidad 1

Programando para resolver problemas



UNIDAD 1
PROGRAMANDO PARA RESOLVER PROBLEMAS

Objetivo

El alumno creará programas de cómputo relacionados con la ciencia y la industria, diseñando algoritmos codificados en un lenguaje de programación para la solución de problemas de su vida cotidiana, desarrollando habilidades de abstracción, concentración, perseverancia y creatividad.

Introducción

Tanto en la ciencia como en la industria se recurre a la informática para resolver aquellos problemas que son catalogados como computables. Esto se realiza a través de la metodología de solución de problemas, que tal como se vió en la unidad 3 de la asignatura de Informática, consta de cuatro fases; cuyo punto de partida surge en el planteamiento y análisis del problema, y culmina con la construcción de un programa o sistema de cómputo que resuelve dicho problema.

Es importante mencionar que para elaborar un programa o sistema de cómputo es necesario hacerlo a través de la programación, entendida ésta como la acción en la que la computadora realiza tareas específicas que son construidas a través de una serie de comandos o instrucciones insertadas en un cierto código que es convertido a lenguaje máquina (código binario) para su propia ejecución.

Las primeras computadoras se programaban exclusivamente mediante el lenguaje máquina, es decir, en código binario. Por lo anterior, se requería que las personas que las programaban tuvieran un conocimiento profundo de sus componentes, así como de su funcionamiento interno.

Posteriormente, se creó el lenguaje ensamblador que utiliza palabras muy cortas (mnemónicos), también llamadas comandos o instrucciones, para hacer referencia a operaciones internas de las computadoras sin tener que utilizar el código binario. Es aquí cuando surge el primer nivel de abstracción en la programación de computadoras, donde con una palabra se podía programar una serie de operaciones. Sin embargo, era necesario todavía saber muchos aspectos internos de la computadora, por ello, se llamó lenguaje de bajo nivel.

Más adelante, surgieron otros lenguajes cuya programación hacía más fácil hacer programas sin necesidad de conocer qué componentes estaban involucrados en la ejecución de las instrucciones, ni cómo manipulaba los datos la computadora internamente. De modo tal que el programador solamente se dedicaba a entender cómo utilizar cada comando. Es así como surgen los lenguajes de alto nivel.

Aquí vamos a hacer un paréntesis para explicar qué son los niveles de abstracción, lo haremos a través del siguiente ejemplo: Si una persona desea aprender a conducir un automóvil ¿es necesario que sepa mecánica? o ¿cómo es que internamente los componentes del auto interactúan uno con el otro para lograr que éste avance? ¿Necesitará construir cada una de las partes del automóvil y ensamblarlas para poder conducirlo? Lo cierto es que no. Lo único que tiene que hacer es ver al automóvil como si fuera una caja negra, en la que no le interesa qué es lo que sucede internamente en ella, de hecho, los componentes mecánicos y eléctricos del automóvil no se encuentran a la vista, sino que están encapsulados dentro de un espacio al que se accede a través del cofre. Así, el aprendiz de conductor deberá interactuar solamente con el volante, los pedales y la palanca de velocidades del automóvil y de esta forma, está abstrayendo todo el funcionamiento del coche.

De manera similar, se ha logrado un nivel de abstracción dentro de la programación, cuyo resultado ha sido una mayor facilidad y accesibilidad para que las personas puedan programar sus computadoras sin necesidad de conocer a detalle la forma en que internamente funcionan los componentes de éstas, ni que tengan que aprenderse el código máquina como tal. Para ello, fue necesario que se crearan varios programas que permitieran, por un lado, que el ser humano pudiera realizar un programa en un código más familiar para él y por otro lado, lo pudieran convertir a un código binario que fuera interpretado y ejecutado por la computadora. A estos programas se les denomina compiladores o intérpretes. La diferencia entre estos dos programas es que el primero convierte todo el archivo del código fuente a código binario, mientras que el segundo lo hace línea por línea. En esta guía haremos referencia únicamente a los compiladores pero están implícitos los intérpretes.

Los compiladores están basados en reglas gramaticales, tanto sintácticas como semánticas, a las cuales se les denomina de manera conjunta como lenguajes de programación. Cierto es que existe una gran variedad de lenguajes de programación con los que se puede programar la computadora, así como, también suele existir una serie de compiladores para un solo lenguaje de programación.

Caso de estudio: la programación en la industria automotriz

Actualmente, los sensores son piezas claves en la industria automotriz y se utilizan para gestionar el funcionamiento del motor e incrementar la seguridad y confort del automóvil. Por ejemplo, un sensor sirve para alertar al pasajero en caso de que no se haya colocado el cinturón de seguridad, o si la gasolina está por terminarse, o bien, si le hace falta aceite o agua al motor, entre otras funcionalidades. Estos sensores miden magnitudes físicas, que posteriormente son convertidas en señales digitales que la computadora puede manipular, y dependiendo del valor de dichas magnitudes, programar alguna acción específica tal como encender una luz de alerta o hacer sonar una alarma. Estos procesos son factibles de realizarse de manera automática gracias a la programación de diversos circuitos electrónicos.

En este contexto, durante el desarrollo de esta unidad, te convertirás en un programador de la industria automotriz. Tu principal función será diseñar programas para simular el manejo de algunos sensores básicos y controlar distintas funcionalidades de un automóvil, así como desarrollar programas relativos a trámites vehiculares.

El lenguaje de programación C

Imagen 1. Ícono Dev C++. (Mendoza, s/f).

Para la codificación de los ejemplos y ejercicios propuestos en esta guía se utilizará el lenguaje de programación C. La edición, compilación, depuración y ejecución de los programas se hará en el entorno de desarrollo integrado Dev-C++, un programa con licencia GNU GPLv2, que se puede descargar de manera gratuita en la siguiente dirección electrónica: https://sourceforge.net/projects/orwelldevcpp/.

C es un lenguaje de programación diseñado por Dennis Ritchie. Se creó originalmente para el sistema operativo UNIX, sin embargo, no está ligado a ningún sistema operativo o máquina concretos. Se caracteriza por ser un lenguaje de empleo general y por la eficiencia en el código que genera.

Se consideró el uso del lenguaje de programación C en este cuaderno de trabajo porque, aunque C es un lenguaje antiguo, actualmente sigue siendo uno de los más usados para la programación de sistemas operativos y de aplicaciones; tiene gran popularidad en el desarrollo loT (internet de las cosas) para codificar la interconexión digital de los objetos cotidianos con internet, además de que muchos lenguajes de programación se basan en la sintaxis de C y heredan gran parte de sus características. Sin embargo, es importante aclarar que todas las actividades de aprendizaje propuestas son susceptibles de ser adaptadas a cualquier otro lenguaje de programación, si así lo considera necesario el profesor.

Estructura general de un programa en C

Antes de conocer los conceptos básicos para programar en lenguaje C, es importante saber la estructura general de un programa, con el fin de colocar cada instrucción en el lugar correcto para que se ejecute con éxito lo que queremos hacer, aunado a no tener errores de sintaxis ni de lógica en la programación.

Todos los programas en C están compuestos de una o más funciones[1]. La función principal *main()* es la que ejecuta el sistema operativo cuando se invoca un programa en C. *main()* identifica el punto de partida del programa, controla todas las acciones que realizará el programa y es la encargada de hacer el llamado al resto de las funciones.

La estructura general de un programa en C se muestra en la Ilustración 1.1

```
Sección de inclusión de archivos (bibliotecas)
        # include <nombre_archivo>
        # include "nombre_archivo"

Sección de definición de constantes
        #define <identificador_constante> <secuencia de caracteres>

Sección de declaración de variables globales
        tipoDato <identificador_variable>;

Declaración de funciones
        tipoDato identificador_función (lista de parámetros);

Función main (Programa principal)
main() {
        Declaración de variables locales
        Bloque de instrucciones
        return 0;
}

Definición de funciones
        tipoDato identificador función (lista de parámetros) {
                Declaración de variables locales
                Bloque de instrucciones
                return resultado;
        }
```

Ilustración 1.1. Estructura general de un programa en lenguaje C

[1] "Una función se define como una secuencia finita de instrucciones que realizan una tarea específica y pueden retornar un valor" (Programación en C++, 2018).

Bibliotecas

En un programa se pueden utilizar bibliotecas, que son archivos que contienen funciones hechas previamente para realizar alguna tarea específica. Su uso permite la reutilización de código en cualquier programa sin necesidad de volver a codificar la función que se desea ejecutar.

C distingue dos clases de bibliotecas:

- Bibliotecas estándares de C. Generalmente incluyen funciones utilizadas habitualmente por cualquier programador, por ejemplo, la biblioteca *stdio.h* que contiene las funciones para hacer operaciones de entrada y salida de datos. Para incluirlas en un programa se debe agregar en la sección de inclusión de archivos la sentencia: *#include <nombre de la biblioteca>*.

- Bibliotecas creadas por el programador. El lenguaje también permite la creación de bibliotecas que contengan funciones desarrolladas por el propio programador, en cuyo caso deberán agregarse al programa con la sentencia: *#include "nombre de la biblioteca"*.

Programa 1. Mostrar un mensaje en el monitor

Vamos a realizar nuestro primer programa en C con el objeto de que te familiarices con el proceso de edición, compilación, depuración y ejecución de un programa mediante el compilador Dev C++.

Para realizar este ejercicio es necesario que descargues e instales en la computadora el entorno de desarrollo integrado Dev-C++ de la dirección electrónica https://sourceforge.net/projects/orwelldevcpp/. En esta página encontrarás las indicaciones para realizar dicho proceso.

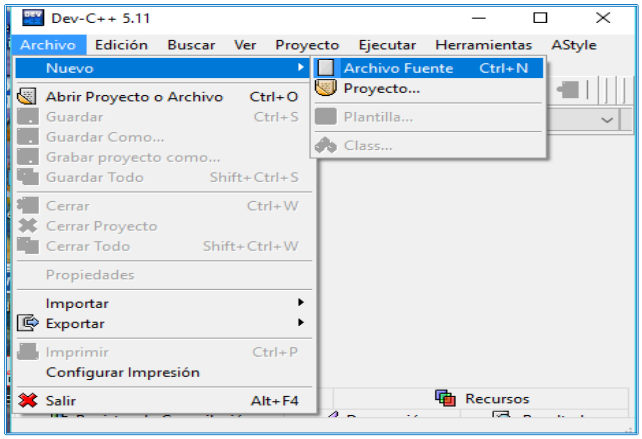


Ilustración 1.2. Creación de un nuevo archivo fuente

La primera actividad que realizaremos será abrir el entorno de desarrollo integrado Dev C++.

Verás una pantalla similar a la Ilustración 1.2, Selecciona la opción *Archivo - Nuevo - Archivo Fuente*, o bien, haz clic en el ícono *Nuevo* y selecciona la opción *Archivo fuente*.

La siguiente actividad consiste en la captura del programa fuente. Un programa fuente es el conjunto de instrucciones escritas bajo la sintaxis de un lenguaje de programación, que realizan una tarea específica.

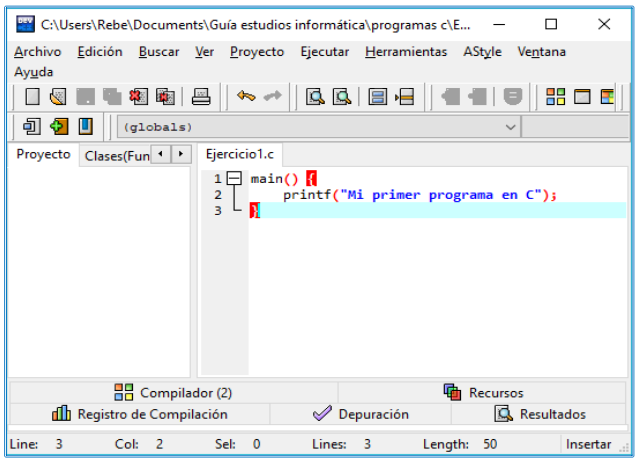


Ilustración 1.3. Captura del programa fuente

Captura[2] el siguiente programa (Ilustración 1.3):

```
main() {
  printf("Mi primer programa de C");
}
```

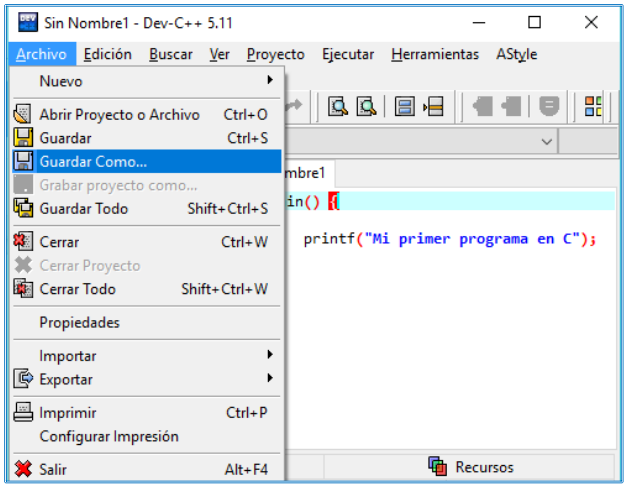


Ilustración 1.4. Guardar un programa fuente

Guarda tu programa con la opción *Archivo-Guardar Como* (Ilustración 1.4).

El compilador Dev C++ es capaz de manejar distintos lenguajes de programación C, así que es importante que cuando te pida el nombre y tipo de archivo, le indiques al compilador el lenguaje que estás empleando.

[2] Debes ser precavido en la captura del código, ya que C es sensible al uso de mayúsculas y minúsculas. Es decir, no es igual capturar, main que Main.

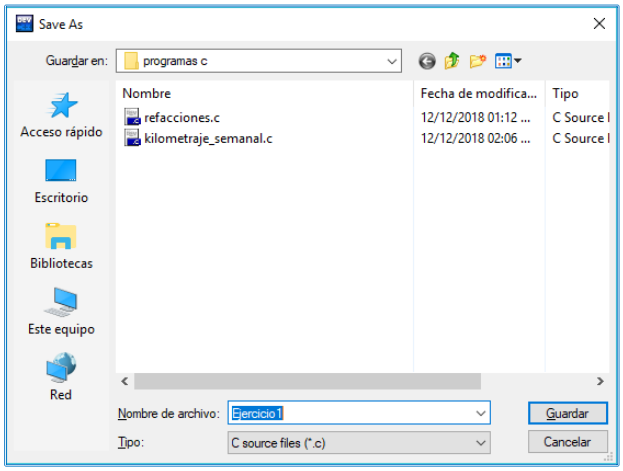


Ilustración 1.5. Selección del tipo de archivo

Por lo anterior, en la ventana de la Ilustración 1.5, selecciona la opción de tipo de archivo *C source files (*.c)*. Esta acción le indicará al compilador que estás programando en lenguaje C estándar.

El programa se almacenará en la computadora con la extensión .c, la cual corresponde a los tipos de archivo código fuente del lenguaje C.

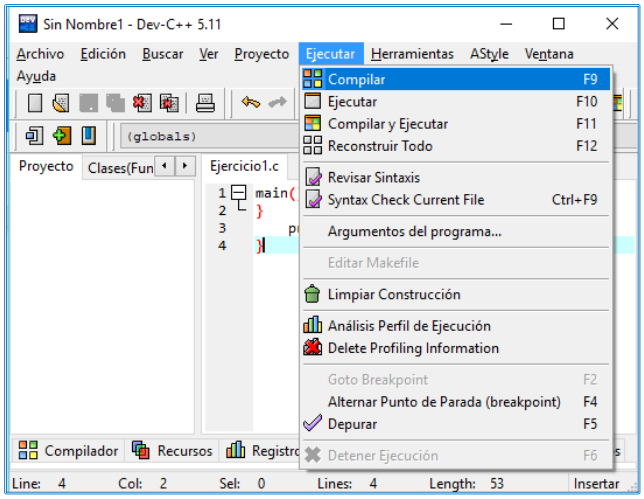


Ilustración 1.6. Proceso de compilación del programa

Después de capturar y guardar tu programa fuente, la siguiente acción será compilarlo.

Para ello, selecciona la opción *Ejecutar-Compilar* (F9) como se muestra en la Ilustración 1.6.

El proceso de compilación se lleva a cabo en varias fases que normalmente son automatizadas y ocultas para el usuario por los entornos de desarrollo. La Figura 1 muestra las fases del proceso de compilación.

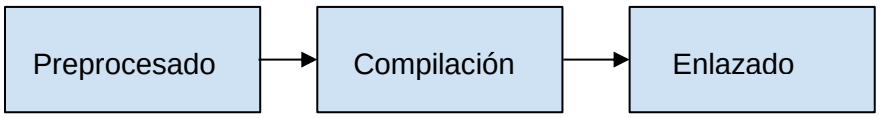


Figura 1.1 Fases del proceso de compilación

- Preprocesado. Consiste en la modificación del código mediante una serie de líneas agregadas de código que provienen de otros archivos (de directivas como: *#include* y *#define*).

- Compilación. Se revisa la sintaxis de las instrucciones y si no hay errores, se genera el código objeto del programa.

- Enlazado. Se enlazan el código objeto con el código que existe en las bibliotecas y se crea el programa ejecutable.

Si durante el proceso de compilación se detecta algún error, éste se mostrará en la parte inferior de la pantalla (Ilustración 1.7).

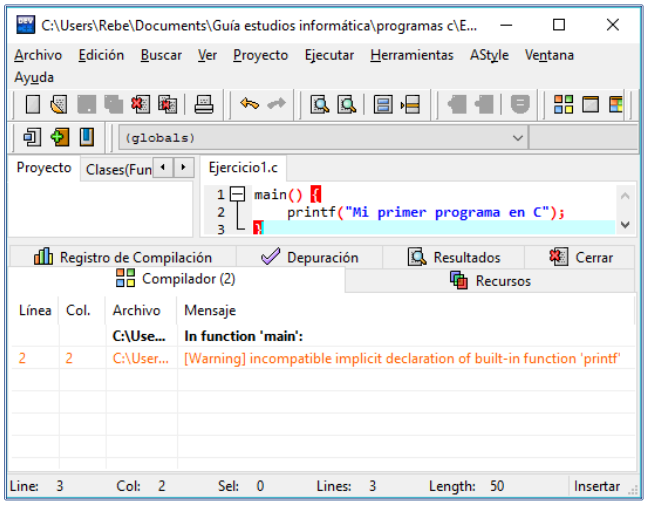


Ilustración 1.7. Proceso de compilación con errores

En este caso, el compilador encontró un error en nuestro programa y el mensaje que envía es que desconoce la función *printf()*.

Cuando el programa tiene errores, éstos se deben corregir. A este proceso se le conoce como depuración del programa.

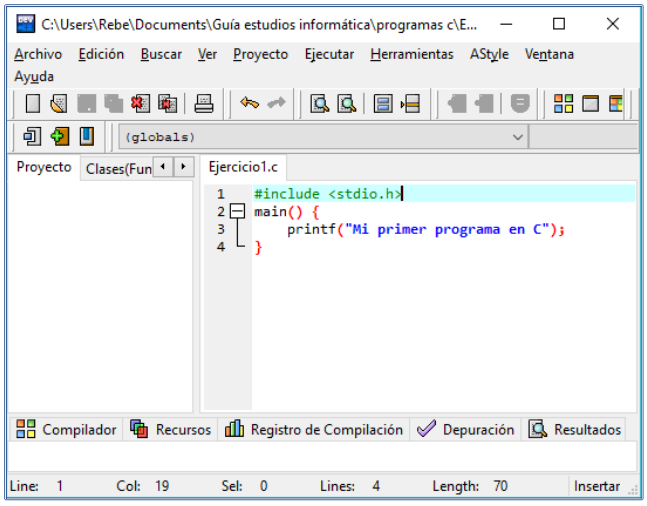


Ilustración 1.8. Depuración del programa fuente

En nuestro ejemplo, el error que cometimos es que estamos empleando la función *printf()*, pero ésta no está definida dentro del programa. Esta función está contenida en una biblioteca estándar de C llamada *stdio.h*, por lo que debemos agregar dicha biblioteca a nuestro programa.

¿Recuerdas en qué parte del programa se incluyen las bibliotecas?

Exacto, en la zona de inclusión de archivos. Procedamos a corregir el programa agregando la línea de código: *#include <stdio.h>* como se muestra en la Ilustración 1.8.

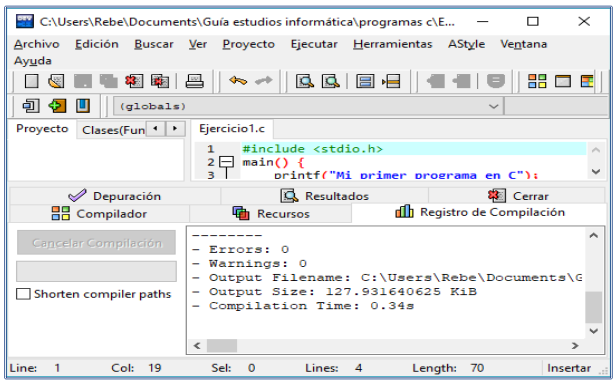


Ilustración 1.9. Proceso de compilación sin errores

Cuando se hace un cambio en el código fuente, es necesario volver a compilar el programa. ¿Recuerdas con qué tecla de funciones se compila el programa?
Sí, con F9. Observa el mensaje que envía el compilador cuando el programa ya no tiene errores (Ilustración 9).

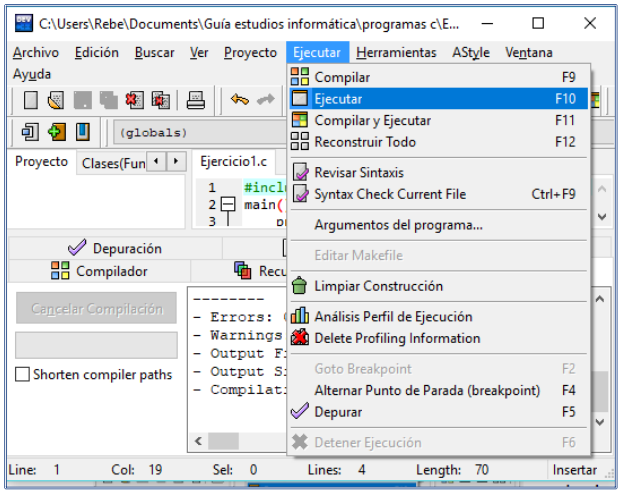


Ilustración 1.10. Ejecución del programa

Una vez que el programa es compilado sin errores, el compilador genera un archivo con extensión *.exe*, es decir, genera el programa ejecutable. Este archivo está escrito en código binario, por lo que para su ejecución ya no será necesario el uso del compilador.

Para ejecutar el programa dentro del ambiente de desarrollo de Dev C++, selecciona la opción *Ejecutar-Ejecutar* (F10) como se observa en la Ilustración 1.10.

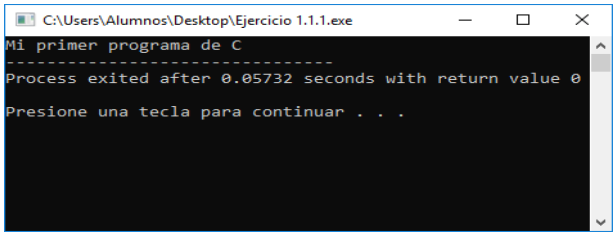


Ilustración 1.11. Ejecución del programa

La Ilustración 1.11 muestra el resultado de la ejecución de tu programa. En este caso, el objetivo del programa era mostrar el mensaje "Mi primer programa de C" en pantalla.

Ejercicio 1.1 En la asignatura de Informática, que cursaste en cuarto grado, revisaste la metodología de solución de problemas computables, algoritmos y diagramas de flujo. Repasa dichos temas y en equipos de 3 personas, responde las siguientes preguntas:

a) Explica cada uno de los pasos de la metodología de solución de problemas computables.
b) Describe el proceso para capturar, compilar, depurar y ejecutar un programa en lenguaje C utilizando el entorno de desarrollo integrado Dev-C++.

Contenidos conceptuales	**Contenidos procedimentales**
1.1 Los datos: tipos de datos, variables locales y globales, expresiones y operadores.  	**Contenidos procedimentales** 1.5 Diseño de programas aplicando la metodología de solución de problemas con ejemplos de automatización en la ciencia y la industria. **Contenidos actitudinales** 1.6 Disposición a la colaboración, cooperación y al trabajo en equipos interdisciplinarios. 1.7 Valoración de la creatividad, empatía, perseverancia, disciplina, innovación e inventiva. 1.8 Capacidad crítica y autocrítica en la toma de decisiones y solución de problemas, mediación de conflictos.

Para resolver un problema mediante un programa es indispensable identificar los datos necesarios para poder solucionarlo, así como las operaciones que se realizarán sobre los mismos para obtener los resultados deseados.

Por ejemplo, si deseas conocer la distancia que recorre un auto que se desplaza en línea recta a una velocidad de 20 km/h durante 45 minutos, ¿cuáles son los datos que intervienen en la solución del problema? ¿qué operaciones se deben realizar sobre los mismos para obtener la distancia? En este caso, los datos y operaciones identificadas serían:

Datos de entrada	Proceso	Datos de salida
velocidad tiempo	distancia=velocidad x tiempo	distancia

En programación, todos los datos deben ser identificados y organizados de tal manera que la computadora pueda almacenarlos en la memoria y realizar con ellos las operaciones matemáticas y lógicas necesarias.

Por lo anterior, revisaremos los conceptos de constantes, variables, tipos de datos, operadores y expresiones, elementos de un programa que nos ayudarán a organizar los datos en la computadora para que puedan ser procesados de forma eficiente.

Definición de dato, variable y constante

En el ámbito de la informática, un **dato** es una cifra, letra, símbolo, palabra, frase u objeto que la computadora almacena en un formato determinado para poder procesarlo.

Una **variable** es un espacio de memoria donde se puede almacenar un dato cuyo valor puede cambiar mientras se ejecuta un programa.

Una **constante** es un dato que no puede cambiar su valor durante la ejecución de un programa.

Declaración de datos en C

Para poder procesar datos en el lenguaje de programación C es necesario que sean declarados antes de su uso. La declaración de un dato origina que el compilador reserve un espacio de memoria para almacenarlo y manipularlo. Para declarar un dato son necesarios dos elementos:

- **Identificador.** Es el nombre que se le asigna a cada espacio de memoria utilizado en un programa. Se forma de la combinación de letras y números. El identificador debe cumplir con los siguientes criterios:
    a. Iniciar con una letra.
    b. Pueden contener caracteres alfabéticos y dígitos.
    c. Está prohibido utilizar caracteres especiales (%, #, &, etc.) con la excepción del guión bajo (_).
    d. No se pueden utilizar palabras reservadas[3] del compilador.
    e. Se distingue entre el uso de letras mayúsculas y minúsculas.
    f. No puede llevar espacios en blanco.
    g. Normalmente las variables y funciones se escriben con minúsculas y las constantes con mayúsculas.

- **Tipo de dato.** Determina el dominio (posibles valores que puede tener el dato) y las operaciones que se pueden realizar sobre el dato.

Tipos de datos primitivos en C

Cuando el dominio y las operaciones se encuentran definidas dentro del lenguaje de programación se le llama tipo de dato primitivo.

[3] Las palabras reservadas son identificadores predefinidos por el compilador que tienen un significado especial y realizan una función específica dentro de la ejecución del programa.

La Tabla 1.1 enlista los tipos de datos primitivos en C.

Tipo	Descripción	Espacio de memoria que ocupa (bytes)
char	Caracter[4]	1
int	Número entero	2
float	Número real en simple precisión	4
double	Número real en doble precisión	8

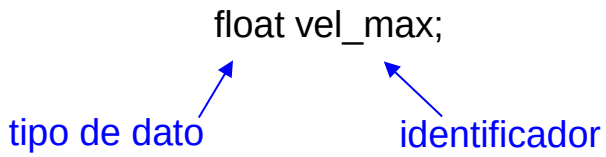
Tabla 1.1 Tipos de datos primitivos en el lenguaje C

Declaración de variables

La sintaxis para declarar una variable en el lenguaje de programación C es:

<tipoDeDato> <identificador1, identificador2, …, identificadorN>;

Es recomendable que los identificadores de las variables sean acordes con lo que se desea almacenar en ella, por ejemplo, si quiero guardar la velocidad máxima recorrida por un automóvil, en una variable de tipo real, una sugerencia de nombre puede ser vel_max. También te podrás percatar que la declaración de una variable termina con un punto y coma (;), no debes olvidar colocarlo en tus programas, porque de lo contrario, el compilador señalará un error de sintaxis. La declaración de la variable vel_max sería:



En este ejemplo, al declarar la variable vel_max, estamos reservando un espacio de memoria para una variable de tipo real, así que se podrán realizar con ella todas las operaciones matemáticas y lógicas válidas para números reales en el lenguaje C.

[4] Un caracter es una letra, un signo de puntuación, un símbolo especial (@, ^, etc.) o un número sobre el que no se pueden realizar operaciones aritméticas.

Algunos ejemplos de cómo declarar diferentes variables en C son:

Descripción del dato	Declaración de la variable
Tipo de licencia de conducir (A, B, C y D).	char tipo_licencia;
Número de pasajeros de un vehículo.	int num_pasajeros;
Litros de gasolina.	float litros;
Rendimiento por litro de un motor.	double rendimiento;

Ejercicio 1.2. Escribe la declaración en lenguaje C de las siguientes variables:

Descripción del dato	Declaración de la variable
Velocidad inicial de un automóvil.	
Velocidad final de un automóvil.	
Codifica en una sola instrucción de C, la declaración de las variables velocidad inicial y final de un automóvil.	
Posición de la palanca de velocidades de un auto (P, R, N, 1, 2 o 3).	
Bolsas de aire que tiene un coche.	

Declaración de constantes

Una constante debe ser declarada previamente a su uso. Las constantes en C tienen un tratamiento diferente al de otros lenguajes de programación. Para poder representarlas se usan *constantes simbólicas*, éstas sustituyen a una secuencia de caracteres, en lugar de representar a un dato almacenado en memoria (valor).

La sintaxis para declarar una constante en C es:

#define <identificador> <secuencia_de_caracteres>

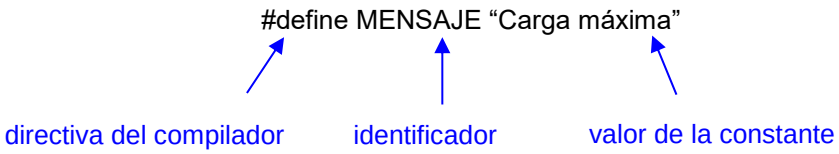
En la declaración de la constante se utiliza la directiva del preprocesador[5] *#define*. Esta directiva indica al preprocesador que debe sustituir en el código fuente del programa todas las coincidencias del <identificador> por la <secuencia_de_caracteres>, antes de realizar la compilación.

Por ejemplo, la declaración de dos constantes habituales en cálculos matemáticos es:

#define PI 3.141592
#define GRAVEDAD 9.81

Es recomendable que los identificadores de las constantes se escriban con mayúsculas, así resulta más fácil ubicarlos en el código de un programa. A diferencia de las variables, en la declaración de una constante simbólica no se escribe punto y coma al final. Tampoco está permitido declarar más de una constante simbólica en una misma línea.

Veamos otro ejemplo dentro de la industria automotriz. Si necesitamos enviar un mensaje que indique que se han cargado los litros máximos de gasolina en un auto, la declaración de la constante puede codificarse:



Ejercicio 1.3. Escribe la declaración en lenguaje C de las siguientes constantes:

Descripción del dato	Declaración de la constante
Capacidad máxima del tanque de combustible de un tráiler (1500 l).	
Mensaje que indique que se ha alcanzado la velocidad máxima.	
Rendimiento de combustible de un automóvil (5.8 l/100 km).	

[5] El preprocesador es un programa perteneciente al compilador que modifica el código fuente previo al proceso de traducción a código binario.

Ámbito de una variable

Se refiere al ciclo de vida que tiene una variable, el cual está definido en función del bloque de código (subrutina o estructura de control) en la que fue declarada. Existen básicamente dos tipos: local y global. También se le llama alcance de una variable.

Cabe mencionar que en algunos lenguajes de programación el alcance de una variable también se puede definir a través de palabras reservadas llamadas modificadores. En este caso, el ciclo de vida de la variable estará en función tanto del lugar de su declaración, como del modificador que el programador le asigne.

Variables globales

Se definen en la sección de declaración de variables globales (ver Ilustración 1.1, pág. 11). Pueden ser utilizadas y modificadas por todas las funciones definidas en el programa.

Variables locales

Se restringe su uso dentro de la estructura de control o función en la que se ha declarado y sólo puede ser utilizada dentro de ese ámbito o entorno, es decir, la variable “nace” con la estructura de control o función, y “muere” cuando la ejecución de cualquiera de éstas termina.

Las variables locales prevalecen sobre las globales, esto es, si existen dos variables con el mismo nombre en distinto ámbito, el compilador hará referencia a la local.

Por el momento, sólo veremos el concepto teórico del ámbito de acción de las variables globales y locales, retomándolo más adelante de manera práctica en el tema de procedimientos y funciones.

Uso de datos en C

Una vez que se ha revisado cómo se hace la declaración de variables y constantes en C, vamos a ver cómo se pueden manipular estos datos para obtener los resultados deseados.

La asignación o modificación del valor de un dato se realiza mediante expresiones.

Expresión

Es una combinación de uno o más valores con uno o más operadores, que al evaluarse nos entregan un valor. Las expresiones pueden ser:

- de asignación
- aritméticas

- relacionales

Expresión de asignación

Una expresión de asignación nos permite asignar un valor a una variable previamente declarada. Se realiza usando el signo "=", escribiendo a la izquierda la variable a la que se le asigna la expresión de la derecha (Figura 1.2), que puede ir desde un simple número hasta una operación más compleja.

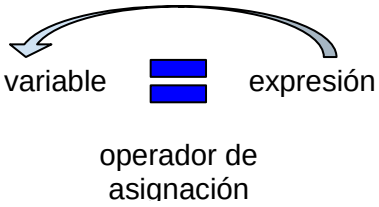


Figura 1.2. Expresión de asignación

En la Tabla 1.2, se muestran algunos ejemplos de expresiones de asignación.

Código	Resultado de la ejecución del código
float vel_max; vel_max=50;	0.0 vel_max 50.0 vel_max La primera instrucción declara la variable vel_max. Como no se inicializó con ningún valor, por default el compilador le asigna el valor 0. Después, la segunda instrucción le indica al compilador que asigne el valor 50 a la variable vel_max, por lo tanto, el 0 (valor anterior) es sobreescrito con 50 (nuevo valor asignado).
float vel_max=50;	50.0 vel_max También es posible hacer la declaración de una variable y en la misma instrucción asignarle un valor inicial.
#define GRAVEDAD 9.8 vel_max=GRAVEDAD ;	9.8 vel_max Otro caso es cuando se asigna a la variable el valor de una constante previamente definida en el programa.

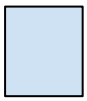
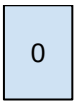
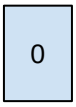
Código	Resultado de la ejecución del código
char tipo_licencia; tipo_licencia='A';	 tipo_licencia  tipo_licencia
int num_pasajeros; num_pasajeros=4;	 num_pasajeros  num_pasajeros
double rendimiento; rendimiento=16.1;	 rendimiento  rendimiento

Tabla 1.2. Ejemplos de expresiones de asignación en lenguaje C

Ejercicio 1.4. Identifica los datos necesarios para resolver cada problemática y llena la siguiente tabla:

a) Se desea conocer la distancia que recorre un auto que se desplaza en línea recta a una velocidad de 20 km/h durante 45 minutos

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato

b) Desde un edificio se deja caer una pelota, que tarda 8 segundos en llegar al piso. ¿Con qué velocidad impacta la pelota contra el piso?

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato

c) Un profesor tiene registrados los promedios de cada uno de sus estudiantes de un grupo de Informática de la ENP y desea calcular el promedio del grupo. También desea saber cuántos alumnos exentaron el examen final, considerando que la calificación mínima para exentar es 8.

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato

Expresión aritmética

Está formada por operandos que pueden ser constantes o variables y por operadores aritméticos.

Se deben considerar aspectos importantes cuando se utilizan expresiones aritméticas como:

- Tomar en cuenta los tipos de datos de las variables y constantes usados en la expresión aritmética para el resultado de la operación, éstos deben ser reales o enteros.
- Realizar la escritura correcta de la expresión, cuidando el orden y los criterios de precedencia y asociatividad de operadores.

Ejemplos:

1) distancia1 + distancia2 + distancia3
2) km_final - km_inicial
3) 16.1 * litros
4) distancia / tiempo

Expresión relacional

Este tipo de expresión compara dos valores y da como resultado un valor de verdad (falso o verdadero). En el caso específico del lenguaje de programación C, cuando se evalúa una expresión relacional y ésta es falsa, el resultado de la expresión es 0 (cero), y si es verdadera, su resultado es diferente de 0.

Ejemplos:

1) temp_interior > temp_exterior
2) capacidad_tanque != reserva
3) mes_verificacion == mes_actual
4) litros >= 5 && litros <= 10

Operadores

Como ya se mencionó en la definición de expresión, se requieren operadores, que definen las operaciones que se aplicarán sobre los valores.

Cuando se hace alguna operación matemática o se hace uso de alguna de las estructuras repetitivas o iterativas, se suelen utilizar diversos tipos de operadores. Las Tablas 1.3, 1.4, 1.5 y 1.6 muestran los operadores más utilizados en el lenguaje C:

Operadores aritméticos	
+	suma
-	Resta
*	Multiplicación
/	División
%	residuo de una división entera

Tabla 1.3. Operadores aritméticos

Operadores unarios	
+	Más unario
-	Menos unario
++	Incremento
--	Decremento
! ~	Negación y complemento

Tabla 1.4. Operadores unarios

Operadores relacionales	
>	mayor que
<	menor que
>=	mayor o igual
<=	menor o igual
==	igual
!=	diferente de

Tabla 1.5. Operadores relacionales

Operadores lógicos	
&&	y (and)
||	o (or)
!	negación (not)

Tabla 1.6. Operadores lógicos

Precedencia y asociatividad de operadores

Se conoce como precedencia a la jerarquía existente en los operadores matemáticos, ya sea aritméticos o lógicos, la cual establece el orden en el que se realizan las operaciones.

La Tabla 1.7 resume las reglas de precedencia y asociatividad de los operadores que se utilizarán en esta guía. La precedencia se ordena de mayor a menor y los operadores colocados en la misma línea tienen la misma precedencia. Por ejemplo: la multiplicación, división y residuo tienen la misma precedencia y ésta es mayor a la de la suma y la resta.

Símbolo del operador	Tipo de operación	Asociatividad
[ ], ()	Expresión	Izquierda a derecha
++, --	Incremento y decremento (post)	Izquierda a derecha
++, --	Incremento y decremento (pre)	Derecha a izquierda
!	Negación	Derecha a izquierda
+, -	Más y menos unarios	Derecha a izquierda
/, *, %	División, multiplicación, residuo	Izquierda a derecha
+, -	Suma y resta	Izquierda a derecha
<, >, <=, >=	Relacionales	Izquierda a derecha
==, !=	Igualdad y desigualdad	Izquierda a derecha
&&	Y lógico (and)	Izquierda a derecha
||	O lógico (or)	Izquierda a derecha
=, +=, -=	Asignación	Derecha a izquierda

Tabla 1.7 Reglas de precedencia y asociatividad de operadores del lenguaje C

Los siguientes ejemplos muestran la aplicación de la precedencia y asociatividad de los operadores en lenguaje C:

1) a = 1 + 4 / 2 * 5 + 3
   a = 1 + 2 * 5 + 3
   a = 1 + 10 + 3
   a = 11 + 3
   a = 14

2) Si x=15, y=18
   a = (x==y)
   a = 0

3) Si x=15, y=18, z=20
   b = (x > y) && (y <= z)
   b = 1 && (y <= z)
   b = 1 && 1
   b = 1

Ahora vamos a ver algunos ejemplos del uso combinado de los distintos tipos de expresiones:

Expresión	Resultado de la expresión

km_inicial = 100;
km_final = 185;
km_recorridos = km_final - km_inicial;
	km_recorridos = 85

rendimiento = 16.1;
km_recorridos = 85;
litros = km_recorridos / rendimiento;
	litros = 5.279503

temp_interior = 11;
temp_exterior = 7;
temp = temp_interior < temp_exterior;
	temp = 0

km_r1 = 50;
km_r2 = 80;
km_r3 = 100;
prom_km = (km_r1 + km_r2 + km_r3) / 3;
	prom_km = 76.666666

litros = 5;
km = 8;
consumo = (litros >= 5 && km < 10);
	consumo = 1

Ejercicio1.5. Calcula el resultado de las siguientes expresiones:

Expresión	Resultado de la expresión

int a = 1;
int b = 2;
int c;
c = a + b * 3 - a * b + 3;	

#define FACTOR 10
int x = 5;
int y = 4;
int z, w;
z = (x + y) * FACTOR;
w = x + y * FACTOR;	

Ejercicio 1.6. Escribe las expresiones en lenguaje C para representar las expresiones matemáticas y lógicas que se muestran a continuación:

Expresión matemática	Expresión en lenguaje C
$a = \Pi r^2$	
$c = \frac{1}{ab}$	
$c = \frac{ab}{c}$	
$v = \frac{d}{t_f - t_i}$	
Expresión relacional	**Expresión en lenguaje C**
a es mayor que b y que c	
temperatura es mayor que 10 y menor que 20	
6<=calificación<=10	
a y b son iguales	
a es igual a b, pero diferente de c	
a es igual a b o b es igual a c	

Entrada y salida de datos

La biblioteca estándar *stdio.h* de C contiene funciones de entrada y salida de datos que permiten la comunicación entre la computadora y los dispositivos estándares de entrada y salida como el teclado y el monitor. Algunas de estas funciones se describen a continuación:

Función printf( )

“Convierte, da formato e imprime los argumentos en la salida estándar. La cadena de control contiene dos tipos de objetos: caracteres ordinarios, que sencillamente se copian a la salida estándar, y especificaciones de conversión, cada una de las

cuales origina la conversión e impresión de los argumentos del printf()" (Kernighan y Ritchie, 1985, pág. 148). Su sintaxis en lenguaje C es:

printf(cadena de control, argumento1, argumento2, …, argumentoN);

Las especificaciones de conversión se muestran en la Tabla 1.8:

Símbolo	Significado
%d	notación decimal
%o	notación octal
%x	notación hexadecimal
%u	notación decimal sin signo
%c	Carácter
%s	cadena de caracteres
%e	notación decimal con el formato [-]m.nnnnnnE[±]xx
%f	notación decimal con el formato [-]mmm.nnnnnn

Tabla 1.8. Especificaciones de conversión

También es común utilizar secuencias de escape en la función printf().

Secuencias de escape

"Se utilizan para definir ciertos caracteres especiales dentro de una cadena de texto" (ccpreference.com, 2018). Las más usuales se representan en la Tabla 1.9:

Símbolo	Acción
\n	Introduce salto de línea
\t	Introduce un tabulador
\"	Muestra el caracter comilla doble
\\	Muestra el caracter diagonal invertida

Tabla 1.9. Secuencias de escape

Programa 2. Escritura de datos primitivos

Veamos un ejemplo del uso de la función printf() en un programa de C. La ilustración 1.12 muestra el resultado de la ejecución del programa.

```
#include <stdio.h> //Biblioteca con las funciones printf y scanf
#include <conio.h>
/* La biblioteca conio.h tiene la función getch(), dicha función se
utiliza para que no se cierre la pantalla al ejecutar el programa */
#define DESPEDIDA "ADIOS"
main( ) {
        char caracter = 'A';
        printf("Programa de prueba de la funcion printf\n");
        printf("El codigo ASCII de %c es %d", caracter, caracter);
        printf("\n %s", DESPEDIDA);
        getch();
}
```

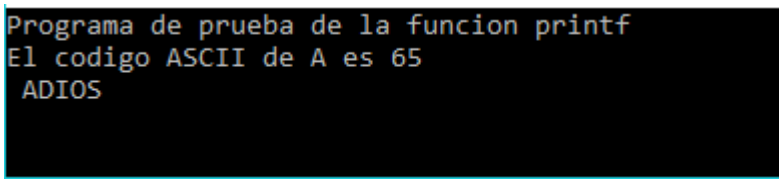


Ilustración 1.12. Resultado de la ejecución del programa 2 Escritura de datos primitivos

Es importante señalar que en la elaboración de todos los ejemplos de programas en C de esta guía se omitió el uso de acentos, ya que éstos se encuentran en los valores del código ASCII que no son contemplados de manera automática por el compilador. Sin embargo, si se desea incluirlos, habrá que hacer referencia al especificador de formato de caracter (%c) y al código ASCII del caracter acentuado a escribir.

Comentarios

Los comentarios son segmentos de código que no son considerados por el compilador y son útiles para documentar el programa y facilitar su comprensión. Se pueden agregar comentarios a un programa de C de dos formas:

- Con el símbolo “//” seguido del texto del comentario, cuando éste ocupa una sola línea, o
- cuando el comentario ocupa más de una línea, usando los símbolos “/*” al principio del texto que se quiere comentar y “*/” al final del texto.

En el código anterior se han usado comentarios para indicar el uso de las bibliotecas *stdio.h* y *conio.h*.

Función scanf( )
Lee caracteres de la entrada estándar, los interpreta de acuerdo con el formato especificado en la cadena de control, y almacena los resultados en los argumentos[6].

scanf(cadena de control, argumento1, argumento2, ..., argumentoN);

Veamos un ejemplo:

scanf("%d", &distancia);

La cadena de control "%d" le indica al compilador que el valor leído es un entero. El símbolo "&", que precede a la palabra distancia, especifica que se envía como argumento la dirección de memoria de la variable distancia. Es muy importante que no olvides capturarlo antes del nombre de la variable, porque si lo omites, el argumento que el compilador utiliza es el valor almacenado en la variable, no su ubicación en la memoria.

Programa 3. Cálculo de los kilómetros recorridos y litros consumidos

Los tableros de control de los automóviles tienen la posibilidad de mostrar al conductor estadísticas de su uso. Por ejemplo, puedes saber cuántos kilómetros recorre tu auto en el trayecto de tu casa a la escuela y cuántos litros de gasolina consume. Vamos a diseñar un programa en C que calcule estos dos valores para cualquier recorrido, considerando que el rendimiento de combustible del auto es de 5.8 l/100 km.

Identifiquemos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada:				
kilómetro inicial del recorrido	km_inicial	variable	números reales	float
kilómetro final del recorrido	km_final	variable	números reales	float
rendimiento por kilómetro	RENDIMIENTO	constante	5.8/100	float

[6] Los argumentos deben ser apuntadores.

Datos de salida:				
kilómetros recorridos del viaje	km_viaje	variable	números reales	float
litros consumidos durante el viaje	l_viaje	variable	números reales	float

El algoritmo de solución del problema es:

INICIO
RENDIMIENTO = 5.8 / 100
Solicitar al usuario el kilómetro inicial del recorrido
Almacenar el valor capturado por el usuario en la variable *km_inicial*
Solicitar al usuario el kilómetro final del recorrido
Almacenar el valor capturado por el usuario en la variable *km_final*
Calcular los km recorridos del viaje: *km_viaje = km_final - km_inicial;*
Calcular los litros consumidos del viaje: *l_viaje = km_viaje * RENDIMIENTO;*
Mostrar al usuario los km recorridos y litros consumidos durante el viaje.
FIN

La codificación del algoritmo en lenguaje C es:

```
/*      Ejemplo 3 Programa que calcula los kilómetros recorridos
        y los litros consumidos por un auto */
#include <stdio.h> //biblioteca con las funciones printf y scanf
#define RENDIMIENTO 5.8/100
main() {
        float km_inicial;
        float km_final;
        float km_viaje;
        float l_viaje;
        printf("Programa que calcula los kilometros recorridos y litros consumidos por
un auto\n");
        printf("Capture el kilometro inicial: ");
        scanf("%f", &km_inicial);
        printf("\nCapture el kilometro final: ");
        scanf("%f", &km_final);
        km_viaje = km_final - km_inicial;
        l_viaje = km_viaje * RENDIMIENTO;
        printf("\nKm recorridos: %2.2f \tlitros consumidos: %2.2f", km_viaje, l_viaje);
        return 0;
}
```

Realicemos la prueba de escritorio del programa para verificar que funcione correctamente:

km_inicial	km_final	RENDIMIENTO	km_viaje	l_viaje
0	0	5.8/100	0	0
0	50		50	2.90

El resultado de la ejecución del programa se muestra en la Ilustración 1.13.

```
Programa que calcula los kilometros recorridos y litros consumidos por un auto
Capture el kilometro inicial: 0

Capture el kilometro final: 50

Km recorridos: 50.00    litros consumidos: 2.90
--------------------------------
Process exited after 14.75 seconds with return value 0
Presione una tecla para continuar . . .
```

Ilustración 1.13. Resultado de la ejecución del programa 3

Ejercicio 1.7. Analiza la codificación en C del “Programa 3. Cálculo de los kilómetros recorridos y litros consumidos” y responde las siguientes preguntas:

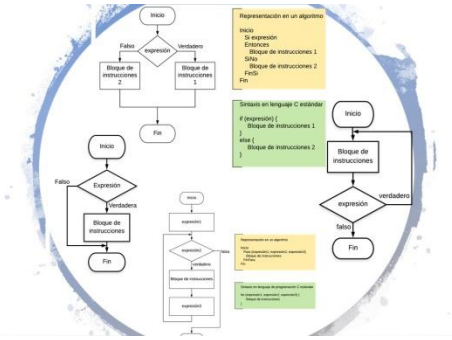
¿Qué biblioteca se está utilizando?	
¿Con qué identificador se nombra a la constante del programa y cuál es el valor que almacena?	
¿Cuáles son los tipos de datos de las variables del programa?	
¿Cuál es el dominio del tipo de datos de cada una de las variables del programa?	
¿Qué secuencias de escape se utilizan en el programa y qué hace cada una de ellas?	
En la cadena de control: "\nKm recorridos: %2.2f \tlitros consumidos: %2.2f", qué significado tiene %2.2f?	

Ejercicio 1.8. Diseña los programas en C que resuelvan los siguientes problemas. Solicita al usuario que capture los datos de entrada que necesites conocer.

a) Distancia que recorre un auto que se desplaza en línea recta a una velocidad constante durante un periodo de tiempo.
b) La velocidad con que impacta un objeto en caída libre.

Preguntas de reflexión

a) Tu profesor te plantea una situación problemática y te pide que la analicen en equipos de trabajo para identificar los datos necesarios para resolverla mediante un programa. ¿qué técnicas utilizarías para desarrollar el trabajo de manera colaborativa?
b) Tu equipo de trabajo comienza a resolver el problema y uno de tus compañeros no ha colaborado en su desarrollo. ¿Qué propondrías para resolver este conflicto?
c) Cuando se analiza una situación problemática que se quiere automatizar, ¿por qué es importante identificar cada dato involucrado en la solución?
d) ¿Qué crees que sucedería si declaras variables que no vas a utilizar?
e) ¿Es posible usar la función *scanf( )*, sin el uso previo de la función *printf( )*?, ¿por qué?

Contenidos conceptuales	**Contenidos procedimentales**
1.2 Las estructuras de control algorítmicas para modificar el flujo de ejecución: selectivas e iterativas. 	1.5 Diseño de programas aplicando la metodología de solución de problemas con ejemplos de automatización en la ciencia y la industria. **Contenidos actitudinales** 1.6 Disposición a la colaboración, cooperación y al trabajo en equipos interdisciplinarios. 1.7 Valoración de la creatividad, empatía, perseverancia, disciplina, innovación e inventiva. 1.8 Capacidad crítica y autocrítica en la toma de decisiones y solución de problemas, mediación de conflictos.

Hasta el momento, los programas de C que has diseñado para solucionar los problemas planteados tienen en común que todas las instrucciones se ejecutan en secuencia, es decir, en el orden en que fueron escritas. Pero no todos los problemas pueden resolverse de este modo. En ocasiones, es necesario ejecutar algún segmento del código de un programa solamente si se cumple una determinada circunstancia, y otro segmento diferente, en caso contrario. Por ejemplo, en un automóvil, si el nivel de gasolina llega a la reserva, se enciende el testigo de alerta de combustible en el tablero, de lo contrario, permanece apagado.

Otra consideración común en la automatización de procesos es cuando se necesitan repetir una o más instrucciones varias veces. Por ejemplo, el aire acondicionado de un vehículo puede configurarse para que la temperatura interior se conserve a 19°C. Para lograrlo, es necesario estar repitiendo la ejecución de varias instrucciones durante el tiempo que se encuentre encendido el auto. En este caso en particular, el proceso que se estaría repitiendo sería: la lectura de la temperatura, comprobar que se encuentra dentro del rango deseado y de no ser así, activar o desactivar el aire acondicionado para regresar al nivel de temperatura deseado.

Como se comentó anteriormente, la solución de este tipo de problemas no se puede programar únicamente con el uso de instrucciones que se ejecutan secuencialmente, para ello se emplean algunas otras estructuras de control.

Estructuras de control

Una estructura de control define el orden en que se ejecutan las instrucciones. Existen tres tipos: secuencial, selectiva e iterativa.

Estructura secuencial

Cuando se hace uso de la estructura de control secuencial, las instrucciones se ejecutan en el mismo orden en que fueron escritas en el programa. Su representación en un algoritmo y un diagrama de flujo se muestra en la Ilustración 1.14.

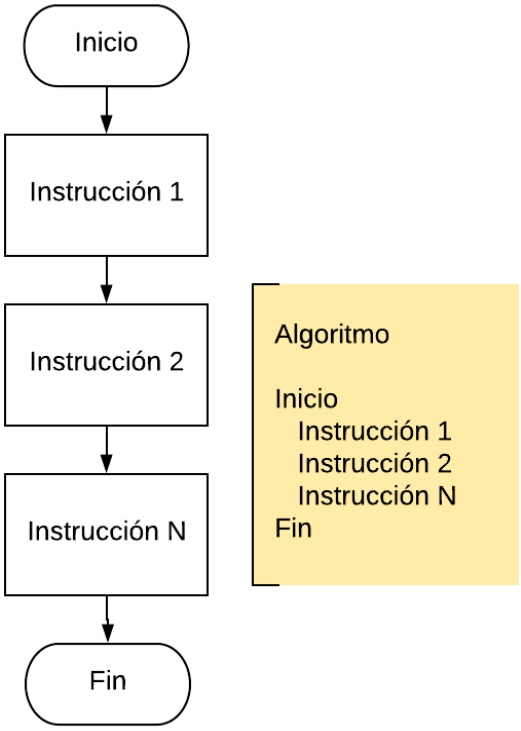


Ilustración 1.14. Estructura de control secuencial

Por ejemplo, el algoritmo para encender un automóvil es:

Inicio

1. Abrir la puerta del auto.
2. Sentarse en el asiento del conductor.
3. Meter la llave al switch de encendido.
4. Girar la llave en el sentido de las manecillas del reloj.

Fin

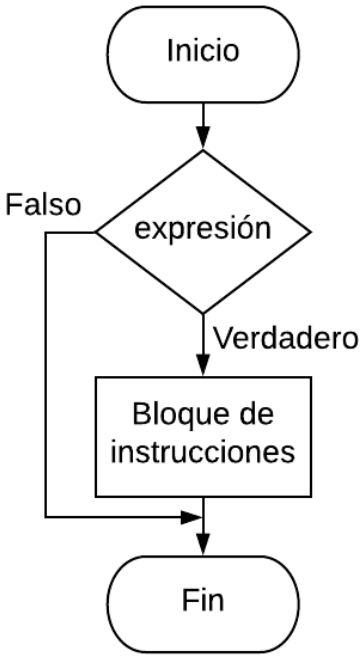
Observa cómo todas las instrucciones se realizan en el orden establecido. Esto es, se está resolviendo el problema mediante una estructura de control secuencial.

Estructuras selectivas

Una estructura de control selectiva realiza la evaluación de una expresión relacional, y dependiendo del resultado de la misma, se ejecutan diferentes segmentos del código de la estructura. Hay tres tipos: *if-then*, *if-then-else* y *switch*.

Estructura if-then (si-entonces)

Funciona de la siguiente forma: se evalúa una expresión. Si el resultado de la expresión es verdadero (para el caso de C, esto se cumple si la expresión tiene un valor distinto de cero), entonces se realiza el segmento de código definido en la estructura *if*, si es falsa (para el lenguaje C la expresión es igual a cero), no se ejecuta ninguna instrucción. En la Ilustración 1.15 se puede observar la representación de esta estructura en un diagrama de flujo y un algoritmo, así como su sintaxis en lenguaje de programación C estándar.




```
Representación en un algoritmo

Inicio
  Si expresión
  Entonces
     Bloque de instrucciones;
  FinSi
Fin
```

```
Sintaxis en lenguaje C estándar

if (expresión ) {
    bloque de instrucciones;
}
```

Ilustración 1.15. Estructura de control selectiva *if-then*

Veamos un ejemplo de una problemática que se resuelve con el uso de la estructura selectiva *if-then*.

Programa 4. Activación automática del sistema desempañador del vidrio trasero de un automóvil

Cuando la temperatura dentro de un automóvil es mayor a la temperatura exterior, se origina un proceso de condensación del agua, lo que produce que los vidrios del auto se empañen e impidan la visibilidad del conductor. Para resolver este problema, algunos automóviles cuentan con un desempañador eléctrico en el vidrio trasero que se activa de manera automática.

Tu tarea como programador será elaborar un programa en C que le avise al conductor si el sistema desempañador está activado. Considera que para simular el proceso deberás solicitar al usuario que capture la temperatura interior y exterior.

Analicemos el problema y obtengamos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada:				
Temperatura exterior del auto	temp_ext	variable	números reales	float
Temperatura interior del auto	temp_int	variable	números reales	float
Datos de salida:				
Mensaje	MENSAJE	Constante	Sistema desempañador activado	string

Una vez identificados los datos, hagamos el algoritmo para determinar el procesamiento que debe realizar el programa sobre los mismos:

Algoritmo
Inicio
  Solicitar al usuario la temperatura exterior del auto
  Almacenar el valor en la variable *temp_ext*
  Solicitar al usuario la temperatura interior del auto
  Almacenar el valor en la variable *temp_int*
  Si *temp_int* > *temp_ext*
  Entonces
    Mostrar mensaje: “Sistema desempañador activado”.
  FinSi
Fin

En primer lugar, solicitamos al usuario que capture la temperatura exterior e interior del auto y guardamos estos datos en las variables *temp_ext* y *temp_int* respectivamente. Después, usamos la estructura *if-then* para mostrar el mensaje solamente en caso de que la temperatura interior del auto sea mayor a la exterior. Observa cómo difieren los resultados que arroja el mismo programa en los dos casos evaluados en la prueba de escritorio:

<table>
<tr><th colspan="2">Prueba de escritorio del programa 4</th></tr>
<tr><td>Caso 1</td><td>Caso 2</td></tr>
<tr><td>
<table>
<tr><th>temp_int</th><th>temp_ext</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>11</td><td>7</td></tr>
</table>
</td><td>
<table>
<tr><th>temp_int</th><th>temp_ext</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>8</td><td>10</td></tr>
</table>
</td></tr>
<tr><td>¿Cuál es la temperatura interior?
11
¿Cuál es la temperatura exterior?
7
Sistema desempañador activado</td><td>¿Cuál es la temperatura interior?
8
¿Cuál es la temperatura exterior?
10</td></tr>
<tr><td>En este caso, la temperatura interior (11) es mayor que la temperatura exterior del auto (7), por lo que el resultado de la expresión temp_int > temp_ext es verdadero (diferente de 0), y por consiguiente, se ejecutará la instrucción que muestra el mensaje en pantalla.</td><td>En el caso 2, como la temperatura interior (8) es menor que la temperatura exterior (10), el resultado de la expresión es falso (0), por lo que no se muestra ningún mensaje, ya que el flujo de control del programa se va directamente a la instrucción FinSi y después termina el programa.</td></tr>
</table>

La codificación en lenguaje C del algoritmo anterior, así como el resultado de la ejecución del programa de los casos mostrados en la prueba de escritorio (Ilustración 1.16) serían:

```c
#include <stdio.h>
#define MENSAJE "Sistema desempañador activado"
int main() {
	float temp_ext;
	float temp_int;
	printf("Programa de activacion del sistema desempañador");
	printf("\nCapture la temperatura interior del auto: ");
	scanf("%f",&temp_int);
	printf("Capture la temperatura exterior del auto: ");
	scanf("%f",&temp_ext);
	if (temp_int>temp_ext) {
		printf("%s",MENSAJE);
	}
	return 0;
}
```

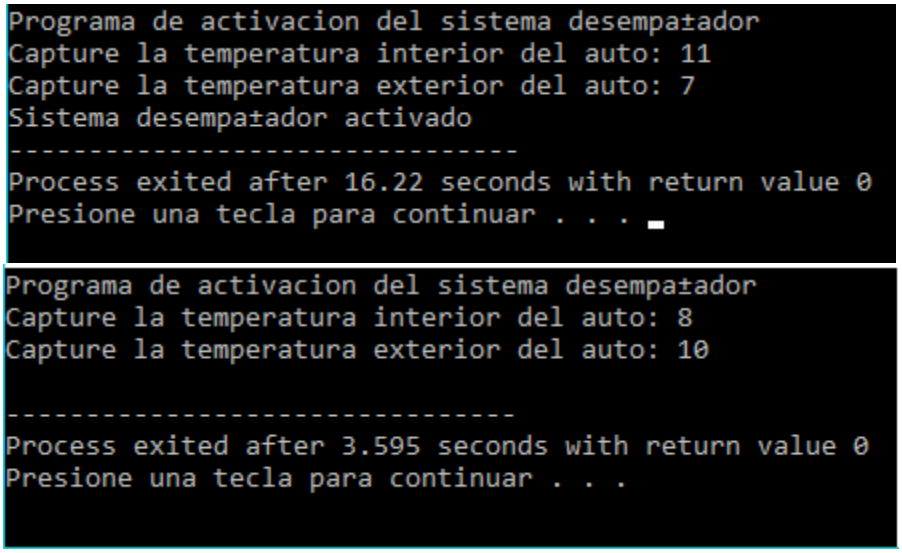


Ilustración 1.16. Resultado de la ejecución del programa 4

Ejercicio 1.9. Captura, compila y ejecuta el programa anterior probando los casos descritos en la prueba de escritorio. ¿Qué resultado arrojará el programa si la temperatura exterior e interior del auto son iguales?

Ejercicio 1.10. Utilizando el método de solución de problemas, diseña los programas en lenguaje C que resuelvan cada una de las situaciones planteadas:

a) Encender el testigo de aire acondicionado de un automóvil si la temperatura interior del auto excede los 20°C.
b) Enviar una advertencia al conductor si la presión de una llanta está fuera del rango: [29, 41] psi.
c) Encender la luz de advertencia de gasolina si el tanque de gasolina contiene entre 0 y 5 litros.

Estructura if-then-else (si-entonces-sino)

Esta estructura evalúa una expresión, si el resultado es verdadero (valor diferente de 0), se realizan el bloque de instrucciones 1, pero si es falsa (valor igual a 0), se realiza el bloque de instrucciones 2 (ver Ilustración 1.17).

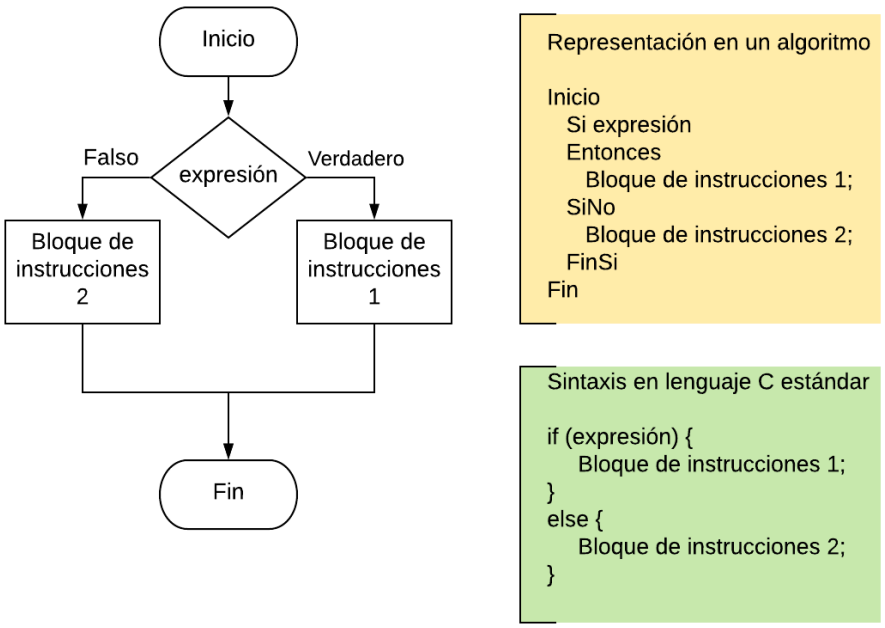


Ilustración 1.17. Estructura de control selectiva if-then-else

Analicemos un caso en donde se aplica la estructura de control selectiva *if-then-else*.

Programa 5. Control del testigo de aceite de un automóvil

En el panel de control de un automóvil, hay un testigo de aceite. Esta luz de advertencia se enciende cuando la presión del aceite del motor es baja. El testigo está conectado a un sensor en el motor que determina si el suministro de aceite es suficiente para mantener lubricadas todas sus partes, porque de lo contrario, podría dañarse.

Como programador de la industria automotriz, deberás diseñar un programa que controle el encendido/apagado del testigo de aceite de un auto, bajo las siguientes consideraciones:

- Debes solicitar al usuario que capture el estado del sensor del motor (0 ó 1).
- El testigo de aceite estará apagado si el estado del sensor del motor es 0 y encendido cuando el estado del sensor sea 1.
- Mostrar en pantalla al usuario el estado del testigo de aceite.

Analicemos el problema y obtengamos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada:				
Estado del motor	estado_motor	variable	0 ó 1	int
Datos de salida:				
Mensaje	MENSAJE	constante	Estado del testigo de aceite:	string
Testigo de aceite	testigo_aceite	variable	encendido o apagado	string

Una vez identificados los datos, determinemos la manera en que los procesará el programa. Observa cómo el valor que se debe asignar a la variable *testigo_aceite* depende del valor de la variable *estado_motor* que el usuario proporcione. Por ello, es necesario evaluar primero qué valor tiene *estado_motor*, para después poder asignar el valor correcto a la variable *testigo_aceite*.

Para programar lo anterior, utilizaremos la estructura de control selectiva *if-then-else* (si-entonces-sino). Así, el algoritmo para resolver este problema sería:

Algoritmo
Inicio

Solicitar al usuario el estado del motor
Almacenar el valor en la variable *estado_motor*
Si *estado_motor*=1
Entonces
testigo_aceite=”encendido”
SiNo
testigo_aceite=”apagado”
FinSi
Mostrar mensaje: “Estado del testigo de aceite:”, *testigo_aceite*
Fin

La codificación en lenguaje C de la solución al problema es:

```c
#include <stdio.h>
#include <string.h> //Biblioteca que contiene la función strcpy
// La función strcpy(s,t) copia la cadena de caracteres t en la cadena s
#define MENSAJE "Estado del testigo de aceite: "
int main() {
	int estado_sensor;
	char testigo_aceite[10];
	printf("Programa que controla el encendido del testigo de aceite");
	printf("\nCapture el estado del sensor del motor: ");
	scanf("%d", &estado_sensor);
	if (estado_sensor) {
		strcpy(testigo_aceite, "encendido");
	}
	else {
		strcpy(testigo_aceite, "apagado");
	}
	printf("%s %s", MENSAJE, testigo_aceite);
	return 0;
}
```

En las primeras dos líneas de la función *main()* se están definiendo las variables del programa: *estado_sensor* de tipo entero (para que pueda almacenar los valores 0 y 1) y *testigo_aceite* de tipo string[7] (para que pueda almacenar los mensajes: “encendido” o “apagado”).

La tercera línea envía un mensaje al monitor del usuario solicitando que capture el estado del sensor. En la cuarta, se almacena el valor capturado por el usuario en la variable *estado_sensor*.

[7] Para muchos lenguajes de programación el tipo de datos string, que corresponde a un conjunto de caracteres, es un tipo de dato primitivo, sin embargo, en el lenguaje C no es considerado como primitivo, por lo que se representa con un arreglo de caracteres. En el tema 1.2 de esta guía analizaremos con mayor detalle los arreglos.

A partir de la quinta línea inicia la estructura de control *if-then-else*. La estructura funciona de la siguiente manera: se evalúa la expresión que está dentro del paréntesis. En este caso, si el valor de la variable *estado_sensor* es igual a 1, se ejecuta el código que se encuentra entre las llaves inmediatas a la expresión, es decir, asignará a la variable *testigo_aceite* la palabra “encendido”. En caso contrario, si el valor de la variable *estado_sensor* es igual a 0, el resultado de la evaluación de la expresión sería falso (valor 0) y se ejecutaría el código que está entre las llaves después de la palabra *else*, es decir, la variable testigo_aceite tendría el valor “apagado”. Finalmente, la última instrucción del programa le muestra al usuario en el monitor el estado del testigo de aceite.

Si se hace la prueba de escritorio de ese programa con dos casos distintos, los resultados serían:

<table>
<tr><th colspan="2">Prueba de escritorio del programa 5</th></tr>
<tr><td>Caso 1</td><td>Caso 2</td></tr>
<tr><td>
<table>
<tr><th>estado_motor</th><th>testigo_aceite</th></tr>
<tr><td>0</td><td></td></tr>
<tr><td>1</td><td>encendido</td></tr>
</table>
</td><td>
<table>
<tr><th>estado_motor</th><th>testigo_aceite</th></tr>
<tr><td>0</td><td></td></tr>
<tr><td>0</td><td>apagado</td></tr>
</table>
</td></tr>
<tr><td>¿Cuál es el estado del sensor?
1
Estado del testigo de aceite:
encendido</td><td>¿Cuál es el estado del sensor?
0
Estado del testigo de aceite:
apagado</td></tr>
<tr><td>En este caso, como la variable estado_sensor es igual a 1, el resultado de la expresión que se evalúa en la estructura if es 1 (verdadero) y por consiguiente, se le asignará el valor encendido a la variable testigo_aceite.</td><td>En el caso 2, como la variable estado_sensor es igual a 0, el resultado de la expresión es 0 (falso) y por consiguiente, se le asigna el valor apagado a la variable testigo_aceite.</td></tr>
</table>

Los resultados de la ejecución de los casos mostrados en la prueba de escritorio se muestran en la Ilustración 1.18.

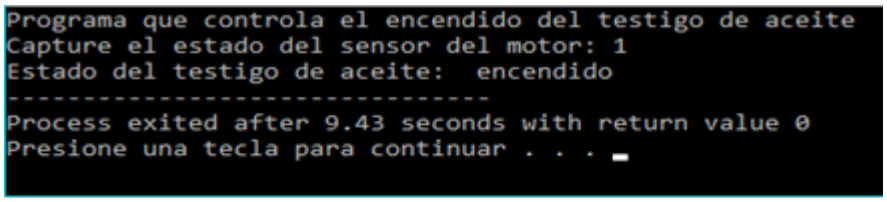


Ilustración 1.18. Resultados de la ejecución del programa 5

Ejercicio 1.11. Captura, compila y ejecuta el programa 5 con los dos casos de prueba. ¿Qué resultado se obtiene si capturas un número distinto de 0 y 1 cuando la computadora te solicita el estado del sensor del motor?, ¿por qué?

Ejercicio 1.12. Utiliza la metodología de solución de problemas para diseñar los siguientes programas en C:

a) Modifica el programa 4 de esta guía para que también avise al conductor si el sistema desempañador está desactivado.
b) Simular la activación de la alerta sísmica de la Ciudad de México, considerando que solamente se debe activar en caso de que el temblor tenga una magnitud superior a 6 grados, en caso contrario, no se activará. Solicita al usuario la magnitud del temblor y con base en ésta, envía el mensaje de si la alerta sísmica se activa o no se activa.
c) Calcular el importe a pagar en la compra de un artículo, considerando que si el monto de compra es superior a $2,000.00, se realiza un descuento del 10%.

Estructura switch (según)

Es útil para tomar decisiones múltiples (Ilustración 1.19). “Comprueba si una expresión iguala uno entre varios valores constantes, y se bifurca a partir de ellos” (Kernighan & Ritchie, 1985, pág.56). Su funcionamiento es:

1) Se evalúa la expresión.
2) Si algún caso corresponde al valor de la expresión en el switch, se ejecutan las instrucciones correspondientes a dicho caso.
3) El caso default se ejecuta si la expresión no es igual a ninguno de los valores de los otros casos. Su definición es opcional, y si no se hace, la estructura switch no realizaría ninguna acción.

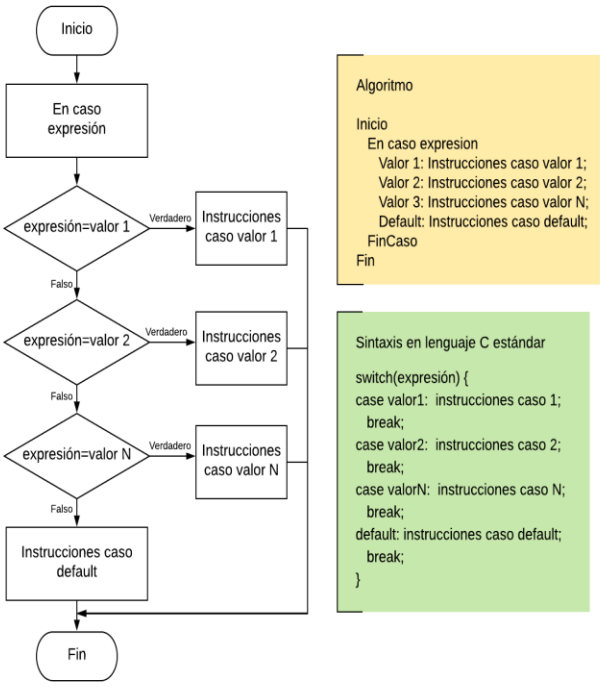


Ilustración 1.19. Diagrama de flujo de la estructura switch.

Revisemos un problema que se soluciona con la aplicación de la estructura de control *switch*.

Programa 6. Programa Hoy No Circula

En la Ciudad de México, a partir de 1990, se inició el programa Hoy No Circula con el fin de apoyar la reducción de emisiones contaminantes producidas por los automotores. Este programa restringe la circulación de los autos en distintos días de la semana de acuerdo con el color del engomado de su placa de circulación (ver Imagen 1.2).

DÍA	LIMITACIÓN DE LA CIRCULACIÓN DE LUNES A SÁBADO
HORARIO	De 5:00 a 22:00 horas
Lunes	Amarillo (5 y 6)
Martes	Rosa (7 y 8)
Miércoles	Rojo (3 y 4)
Jueves	Verde (1 y 2)
Viernes	Azul (9 y 0), permisos y matriculas sin número.
Sábado	El primer y tercer sábado del mes los vehículos con holograma 1, placa terminación impar.
	El segundo y cuarto sábado del mes los vehículos con holograma 1, placa terminación en par.
	Todos los sábados vehículos con holograma 2.

Imagen 1.2. Calendario_HoyNoCircula. (Gobierno del Estado de México, 2018)

Con base en la información anterior, diseña un programa que le indique al propietario de un vehículo cuál es el color del engomado y las terminaciones de placas que no pueden circular en un día determinado.

Analicemos el problema y obtengamos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada:				
Día de la semana	dia	variable	{L,M,I,J,V,S,D}	char
Datos de salida:				
Mensajes de texto con el color del engomado y terminación de las placas según el día de la semana.				

Es importante hacer notar que para la captura del día de la semana se creó una clave, ya que en la estructura *switch* del lenguaje de programación C, “cada opción debe ser etiquetada como un entero, constante de caracter o expresión constante” (Kernighan & Ritchie, 1985, p. 57). También observa que para el día miércoles, la clave es *‘I’*, debido a que tampoco está permitido el uso de valores iguales para distintos casos del switch.

A continuación, definamos el procesamiento que realizaremos sobre los datos de entrada para obtener la información de salida deseada:

Algoritmo
Inicio
  Mostrar en pantalla al usuario las claves del día de la semana.
  Solicitar la clave del día.

Almacenar la clave capturada por el usuario en la variable *dia*.
En caso *dia*
'L': Mostrar mensaje: "Lunes, engomado amarillo, terminación 5 y 6"
'M': Mostrar mensaje: "Martes, engomado rosa, terminación 7 y 8"
'I': Mostrar mensaje: "Miércoles, engomado rojo, terminación 3 y 4"
'J': Mostrar mensaje: "Jueves, engomado verde, terminación 1 y 2"
'V': Mostrar mensaje: "Viernes, engomado azul, terminación 9 y 0, permisos y matrículas sin número"
'S': Mostrar los mensajes: "Vehículos con holograma 2"
"El primer y tercer sábado del mes los vehículos con holograma 1, placa terminación impar"
"El segundo y cuarto sábado del mes los vehículos con holograma 1, placa terminación en par"
'D': Mostrar mensaje: "No hay restricciones. Todos los autos circulan"
Cualquier otro caso: Mostrar mensaje "Error al capturar clave del dia"
Fin caso
Fin

La codificación del algoritmo en el lenguaje de programación C[8] es:

```
#include <stdio.h>
int main() {
  char dia;
  printf("Programa Hoy no circula");
  printf("\nClaves de dia");
  printf("\n\tL - lunes");
  printf("\n\tM - martes");
  printf("\n\tI - miercoles");
  printf("\n\tJ - jueves");
  printf("\n\tV - viernes");
  printf("\n\tS - sabado");
  printf("\n\tD - domingo\n");
  printf("\nCapture la clave del dia: ");
  scanf("%c", &dia);
  switch (dia) {
    case 'L': printf("Los lunes no circulan los autos con engomado amarillo,
terminacion de placas: 5 y 6");
              break;
    case 'M': printf("Los martes no circulan los autos con engomado rosa,
terminacion de placas:  7 y 8");
              break;
    case 'I': printf("Los miercoles no circulan los autos con engomado rojo,
terminacion de placas: 3 y 4");
              break;
```

[8] Algunas instrucciones *printf* ocupan dos renglones por cuestiones de edición del documento, sin embargo, recuerda que deben capturarse en una sola línea.

```
    case 'J': printf("Los jueves no circulan los autos con engomado verde,
terminacion de placas: 1 y 2");
              break;
    case 'V': printf("Los viernes no circulan los autos con engomado azul,
terminacion de placas: 9 y 0");
              printf(", permisos y matriculas sin numero");
              break;
    case 'S': printf("El primer y tercer sabado del mes los vehiculos con holograma
1, placa terminacion impar\n");
        printf("El segundo y cuarto sabado del mes los vehiculos con holograma 1,
placa terminacion en par\n");
printf("Vehiculos con holograma 2");
              break;
    case 'D': printf("No hay restricciones. Todos los vehiculos circulan");
              break;
    default: printf("Error al capturar la clave del dia. Utilice mayusculas");
              break;
  }
return 0;
}
```

Finalmente, realicemos la prueba de escritorio para verificar el correcto funcionamiento del programa:

Prueba de escritorio del programa 6	
Caso 1	Caso 2
dia M	dia x
Programa Hoy no circula Claves de dia L - lunes M - martes I - miercoles J - jueves V - viernes S - sabado D - domingo Capture la clave del día: M	Programa Hoy no circula Claves de dia L - lunes M - martes I - miercoles J - jueves V - viernes S - sabado D - domingo Capture la clave del dia: x

Los martes no circulan los autos con engomado rosa, terminación de placas: 7 y 8.	Error al capturar la clave del dia. Utilice mayusculas.
Como la variable *dia* es igual a 'M', entonces, el programa solamente ejecuta las sentencias que están en el *case* 'M' y después, envía el flujo de control del programa al final de la estructura switch con la instrucción *break*. Fíjate cómo no se ejecutaron ninguna de las instrucciones de los otros casos. En la Ilustración 1.20, se observa el resultado de la ejecución del programa para este caso particular.	La variable *dia* es igual a 'x', por lo que el programa no encuentra ninguna coincidencia con los casos especificados, por lo tanto, ejecuta el código de la declaración *default* y envía el flujo de control del programa al final de la estructura switch con la instrucción *break*.

```
Programa Hoy no circula
Claves de dia
        L - lunes
        M - martes
        I - miercoles
        J - jueves
        V - viernes
        S - sabado
        D - domingo

Capture la clave del dia: M
Los martes no circulan los autos con engomado rosa, terminacion de placas:  7 y 8
--------------------------------
Process exited after 10.23 seconds with return value 0
Presione una tecla para continuar . . .
```

Ilustración 1.20. Pantalla de ejecución del programa 6

Ejercicio 1.13. Realiza la captura, compilación y ejecución del programa 6. Haz la prueba de escritorio para todos los valores posibles de día, además, prueba capturar la clave del día con letra minúscula. ¿Qué mensaje se muestra al usuario?, ¿por qué?

Ejercicio 1.14. Diseña los programas en lenguaje C, utilizando el método de solución de problemas, que resuelva las situaciones planteadas:

a) En un automóvil de transmisión automática existen diferentes velocidades (P, R, N, D, 1, 2 y 3). Elabora un programa que encienda el testigo correspondiente a la velocidad seleccionada por el conductor.
b) Se necesita calcular el importe a pagar de un boleto de cine, considerando los siguientes costos y promociones:

Costo del boleto: $75.00	
Día de la semana	Promoción
Lunes y martes	50% de descuento
Miércoles	2x1
Jueves	20% descuento
Viernes, sábado y domingo	Ninguna

Estructuras de iteración

También se les conoce como ciclos o bucles. Se usan para repetir una o más instrucciones. En el lenguaje de programación C, existen tres: *while*, *for* y *do-while*.

Estructura while (mientras)

Repite un bloque de instrucciones mientras una expresión sea verdadera (diferente de 0 para el lenguaje C). Su representación en un algoritmo y en un diagrama de flujo, así como su sintaxis en lenguaje C estándar se muestra en la Ilustración 1.21.

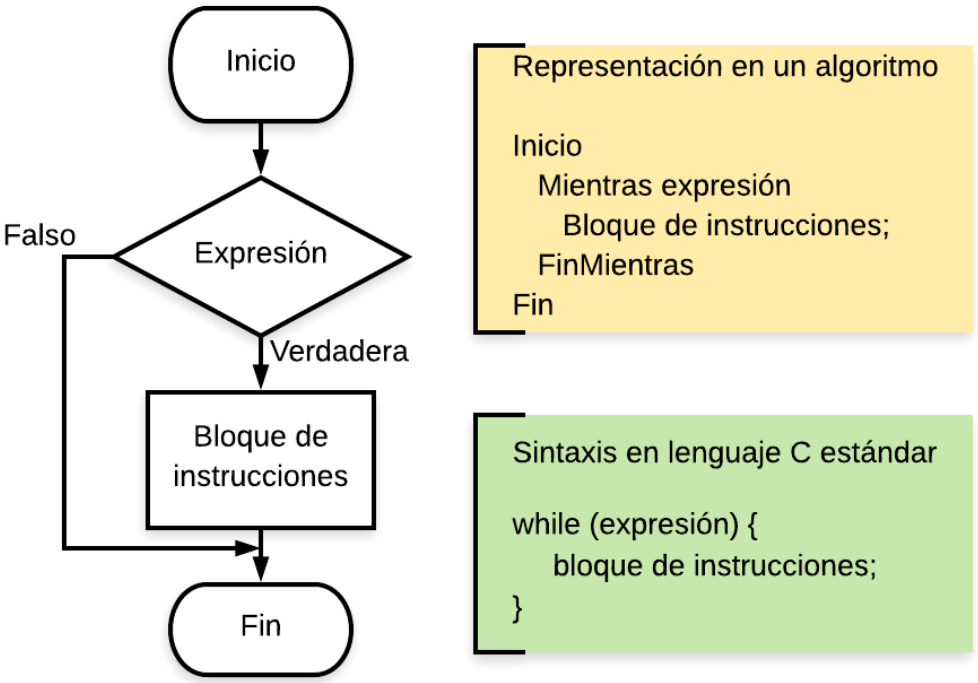


Ilustración 1.21. Estructura de control while (mientras)

El funcionamiento de la estructura *while* es: Se evalúa la expresión. Si es verdadera (el valor de la expresión es diferente de 0), se ejecuta el bloque de instrucciones y se devuelve el control del programa nuevamente a la evaluación de la expresión. Si es falsa (el valor de la expresión es 0), termina la ejecución de la estructura y el control del programa pasa a la siguiente instrucción después del *while*.

Observa que puede darse el caso de que el código dentro del *while* no se llegue a ejecutar en ninguna ocasión en el programa. Esto ocurre si desde la primera vez que se evalúa la expresión, ésta es falsa, de otro modo, se ejecutará el ciclo una o más veces, mientras la expresión sea verdadera.

Programa 7. Pago de multas

Para realizar la verificación vehicular de un automóvil en la Ciudad de México es obligatorio no tener adeudos de multas. El propietario de un vehículo desea conocer el importe total de multas que debe pagar antes de verificar su auto. Diseña un programa en lenguaje C que calcule dicho importe, solicitando al usuario el número de multas y el importe de cada una de ellas.

Identifiquemos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada				
Número de multas a pagar	total_multas	variable	números enteros	int
Importe de cada multa	importe_multa	variable	números reales	float
Datos de salida				
Importe total de multas	importe_total	variable	números reales	float
Variables auxiliares				
Contador del número de veces que se repite el ciclo	cont_multas	variable	números enteros	int

Hagamos el diseño de la solución al problema planteado:

Algoritmo
Inicio
  Inicializar la variable *importe_total* a 0
  Inicializar la variable *cont_multas* a 0
  Solicitar al usuario el número de multas que debe pagar
  Almacenar el valor capturado por el usuario en la variable *total_multas*
  Mientras (cont_multas < total_multas)
    Sumar 1 a *cont_multas*
    Solicitar al usuario el importe de la multa
    Almacenar el valor capturado por el usuario en la variable *importe_multa*
    Agregar al *importe_total* el *importe_multa*
  FinMientras
  Mostrar al usuario el *importe_total*
Fin

La codificación de la solución en lenguaje C es:

```c
#include <stdio.h>
int main() {
	int total_multas;
	int cont_multas = 0;
	float importe_multa;
	float importe_total = 0;
	printf("Programa que calcula el importe a pagar por concepto de multas");
	printf("\nCuantas multas va a pagar: ");
	scanf("%d", &total_multas);
	while (cont_multas < total_multas) {
		cont_multas++;
		printf("\nImporte de la multa %d: ", cont_multas);
		scanf("%f", &importe_multa);
		importe_total = importe_total + importe_multa;
	}
	printf("\nImporte a pagar de %d multas es: $%5.2f", cont_multas, importe_total);
	return 0;
}
```

Realicemos la prueba de escritorio de nuestro programa con los datos:
Número de multas: 3
Importe multa 1: $100.00 Importe multa 2: $200.00 Importe multa 3: $150.00

Observa en la prueba de escritorio cómo se modifican las variables del programa en cada una de las repeticiones del *while*. La Ilustración 1.22 muestra los resultados del programa para este caso en particular.

Prueba de escritorio del programa 7 Pago de multas					
total_ multas	cont_ multas	expresión de relación	resultado expresión	importe_multa	importe_total
0	0			0	0
3	0	0<3	1	100.00	100.00
3	1	1<3	1	200.00	300.00
3	2	2<3	1	150.00	450.00
3	3	3<3	0		

```
Programa que calcula el importe a pagar por concepto de multas
Cuantas multas va a pagar?
3
Importe de la multa  1 : 100
Importe de la multa  2 : 200
Importe de la multa  3 : 150

Importe a pagar de  3  multas es: $ 450.00
--------------------------------
Process exited after 19.43 seconds with return value 0
Presione una tecla para continuar . . .
```

Ilustración 1.22. Resultado de la ejecución del programa 7 Pago de multas

Ejercicio 1.15. Captura, compila y ejecuta el programa 7 para diferentes casos. Prueba qué pasa si el número de multas es 0. En este caso, ¿cuántas veces se ejecutó el ciclo?, ¿por qué?

Ejercicio 1.16. Diseña un programa en C que resuelva las siguientes problemáticas utilizando la metodología de solución de problemas:

a) Modifica el *programa 4 Activación automática del sistema desempañador del vidrio trasero de un automóvil* de esta guía para que el usuario lo ejecute el número de veces que él desee.
b) Modifica el programa del ejercicio 14, inciso a) de esta guía, para que se pueda capturar el estado de la palanca de velocidades de un auto tantas veces desee el usuario.

Estructura for (para)

Es otra de las estructuras iterativas. Su representación en algoritmo, diagrama de flujo y sintaxis en lenguaje C estándar se muestra en la Ilustración 1.23.

Gramaticalmente en C, los tres componentes de un *for* son expresiones. Comúnmente, *expresion1* y *expresión3* son asignaciones, mientras que *expresión2* es una expresión de relación. Es equivalente a:

```
expresión1;
while (expresión2) {
        Bloque de instrucciones
        expresión3;
}
```

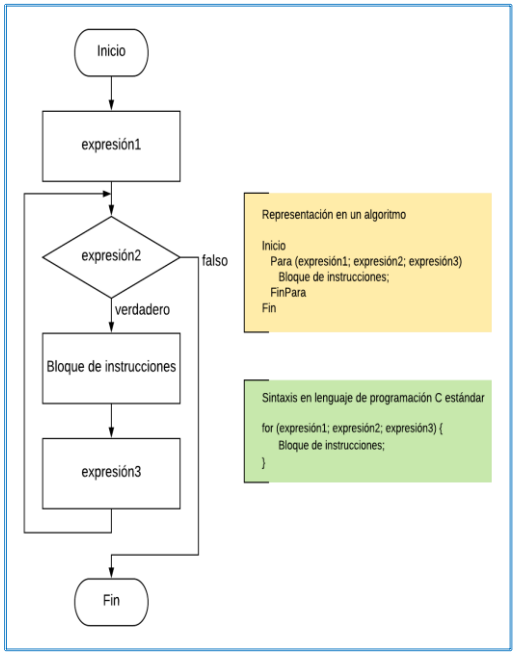


Ilustración 1.23. Estructura de control *for* (para)

Programa 8. Pago de multas (solución con la estructura *for*)

Para ejemplificar el uso de la estructura *for*, usaremos la misma situación problemática descrita en el "programa 7 Pago de multas", sólo que ahora en lugar de resolverlo con la estructura *while*, lo haremos con *for*.

El diseño de la solución sería:

Algoritmo
Inicio
  Inicializar la variable *importe_total* a 0.
  Solicitar al usuario el número de multas que debe pagar
  Almacenar el valor capturado por el usuario en la variable *total_multas*
  Para (*cont_multas* = 0; *cont_multas* <= *total_multas*; *cont_multas*++)
    Solicitar al usuario el importe de la multa
    Almacenar el valor capturado por el usuario en la variable *importe_multa*
    Agregar al *importe_total* el *importe_multa*
  FinPara
  Mostrar al usuario el *importe_total*
Fin

En la Figura 1.3 se muestra cómo se especifican las expresiones de la estructura *for* en el lenguaje C.

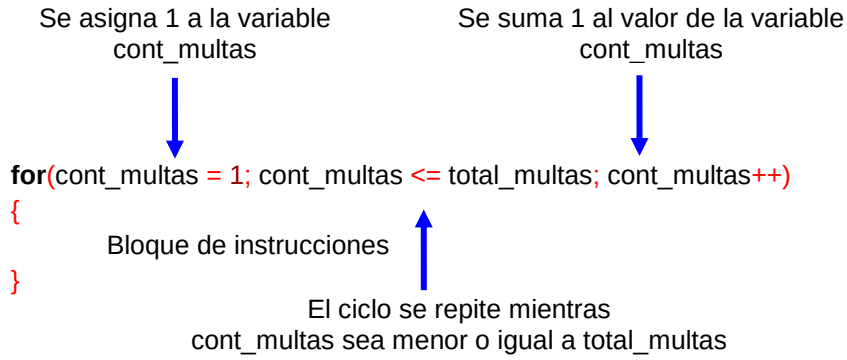


Figura 1.3. Estructura iterativa for

La estructura *for* funciona de la siguiente manera:

1) Asigna a la variable cont_multas el valor de 1.
2) Evalúa la condición cont_multas <= total_multas.
   Si es verdadera (diferente de cero):
3) Ejecuta el bloque de instrucciones del *for*.
4) Incrementa la variable cont_multas en 1.
5) Regresa a evaluar la condición (paso 2).
   Si es falsa (igual a 0):
6) Envía el control del programa al final de la estructura *for*.

Así, la codificación en lenguaje C de la solución al programa 8 es:

```
#include <stdio.h>
int main() {
	int total_multas;
	int cont_multas;
	float importe_multa;
	float importe_total = 0;
	printf("Programa que calcula el importe a pagar por concepto de multas");
	printf("\nCuantas multas va a pagar: ");
	scanf("%d", &total_multas);
	for(cont_multas = 1; cont_multas <= total_multas; cont_multas++) {
		printf("\nImporte de la multa %d: ", cont_multas);
		scanf("%f", &importe_multa);
		importe_total = importe_total + importe_multa;
	}
	printf("\nImporte a pagar de %d multas es: $%5.2f", cont_multas - 1,
importe_total);
	return 0;
}
```

Realicemos la prueba de escritorio con los mismos datos que usamos con el programa que empleaba la estructura *while*:

Número de multas: 3
Importe multa 1: $100.00
Importe multa 2: $200.00
Importe multa 3: $150.00

Prueba de escritorio del Programa 8 Pago de multas (uso de la estructura for)

total_ multas	cont_ multas	expresión de relación	resultado expresión	importe_multa	importe_total
0	0			0	0
3	1	1<=3	1	100.00	100.00
3	2	2<=3	1	200.00	300.00
3	3	3<=3	1	150.00	450.00
3	4	4<=3	0		

La Ilustración 1.24 muestra el resultado de la ejecución del programa.

```
Programa que calcula el importe a pagar por concepto de multas
Cuantas multas va a pagar?
3
Importe de la multa  1 : 100
Importe de la multa  2 : 200
Importe de la multa  3 : 150
Importe a pagar de  3  multas es: $ 450.00
--------------------------------
Process exited after 5.497 seconds with return value 0
Presione una tecla para continuar . . .
```

Ilustración 1.24. Resultado de la ejecución del Programa 8 Pago de multas (solución con la estructura for)

Ejercicio 1.17. Captura, compila y ejecuta el programa 8 para diferentes casos y responde las siguientes preguntas:

a) Si el número de multas es 0, ¿se ejecuta alguna vez el ciclo?, ¿por qué?
b) ¿Por qué en la última instrucción del programa se le resta uno a la variable *cont_multas*?

Ejercicio 1.18. El programa “Pago de multas” que utiliza la estructura *for* da el mismo resultado que el que emplea la estructura *while*, sin embargo, al comparar las pruebas de escritorio se observa que no son iguales. Responde las siguientes preguntas:

a) ¿En qué varían y por qué?
b) ¿Podría empezar el ciclo *for* inicializando la variable *cont_multas* con 0 como en la solución que emplea el *while*? Si es así, ¿qué cambios tendrías que hacer al programa?

Ejercicio 1.19. Diseña los programas en lenguaje C, que resuelvan las problemáticas planteadas utilizando la metodología de solución de problemas:

a) Tu tío es dueño de una pequeña tlapalería. La persona a cargo del mostrador tiene problemas para calcular el importe total que debe pagar un cliente cuando compra varios productos. Por lo anterior, decides ayudarlo haciendo una aplicación que solicita el número total de productos comprados, el importe de cada uno y calcula el total a pagar.
b) Estás realizando una investigación sobre el cambio climático en la CDMX dentro de la Estación Meteorológica de la ENP. Tu profesor de física, quien te asesora en este proyecto, te solicita hacer un programa que determine la temperatura máxima registrada en el día. En tus notas, dispones de 24 lecturas de temperatura diarias.

Estructura do-while (haz-mientras)

Funciona de manera similar al *while*, la diferencia es que con el uso de esta estructura siempre se ejecutará el código que se desea repetir al menos una vez (ver Ilustración 1.25), mientras que en el *while* como lo comentamos anteriormente, puede ocurrir que el código no se ejecute en ninguna ocasión.

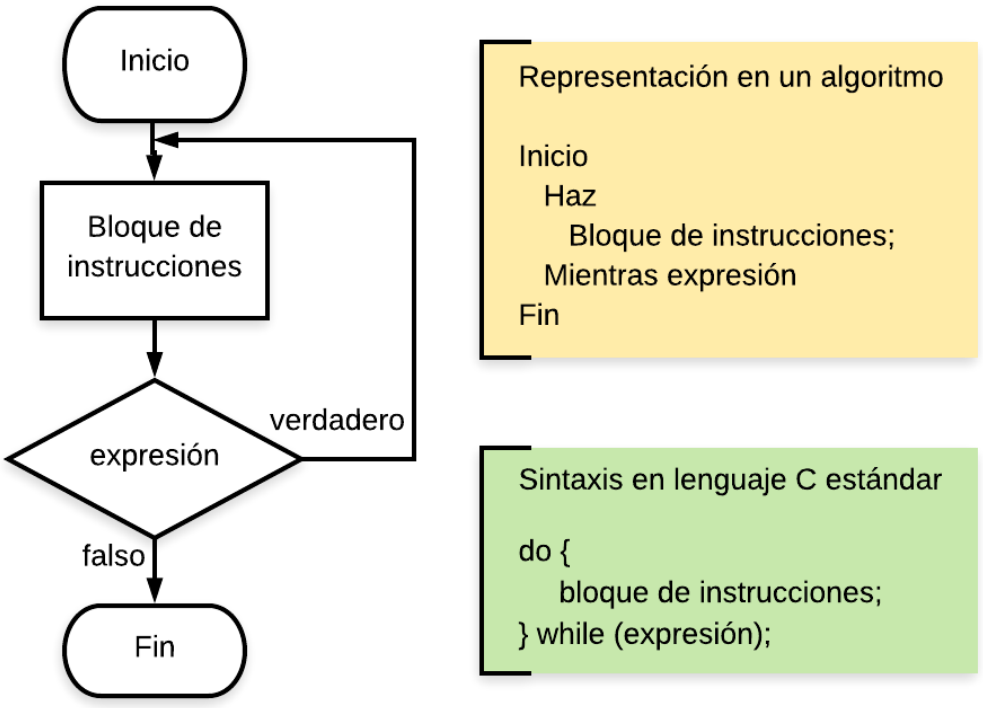


Ilustración 1.25. Estructura de control do-while (haz-mientras)

Funciona de la siguiente forma:

1) Se ejecuta el bloque de instrucciones que está entre las palabras reservadas *do* y *while.*
2) Se evalúa la expresión, si es verdadera (el valor de la expresión es diferente de 0), se regresa el control del programa al inicio de la estructura (instrucción *do*), es decir, se regresa al paso 1.
3) La iteración termina cuando el valor de la expresión cambia a falso (igual a 0).

Veamos la solución a un problema que se resuelve mediante la estructura do-while:

Programa 9. Calcular el importe a pagar por concepto de gasolina

En una gasolinera, las máquinas despachadoras calculan el importe a pagar dependiendo del número de litros que se depositaron en el tanque del automóvil. Este proceso se repite cada vez que llega un nuevo cliente. Vamos a ver cómo podríamos codificar la solución a este problema en lenguaje C utilizando la estructura de iteración *do-while*. Asume que el costo por litro de gasolina es de $19.99.

Identifiquemos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada				
Precio del litro de gasolina	PRECIO_LITRO	constante	19.99	float
litros vendidos	Litros	variable	números reales	float
Datos de salida				
Importe por pagar	Importe	variable	números reales	float
Respuesta si desea realizar otra venta [si/no]	Ventas	variable	‘S’ o ‘N’	char

El diseño de la solución podría ser:

Algoritmo
Inicio
  PRECIO_LITROS = 19.99
  Haz
    Solicitar al usuario capturar los litros de gasolina vendidos
    Almacenar el valor capturado por el usuario en la variable *litros*
    Calcular *importe* = *litros* x *PRECIO_LITROS*
    Mostrar al usuario el *importe*
    Preguntar al usuario se desea realizar otra venta
    Almacenar la respuesta del usuario en la variable *ventas*
Mientras *ventas* = 'S'
Fin

La codificación de la solución en lenguaje C es:

```
#include <stdio.h>
#define PRECIO_LITRO 19.99
int main() {
        float litros;
        float importe;
        char ventas;
        printf("Programa que calcula el importe a pagar por concepto de gasolina");
        do {
                printf("\nLitros vendidos: ");
                scanf("%f", &litros);
                importe = litros * PRECIO_LITRO;
                printf("Importe por pagar: $ %5.2f", importe);
                printf("\nDesea realizar otra venta?b");
                scanf(" %c", &ventas);
        } while (ventas=='S');
}
```

Veamos cómo funciona el programa. Primero, realizamos la definición de constantes y variables. Observa que se definió una variable llamada *ventas* de tipo char para controlar la repetición del ciclo *do-while*. Mientras *ventas* sea igual a la letra 'S' (mayúscula), el código entre las palabras reservadas *do* y *while* se estará repitiendo. Si toma algún otro valor, el programa terminará.

Por lo anterior, las instrucciones de solicitud de captura de los litros vendidos, el cálculo e impresión del importe a pagar y la pregunta si desea realizar otra venta, se ejecutarán al menos una vez en el programa, y se repetirán tantas veces como el usuario responda la letra 'S' a la pregunta. En el momento en que capture cualquier otro caracter, el programa terminará ya que la expresión *ventas=='S'* valdrá 0 (falso).

Realicemos la prueba de escritorio registrando 3 ventas con los siguientes datos:

Venta 1: 15 litros
Venta 2: 30 litros
Venta 3: 50 litros

Prueba de escritorio del programa 9				
litros	Importe	ventas	expresión de relación	resultado de la expresión
0	0			
15	299.85	S	'S'=='S'	1
30	599.70	S	'S'=='S'	1
50	999.50	N	'N'=='S'	0

Programa que calcula el importe a pagar por concepto de gasolina
Litros vendidos: 15
Importe por pagar: $299.85
Desea realizar otra venta? S

Litros vendidos: 30
Importe por pagar: $599.70
Desea realizar otra venta? S

Litros vendidos: 50
Importe por pagar: $999.50
Desea realizar otra venta? N

En la Ilustración 1.26 se observan los resultados del programa para estos datos de prueba:

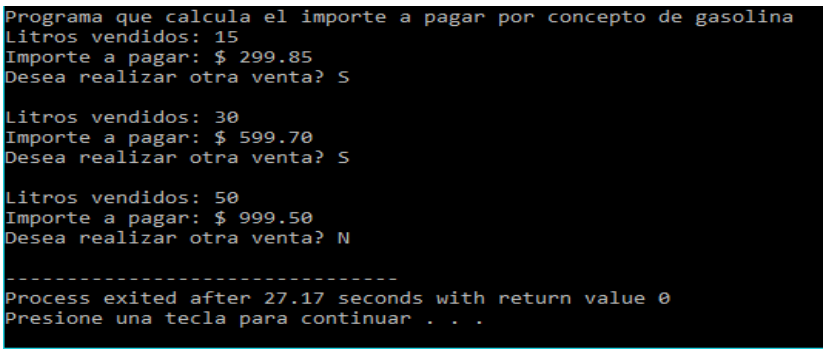


Ilustración 1.26. Resultado de la ejecución del programa 9

Ejercicio 1.20. Captura, compila y ejecuta el programa 9 para que observes los resultados. ¿Puede suceder el caso de que el programa no calcule ningún importe?

Ejercicio 1.21. Diseña un programa en C que resuelva las siguientes problemáticas utilizando la metodología de solución de problemas:

a) Modifica el *programa 4 Activación automática del sistema desempañador del vidrio trasero de un automóvil* de esta guía para que el usuario lo ejecute el número de veces que él desee. Emplea la estructura *do-while* en la solución.
b) Modifica el programa del ejercicio 14, inciso a) de esta guía, para que se pueda capturar el estado de la palanca de velocidades de un auto tantas veces desee el usuario. Emplea la estructura *do-while* en la solución.

Estructuras de control anidadas

Ahora que ya conoces las estructuras de control de secuencia, selección e iteración, te preguntarás si es posible combinarlas en un mismo programa. La respuesta es sí. Muchos programas utilizan una o más estructuras de control para resolver un problema.

En el ámbito de la programación cuando una estructura de control forma parte del cuerpo de otra, se denominan estructuras anidadas. Veamos un ejemplo:

Programa 10. Testigo de alerta de combustible

Una de las preocupaciones más comunes de los conductores es saber cuánta gasolina tiene el auto y hasta dónde puede llegar antes de que se vacíe el tanque. Para ello, los tableros de los automóviles disponen de un testigo de alerta de combustible. Generalmente éste se enciende cuando la gasolina almacenada en el tanque del coche llega a un valor en litros conocido como reserva. En este contexto, diseñarás un programa que controle el encendido del testigo de alerta de combustible de un automóvil, tomando en cuenta las siguientes consideraciones:

- El tanque de gasolina tiene una capacidad de 50 litros.
- El rendimiento de combustible es de 20.97 km/litro.
- La reserva es de 5 litros.

- El programa inicia con el tanque de gasolina lleno y termina cuando el tanque esté vacío.
- El programa calculará el número de litros consumidos de acuerdo con el kilometraje recorrido por el auto, el cual es proporcionado por el usuario.
- El testigo de alerta de combustible deberá encenderse si los litros almacenados en el tanque de gasolina son igual o menor a la reserva.
- Validar que no se hagan recorridos que utilicen más combustible del disponible en el tanque de gasolina.

Identifiquemos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada				
Capacidad del tanque	L_MAXIMO	Constante	50	
Rendimiento km/l	RENDIMIENTO	Constante	20.97	
Reserva	RESERVA	Constante	5	
Kilómetros del recorrido	km_recorrido	Variable	números reales	float
Datos de salida				
Kilómetros disponibles	km_disponibles	Variable	números reales	float
Litros disponibles	l_disponibles	Variable	números reales	float
Litros recorrido	l_recorrido	Variable	números reales	float
Testigo de reserva	Mensaje Testigo de reserva encendido.			

El diseño de la solución del problema es:

```
Algoritmo
Inicio
  L_MAXIMO = 50
  RENDIMIENTO = 20.97
  RESERVA = 5
  l_disponibles = L_MAXIMO
  km_disponibles = l_disponibles x RENDIMIENTO
  Mientras (km_disponibles > 0) haz
      Muestra en pantalla km_disponibles y l_disponibles
      Solicita al usuario los kilómetros de su recorrido
      Guarda el valor capturado por el usuario en la variable km_recorrido
      Si (km_recorrido <= km_disponibles) entonces
          km_disponibles = km_disponibles - km_recorrido
          l_recorrido = km_recorrido/RENDIMIENTO
          Muestra en pantalla km_recorrido y l_recorrido
          l_disponibles = l_disponibles - l_recorrido
          Si (l_disponibles <= RESERVA) entonces
              Mostrar en pantalla el mensaje “Testigo de reserva encendido”
          FinSi
      SiNo
          Muestra en pantalla el mensaje “No hay suficiente gasolina para su
          recorrido”
      FinSi
  FinMientras
  Muestra en pantalla el mensaje “Tanque vacío, ya no puede hacer más recorridos”
Fin
```

La codificación en lenguaje C es:

```c
#include <stdio.h>
#include <string.h>
#define L_MAXIMO 50
#define RENDIMIENTO 20.97
#define RESERVA 5
int main() {
  float l_disponibles = L_MAXIMO;
  float l_recorrido, km_disponibles, km_recorrido;
  km_disponibles = l_disponibles * RENDIMIENTO;
  printf("Programa testigo de alerta de combustible");
  while (km_disponibles > 0) {
      printf("\nKilometros disponibles: %5.2f", km_disponibles);
      printf("\tLitros disponibles: %5.2f", l_disponibles);
      printf("\nCaptura el numero de kilometros de tu recorrido: ");
      scanf("%f", &km_recorrido);
```

```
        if (km_recorrido <= km_disponibles) {
                km_disponibles = km_disponibles - km_recorrido;
                l_recorrido = km_recorrido / RENDIMIENTO;
                printf("Kilometros del recorrido: %5.2f \t", km_recorrido);
                printf("Litros consumidos: %5.2f \n", l_recorrido);
                l_disponibles = l_disponibles - l_recorrido;
                if (l_disponibles <= RESERVA)
                        printf("Testigo de reserva encendido\n");
        }
        else {
        printf("No hay suficiente gasolina para hacer su recorrido\n");
                }
        }
          printf("Tanque vacio, ya no puede hacer mas recorridos");
}
```

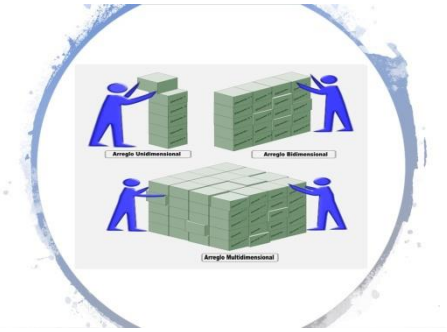
La Ilustración 1.27 muestra un ejemplo de la ejecución del programa.

```
Programa testigo de alerta de combustible
Kilometros disponibles:  1048.50        Litros disponibles:  50.00
Captura el numero de kilometros de tu recorrido: 50
Kilometros del recorrido:  50.00        Litros consumidos:   2.38

Kilometros disponibles:  998.50 Litros disponibles:  47.62
Captura el numero de kilometros de tu recorrido: 900
Kilometros del recorrido:  900.00       Litros consumidos:  42.92
Testigo de reserva encendido

Kilometros disponibles:  98.50  Litros disponibles:   4.70
Captura el numero de kilometros de tu recorrido: 100
No hay suficiente gasolina para hacer su recorrido

Kilometros disponibles:  98.50  Litros disponibles:   4.70
Captura el numero de kilometros de tu recorrido: 90
Kilometros del recorrido:  90.00        Litros consumidos:   4.29
Testigo de reserva encendido

Kilometros disponibles:   8.50  Litros disponibles:   0.41
Captura el numero de kilometros de tu recorrido: 5
Kilometros del recorrido:   5.00        Litros consumidos:   0.24
Testigo de reserva encendido

Kilometros disponibles:   3.50  Litros disponibles:   0.17
Captura el numero de kilometros de tu recorrido: 3.5
Kilometros del recorrido:   3.50        Litros consumidos:   0.17
Testigo de reserva encendido
Tanque vacio, ya no puede hacer mas recorridos
--------------------------------
Process exited after 45.38 seconds with return value 0
Presione una tecla para continuar . . .
```

Ilustración 1.27. Resultado de la ejecución del programa 10

Preguntas de reflexión

1. ¿Consideras que para resolver un problema cuya solución se diseña con estructuras de control selectivas e iterativas es más fácil y rápido iniciar directamente con la codificación en un lenguaje de programación, que seguir los pasos de la metodología de solución de problemas?, ¿por qué?
2. Tu profesor de Informática les deja de tarea resolver en equipos de 4 personas todos los ejercicios del tema 1.2 de esta guía. ¿Cómo se organizarían para realizar la tarea?
3. En clase de Informática realizas una actividad de coevaluación de los programas que usan estructuras cíclicas de uno de tus compañeros. ¿Qué consejos le darías para que te fuera más fácil entender sus programas y detectar posibles errores?
4. De las estructuras de repetición, ¿cuál consideras que es la mejor? Justifica tu respuesta.
5. Tu profesor te solicita diseñar un programa que calcule la potencia de un número entero cualquiera. Llevas varias horas intentando solucionar el problema, pero no puedes. Decides descargar la solución de una página de internet, analizar cómo funciona y entregar ese programa. ¿Tu actitud para resolver esta situación es ética?

Contenidos conceptuales	Contenidos procedimentales
1.3 Declaración, asignación, lectura y escritura de datos estructurados: arreglos unidimensionales y bidimensionales.  	1.5 Diseño de programas aplicando la metodología de solución de problemas con ejemplos de automatización en la ciencia y la industria. **Contenidos actitudinales** 1.6 Disposición a la colaboración, cooperación y al trabajo en equipos interdisciplinarios. 1.7 Valoración de la creatividad, empatía, perseverancia, disciplina, innovación e inventiva. 1.8 Capacidad crítica y autocrítica en la toma de decisiones y solución de problemas, mediación de conflictos.

En la práctica, la información no se encuentra aislada, siempre está estructurada y organizada. Piensa en la estructura que tiene un diccionario o una enciclopedia, son colecciones de datos organizados o estructurados bajo ciertas reglas, si no fuera así, no serían de utilidad. Como puedes ver, tener la información organizada tiene ciertas ventajas, como el acceso, búsqueda y manejo de forma fácil y rápida.

También en la programación es importante decidir cómo organizar la información, seleccionando los tipos y estructuras de datos adecuados para almacenarlos y manipularlos en la computadora de manera eficiente. ¿Conoces los tipos de datos que existen?, ¿qué son los datos estructurados? o ¿cómo accedemos a los datos organizados en cierta estructura? En esta sección daremos respuesta a estas interrogantes.

Tipos de datos primitivos y estructurados

En los lenguajes de programación, se distinguen los tipos de datos básicos (primitivos) y los tipos de datos estructurados. Los tipos de datos primitivos definen la forma en que se almacena la información en la memoria de la computadora, almacenando un valor a la vez y los diferentes procesos que se pueden realizar sobre ella, cuando se requiere almacenar más de un valor, los lenguajes de programación proporcionan mecanismos para combinar los tipos de datos primitivos en estructuras más complejas llamadas tipos de datos estructurados (Figura 1.4) y podemos almacenar entonces más de un valor.

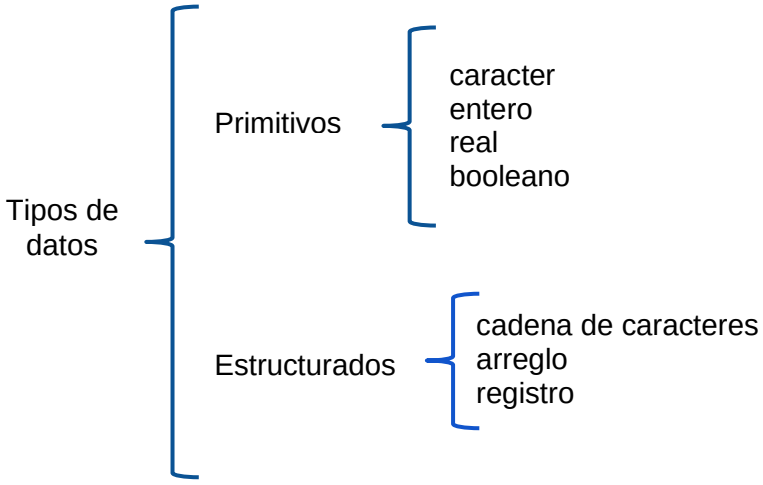


Figura 1.4. Tipos de datos

Es importante señalar que los lenguajes de programación organizan los datos en tipos diferentes según sus características, en lenguaje C, el booleano no es un tipo de dato primitivo.

Estructuras de datos

Una estructura de datos es un conjunto de variables asociadas bajo un mismo nombre, que se relacionan entre sí y que se manejan como una unidad. Para la definición de las estructuras de datos son esenciales los tipos de dato (Aho, HopCroft & Ullman, 1998).

El componente básico de una estructura de datos es la celda. La celda es un espacio de memoria que puede almacenar un valor tomado de algún tipo de dato primitivo. Cuando los datos se almacenan en estas celdas se conocen como datos estructurados como lo muestra la Figura 1.5.

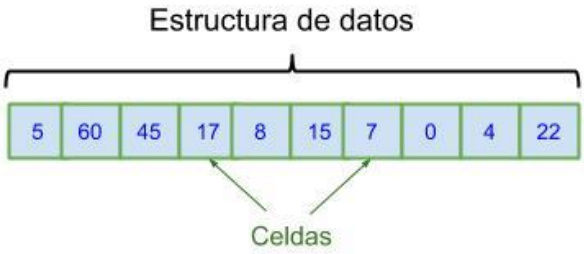


Figura 1.5. Datos estructurados y celdas

Arreglos

Un arreglo es una colección finita de celdas que nos sirve para almacenar datos homogéneos de forma temporal y ordenada. Sus características son:

- Finito. Tienen un límite, un número máximo de elementos que puede almacenar.
- Homogéneo. Todos los elementos del arreglo son del mismo tipo de datos.
- Ordenado. Se sabe cuál es el primero, el segundo y el n-ésimo elemento.

Elementos de un arreglo

Un arreglo tiene los siguientes elementos (Figura 1.6):

- **Nombre**. Es un identificador definido por el programador para poder hacer referencia al arreglo dentro de un programa.
- **Número de elementos**. Especifica cuántos elementos tiene el arreglo.
- **Índice**. Indica la posición de un elemento dentro del arreglo.

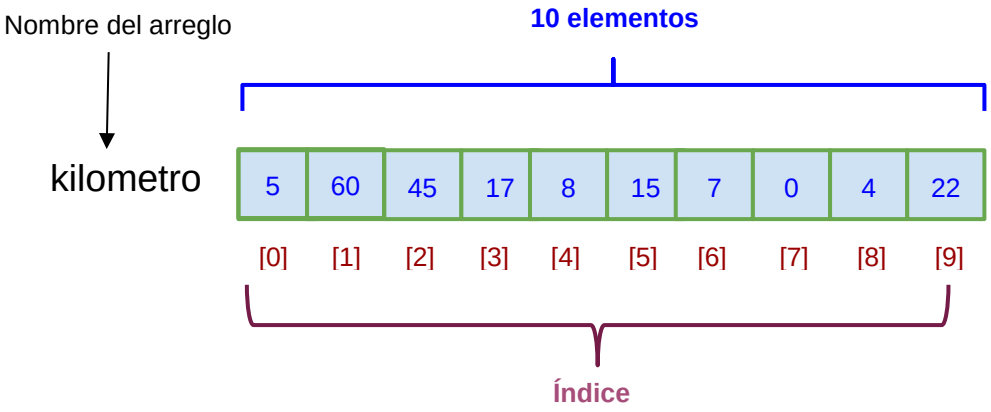


Figura 1.6. Elementos de un arreglo

Declaración de un arreglo

La sintaxis en lenguaje C para declarar un arreglo o array unidimensional es:

```
TipoDeDato IdentificadorArreglo [NúmeroDeElementos];
```

Por ejemplo, si deseamos construir un arreglo unidimensional que almacene los kilómetros que recorrió un automóvil en los 10 trayectos que hizo en el día, su declaración en lenguaje C sería como se muestra en la Figura 1.7.

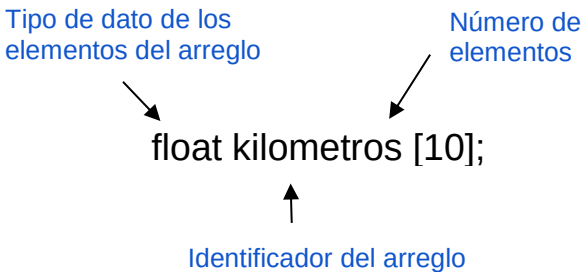


Figura 1.7. Declaración del arreglo kilómetros

Ejercicio 1.22. Escribe la declaración de los siguientes arreglos en lenguaje C:

Descripción de los datos	Declaración del arreglo en C
El kilometraje de 10 viajes	
15 números de motor de vehículos	
7 modelos de vehículos (año)	
Los litros de gasolina gastados en 5 viajes	

Referencia a un elemento de un arreglo

Para referenciar a los datos de un arreglo se debe especificar el nombre del arreglo y entre corchetes el índice de la celda a la que se quiere acceder como lo muestra la sintaxis:

IdentificadorArreglo [Índice];

Por ejemplo, observa el arreglo de la Figura 1.8, si necesitáramos recuperar el dato 2018, entonces su referencia sería:

modelo_auto[1];

Ejercicio 1.23. Realiza las referencias a los elementos del arreglo que se te indican:

modelo auto=

1990	2018	2000	1995	2015
[0]	[1]	[2]	[3]	[4]

Figura 1.8. Arreglo modelo_auto

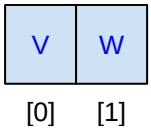
Referencia	Respuesta
Al primer elemento del arreglo	
Al tercer elemento del arreglo	
Al último elemento del arreglo	

Asignación de datos a un arreglo

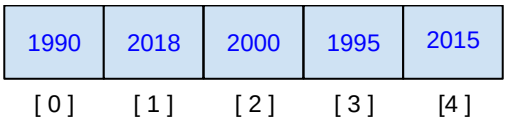
La asignación de valores a los elementos de un arreglo se hace de la misma forma que con una variable de tipo primitivo, es decir, empleando el operador "=".

En C, puedes asignar valores a un arreglo de distintas formas. La Tabla 1.10 muestra algunos ejemplos:

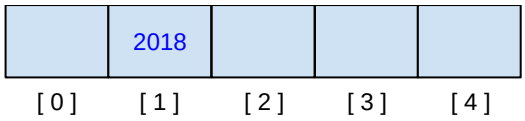
Programación en C	Explicación de la asignación
Asignación en la declaración float km [5]= {8.3, 6.7, 5.6, 4.8, 9.5}; km= 8.3 | 6.7 | 5.6 | 4.8 | 9.5 [0] [1] [2] [3] [4]	En este caso, se está definiendo un arreglo llamado *km* de 5 elementos que puede almacenar números reales. En la misma instrucción en que estamos definiendo el arreglo, se están asignando valores a cada uno de los elementos.
Asignación en la declaración char marca_auto [2] = {'V','W'};	Se declara el arreglo *marca_auto* de 2 elementos y se inicializan cada uno de sus elementos con los caracteres especificados. marca_auto[0] contiene 'V' y marca_auto[1] contiene 'W'

<table>
<tr>
<td>marca_auto= 
</td>
<td></td>
</tr>
<tr>
<td>Asignación a un elemento específico

int modelo_auto[5];
modelo_auto[0] = 1990;
modelo_auto[1] = 2018;
modelo_auto[2] = 2000;
modelo_auto[3] = 1995;
modelo_auto[4] = 2015;

modelo_auto= 
</td>
<td>Observa cómo a diferencia de los dos casos anteriores, aquí se define primero el arreglo modelo_auto de cinco elementos, y posteriormente se asignan valores a cada elemento particular.</td>
</tr>
<tr>
<td>Asignación de valores ingresados por el usuario mediante el teclado a los elementos de un arreglo

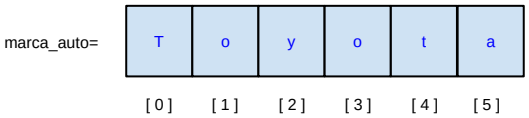
int modelo_auto[5];
scanf(“%d”, &modelo_auto[1]);

modelo auto= 
</td>
<td>En este ejemplo, primero se hace la definición del arreglo y después, se emplea la función scanf() para asignar el valor tecleado por el usuario en el elemento 1 del arreglo.

Supongamos que el usuario ingresó el modelo 2018, entonces, el contenido del arreglo sería como el de la imagen.</td>
</tr>
</table>

Tabla 1.10. Ejemplos de asignación de valores a un arreglo

Ejercicio 1.24. Asigna a los elementos de un arreglo los valores que se te indican:



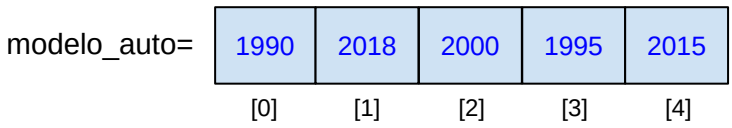
Asignación	Respuesta
En la declaración haciendo referencia a sus elementos	
En la declaración del arreglo	

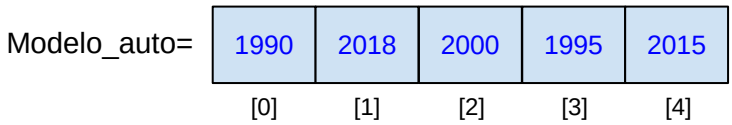
Escritura y lectura de datos en un arreglo

La lectura y escritura de los elementos de un arreglo se maneja de igual forma que una variable de tipo primitivo. Observa los siguientes segmentos de código que se utilizan para realizar operaciones de lectura o escritura de un elemento específico de un arreglo en el lenguaje de programación C estándar:

Escritura de datos en el arreglo	**Explicación**

modelo_auto[0] = 1990;
modelo_auto[1] = 2018;
modelo_auto[2] = 2000;
modelo_auto[3] = 1995;
modelo_auto[4] = 2015;
	Se asigna un valor a cada elemento del arreglo *modelo_auto* indicando el índice del elemento y con la utilización del signo igual = se asigna el valor.

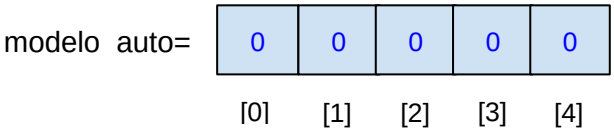
El resultado es:  	
Lectura de datos en el arreglo	**Explicación**
int fecha_auto; fecha_auto = modelo_auto[4];	Con la instrucción *fecha_auto = modelo_auto[4];* se hace la lectura del elemento 5 del arreglo y se asigna dicho valor a una variable

	tipo entero *fecha_auto,* así *fecha_auto y modelo_auto[4] t*ienen el mismo valor (2015).
El resultado es:  fecha_auto=2015	

También es posible realizar operaciones de lectura y escritura de un arreglo utilizando un ciclo de repetición (*for*, *while* o *do-while*) que ayuda a recorrer todos los elementos del arreglo. Por ejemplo, si quisiéramos inicializar con 0 todos los elementos del arreglo *modelo_auto*, el código sería el siguiente:

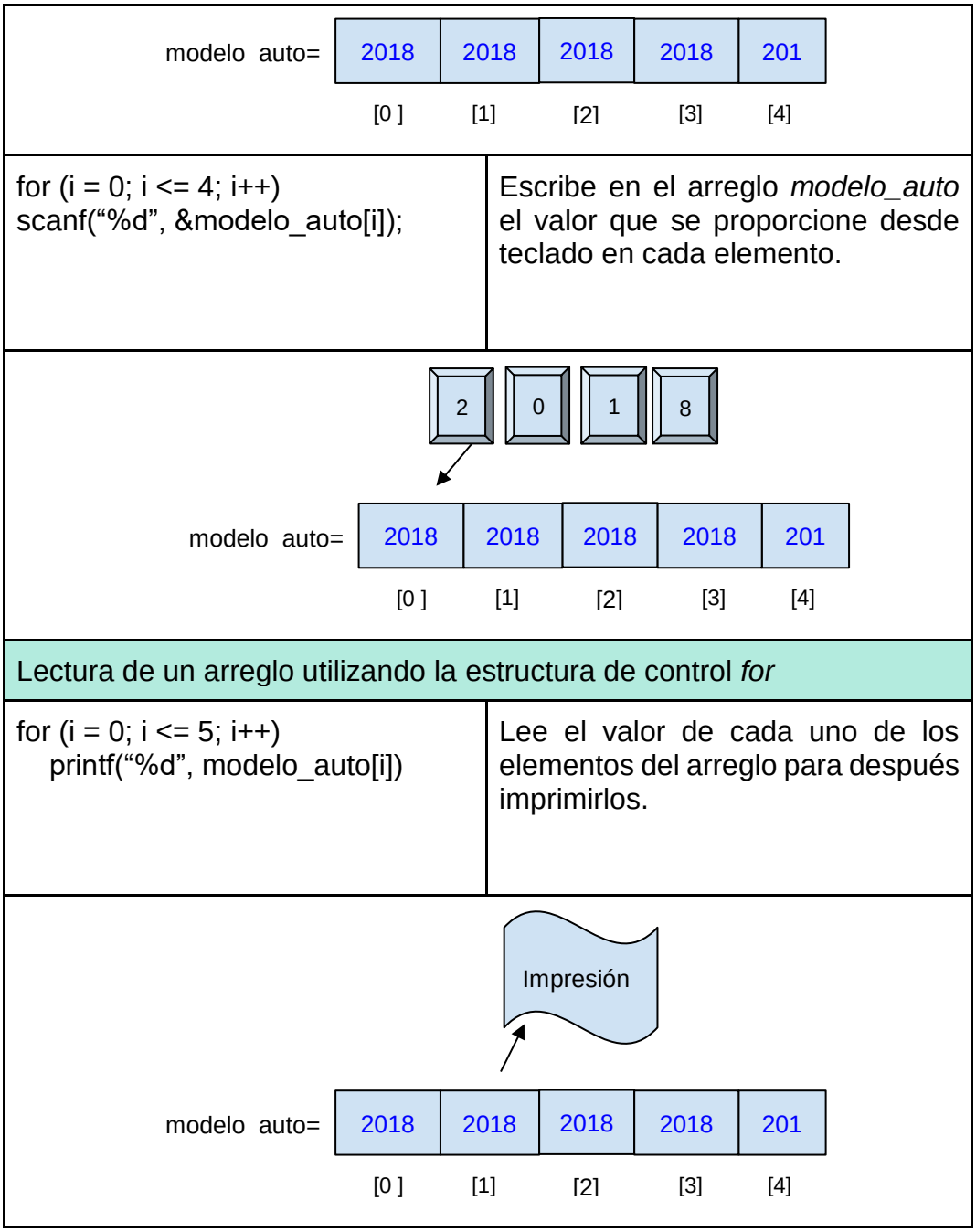
Ejemplo	Explicación

int modelo_auto[5];
int i;
for (I = 0; I <= 4; i++)
modelo_auto[i] = 0;
	Se declara el arreglo *modelo_auto* de 5 elementos.
	Se declara variable *i* como entero para controlar el índice.
	En el ciclo *for*, la *i* lleva el conteo del índice.
	Se asigna 0 a todos los elementos del arreglo *modelo_auto*.

La lectura y escritura de todos los datos de un arreglo empleando la estructura de control de iteración *for* quedaría de la siguiente forma:

Escritura de un arreglo utilizando la estructura de control *for*	
Ejemplo	Explicación
for(i = 0; i <= 4; i++) modelo_auto[i] = 2018;	Escribe en el arreglo *modelo_auto* el valor 2018 en cada elemento.



Programa 11. Asignación del holograma vehicular del Programa Hoy no Circula

El programa Hoy no Circula de la Ciudad de México contempla la revisión del nivel de emisión de contaminantes que produce un coche cuando circula. Este proceso se lleva a cabo en un verificentro donde un empleado coloca en el escape del auto un sensor que mide la cantidad de gases de efecto invernadero que produce el motor del auto por un período de tiempo. Con base en estos datos, se procede al

cálculo del promedio de emisiones contaminantes, y dependiendo de este valor se asigna un holograma vehicular que define los días que el auto podrá circular.

Diseña un programa en C que asigne el holograma correspondiente tomando en cuenta las siguientes consideraciones:

- El programa deberá mostrar al empleado del verificentro el siguiente menú:
  Programa Asignación de holograma vehicular
  1. Lectura de emisión de contaminantes
  2. Generación de comprobante de verificación
  3. Asignación del holograma vehicular
  4. Salir
- El sensor que registra la emisión de contaminantes realiza una lectura por segundo durante un período de 10 segundos.
- El comprobante de verificación vehicular debe incluir todas las lecturas realizadas por el sensor y su promedio.
- La asignación del holograma vehicular se realiza bajo los siguientes criterios:

Holograma vehicular	Lectura de emisión de contaminantes	
0	emisión<=250	 Imagen 3. Holograma vehicular (Verificación vehicular, s/f).
1	250<emisión<=700	
2	emisión>700	
Exento 00 Foráneo	No se consideran en el diseño del programa.	

Análisis del problema y definición de datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Rango de valores	Tipo de dato
Entradas				
lectura de emisión del sensor	emisiones	arreglo de 10 elementos	números reales	float
promedio de las	promedio	variable	números	float

10 emisiones			reales	
Salida				
Mensaje	promedio mensaje	variable	número real	float string
Número de holograma asignado	holograma	variable	número entero	int
Variables auxiliares				
opción del menú a ejecutar	opcion	variable	número entero	int
acumulador de las lecturas de emisiones	suma	variable	número entero	int
bandera para verificar que se haya realizado la lectura de datos	lectura	variable	número entero	int

El algoritmo sería el siguiente:

```
Algoritmo
Inicio
   lectura = 0
   Mientras (opcion != 4 ) haz
      Mostrar en pantalla al usuario menú de opciones
      Solicitar la opcion
      Almacenar la opción capturada por el usuario en la variable opcion
      En caso opcion
        1: Para (i = 0; i <= 9; i++)
              Obtener lectura del sensor
              Guardar valor en arreglo emisiones[i]
           Fin Para
           lectura =1
       2:  Si lectura == 1
           Entonces
                 Para   (I = 0; I <= 9; i++)
```

```
                    Mostrar en pantalla cada una de las emisiones
                    Calcular el promedio de las emisiones
                Fin Para
                Mostrar en pantalla promedio de las emisiones
            SiNo
                Mostrar en pantalla “No se ha realizado la lectura”
            FinSi
        3:  Si lectura == 1
            Si promedio <= 250 Entonces
              holograma = 0
            SiNo
                Si promedio >250 && promedio <=700  Entonces
                    holograma=1
                SiNo
                    Si promedio>700 Entonces
                        holograma=2
                    FinSi
                FinSi
            FinSi
        SiNo
                Mostrar en pantalla “No se ha realizado la lectura ”
        FinSi
        4: Mostrar al usuario el mensaje “Fin del programa”
        Cualquier otro caso: Mostrar mensaje “Error al capturar opcion”
    Fin En caso
  Fin Mientras
Fin
```

El código del programa en lenguaje C sería el siguiente:

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>

int main () {
float emisiones[10] = {0,0,0,0,0,0,0,0,0,0};
int i;
int holograma = 0;
int opcion = 0;
float suma = 0;
float promedio = 0;
char texto[20];
int lectura = 0;
while (opcion != 4)
{
   printf (" \n  Asignacion de holograma vehicular ");
```

```
printf ("\n 1. Lectura de emision de contaminantes");
printf ("\n 2. Generacion de comprobante de verificacion");
printf ("\n 3. Asignacion de holograma vehicular ");
printf ("\n 4. Salir");
printf ("\n ¿Que opcion desea? ");
scanf("%d", &opcion);
 /* Lectura de datos*/

 switch (opcion) {
 case 1: /* lectura de datos*/
         printf ("\n Lectura del sensor de emisiones \n");
         printf ("Proporciona la lectura del sensor de emisiones \n");
         for (i = 0; i <= 9; i++)
               {
                    printf ("Lectura No.%d  del sensor de emisiones: ",i + 1 );
                    scanf("%f", &emisiones[i]);
               }
          printf("\nFin de la lectura\n\n");
          lectura = 1;
         break;
  case 2:  /* Cálculo del promedio*/
         if (lectura == 1)
         {       printf ("\n\tComprobante de verificacion vehicular" );
               for (I = 0; I <= 9; i++)
               {
                        printf ("\n\tLectura del sensor No.%d: %5.2f ",i+1,emisiones[i] );
                         suma = suma + emisiones[i];
               }
               promedio = (suma / 10);
               printf ("\nEl promedio de emisiones contaminantes es: %5.2f
                    \n",promedio);
               }
         else
               printf("\nNo se ha realizado la lectura\n\n");
         break;
 case 3: /* Asignación del holograma*/
        if (lectura == 1) {
               if (promedio <= 250)
               {
                    holograma = 0;
                    strcpy(texto,"Color amarillo \n");
               }
               else {
                     if (promedio > 250 && promedio <= 700)
                    {
                        holograma=1;
                          strcpy(texto," Color naranja\n");
```

```
                    }
                    else {
                            if  (promedio>700 )
                            {    holograma=2;
                                 strcpy(texto," Color rojo");
                            }
                    }
                }
                printf ("\n El holograma correspondiente es: %d %s",holograma,  texto);
            }
          else
           printf ("\nNo se ha realizado la lectura\n\n");
           break;
   case 4: /* salida*/
            printf(" Fin del programa\n");
            break;
   default:  printf("\n Error en la opcion\n\n");
            break;
   } /* fin switch*/
 } /* fin while*/
 getch ();
 return 0;
}
```

Realicemos la prueba de escritorio:

Prueba de escritorio del programa 11		
Para opcion = 1		
i	emisiones[10]={0,0,0,0,0,0,0,0,0,0}	lectura
0 1 2 3 4 5 6 7 8	emisiones{250.5,0,0,0,0,0,0,0,0,0} emisiones{250.5,251.5,0,0,0,0,0,0,0,0} emisiones{250.5,251.5,252.08,0,0,0,0,0,0,0} emisiones{250.5,251.5,252.08,251.0,0,0,0,0,0,0} emisiones{250.5,251.5,252.08,251.0,252.10,0,0,0,0,0} emisiones{250.5,251.5,252.08,251.0,252.10,253.5,0,0,0,0 } emisiones{250.5,251.5,252.08,251.0,252.10,253.5,254.0, 0,0} emisiones{250.5,251.5,252.08,251.0,252.10,253.5,254.0, 255.0,0, emisiones{250.5,251.5,252.08,251.0,252.10,253.5,254.0,	0

	255.0, 254.8, 0}	
9	emisiones{250.5,251.5,252.08,251.0,252.10,253.5,254.0, 255.0, 254.8, 255.18}	1

Para opcion =2 lectura = 1			
i	emisiones [i]	Suma	promedio
0	250.5	250.5	0
1	251.5	502	0
2	252.08	754.08	0
3	251.0	1005.08	0
4	252.10	1257.18	0
5	253.5	1520.68	0
6	254.0	1764.68	0
7	255.0	2019.68	0
8	254.8	2274.48	0
9	255.18	2529.66	252.97

Para opcion = 3					
lectura==1					
Resultado de la expresión	expresión de relación	Resultado de expresión	Promedio	Holograma	Texto
1	252.97< 250	0	252.97	0	
	252.97>250 && 252.97<=700	1	252.97	1	Color naranja
	252.97>700	0	252.97	0	

Lectura ==0
“No se ha realizado la lectura”

La Ilustración 1.28 muestra la ejecución del ejemplo con los datos utilizados en la prueba de escritorio:

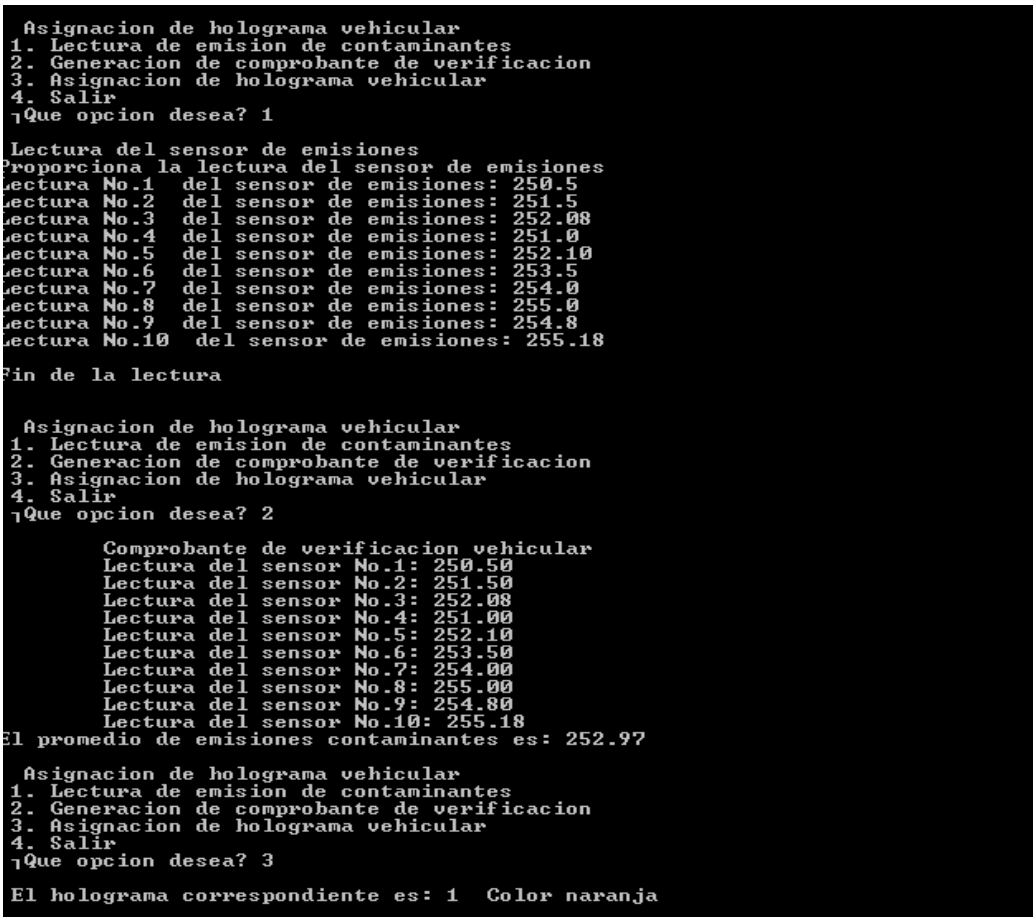


Ilustración 1.28 Ejecución del programa 11 Asignación de holograma vehicular

Ejercicio 1.25. Programa Registro de viajes

Un automóvil realiza en un día 10 viajes, el kilometraje de cada viaje es guardado y al final del día, la computadora del automóvil despliega el total de kilómetros recorridos y pregunta al usuario si desea el listado de los kilómetros de cada viaje.
Si el usuario selecciona la opción de listado, despliega en la pantalla cada viaje y los kilómetros recorridos. Diseña el programa que resuelva este problema.

Arreglos bidimensionales

Los arreglos pueden tener más de una dimensión (multidimensionales) y por lo tanto, más de un índice, los más usuales son los de dos dimensiones y se les conoce como arreglos bidimensionales.

Un arreglo de dos dimensiones equivale a una tabla compuesta por filas y columnas como lo muestra la Figura 1.9.

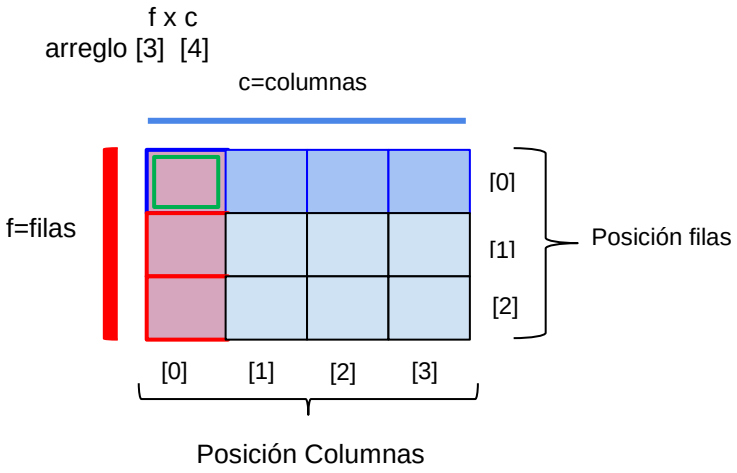


Figura 1.9. Arreglo bidimensional

Declaración de un arreglo bidimensional

La sintaxis en lenguaje C para la declaración de un arreglo bidimensional es:

```
TipoDeDato IdentificadorArreglo[NúmeroDeFilas][NúmeroDeColumnas];
```

Por ejemplo, si deseamos construir un arreglo bidimensional para almacenar los kilómetros que recorre un automóvil en cada uno de los 10 viajes que realiza en un día, en el lapso de una semana, su declaración en lenguaje C sería como se muestra en la Figura 1.10.

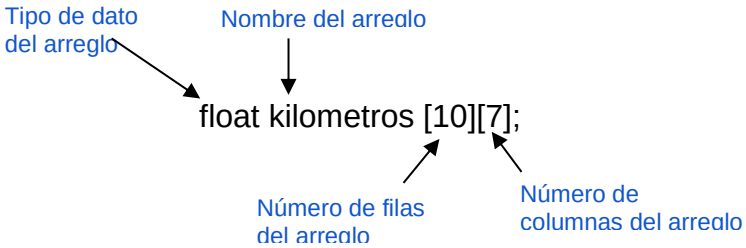


Figura 1.10. Declaración del arreglo bidimensional kilómetros

En donde las filas corresponden a cada uno de los 10 viajes y las columnas a cada día de la semana.

Ejercicio 1.26. Realiza la declaración de los arreglos bidimensionales requeridos para solucionar las siguientes problemáticas:

a) En una fábrica de refacciones, los productos que ya están listos para su distribución se encuentran almacenadas en una bodega. La bodega tiene 6 anaqueles con 3 repisas cada uno. Para identificar las refacciones en la bodega se les asigna un número de inventario (un número de 3 dígitos). Para tener guardados los números de las refacciones, ¿cómo se debe declarar el arreglo?
b) Una empresa automotriz lleva con un programa el orden y el registro de los montos de ventas por mes de 5 sucursales, ¿cómo implemento en un programa de C la estructura para almacenar los montos de ventas?

Referencia a elementos de un arreglo bidimensional

Para localizar un elemento de un arreglo bidimensional se debe especificar la coordenada representada por el número de fila y el número de columna, por lo que para hacer la referencia se utiliza el nombre del arreglo, el índice de la fila entre corchetes y el índice de la columna entre corchetes de la celda a la que se quiere acceder, como lo muestra la sintaxis:

```
IdentificadorArreglo [ÍndiceFila][ÍndiceColumna];
```

Por ejemplo, observa el arreglo de la Figura 1.11, si necesitáramos recuperar el dato 8, entonces su referencia sería:

autos_ventas[1][2];

Ejercicio 1.27. Realiza las referencias a los valores del arreglo que se te indican:

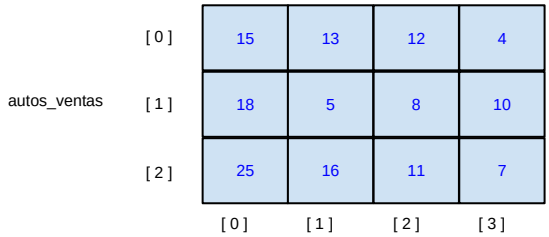


Figura 1.11 Arreglo bidimensional

Referencia	Respuesta
Al valor 12	
Al valor 18	
Al valor 7	
Al valor 16	
Al valor 10	

Asignación de datos a un arreglo bidimensional

A los arreglos bidimensionales se les puede asignar datos de la misma forma que los de una dimensión, con la diferencia de que se deben especificar los índices de la fila y la columna.

Veamos algunos ejemplos:

Suponga que se desea crear el arreglo bidimensional que se muestra en la Figura 1.12 y asignarle los siguientes valores:

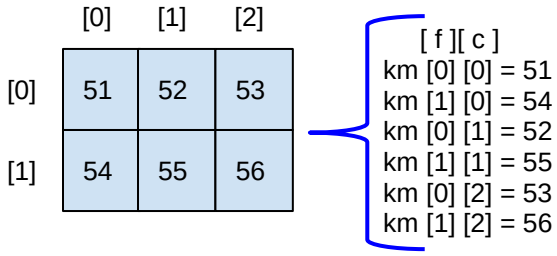


Figura 1.12. Arreglo km

Inicialización en la declaración	La asignación se realiza de la siguiente forma
La inicialización consta de una serie de valores separados por comas y encerradas entre llaves	
int km [2] [3] = {51, 52, 53, 54, 55, 56};	Se asignan los valores indicando en una sola línea de código todos los valores de la primera fila, enseguida los de la segunda fila y así sucesivamente.
int km [2] [3]= { {51, 52, 53} , {54, 55, 56} };	Se asignan los valores indicando entre llaves los valores de cada fila.
int km [2] [3]= { {51, 52, 53} , {54, 55, 56} };	La forma más amigable es indicando las filas en renglones diferentes para visualizar cada fila del arreglo.
Asignación a cada uno de los elementos del arreglo	

int km [2][3];
km [0] [0] = 51;
km [1] [0] = 54;
km [0] [1] = 52;
km [1] [1] = 55;
km [0] [2] = 53;
km [1] [2] = 56;	

Ejercicio 1.28. Tomando como base el arreglo bidimensional del ejercicio 27, realiza la asignación de los valores del arreglo bidimensional como se pide:

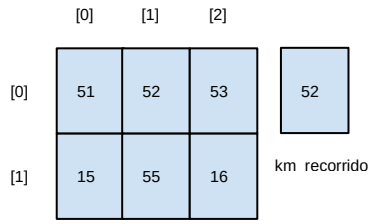
Asignación en la declaración	Respuesta
Describiendo todos los elementos del arreglo en una línea de código	
Con filas en renglones diferentes	

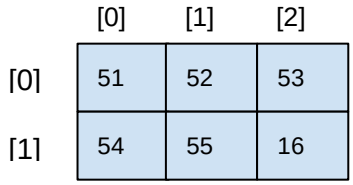
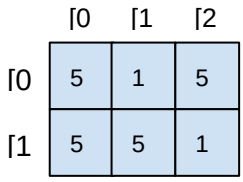
Lectura y escritura de datos en un arreglo bidimensional

Las operaciones de lectura y escritura de un arreglo bidimensional se aplican de igual forma que un arreglo unidimensional, cuidando indicar siempre la fila y la columna [f] [c] del elemento deseado.

A continuación, se muestran algunos ejemplos de estas operaciones sobre el arreglo *km* que tiene la siguiente estructura:

```
int km [2] [3]= { {51, 52, 53} ,
                  {54, 55, 56} };
```

Lectura de datos	**Descripción**
int km_recorrido; km_recorrido = km[0][1] 	Con la instrucción *km_recorrido = km[0][1];* se hace la lectura de un valor del arreglo km (52) y se asigna a la variable tipo entero km_recorrido, así *km_recorrido y km[0][1]* tienen el mismo valor (52)
printf (“El valor es: %d”, km[0][1]);	Con la instrucción printf() escribimos el elemento que se

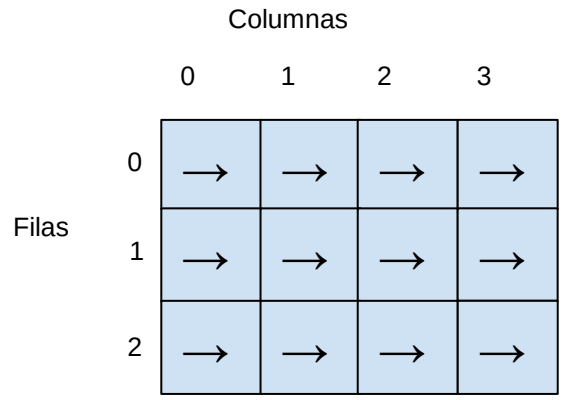
	encuentra en la fila 0 columna 1, resultando: “El valor es 52”.
Escritura de datos	**Descripción**
Por medio del teclado: printf (“Dame el valor: ”) scanf(“%d”, &km[1][2]);  	Con la función *scanf()* se hace la lectura desde el teclado de un valor que se almacena en el elemento ubicado en la fila 1, columna 2 del arreglo al Si desde el teclado se proporciona el valor 16, el elemento km [1][2] contiene 16.
Por una expresión de asignación: km [0][1]= 15;  	Se asigna el valor 15 en el elemento ubicado en la fila 0, columna 1

Recorrido de un arreglo bidimensional

En los arreglos bidimensionales las operaciones de lectura y escritura a todos los elementos del arreglo se puede hacer realizando un recorrido por renglones o por columnas. Para recorrer el arreglo se utiliza la técnica de ciclos anidados. Se debe utilizar un ciclo para manipular las filas y otro para las columnas.

Recorrido por filas

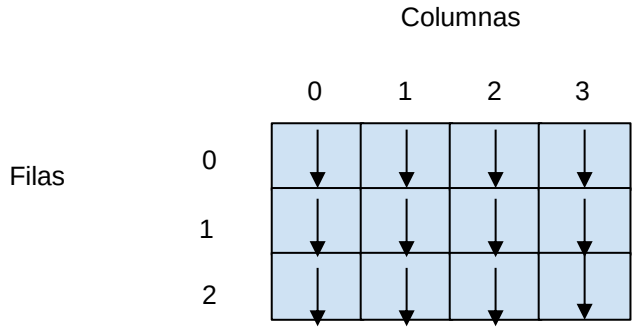
En el recorrido por filas el bucle externo es para el acceso a las filas, y el ciclo interno para las columnas:




```
para todas las filas
        para todas las columnas
                arreglo[fila][ columna]
```

Recorrido por columnas

En el recorrido por columnas el ciclo externo es para el acceso a las columnas, y el ciclo interno para las filas:




```
para todas las columnas
        para todas las filas
                arreglo [fila][ columna]
```

Empleando la estructura de control for, la sintaxis en lenguaje C del recorrido por filas sería la siguiente:

```
// Recorrido por filas
int fila;  /* Para llevar el control del índice de la fila */
int columna;/* Para llevar el control del índice de las columnas*/
for (fila = 0; fila < TotalFilas; fila++)
    for (columna = 0; columna < TotalColumna; columna++)
                arreglo [fila][columna];
```

Y la sintaxis en lenguaje C del recorrido por filas empleando la misma estructura de control sería:

```
// Recorrido por columnas
int fila;  /* Para llevar el control del índice de la fila */
int columna;   /* Para llevar el control del índice de las columnas*/
for (columna = 0; columna < TotalColumna; columna++)
    for (fila = 0; fila < TotalFilas; fila++)
                arreglo [fila][columna];
```

Programa 12. Kilómetros recorridos por semana

La computadora de viaje de un automóvil almacena los recorridos realizados en el día. Supongamos que un transporte escolar realiza 5 viajes al día. Su conductor desea conocer algunas estadísticas de uso semanal. Diseña un programa en C que reporte la siguiente información:

- Total de kilómetros recorridos por día.
- Total de kilómetros recorridos en la semana.
- Día en que se realizó el mayor recorrido.

Identificación de datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Rango de valores	Tipo de dato
Entrada				
kilómetros recorridos por viaje	kilometros	arreglo bidimensional de [5][7]	números reales	float
día de la semana	dia	variable	[0,6] 0-domingo 1-lunes … 6-sábado	int
número de viaje del día	viaje	variable	[0,5]	int
Salida				
Día que más	max_dia	variable	número	entero

kilómetros recorrió el vehículo			entero	
Acumulador de kilómetros recorridos por día	km_dia	variable	número real	float
Auxiliar para saber el número máximo de kilómetros recorridos en un día	max_km	variable	número real	float
Total de kilómetros recorridos en la semana	total_semana	variable	número real	float

El algoritmo para resolver el problema sería:

```
Algoritmo
Inicio
  dia=0
  viaje=0
  max_dia=0
  max_km=0
  km_dia=0
  kilometros[5][7]
  total_semana=0
 opcion=0
 Mientras (opcion <>3) haz
   Mostrar en pantalla al usuario menú de opciones
   Solicitar opción
   Almacenar la opción capturada por el usuario en la variable opcion
   En caso opcion
       1:
            Para dia=0; dia <7; dia++
                 Para viaje =0; viaje<5; viaje++
                      Obtener kilometros del viaje/dia
                      Almacenar valor en arreglo kilometros [viaje][dia]
                 FinPara
            FinPara
       2:
```

```
                max_dia=0
                Para dia=0; dia<7; dia++
                        km_dia=0
                        Para viaje =0; viaje<5; viaje ++
                                km_dia=km_dia+kilometros[viaje][dia]
                        FinPara
                        Mostrar en pantalla km_dia
                        Si km_dia>max_km entonces
                                max_km=km_dia
                                max_dia=dia
                        FinSi
                        total_semana=total_semana+km_dia
                FinPara
                Mostrar en pantalla max_dia y max_km
                Mostrar en pantalla total_semana
        3:
                Mostrar en pantalla el mensaje “Salir”
        FinCaso
  FinMientras
Fin
```

El código en lenguaje C del programa 12 sería:

```c
#include<stdio.h>
#include<conio.h>
int main ()
{
   int dia=0;
   int viaje=0;
   int max_dia=0;
   float max_km=0;
   float km_dia=0;
   float kilometros[5][7]={{0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                           {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                           {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                           {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                           {0.0,0.0,0.0,0.0,0.0,0.0,0.0}};
   float total_semana;
   int opcion=0;
   while (opcion != 3)
  {
    printf (" \n  Bitacora de viajes \n ");
    printf ("\n 1. Lectura de kilometros/semana");
    printf ("\n 2. Reporte semanal");
    printf ("\n 3. Salir");
    printf ("\n ¿Que opcion desea? ");
    scanf("%d",&opcion);
```

```
   /* Lectura de datos*/

         switch (opcion){
         case 1: /* lectura de datos*/
         printf ("Bitácora de viaje\n\n");
       for (dia =0; dia<7; dia++)
       {
                  for (viaje=0; viaje<5; viaje++)
                              {
                              printf ("Kilómetros recorridos,  Dia No. %d viaje No. %d :", dia,
viaje+1);
                              scanf ("%f", &kilometros[viaje][dia]);
                              }
       }
   break;
   case 2:/* cálculo e impresión de datos*/
       total_semana=0;
       max_dia=0;
       for (dia=0; dia<7; dia++)
       {
           km_dia=0;
                     for (viaje=0; viaje<5; viaje++)
                              {
                                        km_dia=km_dia+kilometros [viaje][dia];
                              }
                    printf("\n Total kilometros dia No. %d: %5.2f", dia, km_dia);
                    if (km_dia >= max_km)
                  {
                    max_km=km_dia;
                max_dia=dia;
                    }
                    total_semana=total_semana+km_dia;
       }
         printf("\nDia con mas kilometros ( %d ) con %5.2f kilometros"
,max_dia,max_km);
         printf("\ntotal  kilometros a la semana %.2f",total_semana);
         break;
   case 3:
         printf ("\nFin del programa");
         break;
         default: printf ("\nError en opcion\n\n");
   } /* fin switch*/
}/* fin while*/
         getch();
         return 0;
}
```

Realicemos la prueba de escritorio para el caso 2 (opcion 2 del menú), con los valores siguientes proporcionados desde teclado:

Prueba de escritorio del programa 12 Caso 1							
	kilometros[5][7]=						
viaje	dia 0	dia 1	dia 2	dia 3	dia 4	dia 5	dia 6
0	3.5,	15.6,	9.5,	2.5,	9.20,	4.6,	11.3
1	4.6,	5.6,	4.7,	11.4,	2.6,	2.5,	9.2
2	2.4,	6.8,	7.3,	6.8,	6.8,	7.8,	3.5
3	5.7,	3.6,	3.8,	9.3,	9.3,	9.3,	9.2
4	10.8,	8.2,	8.2,	7.8,	7.8,	10.5,	3.5

Caso 2					
dia	km_dia	max_km	km_dia >= max_km	total_semana	max_dia
	0				
0	27.00	27.0	1	27.0	0
1	39.80	27.0	1	66.8	1
2	33.50	39.80	0	100.3	1
3	37.80	39.80	0	138.1	1
4	35.70	39.80	0	173.8	1
5	34.70	39.80	0	208.5	1
6	36.70	39.80	0	245.2	1

La ilustración 1.29 muestra el resultado de la ejecución del programa.

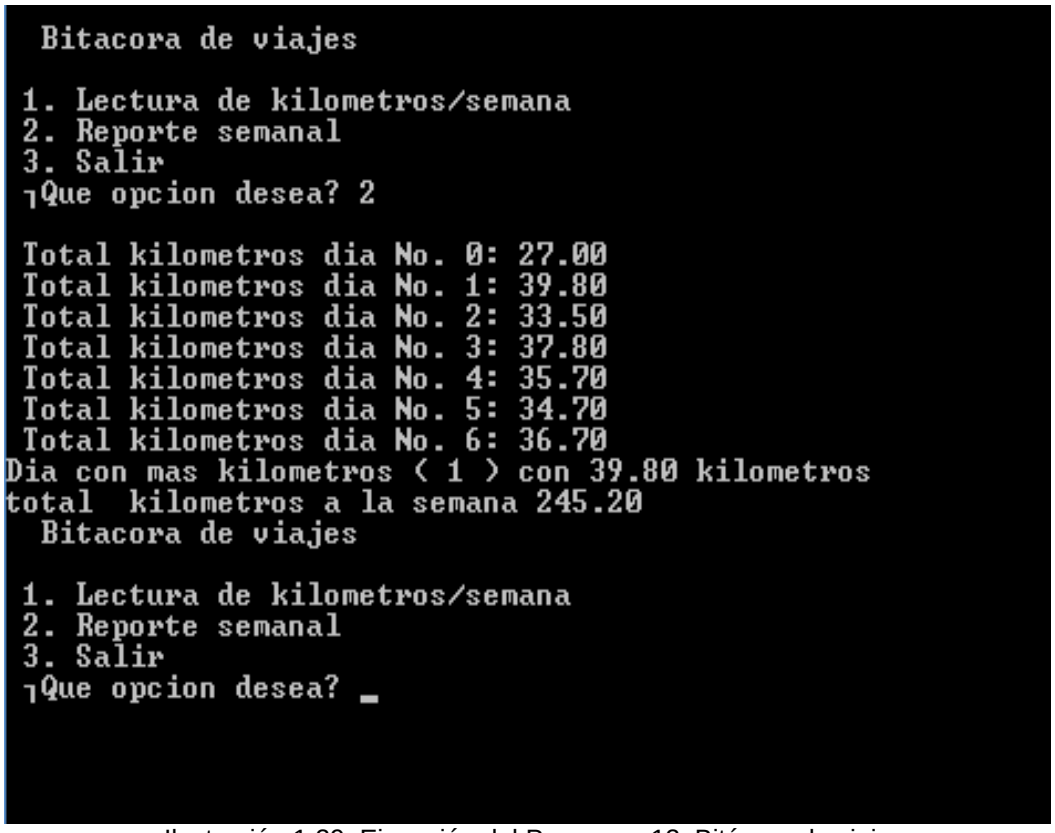


Ilustración 1.29. Ejecución del Programa 12. Bitácora de viaje

Ejercicio 1.29. Programa Refacciones

Una agencia guarda sus refacciones en un almacén que está integrado por varios anaqueles. Cuando se localiza una refacción en el almacén, se identifica primero el anaquel y posteriormente el entrepaño en el que se encuentra. ¿Cómo implementarías la organización y localización de las refacciones utilizando arreglos?

Supongamos que el almacén tiene cuatro anaqueles con dos entrepaños cada uno. Por tanto, se tendría que crear un arreglo bidimensional que representara el almacén. Las columnas serían los anaqueles y las filas los entrepaños (Figura 1.13).

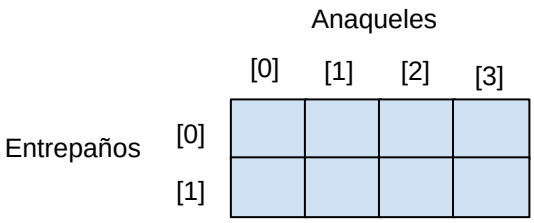


Figura 1.13. Representación del almacén en un arreglo bidimensional.

La agencia desea colocar el precio a cada una de las refacciones de su almacén. Diseña un programa en C que solicite al usuario el precio de la refacción según su ubicación y muestre en pantalla un listado con los precios de todas sus refacciones.

Preguntas de reflexión

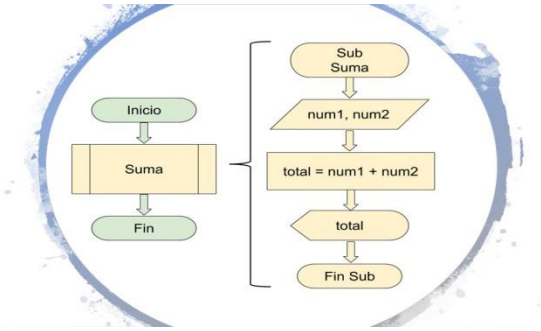
Caso 1. En la escuela te solicitaron realizar un proyecto y tienes que programar un sistema con uso de sensores conectados a una tarjeta electrónica. El profesor te solicita que el proyecto sea innovador. El trabajo es por equipo y cada uno da una opción para realizarlo:

- Investigar la aplicación de los sensores y crear el proyecto de una forma atractiva y novedosa.
- Buscar un proyecto ya realizado en Internet, tomar la idea, adecuarlo a lo que ustedes necesitan y construir el proyecto.
- Pedirle a un amigo que ya está en la facultad que les haga el proyecto.
- Copiar tal cual está un proyecto que encontraron en Internet.

1. ¿Qué decisión toman para realizar el proyecto? Si elegiste la opción 4, ¿qué repercusiones tiene para tu aprendizaje? Justifica tu respuesta.

2. Uno de los integrantes del equipo siempre está en desacuerdo ¿Qué harías para llegar a un acuerdo?

Caso 2. Formas parte de un equipo de trabajo que tiene como proyecto desarrollar el sistema que revisa que los sensores del auto trabajen adecuadamente. Para diseñarlo hay que considerar las normas internacionales, pero si lo hacen repercute en el rendimiento del coche. En reunión con tu jefe y los demás desarrolladores deberán tomar la decisión a seguir. Hay dos posturas: 1) Que se utilicen las normas internacionales y 2) Modificar las normas internacionales. ¿Qué postura apoyarías y por qué?

Contenidos conceptuales	Contenidos procedimentales
1.4 Subprogramas como abstracción de operaciones: procedimientos y funciones con paso de parámetros por valor y por referencia. 	1.5 Diseño de programas aplicando la metodología de solución de problemas con ejemplos de automatización en la ciencia y la industria. **Contenidos actitudinales** 1.6 Disposición a la colaboración, cooperación y al trabajo en equipos interdisciplinarios. 1.7 Valoración de la creatividad, empatía, perseverancia, disciplina, innovación e inventiva. 1.8 Capacidad crítica y autocrítica en la toma de decisiones y solución de problemas, mediación de conflictos.

1.1 Subprogramas como abstracción de operaciones: procedimientos y funciones con paso de parámetros por valor y por referencia

Para el desarrollo de este tema, vamos a retomar el concepto de abstracción en la programación. Recordemos que su fin es separar los detalles de un código para que el programador solamente se enfoque en cómo utilizarlo y se le facilite el trabajo para elaborar nuevos programas. Recordemos el ejemplo de la analogía del aprendiz de conductor de un automóvil, donde éste utiliza e interactúa tan solo con ciertos elementos del coche como el volante, los pedales y la palanca de velocidades, sin conocer a detalle cada uno de los elementos mecánico-eléctricos que lo componen. Así mismo, se pensó en una forma en la que los programadores pudieran encapsular (separar) código, como si fuera una caja negra, para concentrar la atención en cómo utilizarlo, sin necesidad de entrar a detalle sobre cómo fue realizado. Fue entonces que se creó el concepto de la subrutina. Cabe señalar que en varios libros de lenguaje C utilizan el concepto de módulo, en lugar de subrutina.

Subrutinas

La programación modular se basa en que es más fácil escribir programas si se divide en pequeñas partes llamadas subrutinas. La subrutina es una secuencia finita de instrucciones, separada del código principal del programa e identificada bajo un nombre, que resuelve una tarea específica.

Por lo anterior, generalmente al diseñar la solución a un problema se codifica una función principal, la cual llama o invoca a diferentes subrutinas que resuelven partes específicas del problema general. Cuando se ejecuta el llamado a una subrutina, el control se transfiere a su definición, la subrutina realiza las acciones

para la que fue programada, y cuando termina, el control es devuelto al punto del programa de donde se llamó originalmente a la función.

Este tipo de programación aporta las ventajas de poder reutilizar código, encontrar más fácilmente los errores de un programa, llamar a todo un código con solo mencionar el nombre de la subrutina, así como dar un orden y estructura a los programas.

En el lenguaje C, se llevan a cabo tres acciones para la implementación de las subrutinas tal como muestra en la Ilustración 1.30:

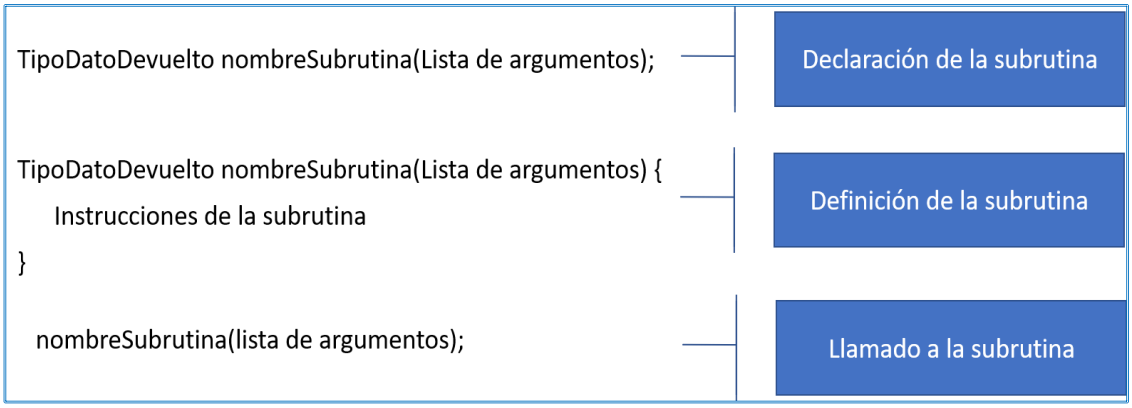


Ilustración 1.30. Declaración de una subrutina en lenguaje C

El uso de las subrutinas no es nuevo para tí, ya que has venido utilizando algunas de ellas en los ejercicios que has realizado, por ejemplo, has estado definiendo una subrutina muy importante denominada *main()*, siendo ésta la que nunca debe faltar en los programas de lenguaje C, porque es la primera que se ejecuta después de compilado el código fuente. También has estado llamando a subrutinas que tú no definiste, sino que ya estaban definidas dentro de bibliotecas estándares de C, como por ejemplo, la función *printf()*.

Procedimientos

El procedimiento es un tipo de subrutina que no regresa ningún valor.

A pesar de que en el lenguaje C no existe el concepto de procedimiento, en la presente guía se mostrará el manejo de una subrutina que funciona de manera similar al procedimiento. Lo anterior se debe a que en otros lenguajes de programación sí se contempla el uso de procedimientos, por lo que es importante que tengas conocimiento de este tipo de subrutina.

Por lo general, en los demás lenguajes de programación existe una palabra reservada destinada para indicar que se trata de un procedimiento, en lenguaje C, lo que se debe hacer es siempre anteponer la palabra *void* al nombre de la subrutina. Cabe señalar, que la palabra *void* es un tipo de datos que significa vacío o nada. De esta forma, se define que la subrutina no devuelve ningún valor que

pertenezca a los tipos de datos primitivos definidos en el lenguaje de programación, por ello, se le puede considerar un procedimiento. Su sintaxis se muestra a continuación:

```
void NombreDelProcedimiento(lista de argumentos)
{
    <En este espacio se escribirá el código del procedimiento>
}
```

Donde:

- *NombreDelProcedimiento*. Es el identificador asignado al procedimiento.
- *Lista de argumentos.* Son los datos que el programa que invoca al procedimiento envía al mismo.
- *Código del procedimiento.* Es el conjunto de instrucciones que llevan a cabo la tarea específica para la cual ha sido creado el procedimiento.

La llamada o ejecución de un procedimiento se realiza indicando su nombre, seguido de la lista de argumentos encerrados entre paréntesis (si es que los tiene) y termina con el signo punto y coma, como se muestra a continuación:

```
NombreDelProcedimiento(lista de argumentos);
```

Programa 13. Cálculo de los kilómetros recorridos y litros consumidos a través del uso de procedimientos

Para ejemplificar el uso de procedimientos, vamos a retomar el *programa 3 Cálculo de los kilómetros recorridos y litros consumidos*. Cabe señalar que el análisis del problema se encuentra en las páginas 32 a 34, así como la salida del programa se encuentra en la ilustración 1.13. Lo anterior con la idea que puedas cotejar la diferencia que hay únicamente en el código de un programa hecho con un procedimiento y otro sin él.

```
/* Ejemplo 13
Programa que calcula los kilómetros recorridos y los litros consumidos por un auto.
La solución se realiza  aplicando el concepto de procedimiento
*/
#include <stdio.h>
#define RENDIMIENTO 5.8/100

void rendim_combus(); //Declaración del procedimiento.
```

```
int main() { //Declaración del programa principal
        rendim_combus(); //Se llama o ejecuta al procedimiento
        getch( );
        return 0;
}

//Definición procedimiento
void rendim_combus() //Cabecera de la declaración del procedimiento
{                                   //Cuerpo del procedimiento
   float km_inicial;
   float km_final;
   float km_viaje;
   float l_viaje;
   printf("Programa que calcula los kilometros recorridos y litros consumidos por un
auto\n");
   printf("Capture el kilometro inicial: ");
   scanf("%f", &km_inicial);
   printf("\nCapture el kilometro final: ");
   scanf("%f", &km_final);
   km_viaje = km_final - km_inicial;
   l_viaje = km_viaje * RENDIMIENTO;
   printf("\nKm recorridos: %2.2f \tlitros consumidos: %2.2f", km_viaje, l_viaje);
} //Fin de la definición del procedimiento
```

La subrutina *main()* puede ser considerada como un procedimiento cuando lleva la palabra *void* al principio de su declaración, ya que ésta no va a regresar un valor.

Funciones

La función es una subrutina muy similar al procedimiento con la diferencia de que ésta sí devuelve un valor, el cual será del tipo de dato que se haya definido al crear la función. Por lo tanto, no debe codificarse la palabra *void* al principio, sino que se deberá escribir el tipo de dato que va a regresar. Dentro del cuerpo de la función siempre se debe agregar la instrucción *return* seguida del valor que se devuelve al punto del programa donde se hizo el llamado a la función.

La sintaxis para declarar una función en C es:

```
<TipoDatoDevuelto> NombreDeLaFunción(lista de argumentos)
{
        <En este espacio se escribirá el código de la función>
        return <ValorDevuelto>;
}
```

Donde:

- *TipoDatoDevuelto.* Es el tipo del valor devuelto por la función.
- *NombreDeLaFunción*. Es el nombre o identificador asignado a la función.
- *Lista de argumentos*. Son los valores de entrada que utiliza la función.
- *Código de la función*. Es el conjunto de sentencias que lleva a cabo la tarea específica para la cual ha sido creada la función.
- *ValorDevuelto.* Mediante la palabra reservada *return*, se devuelve este valor al punto del programa donde se llamó a la función.

La función puede (y es recomendable) utilizarse en expresiones de asignación, relacionales o aritméticas, teniendo cuidado que el tipo de dato del valor devuelto por la función corresponda con el de la variable que recibe el valor. Veamos el siguiente ejemplo:

```
#include <stdio.h>
#include <stdlib.h>
//Declaración de la función
int numero_entero();

//Definición de la función
int num_entero() {          //Cabecera de la función
        return rand();      //Cuerpo de la función
}
int main() {
        int numero;
        numero = num_entero();    //Llamado de la función
        printf("Número aleatorio generado: %d", numero);
}
```

En este programa primero se declara la función *num_entero* que devolverá un valor entero. Después, se hace la definición de esta función cuyo objetivo es generar un número aleatorio, lo cual se logra mediante la función *rand*. Observa que esta función no está declarada dentro del programa como se hizo con *num_entero*, su declaración está en la biblioteca estándar de C *stdlib.h* y el programa simplemente la está invocando. En este caso, no conocemos el tipo de dato del valor devuelto por esta función, pero lo podemos consultar en cualquier manual de referencia del lenguaje de programación C. De ese modo, sabemos que la función *rand* devuelve un entero. Este valor entero es a su vez devuelto por la función *numero_entero* al punto del programa donde se hace el llamado a la misma.

El llamado de la función se hace dentro de la función *main()*, en la expresión de asignación *numero = num_entero();* donde se asigna el resultado de la función *num_entero()* que como se explicó en el párrafo anterior, devuelve un valor de tipo entero, a la variable *numero,* que es del mismo tipo.

En el caso de la subrutina *main()* definida en este programa, se puede considerar una función porque cumple la condición de definir el tipo de datos de su

valor devuelto diferente de void, en este caso, como regularmente se suele escribir, se definió entero (*int).* Cabe señalar que de manera implícita se considera que una función regresa un entero, por ello si no se escribe ningún tipo de dato al principio, va a regresar un entero.

Programa 14. Cálculo de la distancia a partir de conocer la velocidad promedio y el tiempo recorrido utilizando funciones

A continuación se presenta un programa que ejemplifica el uso de funciones, el cual muestra la distancia de recorrido en el tablero del auto, a partir de conocer la velocidad promedio y el tiempo de recorrido. Cabe señalar que en dicho código fuente se utilizan dos variables locales, dentro de la función *calcula_distancia* y la variable local *distancia* dentro de la función main().

Identifiquemos los datos de entrada y salida:

Descripción	Identificador	Variable/ Constante	Dominio	Tipo de dato
Datos de entrada				
Velocidad promedio	Velocidad	variable	números reales	float
Tiempo recorrido	Tiempo	variable	números reales	float
Datos de salida				
Distancia recorrida	Distancia	variable	números reales	float

El diseño de la solución en lenguaje C es:

```
#include <stdio.h>
/*Declaración de la función calcula_distancia
 Observa que el valor de retorno de la función se define del tipo de  datos float
*/
float calcula_distancia();

/* Definición de la función calcula_distancia, es importante visualizar que tiene dos variables locales: velocidad y tiempo */
float calcula_distancia()
 {
        float velocidad, tiempo; //variables locales
```

```
        printf("Escribe la velocidad promedio:\n");
        scanf("%f",&velocidad);
        printf("Escribe el tiempo del recorrido:\n");
        scanf("%f",&tiempo);
        return velocidad*tiempo; //El resultado de la expresión es el valor devuelto
} // Finaliza la función calcula_distancia

//Declaración de la función main
//El valor devuelto por la función es de tipo entero
int main() {
        float distancia;
        printf("El presente programa obtiene la distancia recorrida a partir de conocer la velocidad y el tiempo de recorrido\n\n");
        /*  Se hace el llamado (ejecución) de la función calcula_distancia
                La función devolverá un valor de tipo float, que es al igual al tipo de datos de la variable distancia
        */
                distancia=calcula_distancia();
                printf("La distancia recorrida es %f.\n", distancia);
        getch( );
        return 0;
} //Finaliza la funcion main
```

La ejecución del programa con los datos de prueba velocidad=50 y tiempo=0.5 se muestra en la Ilustración 1.31.

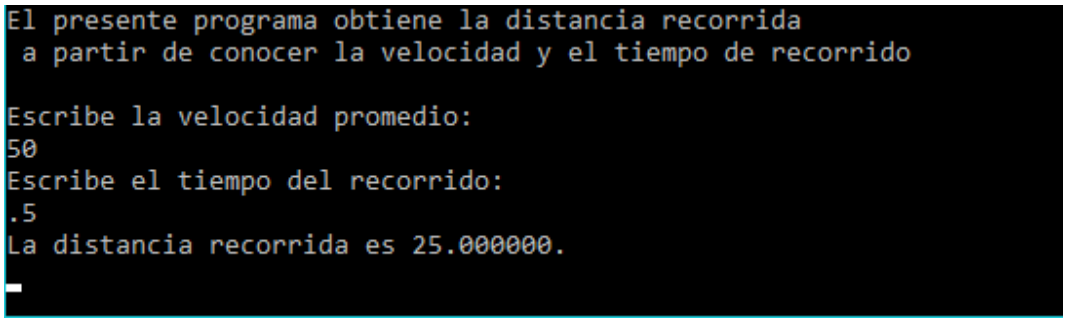


Ilustración 1.31. Resultado de la ejecución del programa 14

Ámbito de las variables

Antes de explicar el tema de parámetros es importante recordar el concepto de ámbito de las variables, visto en el tema 1.1 de esta guía, porque es en el uso de las subrutinas con parámetros donde se puede observar con mayor detalle la diferencia entre las variables locales y globales.

Como recordarás, una variable global puede ser utilizada en cualquier subrutina del programa conservando su valor almacenado. En cambio, una variable local, sólo puede ser usada dentro de la subrutina en la que fue declarada y su ciclo

de vida termina al concluir la ejecución de la subrutina. Aquí es importante mencionar un tipo de variable muy peculiar, asociado a las subrutinas, llamado parámetro.

Parámetros y argumentos

Los parámetros y argumentos son utilizados para transferir datos entre subrutinas. Algunas fuentes bibliográficas manejan ambos términos como sinónimos, mientras que otras hacen una distinción entre los dos conceptos y los definen como:

- **Argumento** es el valor que es pasado a una subrutina cuando ésta es invocada.
- **Parámetro** es una variable definida en una subrutina que recibe un valor cuando la subrutina es invocada.

En la explicación de esta guía, los términos argumento y parámetro serán tratados como sinónimos.

Parámetros por valor

Cuando un parámetro por valor se pasa a una subrutina se crea en memoria una copia temporal del dato, sin alterar el valor de la variable original. Esta copia sólo se utiliza dentro de la subrutina. En la ilustración 1.32 se muestra el proceso de escritura en memoria principal que se hace internamente al utilizar un paso de parámetro por valor.

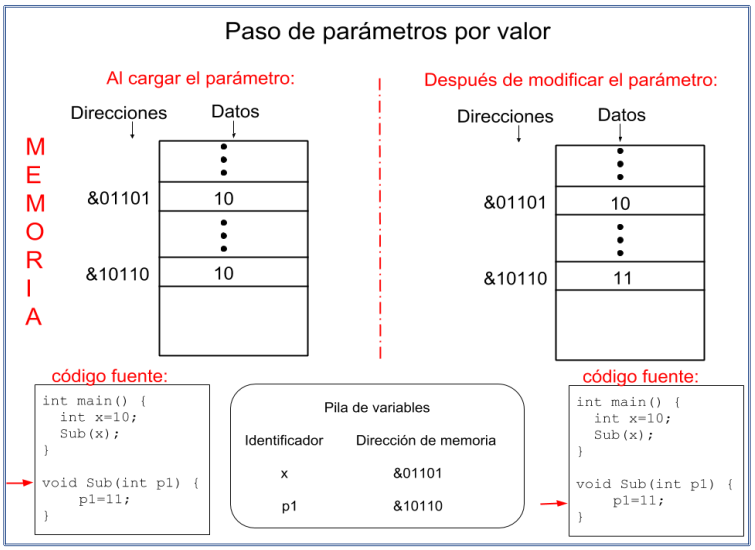


Ilustración 1.32. Parámetros por valor

Programa 15. Activación automática del sistema desempañador del vidrio trasero de un automóvil utilizando funciones

Para declarar una subrutina con paso de parámetros por valor, vamos a retomar el código del *programa 4 Activación automática del desempañador del vidrio trasero de un auto* realizado previamente en las páginas 39 a 41 de esta guía, donde encontrarás el cuadro de datos de entrada y salida, el algoritmo, la prueba de escritorio y dos pantallas de salida del programa dentro de la ilustración 1.16. Para que te puedas apoyar en la revisión del presente programa.

La idea es utilizar una subrutina llamada *sistema_vidrio_trasero* que encapsulará la tarea principal de dicho programa. De manera que puedas comparar el utilizar código con subrutinas y sin ellas.

Diseñaremos la función *sistema_vidrio_trasero* con dos parámetros, que corresponden a las dos variables de entrada del programa. Para ello, primero se declarará la función con sus dos parámetros, después se definirá la función *main()* que a su vez hará una llamada a la subrutina con el paso de parámetros por valor y por último, se definirá dicha subrutina. A continuación se muestra el código en lenguaje C:

```
#include <stdio.h>
 /* Se declara la subrutina indicando el tipo de datos de cada parámetro */
void sistema_vidrio_trasero(float, float);

int main() {
        float temp_exterior;
        float temp_interior;
        printf("¿Cuál es la temperatura interior del auto?\n");
        scanf("%f", &temp_interior);
        printf("¿Cuál es la temperatura exterior del auto?\n");
        scanf("%f", &temp_exterior);
/* Se hace la llamada a la subrutina escribiendo las variables que le pasarán el valor
a los parámetros*/
        sistema_vidrio_trasero(temp_exterior, temp_interior);
        return 0;
}

/* Se define la subrutina con sus dos parámetros */
void sistema_vidrio_trasero(float t_ext, float t_int)
 {
        #define MENSAJE "Sistema desempañador activado"
        if (t_int > t_ext) {
                printf("%s", MENSAJE);
        }
} // Finaliza la subrutina
```

Parámetros por referencia

Cuando una variable pasa a una subrutina como un parámetro por referencia, los cambios realizados en ésta dentro de la subrutina permanecen incluso después de que la subrutina haya terminado. Esto se debe a que realmente no se está pasando como parámetro el valor de la variable, sino la dirección de memoria donde se almacena dicha variable. Esto se puede observar en la Ilustración 1.33.

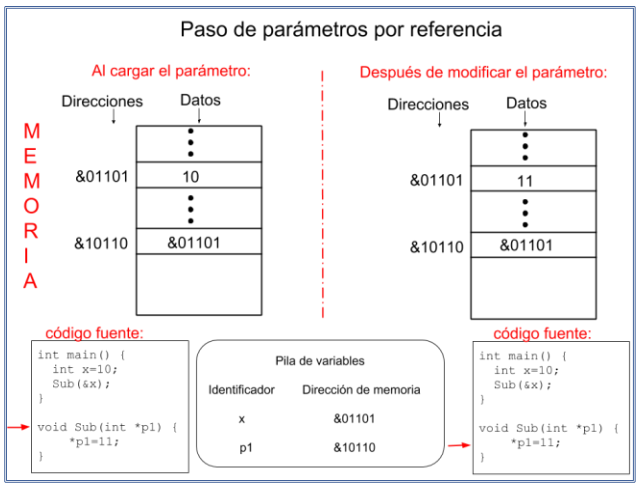


Ilustración 1.33. Parámetros por referencia

Programa 16. Kilómetros recorridos por semana (solución con subrutinas, paso de parámetros y variables globales y locales)

Revisemos otra solución del *programa 12 Kilómetros recorridos por semana* de esta guía, ahora aplicando todos los conceptos del tema 1.4 Subprogramas como abstracción de operaciones: procedimientos y funciones con paso de parámetros por valor y por referencia.

Se utiliza una función, dos procedimientos, una variable global, varias variables locales y paso de parámetros tanto por valor como por referencia. Todos estos elementos aparecen en el código fuente marcados con comentarios para que los puedas identificar.

Cabe señalar que para el paso de parámetros por referencia en el lenguaje C, en la declaración de la subrutina se debe agregar antes del identificador del parámetro un asterisco (*) y cuando se hace la llamada a la subrutina, se debe escribir un ampersand (&) antes del identificador del parámetro, porque lo que se está enviando como argumento es la dirección de memoria de la variable. Por ello, se nombró a los parámetros incluyendo las frases _paso_por_valor o _paso_por_referencia, según sea el caso, para que identifiques fácilmente el tipo de paso de parámetro.

Asimismo, es importante mencionar que se tiene una variable global que es kms[ ][ ], la cual se refiere a un arreglo. Recuerda que este tipo de variable conserva sus valores independientemente de la subrutina en la que se encuentren, contrario

a las variables locales, las cuales se destruyen (por así decirlo) después de finalizar la subrutina.

```
#include <stdio.h>
#include <conio.h>
/* Variable global, llamada kms que se refiere a un arreglo que almacenará los
recorridos por días. Puede ser utilizada en cualquier subrutina del programa*/

float kms[5][7] = {{0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                   {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                   {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                   {0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                   {0.0,0.0,0.0,0.0,0.0,0.0,0.0}};

/* Declaración de la función calcula_km con cinco parámetros:
     viajes_paso_por_valor.- Es un parámetro entero de paso por valor
     dias_paso_por_valor.- Es un parámetro entero de paso por valor
     *maxK_paso_por_referencia.- Es un parámetro de punto flotante de paso por
referencia
     *maxDia_paso_por_referencia.- Es un parámetro de punto flotante de paso por
referencia
     El valor devuelto por la función es de tipo real */
float calcula_km(int viajes_paso_por_valor, int dias_paso_por_valor, float
*maxK_paso_por_referencia, int *maxDia_paso_por_referencia);

/* Declaración del procedimiento lectura_km con dos parámetros:
     viajes_paso_por_valor.- Es un parámetro entero de paso por valor
     dias_paso_por_valor.- Es un parámetro entero de paso por valor
  Como es un procedimiento, no devuelve ningún valor.
*/
void lectura_km(int viajes_paso_por_valor, int dias_paso_por_valor);

/* Declaración del procedimiento menu. No tiene parámetros */
void menu();

/* Definición de la  función calcula_km*/
//Cabecera de la función
float calcula_km(int viajes_paso_por_valor, int dias_paso_por_valor, float
*maxK_paso_por_referencia, int *maxDia_paso_por_referencia)
//Cuerpo de la función
{
   //Variables locales de la función. Son cuatro: dia, viaje, km_dia y tot_semana
   int dia = 0;
   int viaje = 0;
   float km_dia = 0.0;
   float tot_semana = 0.0;
```

```
    for (dia = 0; dia < dias_paso_por_valor; dia++) {     //For anidados
        km_dia = 0.0;
        for (viaje = 0; viaje< viajes_paso_por_valor; viaje++) {
            km_dia = km_dia + kms[viaje][dia];
        }  //Fin del for interno
        printf("\n Total kilometros dia No. %d: %5.2f", dia+1, km_dia);
        if (km_dia >= *maxK_paso_por_referencia) {
          *maxK_paso_por_referencia = km_dia;
          *maxDia_paso_por_referencia=dia;
        } //Fin del if
        tot_semana = tot_semana + km_dia;
    }   // fin del for externo
    return tot_semana;
}  //Fin de la función calcula_km

// Definición del procedimiento lectura_km
//Cabecera del procedimiento
void lectura_km(int viajes_paso_por_valor, int dias_paso_por_valor)
//Cuerpo del procedimiento
{
    // Variables locales. Son dos: dia y viaje
    int dia = 0;
    int viaje = 0;
    printf ("Lectura de datos\n\n");
    for (dia  = 0; dia < dias_paso_por_valor; dia++)  {   // for anidado
        for (viaje = 0; viaje< viajes_paso_por_valor; viaje++) {
            printf ("Kilómetros recorridos, dia No. %d, viaje No. %d :", dia+1, viaje+1);
            scanf ("%f", &kms[viaje][dia]);
        } /* fin del for interno*/
    } /* fin del for externo*/
 } /* Fin del procedimiento lectura_km */

/* Definición del procedimiento menu*/
//Cabecera del procedimiento
void menu()
//Cuerpo del procedimiento
{
    // Variables locales. Son seis
    int n_viajes_paso_por_valor = 5;
    int n_dias_paso_por_valor = 7;
    int max_dia_paso_por_referencia = 0;
    float max_km_paso_por_referencia = 0;
    float total_semana = 0.0;
    int opcion = 0;
    while (opcion != 3) {
            printf (" \n Bitacora de viajes\n");
```

```
            printf ("\n 1. Lectura de kilometros/semana");
            printf ("\n 2. Reporte semanal");
            printf ("\n 3. Salir");
            printf ("\n Que opcion desea?");
            scanf("%d", &opcion);
            switch (opcion){
              case 1:
/* Al seleccionar la opción 1 el programa hará una llamada al procedimiento
lectura_km, cuyos paso por parámetros son: las variables viajes_paso_por_valor y
dias_paso_por_valor que son paso por valor. */
                lectura_km(n_viajes_paso_por_valor, n_dias_paso_por_valor);
                break;
             case 2:
/* Al seleccionar la opción 2 se ejecutará el siguiente código, donde hace una
llamada de la función calcula_km cuyas variables max_km y max_dia están
pasando por referencia, es por ello que llevan el símbolo de & al inicio. */
                total_semana = calcula_km(n_viajes_paso_por_valor,
n_dias_paso_por_valor, &max_km_paso_por_referencia,
&max_dia_paso_por_referencia);
                printf("\nDia con mas kilometros ( %d ) con %5.2f kilometros" ,
max_dia_paso_por_referencia + 1, max_km_paso_por_referencia);
                printf("\ntotal  kilometros a la semana %.2f", total_semana);
                break;
             case 3:
                printf("\nFin del programa");
                break;
             default:
                printf("\nError en opcion\n\n");
                break;
           }  /* Fin switch*/
    }  /* Fin while*/
} /* fin del procedimiento menu()*/

/* Inicio de la función main()*/
int main()
{
     menu(); /* Hace una llamada al procedimiento menu()   */
     return 0;
} /* Fin de la función main()*/
```

Ejercicio 1.30. Según las siguientes declaraciones de funciones en lenguaje C, indica el tipo de valor que devuelve, los parámetros (si es que tiene) y de qué tipo son éstos, ya sea paso por valor o paso por referencia.

a) int calculaEdad(int dianacimiento, int mesnacimiento, int anionacimiento)

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?

b) void intercambioDeValores(int *val1, int *val2)

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?

c) float promediaCalificaciones(float calif[100], int numvalores)

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?

d) void punto(float x1, float y1, float x2, float y2, float *pendiente, float *ordenada)

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?

e) void estadisticas(int datos[100], int num, float *promedio, int *max, int *min)

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?

	
Preguntas de reflexión 1. ¿Crees que las subrutinas permitan el trabajo colaborativo en la programación? Argumenta tu respuesta. 2. ¿Es adecuado el exceso de funciones en los programas?, ¿por qué? 3. Aprender a hacer programas utilizando subrutinas ¿facilita o complica la programación?	

Autoevaluación

1. Supongamos que te piden calcular el área de un triángulo, sabiendo que la fórmula es area =base*altura/2, si introduces valores con punto decimal para la base y la altura, ¿cómo se mostrará el resultado al utilizar la variable *area* de tipo entero?

   a) con punto decimal
   b) truncado el valor decimal
   c) solo el valor decimal
   d) no mandará el resultado

2. ¿Cuál de los siguientes programas NO tiene errores?

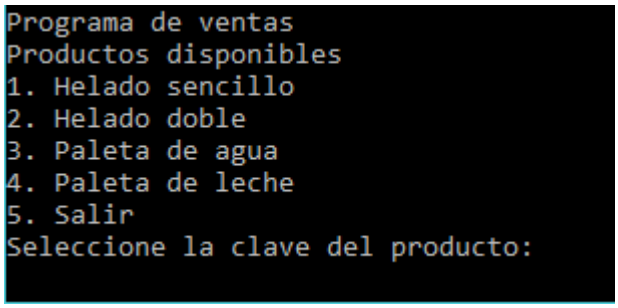
#include <stdio.h>
main()
{
int numero=5;
printf("\t \tEste es mi primer programa");
printf("\nEl valor de la variable numero es = %d", numero);
}
	#include <stdio.h>
	main()
	{
	int numero==5;
	printf("\t \tEste es mi primer programa");
	printf("\nEl valor de la variable numero es = %n", numero);
	}
		#include <stdio.h>
		main()
		{
		int numero=5;
		printf("\t \tEste es mi primer programa");
		printf("\nEl valor de la variable numero es = %f", número);
		}
			#include <studio.h>
			main()
			{
			int numero=5;
			printf("\t \tEste es mi primer programa");
			printf("\nEl valor de la variable numero es = %c", número);
			}

Concluiste tu tercer año de preparatoria y tienes varios meses de vacaciones, así que decides buscar un empleo de verano antes de ingresar a la universidad. Una pequeña heladería te contrata para que le diseñes un sistema de cómputo que automatice su proceso de ventas.

Los productos que ofrece la heladería son:

Clave	Producto	Precio
1	Helado sencillo	$20.00
2	Helado doble	$35.00
3	Paleta de agua	$15.00
4	Paleta de leche	$20.00

El propietario desea que el sistema muestre en pantalla un menú que contenga todos los productos disponibles para que el cajero pueda seleccionar el producto que comprará el cliente, o bien, terminar el programa, como se muestra en la siguiente pantalla:



Responde las siguientes preguntas:

3. ¿Cuál es el dominio adecuado para la variable que almacene la clave del producto?

   a) numeros enteros
   b) caracteres
   c) números reales en simple precisión
   d) números reales en doble precisión

4. ¿Cómo declararías en tu programa los precios de los helados?

   a) float h1, h2;
   b) #define h1 20
      #define h2 35
   c) #define h1 $20.00
      #define h2 $35.00
   d) int h1, h2;

5. Dados los valores de helados *h1=20* y *h2=35*, ¿qué resultado arrojaría la siguiente expresión: *b = h1 == h2*?

   a) 0
   b) 1
   c) h2
   d) La sintaxis de la expresión es incorrecta

6. Suponga que *clave* es una variable de tipo entero que almacena la clave del producto seleccionado por el usuario en el menú. ¿Cuál sería el segmento de código que emplearías para conocer la clave que capturó el cliente?

   a) while (clave) {
      if 1 : printf("\nOpción 1 seleccionada");
      break;
      if 2 : printf("\nOpción 2 seleccionada");
      break;
      if 3 : printf("\nOpción 3 seleccionada");
      break;

```
        if 4 :  printf("\nOpción 4 seleccionada");
                break;
        if 5 :  printf("\nOpción 5 seleccionada");
                break;
        else: printf("\nOpción inválida");
                break;
      }
```

b)
```
switch(clave) {
        case 1 : printf("\nOpción 1 seleccionada");
                 break;
        case 2 : printf("\nOpción 2 seleccionada");
                 break;
        case 3 : printf("\nOpción 3 seleccionada");
                 break;
        case 4 : printf("\nOpción 4 seleccionada");
                 break;
        case 5 : printf("\nOpción 5 seleccionada");
                 break;
        default: printf("\nOpción inválida");
                  break;
      }
```

c)
```
for (clave=1; clave<=5,clave++) {
        printf(“%s %d %s, “Opción”, clave, “ seleccionada”);
}
```

d)
```
for (clave=1; clave=5, clave++) {
        printf(“%s %d %s, “Opción”, clave, “ seleccionada”);
}
```

7. ¿Cuál es el algoritmo que permite calcular el total a pagar cuando el cliente compra más de un producto?

a) Inicio
   1. Definir como constantes los precios de los productos
   2. Inicializar el total a pagar a 0
   3. Inicializar clave a 0
   4. Mostrar las opciones del menú al usuario
   5. Solicitar que seleccione la clave del producto
   6. Almacenar la clave del producto en la variable clave
   7. Calcular el total a pagar dependiendo del producto seleccionado
   8. Mostrar al usuario el total a pagar

   Fin

b) Inicio
    1. Definir como constantes los precios de los productos
    2. Inicializar el total a pagar a 0
    3. Inicializar clave a 0
    4. Mostrar las opciones del menú al usuario
    5. Solicitar que seleccione la clave del producto
    6. Almacenar la clave del producto en la variable clave
    7. Mientras (clave != 5) haz
    8. Calcular el total a pagar dependiendo del producto seleccionado
    9. FinMientras

   Fin

c) Inicio
    1. Mientras (clave != 5) haz
    2. Definir como constantes los precios de los productos
    3. Inicializar el total a pagar a 0
    4. Inicializar clave a 0
    5. Mostrar las opciones del menú al usuario
    6. Solicitar que seleccione la clave del producto
    7. Almacenar la clave del producto en la variable clave
    8. Calcular el total a pagar dependiendo del producto seleccionado
    9. FinMientras

   Fin

d) Inicio
    1. Definir como constantes los precios de los productos
    2. Inicializar el total a pagar a 0
    3. Inicializar clave a 0
    4. Mientras (clave !=5) haz
    5. Mostrar las opciones del menú al usuario
    6. Solicitar que seleccione la clave del producto
    7. Almacenar la clave del producto en la variable clave
    8. Calcular el total a pagar dependiendo del producto seleccionado
    9. FinMientras
    10. Mostrar al usuario el total a pagar

   Fin

8. La heladería tiene una promoción del 20% de descuento en los helados sencillos los lunes. ¿Qué segmento de código emplearías en tu programa para validar esta situación?

a)
```
if (dia == 'l') {
  total_pagar = total_pagar + HELADO_S*0.80;
}
else {
  total_pagar = total_pagar + HELADO_S;
}
```

b)
```
if (dia = 'l') {
  total_pagar = total_pagar + HELADO_S*0.20;
}
else {
  total_pagar = total_pagar + HELADO_S;
}
```

c)
```
if (dia != 'l') {
  total_pagar = total_pagar + HELADO_S*0.20;
}
else {
  total_pagar = total_pagar + HELADO_S;
}
```

d)
```
if (dia = 'l') {
  total_pagar = total_pagar + HELADO_S- HELADO_S*0.20;
}
else {
  total_pagar = total_pagar + HELADO_S;
}
```

Estás realizando una investigación sobre el consumo de tabaco en los adolescentes. Decides entrevistar a TODOS tus compañeros de clase (50 personas) para obtener el total de jóvenes que fuman y el total que no lo hacen. Para ello, diseñas un programa, en lenguaje de programación C, que te ayude en el proceso de recopilación y procesamiento de la información.

Responde las siguientes preguntas:

9. ¿Cómo controlarías el final del programa?

a)
```
for (alumnos = 1; alumnos < 50, alumnos++) {
 instrucciones para realizar el conteo de fumadores y no fumadores
}
```

b)
```
char total_alumnos;
while (total_alumnos='S') {
   instrucciones para realizar el conteo de fumadores y no fumadores
   printf("Hay más alumnos: );
   scanf(" %c", &total_alumnos);
}
```

c)
```
for (alumnos = 1; alumnos <= 50, alumnos++) {
 instrucciones para realizar el conteo de fumadores y no fumadores
}
```

d)
```
char total_alumnos;
do {
    instrucciones para realizar el conteo de fumadores y no fumadores
    printf(“Hay más alumnos: );
scanf(“ %c”, &total_alumnos);
    } while (total_alumnos=’S’);
```

10. ¿Cómo programarías el conteo del número de fumadores y no fumadores?

a)
```
if (fumas = ’S’) {
    si_fuma = si_fuma + 1;
}
if (fumas = ’N’) {
    no_fuma = no_fuma + 1;
}
```

b)
```
if (fumas == 'S') {
    si_fuman = si_fuman + 1;
}
else {
    no_fuman = no_fuman + 1;
}
```

c)
```
switch (fumas) {
  case S: si_fumas = si_fumas + 1;
          break;
  case N: no_fumas = no_fumas + 1;
    break;
  default : printf(“Error al seleccionar la opción”);
}
```

d)
```
if (fumas = 'S') {
    no_fuman = no_fuman + 1;
}
else {
    si_fuman = si_fuman + 1;
}
```

En una escuela primaria los grupos están formados por 25 alumnos. Después de realizar exámenes y tomar en cuenta todas las calificaciones, los profesores utilizan un programa de cómputo para capturar los promedios obtenidos.

11. Para almacenar la calificación de los 25 alumnos ¿qué es lo más conveniente que el programa utilice?

    a) Una variable entera para cada alumno
    b) Una constante entera para cada alumno

c) Un arreglo de numeros reales para todos los alumnos
d) Un arreglo de constantes reales para todos los alumnos

12. El programa utiliza un arreglo de números reales para almacenar las calificaciones de todos los alumnos y para garantizar que el programa no guarde basura en las celdas del arreglo, se inicializa cada una con el valor 0. ¿Cuál es la forma correcta de incializarlo en la declaración del arreglo?

a) int calificaciones [25]=0;
b) float calificaciones [25]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
c) float calificaciones [25]=0.0;
d) float calificaciones [25]="0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"

13. El programa utiliza un arreglo de números reales para almacenar las calificaciones de todos los alumnos. ¿Cuál es el código correcto para que los profesores capturen las calificaciones?

a)
```
int alumno;
float calificaciones [25];
for (alumno = 0; alumno < 25; alumno++)
    {scanf ("%f", &calificaciones [alumno]); }
```

b)
```
int alumno [25];
int calificaciones;
for (alumno = 0; alumno < 25; alumno++)
    {scanf ("%f", &calificaciones [alumno]); }
```

c)
```
int calificaciones [25];
int alumno;
for (alumno = 0; alumno < 25; alumno++)
    {scanf ("%f", &alumno [calificaciones]); }
```

d)
```
float alumno [25];
int calificaciones;
for (alumno=0; alumno <25; alumno++)
    {scanf ("%f", &alumno [calificaciones]); }
```

14. Ahora que los profesores capturaron los promedios en el arreglo claificaciones de todos los alumnos, desean saber el número de alumnos aprobados, reprobados y el promedio general del grupo. ¿Cómo hay que escribir las instrucciones que calculan los datos solicitados?

a)
```
float calificaciones [25];
float alumno;
int  promedio = 0;
int aprobados = 0;
int reprobados = 0;
```

```
for (alumno = 0; alumno < 25; alumno++)
    {       if (calificaciones[alumno] >= 6) {
            aprobados = aprobados + 1;
            reprobados = reprobados + 1;
            promedio = promedio + calificaciones[alumno]}
    }
    promedio=promedio / 25;
    printf("%d", aprobados);
    printf("%d", reprobados);
    printf("%f", prom);
```

b)
```
float promedio = 0;
int aprobados = 0;
int reprobados = 0;
for (alumno = 0; alumno < 25; alumno++)
    {       if (calificaciones[alumno] >= 6)
                aprobados = aprobados + 1;
            else
                reprobados = reprobados + 1;
            promedio = promedio + calificaciones[alumno];
    }
printf("%d",aprobados);
printf("%d",reprobados);
printf("%f", promedio);
```

c)
```
float calificaciones [25];
 int alumno;
 float suma= 0;
 float promedio = 0;
 int aprobados = 0;
 int reprobados = 0;
 for (alumno = 0; alumno < 25; alumno++)
 {   if (calificaciones[alumno] >= 6)
                aprobados = aprobados + 1;
            else
                reprobados = reprobados + 1;
            suma = suma + calificaciones[alumno];
 }
 promedio=suma / 25;
 printf("%d", aprobados);
 printf("%d", reprobados);
 printf("%f", promedio);
```

d)
```
float prom = 0;
float promedio = 0;
int aprobados = 0;
int reprobados = 0;
```

```
for (calificaciones = 0; calificaciones < 25; calificaciones++)
{    if (calificaciones[alumno] >= 6)
                aprobados = aprobados + 1;
        else
                reprobados = reprobados + 1;
        promedio = promedio + calificaciones[alumno];
}
prom = promedio / 25;
printf("%d",aprobados);
printf("%d",reprobados);
printf("%f", prom);
```

Un programa utiliza en su código un arreglo bidimensional de 4x12 que almacena las ventas mensuales por semana, donde los renglones representan las semanas y las columnas los meses. El programador necesita imprimir todos los montos de venta y desea realizar el barrido del arreglo por mes.

15. ¿Cómo debe de estructurarse el barrido del arreglo para darle prioridad a los meses?

a) Para todas las semanas
   Para todos los meses
      imprimir arreglo [semanas][meses]

b) Para todos los meses
   Para todas las semanas
      imprimir arreglo [meses][semanas]

c) Para todas las semanas
      Para todos los meses
         imprimir arreglo[meses][semanas]

d) Para todos los meses
   Para todas las semanas
      imprimir arreglo [semanas][meses]

El profesor de cierta asignatura desea hacer uso de su computadora para llevar a cabo la captura de calificaciones, obtener el promedio de calificaciones de sus estudiantes y calcular aspectos estadísticos. Para ello, se acerca a tí, que estudias la asignatura de Informática aplicada a la ciencia y la industria, para pedirte si le puedes realizar un programa de cómputo donde pueda resolver este tipo de problemas.

Decides diseñar el programa en lenguaje C utilizando subrutinas porque deseas crearlo de manera colaborativa con varios de tus compañeros de clase. De acuerdo con esta problemática, responde las siguientes preguntas:

16. ¿Cuál es la diferencia entre un procedimiento y una función?

    a) El primero es una subrutina y el segundo no lo es
    b) El primero regresa un valor y el segundo no
    c) El primero no es subrutina y el segundo sí lo es
    d) El primero no regresa un valor y el segundo sí

17. ¿Cuál de las siguientes definiciones de subrutinas en lenguaje C, no regresa ningún valor?

    a) int promedioCalificaciones(void);
    b) void promedioCalificaciones(float cal1, float cal2, float cal3);
    c) float promedioCalificaciones(int cal1, int cal2, int cal3);
    d) char promedioCalificaciones(void, void, void);

18. Piensas almacenar las calificaciones en arreglos de datos y utilizarlas en las subrutinas. ¿Cuál de las siguientes afirmaciones es verdadera en el lenguaje de programación C?

    a) Todos los arreglos son considerados variables globales automáticamente, sin importar dónde sean declarados.
    b) Es necesario hacer un paso de parámetros, el cual se hace automáticamente como paso por valor.
    c) Es necesario hacer un paso de parámetros, el cual se hace automáticamente como paso por referencia.
    d) Todos los arreglos deben ser declarados como variables locales en cada subrutina y automáticamente éstas recibirán los datos de otros arreglos.

19. Para declarar un parámetro por referencia, en el encabezado de una subrutina ¿qué prefijo debe escribirse en el identificador del parámetro?

    a) *
    b) &
    c) %
    d) \

20. ¿Cuál ámbito de variable conserva su valor independientemente de la subrutina en la que se utilice?
    a) Global
    b) Local
    c) Parámetro por valor
    d) Parámetro por referencia

Unidad 2

Análisis de los datos e interpretación de resultados

UNIDAD 2
ANÁLISIS DE LOS DATOS E INTERPRETACIÓN DE RESULTADOS

Objetivo

El alumno organizará los datos obtenidos de los procesos automatizados con el uso de programas computables y software específico de control de componentes electrónicos para presentar información que permita emitir conclusiones, desarrollando habilidades de pensamiento crítico en la toma de decisiones y favoreciendo los valores de manera ética, responsable y segura en el manejo de la información.

Introducción

Observar y cuantificar datos de fenómenos que nos rodean nos permite identificar qué pasa en la naturaleza o en algún fenómeno específico, los cuales es importante conocer y comprender para tomar mejores decisiones en nuestra vida y entorno. Estos datos pueden ser agrupados por rangos, según lo que tengamos que estudiar de los mismos, así como aplicarles diferentes tipos de cálculos estadísticos como son: media, mediana, moda, hacer tablas de frecuencias y conocer la veracidad de los datos con el coeficiente de correlación, para entender el comportamiento de algún fenómeno y lograr una conclusión.

Con el apoyo de los cálculos estadísticos y los resultados obtenidos, podemos no sólo entender las situaciones actuales con las que contamos, sino también proyectar comportamientos de los datos del fenómeno a futuro.

Así mismo, los datos pueden ser expresados en forma gráfica con diversos estilos para una mejor comprensión. La frecuencia nos indica qué tan distribuidos están los datos en una gráfica, ayudando a identificar en dónde se encuentra la mayor concentración de estos.

Por ejemplo, los parques de diversiones con juegos mecánicos suelen colocar pantallas con el tiempo de espera para subir a una atracción. Si en lugar del tiempo se pusiera una gráfica de campana, se podría decidir rápidamente si esperas o mejor vas a otro juego. Esta imagen nos estaría mostrando el tiempo de espera el cual estaremos en la fila. Como puedes ver en nuestra vida, la estadística está inmersa.

Contenidos conceptuales	Contenidos procedimentales
2.1 Tipos de datos estadísticos en la recolección de datos: categórico o cualitativos y cuantitativos o numéricos.  Social-sampling (depositphoto, 2018). Recuperado de: https://goo.gl/gpNxiR	2.4 Recopilación de datos en una investigación académica. **Contenidos actitudinales** 2.7 Uso responsable y ético de tecnologías, aplicaciones y prácticas para la recolección, integración, análisis, selección, interpretación y presentación de la información.

Para qué nos sirve saber ¿cuánto ha crecido la población cada 10 años? o ¿cuál es el refresco favorito de las personas entre 13 y 18 años? o ¿cuál es la temperatura que se ha registrado en la Ciudad de México en los últimos meses? Conocer esta información nos permite tomar decisiones, como qué tipo de vestimenta sería adecuada para el día, o cómo puedo aumentar la venta de refrescos. A los datos que recolectamos podemos darle una interpretación aplicando la **estadística**. Cuando transformamos los datos en información los podemos usar para tomar decisiones.

¿Pero en qué más nos puede apoyar la estadística? Revisemos el tema más a fondo.

¿Qué es la estadística?

La estadística es la ciencia que se encarga de planear estudios y experimentos para obtener datos o recopilar información, que se utiliza posteriormente para organizar, resumir, presentar, analizar e interpretar información con el fin de llegar a conclusiones en el área de la mercadotecnia, en la ciencia, en la vida cotidiana, entre otros rubros.

La estadística se divide en dos ramas: inferencial y descriptiva. Inferir es deducir a partir de la evidencia. La inferencia informal es imprecisa, y el objeto de la **Estadística Inferencial** es aportar mayor precisión a la obtención de conocimiento, es decir, deducir características de las poblaciones a partir de la evidencia obtenida en las muestras (Universitat de Valencia, s/f).

La **Estadística Descriptiva** sirve para organizar datos usando métodos numéricos, tabulares y gráficos para poder interpretarlos de manera correcta, o como lo menciona Johnson (1990) consiste en el área de la estadística dedicada a la recolección, presentación y descripción de datos numéricos.

Pero te preguntarás cómo se obtienen estos datos, existen diferentes métodos para su recolección, como técnicas aleatorias donde cada elemento tiene la misma probabilidad de ser seleccionado, o el método estratificado donde se divide la población que se va a estudiar en subconjuntos y se van tomando muestras de cada subconjunto.

¿Y qué son los datos? Son el conjunto de valores recolectados que pueden ser cuantitativos o cualitativos, es decir, números o características, usando diferentes métodos en una población o muestra. Veamos estos conceptos.

Población y muestra

- **Población**. Es el conjunto de todos los elementos que vamos a estudiar, con las características necesarias para obtener los datos.
- **Muestra**. Es un subconjunto de la población, una muestra representativa debe reflejar las características de la población.

Imagen 2.1 CMO (s.f.). Meet the 7 persons of modern CMO. Recuperado de: https://cmo.com/

Ejemplo:

Se va a estudiar cuál es el candidato presidencial por el cual votarán los mexicanos en las próximas elecciones, se toma una muestra de 10,000 personas de todo el país con derecho al voto. La pregunta que se va a realizar es la siguiente: ¿por quién votará en las próximas elecciones presidenciales? Determine la población y la muestra.

- Población: La población son todas las personas de nacionalidad mexicana con derecho al voto.
- Muestra: Es el conjunto de 10,000 mexicanos que forman parte de la población y tiene derecho a voto, es decir, son mayores de 18 años.

Imagen 2.2 Población y muestra. Recuperado: easel.ly (Cruz, I., 2018).

Tipos de datos
Los datos se pueden clasificar en:

Cuantitativos o numéricos. Se expresan mediante un número cuyo valor representa una magnitud- Pueden ser conteos o mediciones.

Ejemplos:
- El número de alumnos de un grupo de la Escuela Nacional Preparatoria.
- Los centímetros que mide el pizarrón.

Los datos cuantitativos se describen con mayor detalle cuando los clasificamos en dos tipos:

Discretos. Los datos discretos resultan cuando los valores que estamos recolectando tienen unidades o partes separadas unas de las otras.

Ejemplos:
- Número de hijos de una familia.
- El número de goles que anotó un equipo de fútbol.

Continuos. Los datos continuos son los que pueden tomar un número infinito de valores entre dos números.

Ejemplos:

- Promedio de calificaciones de los alumnos de sexto año de toda la Escuela Nacional Preparatoria.
- El registro de mi peso en un mes.

Cualitativos o categóricos. Expresan una cualidad del objeto de estudio. No se pueden medir con números, por lo tanto, no representan conteos o mediciones. Reflejan una característica o un atributo.

Ejemplos:
- El color de ojos de los alumnos del grupo 603.
- El estado civil de una persona.

Los datos cualitativos se describen con mayor detalle cuando los clasificamos en dos tipos:

Ordinales. No representan valores numéricos, pero existe un orden.

Ejemplos:
- Los datos de una encuesta de satisfacción (muy satisfecho, satisfecho, normal, deficiente).
- Lugares que obtienen los jugadores en las competencias.

Nominales. No representan valores numéricos, ni orden.

Ejemplos:
- Lugar de nacimiento de mis padres.
- Escuela a la que asisto.

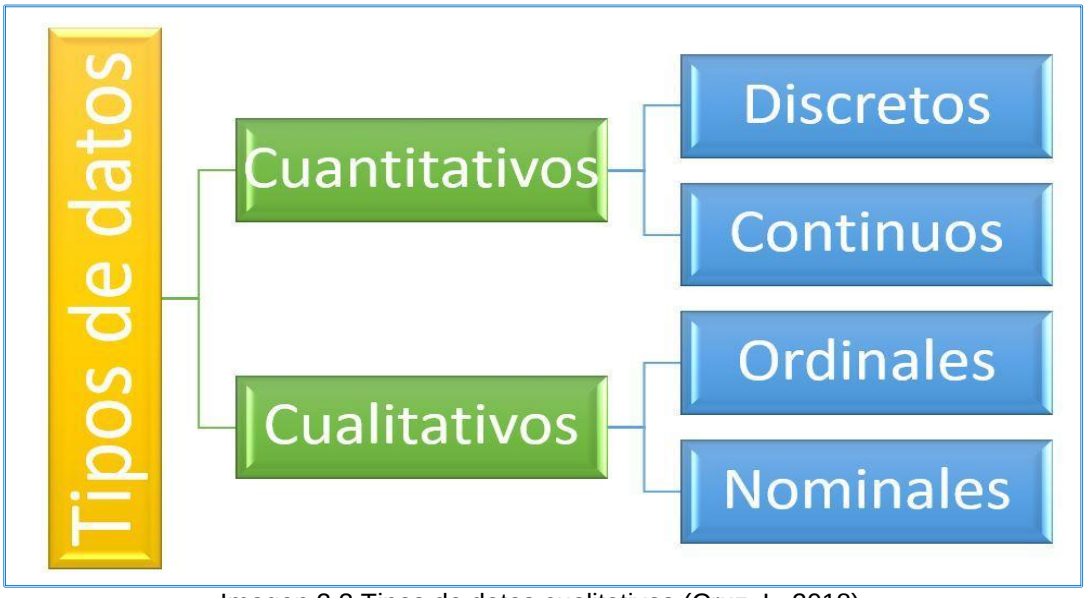


Imagen 2.3 Tipos de datos cualitativos (Cruz, I., 2018).

Ejercicio 2.1

Indica cuál es la población y la muestra para cada problema.

a) Se requiere conocer el número de alumnos de sexto año de la Escuela Nacional Preparatoria que elegirán la carrera de medicina y cumplen con los requisitos de ingreso, que son: terminar la preparatoria en 3 años y obtener un promedio de 9.0. Se realizó un estudio con 5000 alumnos de los cuales solo el 10% cumplieron con los requisitos.
¿Cuál es la muestra y la población?

- Población:
- Muestra:

b) El laboratorio médico Shering-Plough desea conocer la proporción de personas cuya diabetes tipo 1, puede ser controlada con un nuevo fármaco. Se realizó un estudio en 1500 personas con esta diabetes, y se encontró que el 65% de ellas pudo controlar su enfermedad con este fármaco. Se asume que estas 1500 personas son representativas del grupo de pacientes de diabetes tipo 1.
¿Cuál es la muestra y la población?

- Población:
- Muestra:

Ejercicio 2.2

Indica si el dato es cuantitativo o cualitativo marcando con una X la respuesta correcta.

Enunciado	Dato Cuantitativo	Dato Cualitativo
El número de mis seguidores en Twitter.		
El color de cabello de mi mejor amigo.		
La edad de mis padres.		
El lugar que obtuviste en una carrera de atletismo.		
Mi grupo favorito de música.		

Ejercicio 2.3

Indica si el dato es cuantitativo discreto, cuantitativo continuo, cualitativo ordinal, o cualitativo nominal marcando con una X la respuesta correcta.

Enunciado	Cuantitativo discreto	Cuantitativo continuo	Cualitativo ordinal	Cualitativo nominal
La clase favorita de los alumnos del grupo de sexto año.				
Los pares de zapatos que guardo en mi closet.				
La estatura de los niños y niñas de una escuela primaria.				
La cantidad de clientes atendidos en una tienda durante un día.				
El lugar que ocupa un alumno en la Olimpiada del Conocimiento.				
El volumen de agua dentro del tinaco.				
El peso de los animales de una granja.				
El número de pétalos que tiene una flor.				
El color de ojos de los alumnos de quinto año de la ENP.				
El tiempo que dura una llamada en un call center.				

Ejercicio 2.4

Cuando llegas a comprar zapatos, lo primero que te pregunta el vendedor es cuáles zapatos, de qué talla y color, marca, modelo, etc. Registra tu respuesta en la siguiente tabla según consideres que sean datos cualitativos o cuantitativos.

Datos	
Cuantitativos	Cualitativos

Ejercicio 2.5

Completa los siguientes espacios con los tipos de datos correspondientes:

Los datos cualitativos se pueden clasificar en dos tipos, los cuales son: _____ y _____, los primeros no tienen una característica implícita en su valor, como puede ser lugar de nacimiento, color favorito, música favorita, etc., mientras que los otros tienen valores, ya sea de orden _____ o tienen dos opciones de respuesta, por ejemplo, estado civil (soltero, casado, viudo, divorciado u otro) o bien fuma (sí o no) o género (hombre o mujer).

Mientras que los datos cuantitativos se pueden dividir en _____ y _____, siendo los primeros valores los que toman un número finito de valores por ejemplo número de amigos que se tienen, número de hijos. Y el segundo son aquellos que pueden tener un número infinito de valores dentro de sus mediciones, por ejemplo, estatura, promedio, etc.

Preguntas de reflexión

Hasta ahora hemos hablado de los tipos de datos, pero te preguntarás ¿por qué son importantes? Los datos nos permiten tomar decisiones, ya que son herramientas para realizar análisis y comprobar hipótesis o resolver problemas, siempre y cuando conozcamos cuál es el contexto de los datos, de qué fuente se obtuvieron y cómo se recabaron. Por ejemplo, observa la siguiente tabla:

X	56	67	57	60	64
Y	53	66	58	61	68

¿Qué podemos concluir? Anótalo a continuación:

_____
_____

Ahora bien, qué pasa si la tabla la relacionas con el siguiente contexto:

La tabla anterior se tomó del artículo “Los estudiantes aumentan de peso en su primer año de universidad”. Este estudio se realizó a estudiantes de la Facultad de Ingeniería. La variable X corresponde al peso de cinco estudiantes registrado en el mes de agosto, en su primer semestre de estudios, y la variable Y representa el peso registrado de esos mismos estudiantes en el mes de diciembre, al final de su semestre.

Al conocer el contexto de los datos y la fuente donde se obtuvieron ¿qué podemos concluir? Anótalo a continuación.

_____
_____

Ahora que tienes las dos conclusiones, responde lo siguiente:
¿La conclusión 1 y 2 son diferentes?

_____

¿Por qué?

_____
_____

Describe una situación personal donde con ayuda de los datos tomaste una decisión o llegaste a una conclusión:

Contenidos conceptuales	Contenidos procedimentales
2.2 Análisis de datos y variables con pruebas de normalidad, comparación de las medias, análisis de varianza y de correlación.  Datos-estadísticos (freepik, 2018). Recuperado de: https://goo.gl/5o5Q2K	2.5 Elaboración de inferencias sobre la realidad a partir de la información obtenida con estimaciones. 2.6 Graficación e interpretación de información para la formulación de conclusiones. **Contenidos actitudinales** 2.7 Uso responsable y ético de tecnologías, aplicaciones y prácticas para la recolección, integración, análisis, selección, interpretación y presentación de la información. 2.8 Respeto y tolerancia por las opiniones de sus compañeros en los debates y discusiones de temas específicos. 2.9 Disposición para el trabajo colaborativo.

En este tema recordaremos cuando en la primaria veíamos cómo se clasificaban los animales, los vegetales, los ríos, las montañas, o bien, en cursos posteriores como química aprendimos que los elementos de la tabla periódica están clasificados según sus características particulares, y sus semejanzas o diferencias

De esta manera también tenemos clasificados los cajones y guardarropa de nuestra habitación. Estas variedades o diferencias de nuestra ropa, animales, ríos, caminos, etc. nos conducen a tomar decisiones de acuerdo con la pertinencia de lo que vayamos a concluir.

Como comentamos en el tema anterior, la estadística descriptiva nos permite organizar la información y presentarla de manera sistematizada para entender el comportamiento de un fenómeno o suceso. Las técnicas más usadas para realizar estas acciones son el cálculo de las medidas descriptivas, las tablas de frecuencias y las gráficas.

La aplicación de estas técnicas de la estadística descriptiva puede hacerse sobre sobre datos no agrupados y agrupados, dependiendo de la información que necesitemos conocer. Por ello, primero definamos la diferencia entre ambos:

Los datos no agrupados	Los datos agrupados
Son aquellos que se presentan tal y como fueron recabados.	Son los datos recabados que se encuentran ordenados y clasificados en grupos.

Es muy importante que identifiques si tus datos están agrupados o no, porque de ello depende qué fórmula matemática debes utilizar para conocer el comportamiento de la variable que estás estudiando.

Medidas descriptivas

Las medidas descriptivas son valores numéricos calculados que resumen el comportamiento o característica del objeto de estudio.

Valdez, I. y Gómez M. (2017), indican que “las medidas que pueden describir a un conjunto de datos se clasifican en medidas de tendencia central, de dispersión y de forma. Las medidas descriptivas pueden calcularse a partir del conjunto de datos tal cual aparecieron al recolectar la información (datos sin agrupar), o bien, a partir de tablas de frecuencias (tablas de datos agrupados)”.

Imagen 2.4 Estadística-descriptiva. (Valdez, I. y Gómez, D., 2019).

También es importante conocer si se disponen de todos los datos de una población, o si sólo se disponen de datos pertenecientes a una muestra.

Las medidas que describen a una población se denominan **parámetros**, mientras que las medidas de una muestra se denominan **estadísticos** (Valdez. I y Gómez, M., 2019.

Medidas de tendencia central

Aún cuando no lo percibas, hace tiempo que utilizas las medidas de tendencia central, por ejemplo, cuando estás en la escuela y es fin de cursos, generalmente obtienes tu promedio final de calificaciones, tu promedio en cada periodo o por materia, también obtienes la calificación que apareció con más frecuencia en tu boleta, entre otros ejemplos. Inevitablemente comparas tus calificaciones con las del grupo, cuántos aprobaron y quiénes reprobaron. Podemos decir que de manera empírica trabajas con la estadística, realizas cálculos estadísticos en tu vida, específicamente, usas las medidas de tendencia central.

Las medidas de tendencia central permiten calcular valores a partir datos y resumirlos en un solo valor para su interpretación. Representan un centro en torno al cual se encuentra ubicado el conjunto de datos (Revista Biomédica, 2011). Las más utilizadas son: la media aritmética, la mediana y la moda.

Veamos el concepto y las fórmulas para calcular estas medidas para datos no agrupados. Si necesitaras calcular las medidas de tendencia central para datos agrupados, puedes consultar el formulario de la página 259 de esta guía.

Media aritmética

También conocida como promedio, es la suma de todos los valores observados dividido entre el total de datos obtenidos. Su fórmula es:

Para una población	Para una muestra
$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$	$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$
Donde: μ = media aritmética de la población x_i = valor i-ésimo de la población N = Número de datos de la población	Donde: $\bar{X}$ = media aritmética de la muestra x_i= valor i-ésimo de la muestra n= Número de datos de la muestra

Ejemplos:

a) Los siguientes datos representan el número de visitas hechas al museo en los últimos 6 meses: 10, 5, 9, 7, 6, 4. Calcula la media aritmética de los datos.
   Media= (10+5+9+7+6+4) / 6
   Media= 41/6
   Media =6.8
b) Los siguientes datos representan el número de medios de transporte público que utilizan diez estudiantes para trasladarse a la escuela (metro, taxi, metro bus, etc.). Calcula la media aritmética.
   1, 1, 2, 2, 2, 2, 2, 3, 3, 3
   Media= (1+1+2+2+2+2+2+3+3+3) /10

Media= 21/10
Media= 2.1
Los diez estudiantes no pueden utilizar 2.1 medios de transporte, pero en promedio utilizan 2.

Mediana

Representa el valor central de la variable cuando los datos están ordenados. Para determinar la mediana debes ordenar los datos de mayor a menor o viceversa. La mediana es el dato que se encuentra a mitad de la lista. Si el número de datos es par, entonces la mediana será la media de los dos datos que se encuentran a la mitad de la lista.

En los siguientes ejemplos se obtiene la mediana de la muestra de la talla de zapatos de alumnos de sexto año de bachillerato:

a) Si la muestra es **impa**r (11 números)
21, 23.5, 23.5, 23.5, 24, **24**, 24, 24, 25, 25, 26
Mediana=24

b) Si la muestra es **par** (12 números), la mediana es la media de los dos datos que se encuentran a la mitad.
21, 21, 23.5, 23.5, 23.5, **24, 24**, 24, 24, 25, 25, 26
24+24= 48
48/2 = 24
Mediana=24

Moda

Es el valor de los datos que ocurre con mayor frecuencia, puede ser un solo dato (una sola moda) o bien pueden ser bimodales, trimodales y multimodales, si la mayor frecuencia se repite en los datos dos, tres o más veces.

Ejemplos:

a) Continuando con el ejemplo de la talla de zapatos de los alumnos de sexto año de nivel bachillerato:

21, 21, 23.5, 23.5, 23.5, **24, 24, 24, 24**, 25,25, 26

La talla 24 es la que se repite más veces, aparece en 4 ocasiones. La moda en talla de zapatos de los alumnos de sexto año es 24.

b) A continuación, se muestran las estaturas de 14 alumnas de cuarto año de bachillerato.

1.50, 153, **1.60, 1.60, 1.60**, 1.61, **1.62, 1.62, 1.62**, 165,165, 1.67, 1,69, 1.70

En este caso, observamos que las estaturas 1.60 y 1.62, tienen la misma frecuencia (tres veces) y es la mayor frecuencia de los datos, entonces se puede afirmar que hay dos modas.

Ejercicio 2.6

a) Capítulos de series vistos por día. Se aplicó a un grupo de 10 estudiantes de sexto año de bachillerato un cuestionario, del que se obtuvo el número de capítulos que ven en un día de la serie televisiva de moda. Los capítulos tienen una duración de media hora. Los resultados obtenidos se muestran en la siguiente tabla. Ordena la muestra y obtén las medidas solicitadas.

Obtén la media, mediana y moda de los datos propuestos.

Muestra	5	3	4	3	5	2	6	4	4	2
Muestra ordenada										
Media:										
Mediana:										
Moda:										

b) Número de libros leídos al mes. Diez de tus compañeros de clase, desarrollaron una tabla en la que listan el número de libros que han leído durante el mes (incluyendo los que les dejaron de tarea). Ordena la muestra y obtén las medidas solicitadas.

Muestra	6	4	7	5	6	4	1	6	3	2
Muestra ordenada										
Media:										
Mediana:										
Moda:										

c) Durante el ciclo escolar se le solicitó a un grupo de sexto año que realizara varias visitas académicas a museos dentro de la ciudad. Estas fueron el número de visitas realizadas. Ordena la muestra y obtén las medidas solicitadas.

Muestra	10	12	10	8	9	12	10	6	5	9
Muestra ordenada										
Media:										
Mediana:										
Moda:										

Medidas de dispersión

Las medidas de dispersión indican, de manera general, cómo se alejan o acercan los datos respecto de la media aritmética. Las más utilizadas son: el rango, la varianza y la desviación estándar. A continuación, se presentan los conceptos y fórmulas para calcularlos cuando los datos están sin agrupar. Si deseas consultar las fórmulas para datos agrupados, puedes acudir al formulario de la página 259.

Rango

Es la diferencia entre el valor mayor y el valor menor del conjunto de datos.

$$Rango = x_{max} - x_{min}$$

Donde:
x_{max} = es el dato de mayor valor del conjunto de datos
x_{min} = es el dato de menor valor del conjunto de datos

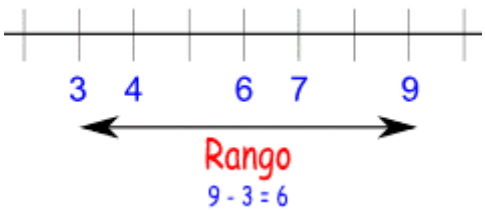


Imagen 2.5 Rango. (Disfruta las matemáticas, 2011).

Varianza

Es la medida que cuantifica la variabilidad de los datos respecto al valor de la media, es decir, es el promedio del cuadrado de las distancias entre cada observación y la media aritmética del conjunto de datos.

La fórmula para calcular la varianza de datos no agrupados es:

Para una población	Para una muestra
$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$	$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n-1}$
Donde: σ^2 = varianza de la población μ = media aritmética de la población x_i = valor i-ésimo de la población N = Número de datos de la población	Donde: S^2 = varianza de la muestra $\bar{X}$ = media aritmética de la muestra x_i= valor i-ésimo de la muestra n= Número de datos de la muestra

Desviación estándar

También se le conoce como desviación típica. Se calcula obteniendo la raíz cuadrada de la varianza.

Para una población	Para una muestra
$\sigma = \sqrt{\sigma^2}$	$S = \sqrt{S^2}$
Donde: σ = desviación estándar de la población σ^2 = varianza de la población	Donde: S = desviación estándar de la población S^2 = varianza de la muestra

Medidas de forma

Se utilizan para tener una idea general de la forma de la gráfica de los datos observados. Nos permiten observar la manera en que se separan o concentran los valores de acuerdo a su representación gráfica. La utilidad de estas medidas es que se pueden identificar las características de la distribución sin necesidad de generar el gráfico. Las medidas de dispersión son el coeficiente de sesgo y el coeficiente de curtosis.

Coeficiente de sesgo

Se usa para medir el grado asimetría de un conjunto de datos. Se considera que los datos están distribuidos simétricamente cuando existe el mismo número de observaciones a la derecha y a la izquierda de la media aritmética del conjunto de datos. Hay asimetría positiva si existen más observaciones del lado derecho de la media y asimetría negativa si existen más observaciones del lado izquierdo de la muestra.

El coeficiente de sesgo para datos no agrupados se calcula con la siguiente fórmula:

Para una población	Para una muestra
$\alpha_3 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3}{\sigma^3}$	$a_3 = \frac{n}{(n-1)(n-2)}\frac{\sum_{i=1}^{n}(x_i - \bar{X})^3}{S^3}$
Donde: α_3 = coeficiente de sesgo de la población σ = desviación estándar de la población μ = media aritmética de la población x_i = valor i-ésimo de la población N = Número de datos de la población	Donde: a_3 = coeficiente de sesgo de la muestra S = desviación estándar de la muestra $\bar{X}$ = media aritmética de la muestra x_i= valor i-ésimo de la muestra n= Número de datos de la muestra

Interpretación del coeficiente de sesgo

Si $\alpha_3 = 0$ o $a_3 = 0$, entonces la distribución es simétrica con respecto a la media.
Si $\alpha_3 < 0$ o $a_3 < 0$, entonces la distribución de los datos está sesgada a la izquierda.
Si $\alpha_3 > 0$ o $a_3 > 0$, entonces la distribución de los datos está sesgada a la derecha.

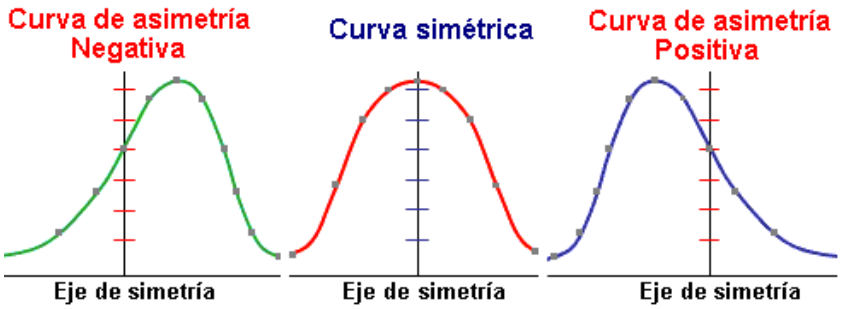


Imagen 2.6 Tipos de asimetrías de un conjunto de datos. (Martínez, A., 2015).

Coeficiente de curtosis

La curtosis es una medida de apuntalamiento. Una curtosis grande (picuda) indica que la dispersión de los datos es pequeña, mientras que una curtosis pequeña (aplanada) indica que los datos están más dispersos.

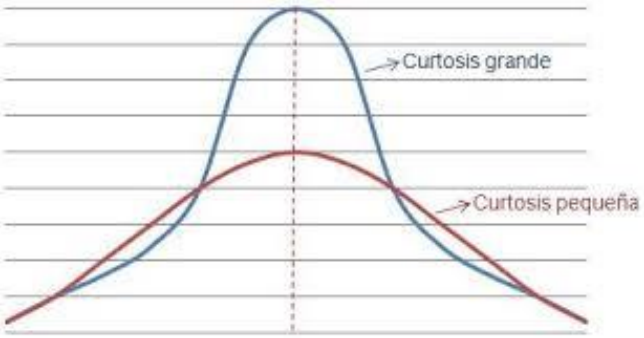


Imagen 2.7 Curtosis. (Universo Fórmulas, s/f).

El coeficiente de curtosis para datos no agrupados se calcula con la siguiente fórmula:

Para una población	Para una muestra
$\alpha_4 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4}{\sigma^4}$	$a_4 = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \frac{\sum_{i=1}^{n}(x_i - \bar{X})^4}{S^4}$
Donde: α_4 = coeficiente de curtosis de la población σ = desviación estándar de la población μ = media aritmética de la población x_i = valor i-ésimo de la población N = Número de datos de la población	Donde: a_4 = coeficiente de curtosis de la muestra S = desviación estándar de la muestra $\bar{X}$ = media aritmética de la muestra x_i= valor i-ésimo de la muestra n= Número de datos de la muestra

Interpretación del coeficiente de curtosis

Si $\alpha_4 = 3$ o $a_4 = 3$, la distribución de datos es mesocúrtica (ni aplanada, ni picuda)
Si $\alpha_4 < 3$ o $a_4 < 3$, la distribución de datos es platicúrtica (muy aplanada)
Si $\alpha_4 > 3$ o $a_4 > 3$, la distribución de datos es leptocúrtica (muy picuda)

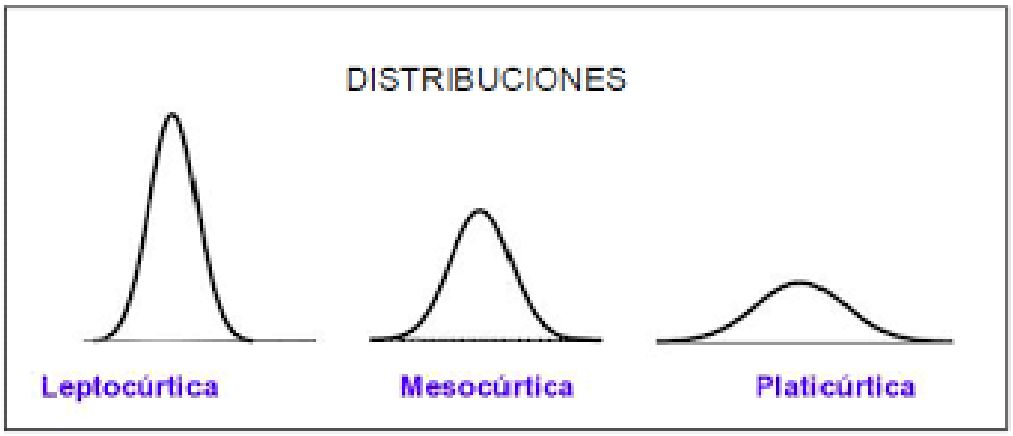


Imagen 2.8 Distribuciones. (Glosario de términos estadísticos, 2018).

Tabla de distribución de frecuencias

Es una herramienta que permite organizar los datos mediante una tabla que muestra la distribución de los datos con base en sus frecuencias. Veamos cómo se construye la tabla de frecuencias para datos no agrupados:

Valores de los datos	Frecuencia Absoluta f_i	Frecuencia Relativa f_r	Frecuencia Acumulada F_i	Frecuencia Relativa Acumulada F_r	Frecuencia Porcentual $f_{\%}$	Frecuencia Porcentual acumulada $F_{\%}$
Datos que forman la encuesta o estudio	Cantidad de veces que aparece un valor en un estudio	Es la fracción o proporción de elementos con relación al total, es decir se divide la frecuencia absoluta entre el total de elementos	Es el acumulado de las frecuencias absolutas	Es el acumulado de las frecuencias relativas	Es el porcentaje de veces que aparece un valor en el estudio respecto al total. FP = (Frecuencia absoluta/Total) * 100 o FP = Frecuencia Relativa * 100%	Es el acumulado de la frecuencia porcentual

Tabla 2.1. Frecuencias de datos no agrupados

Ejemplo:

De acuerdo a un grupo de jóvenes varones de 17 años, se ha obtenido una muestra de 44 datos, que corresponde a la estatura de cada uno de ellos. Se pretende identificar las tallas que requieren producir las marcas de ropa para esa población. Para poder tomar este tipo de decisión es conveniente apoyarse de algunas medidas estadísticas.

La siguiente tabla muestra los datos no agrupados:

160	161	157	162
161	179	169	178
188	187	180	186
184	156	190	175
183	188	174	156
167	157	170	158
161	166	177	178
184	181	189	171
189	184	159	171
161	180	188	190
188	170	164	178

Tabla 2.2. Estaturas de 44 jóvenes.

Procedamos a elaborar la tabla de frecuencias, organizando los datos en orden ascendente y calculando las frecuencias correspondientes a cada valor del conjunto de datos:

Valores de los datos	Frecuencia Absoluta f_i	Frecuencia Relativa f_r	Frecuencia Relativa Acumulada F_r	Frecuencia Acumulada F_i	Frecuencia Porcentual acumulada $F_{\%}$	Frecuencia Porcentual $f_{\%}$
156	2	0.045	0.045	2	0.045	4.545
157	2	0.045	0.091	4	0.091	9.091
158	1	0.023	0.114	5	0.114	11.364
159	1	0.023	0.136	6	0.136	13.636
160	1	0.023	0.159	7	0.159	15.909
161	4	0.091	0.250	11	0.250	25.000
162	1	0.023	0.273	12	0.273	27.273
164	1	0.023	0.295	13	0.295	29.545
166	1	0.023	0.318	14	0.318	31.818
167	1	0.023	0.341	15	0.341	34.091
169	1	0.023	0.364	16	0.364	36.364
170	2	0.045	0.409	18	0.409	40.909
171	2	0.045	0.455	20	0.455	45.455
174	1	0.023	0.477	21	0.477	47.727
175	1	0.023	0.500	22	0.500	50.000
177	1	0.023	0.523	23	0.523	52.273
178	3	0.068	0.591	26	0.591	59.091
179	1	0.023	0.614	27	0.614	61.364
180	2	0.045	0.659	29	0.659	65.909
181	1	0.023	0.682	30	0.682	68.182
183	1	0.023	0.705	31	0.705	70.455
184	3	0.068	0.773	34	0.773	77.273
186	1	0.023	0.795	35	0.795	79.545
187	1	0.023	0.818	36	0.818	81.818
188	4	0.091	0.909	40	0.909	90.909
189	2	0.045	0.955	42	0.955	95.455
190	2	0.045	1.000	44	1.000	100.000

Tabla 2.3. Frecuencia de estaturas (datos no agrupados).

Ahora, mostraremos el procedimiento para elaborar la tabla de distribución de frecuencias agrupando los datos:

Intervalo o clase i	Límite inferior $Linf_i$	Límite superior $Lsup_i$	Frecuencia Absoluta f_i	Frecuencia Relativa f_r	Frecuencia Acumulada F_i	Frecuencia Relativa Acumulada F_r	Frecuencia Porcentual $f_{\%}$	Frecuencia Porcentual acumulada $F_{\%}$
Conse-cutivo del intervalo o clase	Datos que forman en el intervalo inferior	Datos que forman en el intervalo superior	Cantidad de veces que aparece un valor (entre los valores establecidos) en un estudio	Es la fracción o proporción de la frecuencia absoluta entre el total de elementos	Es el acumulado de las frecuencias absolutas (columna 4)	Es el acumulado de las frecuencias relativas	Es el porcentaje de veces que aparece un valor en el estudio respecto al total FP = (Frecuencia absoluta/Total) * 100 o FP = Frecuencia Relativa * 100%	Es el acumulado de la frecuencia porcentual

Tabla 2.4. Tabla de frecuencias de datos agrupados.

Para ejemplificar la elaboración de una tabla de frecuencias de datos agrupados, tomaremos los datos de la Tabla 2.3 mostrada con anterioridad.

Primero, calcularemos el número de intervalos o clases, que depende del número de datos. A una mayor cantidad de datos se requiere un número mayor de intervalos. Por lo general, el número de intervalos se recomienda entre 5 y 15.

Aunque no existe una regla formal para determinar el número de clases y el tamaño de éstos, existen reglas empíricas que resultan útiles en esta decisión. Una de ellas es la Regla de Sturges:

$$k = 1 + 3.3\log_{10}(n)$$

Donde:
k=número de clases
n=total de datos

Así, para nuestro ejemplo, el número de clases se calcularía:

$k=1+3.3\log_{10}(44)$
$k=6.42$

Una vez calculado el número de clases, procedemos a calcular la longitud de cada clase mediante la fórmula:

$$c = \frac{rango\ de\ datos}{k}$$

El rango de datos se calcula restando el valor máximo**,** es decir, el valor más alto representativo de todos los datos y el valor mínimo**,** esto es, el valor más bajo representativo de todos los datos de la muestra.

Rango = Valor máximo – valor mínimo
Rango = 190 -156
Rango=34

En este caso, la longitud de la clase se calcularía: C=34/6.42=5.29. Para cuestiones prácticas, por ser los datos la estatura de los jóvenes, determinaremos intervalos de longitud 5 unidades.

Así, la tabla de frecuencias tendrá 7 clases (k=6.42) y la longitud de clase será de 5 (c=5.29). Con estos datos, calculamos el límite inferior y superior de cada clase, así como sus frecuencias respectivas. La tabla de frecuencia de los datos agrupados sería:

Clase	Estaturas		f_i	f_r	F_i	F_r	$F_{\%}$
	Lim_{Inf}	Lim_{sup}					
1	156	160	7	0.159	7	0.159	15.909
2	161	165	6	0.136	13	0.295	29.545
3	166	170	5	0.114	18	0.409	40.909
4	171	175	4	0.091	22	0.500	50.000
5	176	180	7	0.159	29	0.659	65.909
6	181	185	5	0.114	34	0.773	77.273
7	186	190	10	0.227	44	1	100
			44	1			

Tabla 2.5. Frecuencia de estaturas (datos agrupados)

Analicemos otro ejemplo de construcción de tablas de frecuencia de datos:

Carlos es un estudiante de la carrera de Enfermería y se encuentra haciendo su servicio social. Su jefe de turno le muestra la siguiente tabla de frecuencias absolutas sobre el peso de las personas y le pide determinar y graficar la frecuencia acumulada y complementaria:

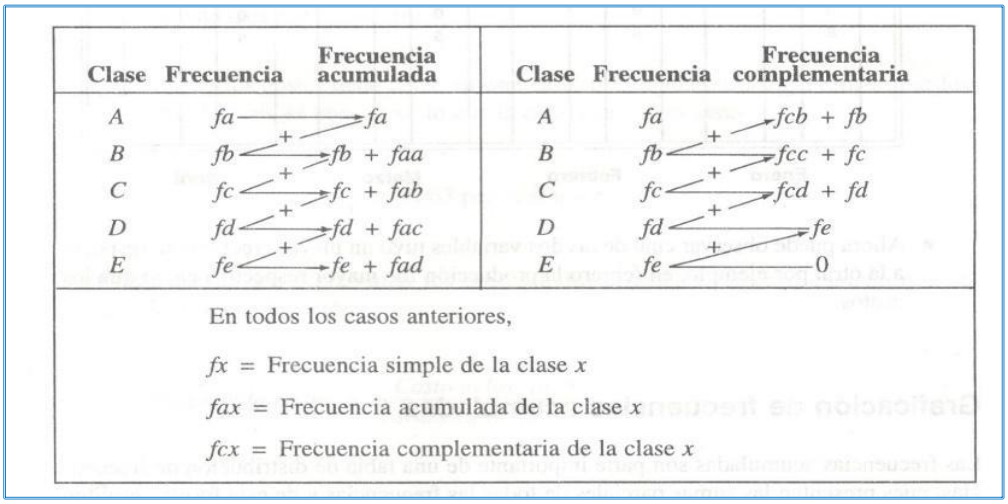
Clase	Límite Inferior	Límite Superior	Límite Inferior Exacto	Límite Superior Exacto	Marca	Frecuencia
A	40	59	39.5	59.5	49.5	385
B	60	79	59.5	79.5	69.5	292
C	80	99	79.5	99.5	89.5	475
D	100	119	99.5	119.5	109.5	129
E	120	139	119.5	139.5	129.5	48

Tabla 2.6. Frecuencias absolutas del peso de las personas

Recuerda que la Frecuencia Acumulada *es conocida como* ***fa****, frecuencia acumulada ascendente define los puntos de la gráfica ascendente y permite contestar preguntas del tipo ¿Cuántos elementos de la población o muestra en estudio tienen menos de cierto nivel de la variable que se analiza?*

La Frecuencia Complementaria *es conocida como* ***fc****, frecuencia acumulada descendiente define los puntos de la gráfica descendente.*

La forma de obtenerlas es haciendo los siguientes cálculos, observa la imagen siguiente que muestra el procedimiento para determinar la frecuencia acumulada y complementaria:



Clase	Frecuencia	Frecuencia acumulada
A	*fa*	*fa*
B	*fb*	*fb* + *faa*
C	*fc*	*fc* + *fab*
D	*fd*	*fd* + *fac*
E	*fe*	*fe* + *fad*

Clase	Frecuencia	Frecuencia complementaria
A	*fa*	*fcb* + *fb*
B	*fb*	*fcc* + *fc*
C	*fc*	*fcd* + *fd*
D	*fd*	*fe*
E	*fe*	0

En todos los casos anteriores,

fx = Frecuencia simple de la clase *x*

fax = Frecuencia acumulada de la clase *x*

fcx = Frecuencia complementaria de la clase *x*

Imagen 2.9 Frecuencia acumulada y complementaria (Sánchez, 1996).

Es decir:

Frecuencia acumulada de clase	=	*Frecuencia acumulada de la clase anterior*	+	*Frecuencia de la clase*

Frecuencia acumulada de la primera clase	=	*Frecuencia de la primera clase*

Frecuencia complementaria de clase	=	*Frecuencia acumulada de la última clase*	-	*Frecuencia acumulada de la clase*

Por lo tanto:

Frecuencia Acumulada de la clase A = 385
Frecuencia Acumulada de la clase B = 385 + 292 = 677
Frecuencia Acumulada de la clase C = 677 + 475 = 1152
Frecuencia Acumulada de la clase D = 1152 + 129 = 1281
Frecuencia Acumulada de la clase E = 1281 + 48 = 1329

Y la frecuencia complementaria:

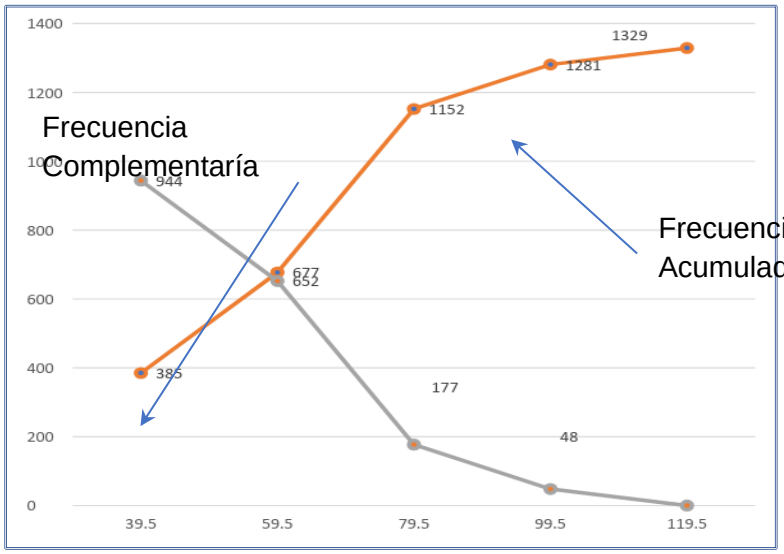
Frecuencia complementaría de la clase A = 1329 – 385 = 944
Frecuencia complementaría de la clase B = 1329 – 677 = 652
Frecuencia complementaria de la clase C = 1329 – 1152 = 177
Frecuencia complementaria de la clase D = 1329 – 1281 = 48
Frecuencia complementaría de la clase E = 1329 – 1329 = 0

La tabla de Frecuencias acumulada y complementaria es la siguiente:

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaría
A	39.5	59.5	385	385	944
B	59.5	79.5	292	677	652
C	79.5	99.5	475	1152	177
D	99.5	119.5	129	1281	48
E	119.5	139.5	48	1329	0
TOTAL			1329		

Tabla 2.7. Frecuencias acumulada y complementaria

Y al graficarlas, podemos observar el comportamiento de cada una de ellas.



Gráfica 2.1. Gráfica resultante de la Frecuencia acumulada y complementaria

Ahora, el subdirector del turno matutino le pide a Carlos que, con base en la siguiente tabla de frecuencias acumuladas y complementarias, determine la frecuencia relativa de una muestra de datos referente a la estatura de las personas que se presentaron a consulta en un día normal en la unidad clínica familiar.

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia relativa	Frecuencia Acumulada	Frecuencia Complementaría
A	-------	120.0		0.0216	0.9784
B	1.21	1.50		0.2151	0.7849
C	1.51	1.70		0.8377	0.1623
D	1.71	180		0.9485	0.0515
E	1.81	------		1.000	0
TOTAL					

Tabla 2.8. Frecuencias relativas, acumuladas y complementarias

La frecuencia relativa *es el cociente entre la frecuencia absoluta (frecuencia) de un determinado valor o clase y el número total de datos, la suma total de la frecuencia relativa te da el 100 por ciento o 1*.

Clase o Intervalo de clase *son divisiones o categorías en las cuales se agrupan un conjunto de datos ordenados con características comunes. En otras palabras, son fraccionamientos del rango o recorrido de la serie de valores para reunir los datos que presentan valores comprendidos entre dos límites.*

Para organizar los valores de la serie de datos hay que determinar un número de clases que sea conveniente. En otras palabras, que ese número de intervalos no origine un número pequeño de clases ni muy grande. Un número de clases pequeño puede ocultar la naturaleza natural de los valores y un número muy alto puede provocar demasiados detalles como para observar alguna información de gran utilidad en la investigación.

Para resolver este problema, Carlos solo despeja de la ecuación la *Frecuencia de la Clase*:

Frecuencia acumulada de clase	=	*Frecuencia acumulada de la clase anterior*	+	*Frecuencia de la clase*

Quedando de la siguiente forma:

Frecuencia de la Clase	=	*Frecuencia acumulada de la clase*	-	*Frecuencia acumulada de la clase anterior*

Frecuencia de la clase A = 0.0216
Frecuencia de la clase B = 0.2151 – 0.0216 = 0.1935
Frecuencia de la clase C = 0.8377 – 0.2151 = 0.6226
Frecuencia de la clase D = 0.9485 – 0.8377 = 0.1108
Frecuencia de la clase E = 1.000 – 0.9485 = 0.0515

Recuerda que la frecuencia acumulada de la primera clase es la misma que la frecuencia relativa en este caso, por lo tanto, la tabla completa queda:

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia relativa	Frecuencia Acumulada	Frecuencia Complementaria
A	-------	120.0	0.0216	0.0216	0.9784
B	1.21	1.50	0.1935	0.2151	0.7849
C	1.51	1.70	0.6226	0.8377	0.1623
D	1.71	180	0.1108	0.9485	0.0515
E	1.81	------	0.0515	1.000	0
TOTAL			1.000		

Tabla 2.9. Frecuencias relativas, acumuladas y complementarias

Ejercicio 2.7

Para reafirmar estos conocimientos de tablas de frecuencias o distribución de frecuencias, realiza los siguientes ejercicios:

a) Indaga los precios de computadoras de las siguientes marcas Dell, Azuz, Hp, Lenovo y Acer. Realiza la tabla de frecuencia de cuántas se han vendido el fin de semana en un establecimiento dedicado a su venta.

b) Se están haciendo encuestas de la frecuencia de visitas a las cafeterías para conocer su popularidad. Se obtuvieron los siguientes datos. Realiza la tabla de frecuencia.

Edad (años)		Número de Visitas
límite inferior	límite superior	
10	12	40
13	15	17
16	18	40
19	23	17
24	26	38
27	30	24
31	25	37
36	40	24
41	45	11
46	50	27
51		7

Ejercicio 2.8

a) Determina la frecuencia acumulada, complementaria y realiza la gráfica de la siguiente tabla de distribución de frecuencias:

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia Relativa
A	37.7	50.2	23
B	50.3	62.8	19
C	62.9	75.4	22
D	75.5	88	21

b) La siguiente tabla muestra la distribución de sueldos de una empresa, determina la frecuencia complementaria.

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia Relativa
1	$ 360.0	$ 399.0	258
2	$ 400.0	$ 439.0	261
3	$ 440.0	$ 479.0	162
4	$ 480.0	$ 519.0	301
5	$ 520.0	$ 559.0	321
6	$ 560.0	$599.0	341
7	$ 600.0	$599.0	208

Distribución normal

Muchos de los fenómenos naturales y cotidianos al ser estudiados se ha comprobado que tienen una distribución muy aproximada a una forma de campana, a la que se le denomina distribución normal o distribución de Gauss (Pértegaz y Fernández, 2001). Por lo anterior, muchas pruebas estadísticas se realizan tomando como parámetro este tipo de distribución.

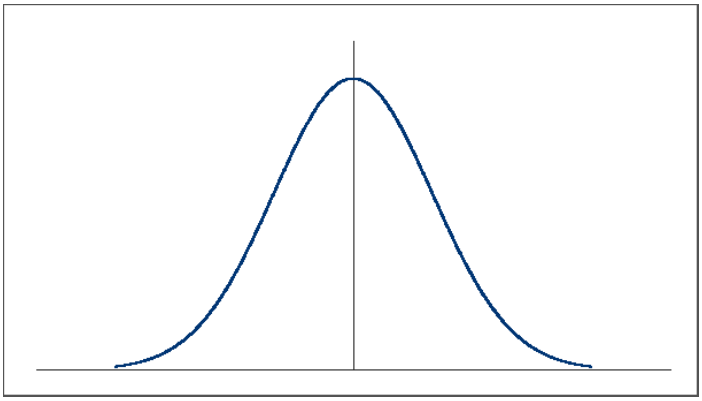


Imagen 2.10 Curva normal. (López, A., s/f)

Pruebas de normalidad

Son pruebas que se utilizan para determinar si un conjunto de datos se distribuye de una manera consistente con una distribución normal. Se mide el grado de ajuste de un modelo normal de datos, si el ajuste se aleja del centro de la campana de Gauss, entonces los datos no están modelados con respecto a una distribución normal. Se pueden realizar con variables cuantitativas u ordinales como la prueba de normalidad Kolmogorov-Smirnov.

Análisis de varianza 

Es un método que utiliza las varianzas para comparar medidas de diferentes grupos o conjunto de datos, establecer si las medidas son significativamente diferentes o manejan valores parecidos. Con el análisis de varianza es posible determinar si diversos conjuntos de muestras aleatorias de una determinada variable proceden de la misma población o de poblaciones distintas.

Lo que nos indica el valor de la varianza es si los valores que se están midiendo se encuentran cerca o dispersos en relación a la curva propuesta como media. Si el resultado de la medición es pequeño, indica que los valores están agrupados, si el resultado de la varianza es grande, significa que los datos están dispersos. Esto nos da a conocer que, si se está haciendo un estudio estadístico, la varianza nos indica qué tanto se están ajustando los datos a la función propuesta como solución del estudio.

Análisis de correlación 

En estadística, la correlación se refiere a la existencia de un vínculo o dependencia entre varios eventos. Una de las herramientas que permite inferir si existe dicha dependencia es el análisis de correlación. Su propósito es indicar si existe relación entre dos eventos (dos variables) y determinar el grado de relación que tienen.

Para realizar el análisis correlacional se debe calcular el **coeficiente de correlación,** que es la forma numérica con la que es posible evaluar la relación que existe entre dos o más variables, es decir, mide la dependencia de una variable con respecto a otra. Si los valores de una variable se modifican con respecto a los valores de la otra, se dice que ambas variables se encuentran correlacionadas.

El coeficiente de correlación lineal se calcula mediante la siguiente fórmula:

$$r = \frac{S_{xy}}{S_x S_y}$$

Donde:
r = coeficiente de correlación lineal
S_x = desviación estándar de la variable x

S_y = desviación estándar de la variable y
S_{xy} = Covarianza entre las variables xy

La covarianza S_{xy} se calcula con la fórmula:

$$S_{xy} = \frac{\sum_{i=1}^{n} f_i x_i y_i}{n} - \bar{X}\bar{Y}$$

Donde:
S_{xy} = Covarianza entre las variables xy
f_i = frecuencia absoluta del producto de $x_i y_i$
x_i = valor i-ésimo de la variable x
y_i = valor i-ésimo de la variable y
$\bar{X}$ = media aritmética de la variable x
$\bar{Y}$ = media aritmética de la variable y
n = número de datos de la muestra

Interpretación del coeficiente de correlación lineal

El valor del coeficiente de correlación lineal puede variar entre -1 y +1, y se interpreta de la siguiente forma:

Si r = -1, entonces hay una correlación perfecta negativa
Si r = 0, entonces no hay correlación
Si r = +1, entonces hay una correlación perfecta positiva

Una correlación positiva o directa significa que, si el valor de *x* aumenta, el valor de *y* aumentará con la misma intensidad. Una correlación negativa o inversa significa que siempre que el valor de *x* aumenta, el valor de y disminuye con la misma intensidad. Es importante señalar que esto no quiere decir que lo hagan en la misma proporción, ya que esto ocurre sólo si ambas variables tienen la misma desviación estándar.

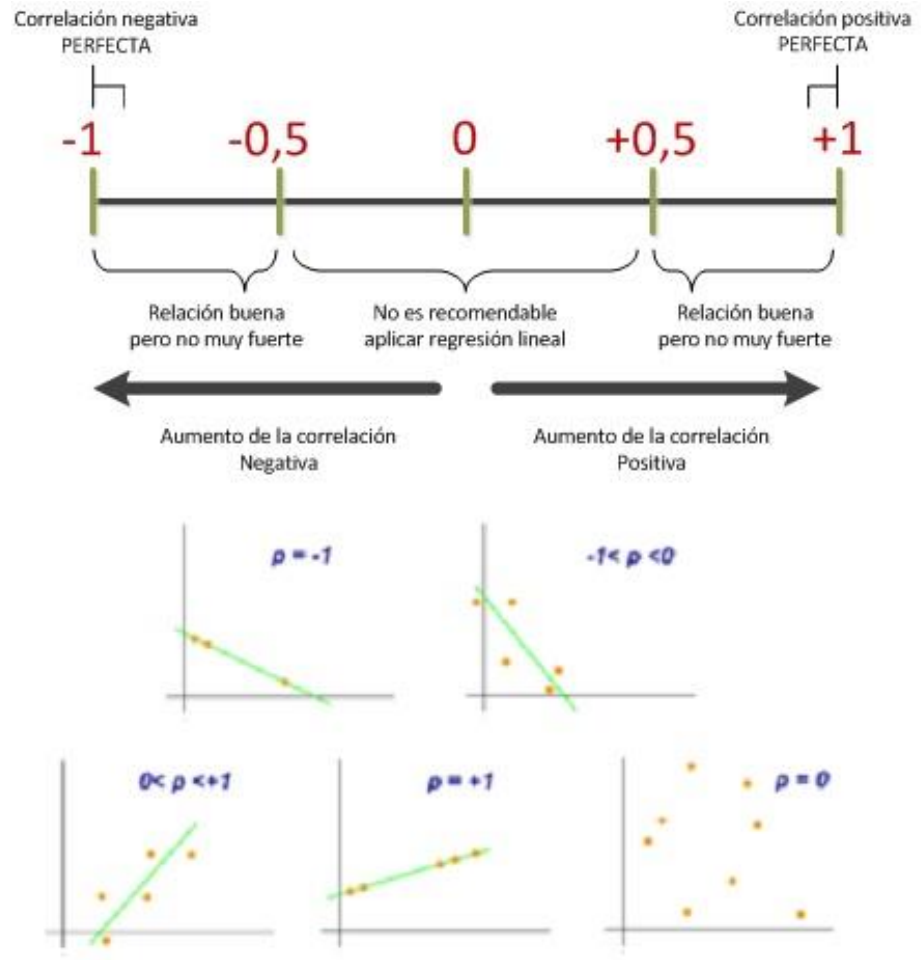


Imagen 2.11 Coeficiente de correlación. (Buján, A., 2017)

Ejemplo:

Las calificaciones de 12 alumnos de la clase de informática y matemáticas son:

Informática	Matemáticas
2	1
3	3
4	2
4	4
5	4
6	4
6	6
7	4
7	6
8	7
10	9
10	10

Hallar el coeficiente de correlación de la distribución e interpretarlo.

x_i	y_i	$x_i\ y_i$	x_i^2	y_i^2	$(x_i - \bar{X})^2$	$(y_i - \bar{Y})^2$
2	1	2	4	1	16	16
3	3	9	9	9	9	4
4	2	8	16	4	4	9
4	4	16	16	16	4	1
5	4	20	25	16	1	1
6	4	24	36	16	0	1
6	6	36	36	36	0	1
7	4	28	49	16	1	1
7	6	42	49	36	1	1
8	7	56	64	49	4	4
10	9	90	100	81	16	16
10	10	100	100	100	16	25
72	60	431	504	380	72	80

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{72}{12} = 6 \qquad \bar{Y} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{60}{12} = 5$$

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{X})^2}{N}} = \sqrt{\frac{72}{12}} = \sqrt{6} = 2.45 \qquad \sigma_y = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \bar{Y})^2}{N}} = \sqrt{\frac{80}{12}} = \sqrt{6.66} = 2.58$$

$$S_{xy} = \frac{431}{12} - 6 * 5 = 35.91 - 30 = 5.92$$

$$r = \frac{S_{xy}}{S_x S_y} = \frac{5.92}{2.45 * 2.58} = \frac{5.92}{6.321} = 0.94$$

Como r es positivo, la correlación es directa y como su valor está muy cercano a 1, significa que la correlación entre las variables es muy fuerte. Por lo anterior, podemos concluir que la calificación obtenida por un alumno en la materia de informática está correlacionada directamente con la calificación que obtiene en matemáticas.

Variables bivariadas

Se le llama así al conjunto de datos observados al mismo tiempo los cuales forman caracteristicas propias nombrados A y B, que se representan por un conjunto de parejas ordenadas (x, y).

Se colocan en una matriz en la que los renglones representan las características de las x y las columnas las características de las y, por ejemplo:

Variable y				
Variable x	x/y	Y_1	Y_2	Y_3
	X_1	F_{11}	F_{12}	F_{13}
	X_2	F_{21}	F_{22}	F_{23}
	X_3	F_{31}	F_{32}	F_{33}

Donde f_{xy} son las frecuencias de las variables bivariadas

Lo importante de estas variables es que se deben analizar en conjunto.

Ejemplo:

Se ha obtenido una muestra de las estaturas de un grupo de 66 jóvenes varones de 17 años para identificar las tallas que requieren producir las marcas de ropa para esa población.

La tabla de datos es la siguiente:

Estaturas de jóvenes en centímetros					
160	161	162	157	187	174
161	179	178	169	173	187
188	187	186	180	185	158
184	156	175	190	160	159
183	188	156	174	181	176
167	157	158	170	187	157
161	166	178	177	173	157
184	181	171	189	187	172
189	184	171	159	181	163
161	180	190	188	159	179
188	170	178	164	189	185

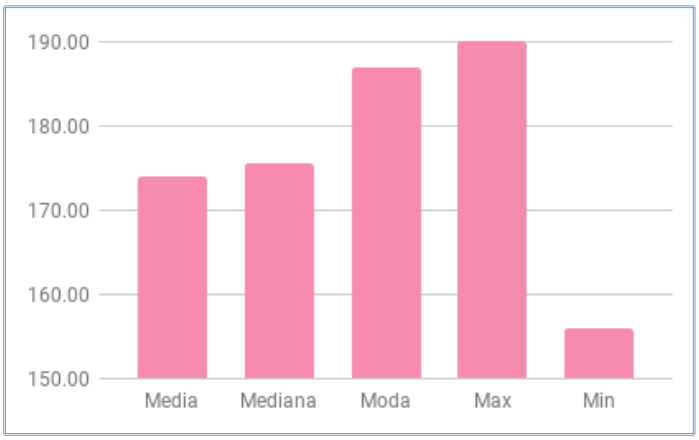
Tabla 2.10. Estaturas de 66 jóvenes.

Calculamos las siguientes medidas

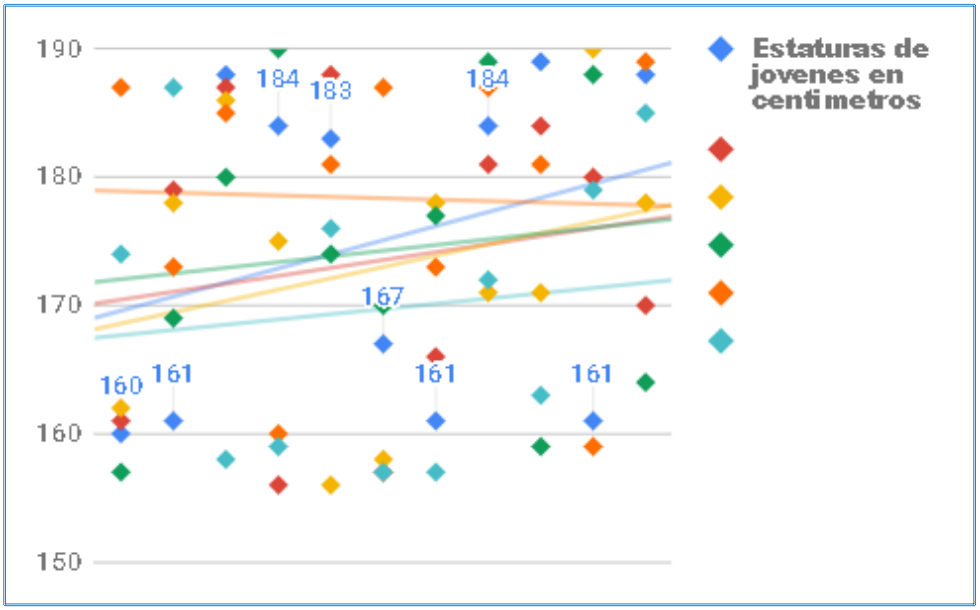
Media	Mediana	Moda	Max	Min
174	175.5	187	190	156

En estas medidas se observa que en la muestra estudiada la estatura mínima es 156 cm, y la máxima es 190 cm. La media es de 174 cm.

La siguiente gráfica muestra estas medidas de tendencia central.



Gráfica 2.2. Estaturas



Gráfica 2.3. Estaturas de jóvenes en centímetros

En la Gráfica 2.3, aunque el promedio es de 174 cm, se observa que la tendencia predominante, que son las líneas de colores, es de 178 cm (línea de tendencia amarilla).

Por lo anterior, concluimos que la tendencia de estatura es de 178 cm, por lo que se tienen que diseñar tallas más grandes de la media que se está identificando en los datos, pero sin dejar de lado a las personas por debajo de la media de estatura actual. Por lo tanto, se debe considerar y empezar a confeccionar tallas para gente más alta.

Preguntas de reflexión

Para poder tomar decisiones asertivas es necesario hacerlo de forma reflexiva, por lo cual, es importante hacer cálculos para determinar la mejor opción, para esto, los datos estadísticos nos ayudan a obtener elementos numéricos que ayudan a este tipo de decisiones.

Si tienes que tomar una decisión de cómo obtener las mejores notas de tus próximos estudios, para concluir en el tiempo que te estipula la carrera. ¿Qué es lo que harás?

_____
_____

¿Tomarías en cuenta cuántos semestres tienen tus estudios o el tiempo máximo en la cual puedes cursarla?

_____
_____

En el caso de que tomes en cuenta los semestres del plan de estudios ¿cuáles son las ventajas y desventajas?

_____
_____

Y si no, ¿cuáles son las ventajas y desventajas?

_____
_____

En todos los casos ¿qué tipo de conceptos estás utilizando?

_____
_____

Contenidos conceptuales	**Contenidos procedimentales**
2.3 Presentación y evaluación de datos estadísticos con Tablas de datos y Gráficas de pastel, barras e histogramas.  Data Analysis (depositphotos, 2018). Recuperado de: https://goo.gl/umvFPh	2.4 Recopilación de datos en una investigación académica. 2.5 Elaboración de inferencias sobre la realidad a partir de la información obtenida con estimaciones. 2.6 Graficación e interpretación de información para la formulación de conclusiones. **Contenidos actitudinales** 2.7 Uso responsable y ético de tecnologías, aplicaciones y prácticas para la recolección, integración, análisis, selección, interpretación y presentación de la información. 2.8 Respeto y tolerancia por las opiniones de sus compañeros en los debates y discusiones de temas específicos. 2.9 Disposición para el trabajo colaborativo.

A lo largo de este trabajo, has conocido diferentes conceptos como el de estadística y por qué es importante conocer sus distintas herramientas y aplicarlas. También te has dado cuenta de que puedes recolectar datos de situaciones de tu día a día, por ejemplo:

¿Qué datos necesitarías recolectar para obtener tu promedio de calificaciones? ¿Qué calificación aparece más veces? ¿En qué materias debes aplicarte más?

En cuanto a tu presupuesto mensual, para apoyarte a administrar mejor tus gastos, ¿qué productos compras más?, ¿qué días del mes inviertes más en material? En promedio, ¿cuánto gastas en comida?, ¿en pasajes?, ¿en material para la escuela? Pensando en los resultados que podrías obtener, ¿consideras que tendrías la información necesaria para reasignar tu presupuesto para gastarlo de una mejor manera? ¿Encontraste gastos mayores de los que no te habías percatado?

Analiza las posibilidades que te da la estadística a nivel individual. Ahora piensa en toda la información que podrían obtener las instituciones, los centros de investigación, las escuelas, o las empresas al trabajarla con estas poderosas herramientas.

Empresas de software han desarrollado herramientas especializadas en funciones estadísticas para automatizar la generación de resultados. En este apartado revisaremos específicamente las hojas de cálculo, qué funciones podemos utilizar, además de trabajar gráficas, que como ya hemos visto, nos permiten una mejor comprensión de los resultados obtenidos.

Cálculo de la media, mediana y moda usando Excel

Como tarea escolar Pedro preguntó a sus compañeros de grupo por la talla de su calzado y debe obtener las medidas descriptivas de moda, mediana y media. Los datos que recabó fueron: 23, 25, 27, 28, 22, 26, 28, 24, 25, 26, 27, 28, 29, 28, 23, 25, 24, 26, 27, 28, 27, 27, 23, 22, 25, 26.

Recuerda las definiciones de moda, media y mediana

- La **moda** es el valor con mayor frecuencia en una distribución de datos.
- La **mediana** representa el valor de la variable de posición central en un conjunto de datos ordenados.
- La **media** representa el promedio de un conjunto de datos, se conoce también como ***media aritmética***.

Para ayudarse en la obtención automatizada de los valores de medidas de tendencia central Pedro utiliza las siguientes fórmulas de Excel:

Para determinar la ***media,*** utiliza la función PROMEDIO.

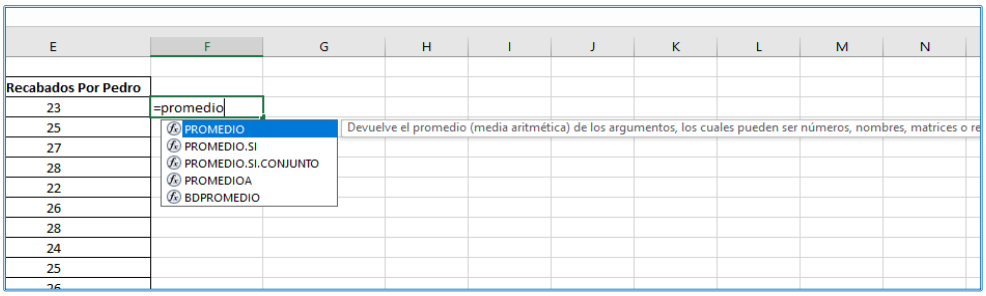


Imagen 2.12 Cálculo de la media con Excel

El resultado es 25.7307692307692 ≈ 25.73

Para determinar la ***mediana,*** utiliza la función MEDIANA.

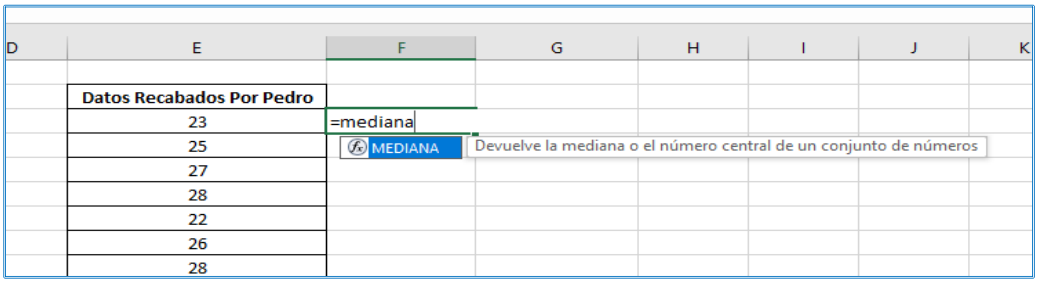


Imagen 2.13 Cálculo de la moda con Excel

El resultado es 26.

Para determinar la ***moda***, utiliza la función MODA.UNO

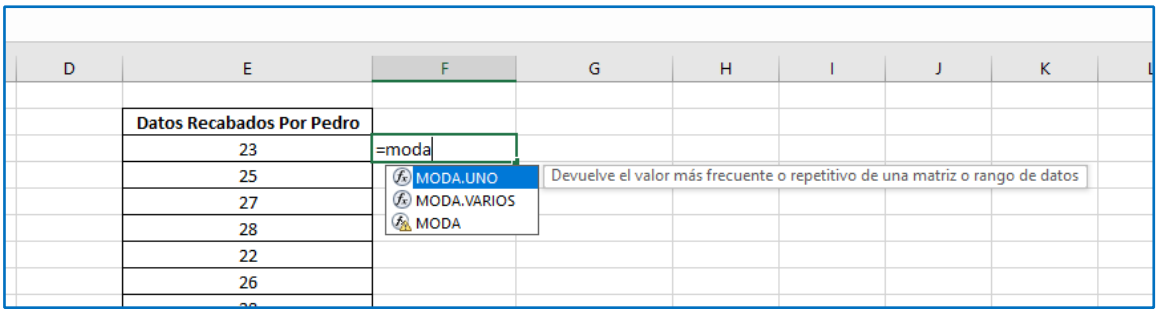


Imagen 2.14 Cálculo de la moda con Excel

El resultado es 27, ten en cuenta que, si hay más de una moda, Excel sólo te muestra la primera que encuentra.

Ejercicio 2.9

Realiza los siguientes ejercicios para determinar la ***moda, media y mediana*** de los siguientes grupos de datos utilizando fórmulas de Excel:

a) Los marcadores del equipo de baloncesto de la escuela son los siguientes: 23, 64, 48, 72, 42, 37, 43, 26, 68, 52, 29, 70, 42, 62, 60, 58, 48, 50, 64, 64, 53, 55, 29, 28, 45.

b) De la siguiente muestra de autos vendidos recientemente por una agencia; azul, blanco, verde, azul, verde, rojo, blanco, blanco, verde, blanco, rojo, azul, verde, blanco, azul, rojo, ¿cuál es la moda de la muestra de automóviles vendidos?

Análisis de datos con Excel

Excel incorpora un complemento que permite ahorrar pasos al momento de hacer análisis estadísticos o cálculos complejos, instalando las Herramientas para análisis de datos de manera automática. Se deberán proporcionar los datos y parámetros para cada análisis específico, así el complemento empleará una serie de funciones predeterminadas para realizar los cálculos automáticamente y mostrar los resultados en una tabla. Algunas herramientas de este complemento también permiten generar gráficos.

Las herramientas para el análisis de datos incluyen el cálculo de valores como desviación estándar, desviación media, coeficiente de correlación, curtosis,

entre otros, conceptos que aprendiste en el tema 2.1. Revisemos su uso con el siguiente ejemplo:

A continuación, tenemos los datos de precipitaciones a nivel nacional de octubre y noviembre por entidad federativa.

Precipitaciones (mm) a nivel nacional por entidad federativa		
Entidad Federativa	**Octubre**	**Noviembre**
Aguascalientes	0.40	2.10
Baja California	15.20	1.90
Baja California Sur	17.00	14.60
Campeche	86.00	31.00
Chiapas	228.00	37.00
Chihuahua	22.12	25.00
Coahuila de Zaragoza	29.73	22.88
Colima	37.00	27.78
Ciudad de México	113.00	27.97
Durango	116.30	23.68
Guanajuato	32.48	24.07
Guerrero	23.68	22.23
Hidalgo	32.04	25.40
Jalisco	14.60	20.90
México	157.02	57.20
Michoacán de Ocampo	14.60	19.00
Morelos	25.00	22.00
Nayarit	31.03	50.00
Nuevo León	25.80	24.07
Oaxaca	58.60	228.00
Puebla	56.00	24.07
Querétaro	50.00	47.60
Quintana Roo	17.20	44.60
San Luis Potosí	13.50	24.00
Sinaloa	51.50	7.60
Sonora	35.50	0.00
Tabasco	48.90	35.90
Tamaulipas	228.00	0.00
Tlaxcala	25.40	0.00
Veracruz de Ignacio de la Llave	53.60	52.80
Yucatán	29.73	52.80
Zacatecas	26.70	14.60

Tabla 2.11. Precipitaciónes (mm) a nivel nacional por Entidad Federativa

Usaremos la herramienta de estadística descriptiva para realizar el análisis de datos.

El primer paso es activar el análisis de datos en Excel: ve a la opción Archivo/opciones/complementos donde se mostrará la siguiente ventana:

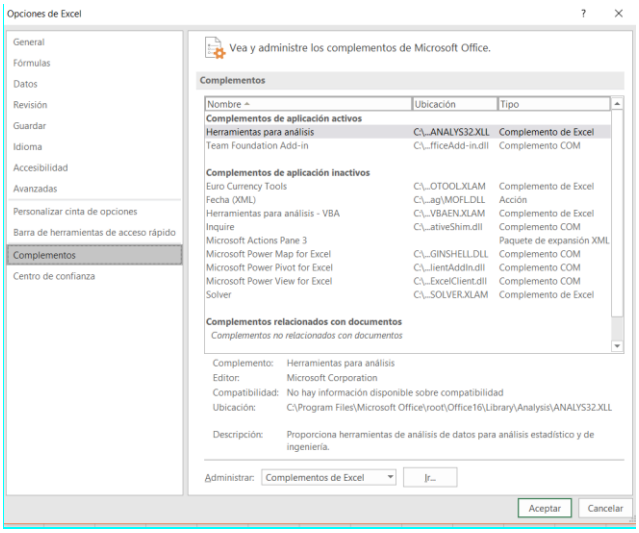


Imagen 2.15 Ventana de opciones de Excel

En la opción Administrar, selecciona Complementos de Excel y da clic en el botón Ir, se mostrará la siguiente ventana:

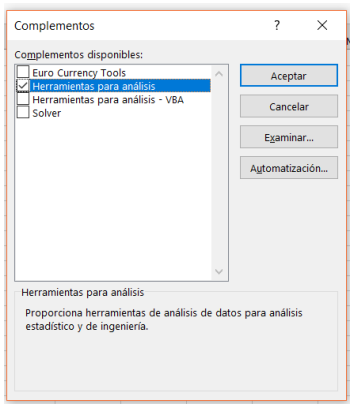


Imagen 2.16 Ventana de complementos de Excel

Habilita la opción Herramientas para análisis y da clic en aceptar. Ahora en la pestaña Datos de Excel se mostrará la opción de análisis de datos:

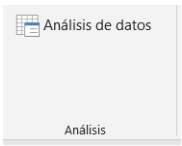


Imagen 2.17 Opción Análisis de datos.

A continuación, captura los datos de la tabla precipitaciones a nivel nacional de octubre y noviembre por entidad federativa. Una vez capturados da clic en la opción Análisis de datos que mostrara la siguiente ventana.

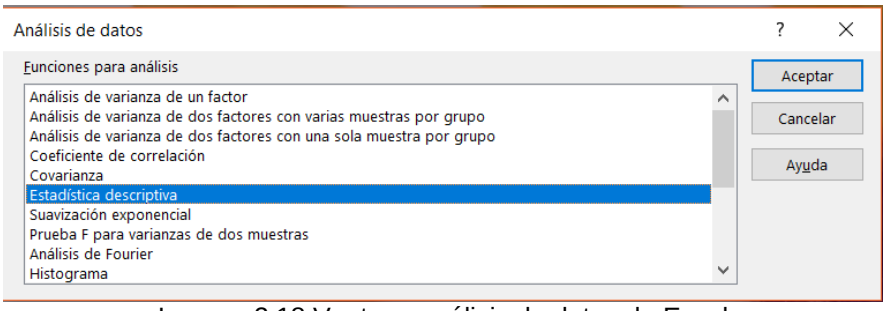
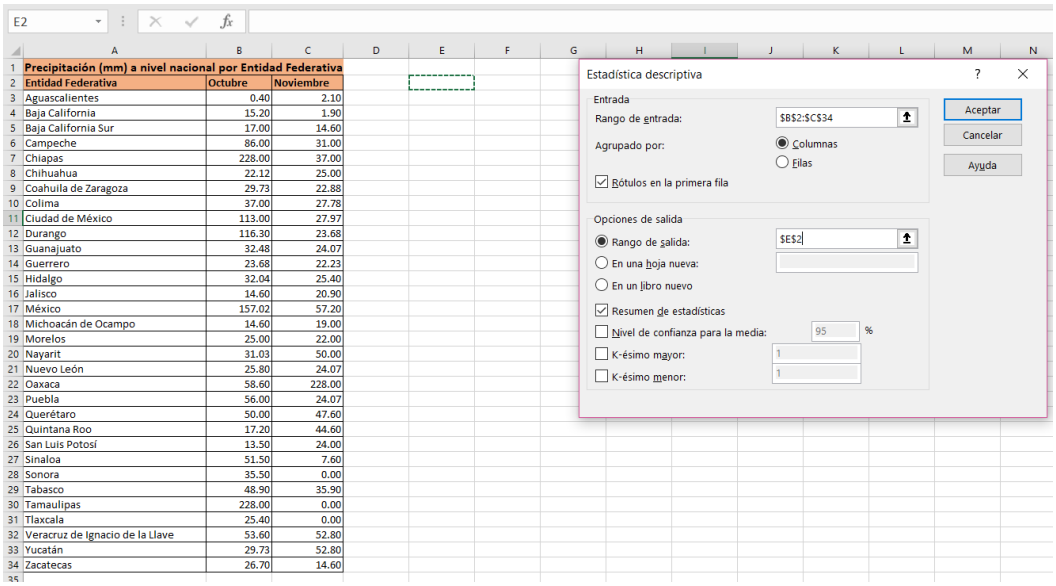


Imagen 2.18 Ventana análisis de datos de Excel

Seleccina Estadística descriptiva y da clic en Aceptar. Se mostrá esta ventana donde seleccionaremos los siguientes parámetros:

- Rango de entrada: son los datos con los que se van a realizar los cálculos, en este caso son las columnas de octubre y noviembre que van desde la celda B2 hasta la celda C34. Como vamos a sacar las estadísticas por mes en el parámetro de Agrupado por seleccionaremos Columnas.
- Rango de salida: es la celda donde va a mostrar el análisis de datos, puedes selccionar cualquier celda vacía, para este ejemplo es E2.

Debes habilitar la opción Resumen de estadísticas. Al configurar estos parámetros da clic en Aceptar.



Precipitación (mm) a nivel nacional por Entidad Federativa		
Entidad Federativa	Octubre	Noviembre
Aguascalientes	0.40	2.10
Baja California	15.20	1.90
Baja California Sur	17.00	14.60
Campeche	86.00	31.00
Chiapas	228.00	37.00
Chihuahua	22.12	25.00
Coahuila de Zaragoza	29.73	22.88
Colima	37.00	27.78
Ciudad de México	113.00	27.97
Durango	116.30	23.68
Guanajuato	32.48	24.07
Guerrero	23.68	22.23
Hidalgo	32.04	25.40
Jalisco	14.60	20.90
México	157.02	57.20
Michoacán de Ocampo	14.60	19.00
Morelos	25.00	22.00
Nayarit	31.03	50.00
Nuevo León	25.80	24.07
Oaxaca	58.60	228.00
Puebla	56.00	24.07
Querétaro	50.00	47.60
Quintana Roo	17.20	44.60
San Luis Potosí	13.50	24.00
Sinaloa	51.50	7.60
Sonora	35.50	0.00
Tabasco	48.90	35.90
Tamaulipas	228.00	0.00
Tlaxcala	25.40	0.00
Veracruz de Ignacio de la Llave	53.60	52.80
Yucatán	29.73	52.80
Zacatecas	26.70	14.60

Imagen 2.19 Ventana de estadística descriptiva en Excel

El resultado final se muestra en la Imagen 2.20. Como puedes observar, se calcularon datos como varianza, desviación estándar, curtosis, media, mediana,

moda entre otros. Con estos datos podemos realizar conclusiones sobre el comportamiento de lluvia en el país en los meses de octubre y noviembre y tomar decisiones al respecto.

	A	B	C	D	E	F	G	H
1	**Precipitación (mm) a nivel nacional por Entidad Federativa**							
2	**Entidad Federativa**	**Octubre**	**Noviembre**		**Octubre**		**Noviembre**	
3	Aguascalientes	0.40	2.10					
4	Baja California	15.20	1.90		**Media**	53.61328991	**Media**	31.58605903
5	Baja California Sur	17.00	14.60		**Error típico**	10.03443886	**Error típico**	6.939421428
6	Campeche	86.00	31.00		**Mediana**	31.535	**Mediana**	24.07
7	Chiapas	228.00	37.00		**Moda**	228	**Moda**	24.07
8	Chihuahua	22.12	25.00		**Desviación estándar**	56.76335811	**Desviación estándar**	39.25529559
9	Coahuila de Zaragoza	29.73	22.88		**Varianza de la muestra**	3222.078824	**Varianza de la muestra**	1540.978232
10	Colima	37.00	27.78		**Curtosis**	4.30322828	**Curtosis**	21.32775576
11	Ciudad de México	113.00	27.97		**Coeficiente de asimetría**	2.163545328	**Coeficiente de asimetría**	4.236372588
12	Durango	116.30	23.68		**Rango**	227.6	**Rango**	228
13	Guanajuato	32.48	24.07		**Mínimo**	0.4	**Mínimo**	0
14	Guerrero	23.68	22.23		**Máximo**	228	**Máximo**	228
15	Hidalgo	32.04	25.40		**Suma**	1715.625277	**Suma**	1010.753889
16	Jalisco	14.60	20.90		**Cuenta**	32	**Cuenta**	32
17	México	157.02	57.20					
18	Michoacán de Ocampo	14.60	19.00					
19	Morelos	25.00	22.00					
20	Nayarit	31.03	50.00					
21	Nuevo León	25.80	24.07					
22	Oaxaca	58.60	228.00					
23	Puebla	56.00	24.07					
24	Querétaro	50.00	47.60					
25	Quintana Roo	17.20	44.60					
26	San Luis Potosí	13.50	24.00					
27	Sinaloa	51.50	7.60					
28	Sonora	35.50	0.00					
29	Tabasco	48.90	35.90					
30	Tamaulipas	228.00	0.00					
31	Tlaxcala	25.40	0.00					
32	Veracruz de Ignacio de la Llave	53.60	52.80					
33	Yucatán	29.73	52.80					
34	Zacatecas	26.70	14.60					

Imagen 2.20 Resultado de análisis de datos en Excel

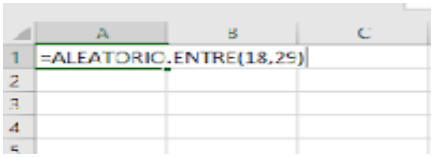
Veamos otro ejemplo de análisis de datos con Excel:

Supongamos que en una fiesta hemos hecho una encuesta preguntando la edad de un grupo de jóvenes, en total se tiene un listado de 100 personas con sus respectivas edades:

	A	B
1	persona	edad
2	per1	19
3	per2	24
4	per3	20
5	per4	21
6	per5	24
7	per6	19
8	per7	19
9	per8	24
10	per9	19
11	per10	23
12	per11	23
13	per12	20

Tabla 2.12. Edad de los jóvenes

En el caso de no poder capturar las edades de los 100 jóvenes, para fines prácticos puedes utilizar la función:



y repetirla para las 100 personas, después copiar y pegar los datos en formato de valor para evitar la modificación de los datos.

Una vez capturados los datos, selecciona del menú, la pestaña datos y posteriormente la opción Análisis de datos y elige la función que necesites para automatizar cálculos estadísticos y complejos. En este caso, se utilizan las funciones para ***Estadística descriptiva e Histograma***, que refieren a la unidad.

Herramienta de Estadística descriptiva

Comenzamos con la opción Estadística descriptiva, trabajarás con el rango o las celdas donde están las 100 edades de los jóvenes o personas, recuerda que sólo funciona con valores numéricos.

	A	B	C	D
1	persona	edad		
2	per1	19		
3	per2	24		
4	per3	20		
5	per4	21		
6	per5	24		
7	per6	19		
8	per7	19		
9	per8	24		
10	per9	19		
11	per10	23		
12	per11	23		
13	per12	20		
14	per13	22		
15	per14	21		
16	per15	22		
17	per16	25		
18	per17	20		
19	per18	20		
20	per19	20		
21	per20	20		
22	per21	22		
23	per22	24		
24	per23	18		
25	per24	20		
26	per25	19		
27	per26	18		
28	per27	23		
29	per28	25		
30	per29	20		
31	per30	25		

Imagen 2.21 Captura de edades en Excel

Al elegir la opción ***Estadística descriptiva***, aparecerá el siguiente recuadro, donde podrás seleccionar el rango de los datos de entrada y los datos de salida.

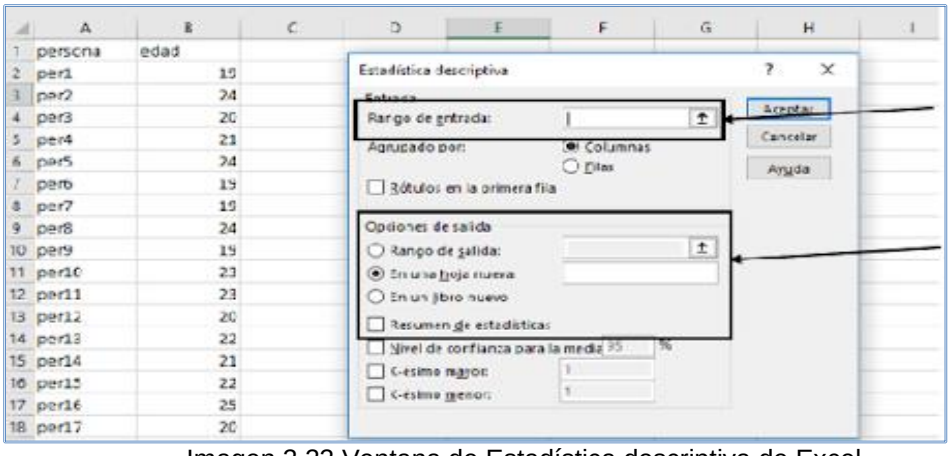
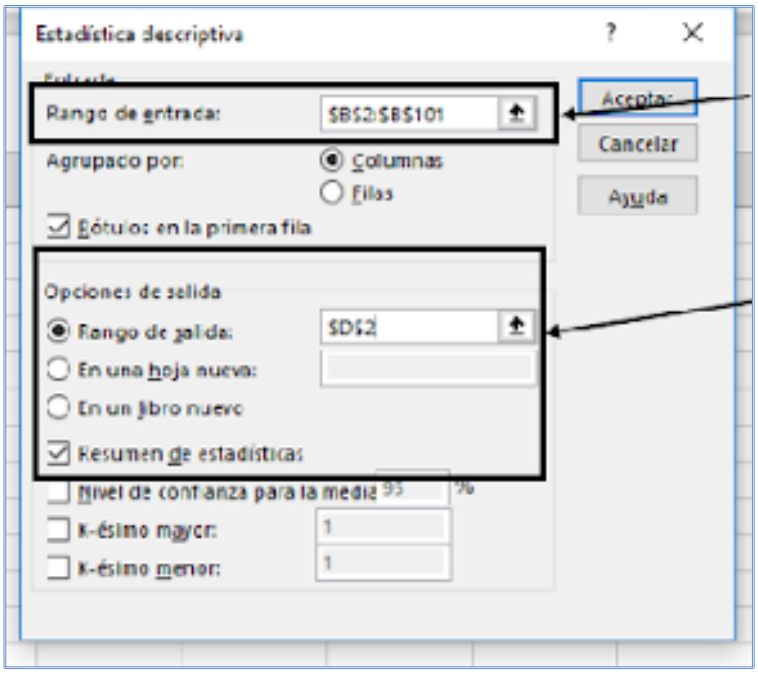


Imagen 2.22 Ventana de Estadística descriptiva de Excel

En esta ventana capturamos las celdas donde se encuentran los datos de entrada para realizar los cálculos y el análisis correspondiente de manera automática y/o automatizada, indicamos también, la opción de salida que será donde se mostrará la tabla de valores referidos a la estadística descriptiva.



Es importante seleccionar la casilla “Resumen de estadística” y para que el resumen te aparezca en la misma hoja del libro, seleccionar un “Rango de salida”.

Imagen 2.23 Captura de parámetros de estadística descriptiva

El resultado que se genera al dar aceptar es el siguiente:

	A	B	C	D	E
1	persona	edad			
2	per1	19		***Cálculos***	
3	per2	24			
4	per3	20		Media	21.2727273
5	per4	21		Error típico	0.22647769
6	per5	24		Mediana	21
7	per6	19		Moda	20
8	per7	19		Desviación estándar	2.25342452
9	per8	24		Varianza de la muestra	5.07792208
10	per9	19		Curtosis	-1.18063288
11	per10	23		Coeficiente de asimetría	0.16549509
12	per11	23		Rango	7
13	per12	20		Mínimo	18
14	per13	22		Máximo	25
15	per14	21		Suma	2106
16	per15	22		Cuenta	99
17	per16	25			
18	per17	20			
19	per18	20			
20	per19	20			

Revisa los valores que te proporciona la tabla, reflexiona el proceso de la automatización de los datos, ya sea de manera individual, con tus pares o en clase con el profesor

Imagen 2.24 Cálculos de estadística descriptiva

Herramienta de Histograma

Para realizar histogramas, es necesario agrupar los datos por clases. Recuerda que por clases nos referimos a las diferentes “categorías” en las que se clasificarán los datos que en nuestro ejemplo son las edades 18, 19, 20, etc. El objetivo del histograma es representar gráficamente cuántos elementos pertenecen a cada una de dichas clases.

Si no conoces las clases de los datos de entrada, puedes hacer una copia de todo el rango de datos correspondientes a la edad en la columna destinada para clase, utilizas la función *Quitar duplicados* que está en el menú de datos, esto para obtener una lista de valores únicos para esta columna, los cuales indicarán tus clases, recuerda que deben estar ordenados.

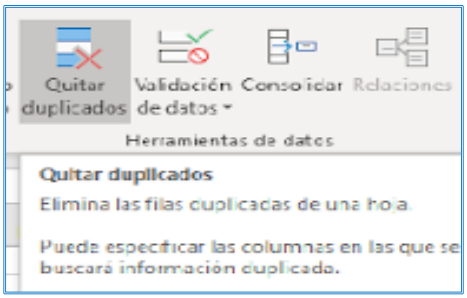


Imagen 2.25 Opción Quitar duplicados

	A	B	C	D
1	persona	edad	Clase	
2	per1	19	18	
3	per2	24	19	
4	per3	20	20	
5	per4	21	21	
6	per5	24	22	
7	per6	19	23	
8	per7	19	24	
9	per8	24	25	
10	per9	19		

Imagen 2.26 Agrupamiento por clases

En caso de usar la función de *Quitar duplicados* conocemos de antemano que el rango de las edades recolectadas en la encuesta está entre 18 y 25 años, así que lo ingresaremos manualmente en la columna C para generar el agrupamiento por clases (para crear el histograma puedes utilizar todos los datos recolectados, Excel en automático te generará el agrupamiento por clases).

Ahora, seleccionamos del menú, la pestaña datos y posteriormente la opción Análisis de datos, elegimos ***Histograma***, seleccionamos el rango de entrada y clase de acuerdo con la imagen siguiente, la celda de salida y activamos la casilla de crear gráfico:

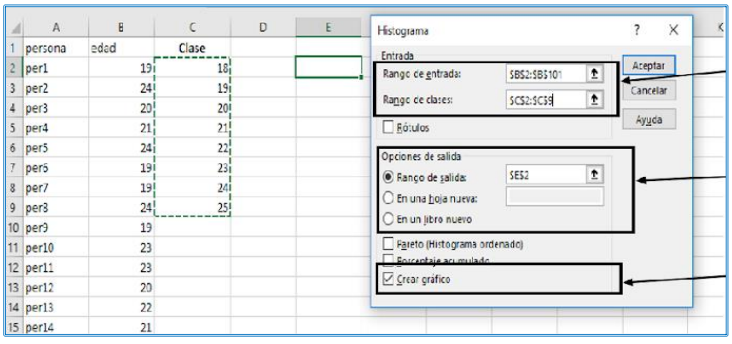


Imagen 2.27 Parámetros para generar el histograma

El resultado que se obtiene es el siguiente:

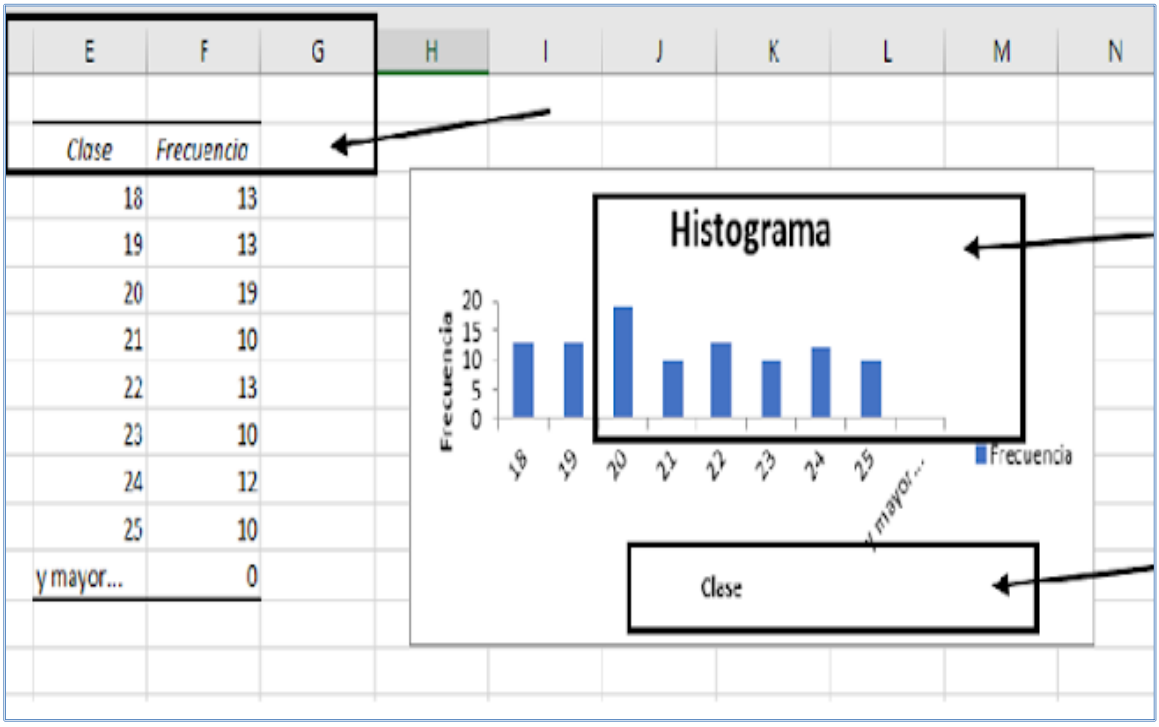


Imagen 2.28 Histograma generado por Excel

Nota: Excel por default crea una gráfica de barras, a la cual se le tendrá que dar formato para crear el histograma. Para ello, seleccionas cualquiera de las barras de la gráfica y con el botón derecho del ratón eliges la opción "***Dar formato a serie de datos…***"

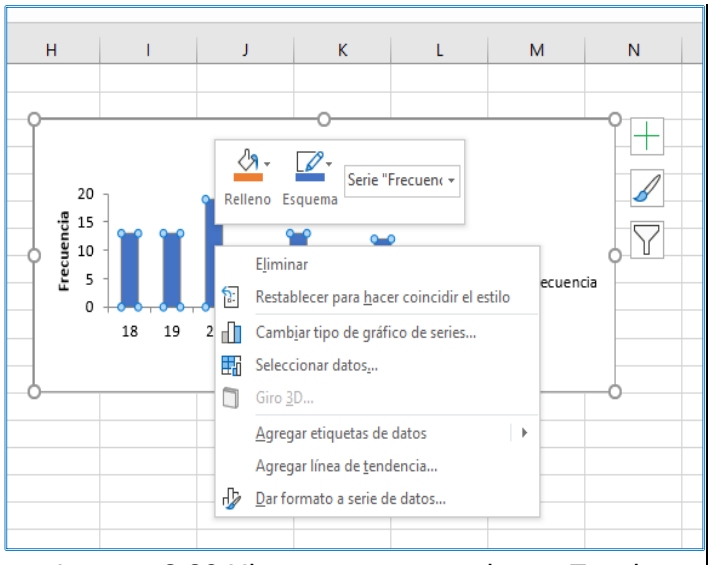


Imagen 2.29 Histograma generado por Excel

En el lado derecho de la hoja de cálculo, se despliega el siguiente menú, en "Ancho del rango" le pones cero por ciento.

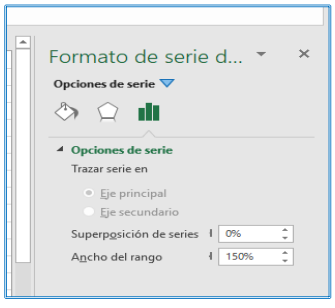


Imagen 2.30 Formato de serie

La gráfica queda de la siguiente forma:

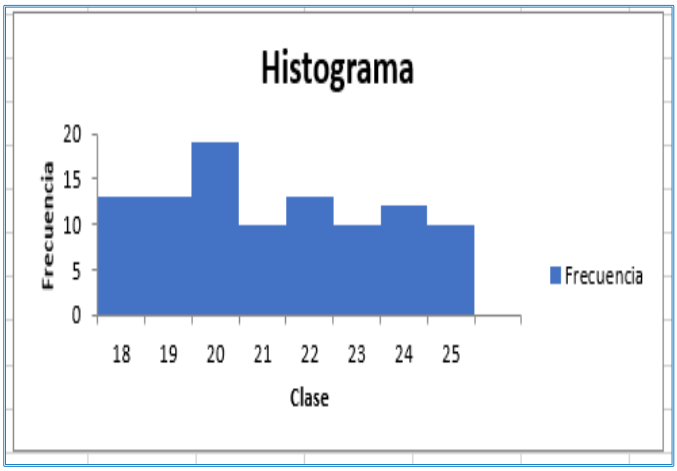


Imagen 2.31 Histograma generado por Excel

Imagen 2.32 Opción bordes y relleno

Por último, repites el procedimiento anterior, eliges la opción de bordes y relleno, para que aparezcan los bordes de las barras.

Finalmente, el histograma generado por Excel con el formato correspondiente es el siguiente:

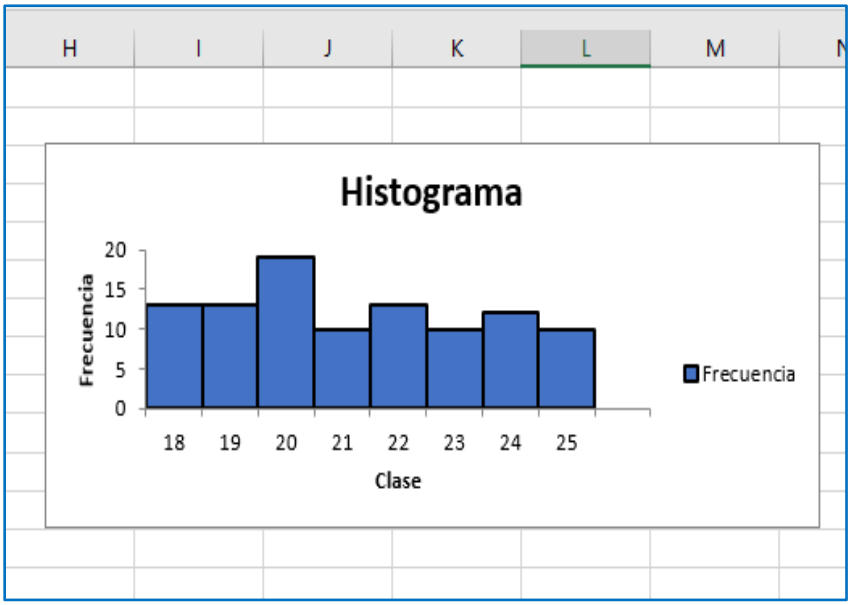


Imagen 2.33 Histograma generado por Excel

Tablas de frecuencias de datos no agrupados en Excel utilizando fórmulas

Como lo revisamos en el tema 2.2, una tabla de frecuencias nos permite trabajar con los datos que se obtuvieron, de una manera ordenada agrupándolos acorde a las apariciones de cada uno de éstos (por su frecuencia) en la muestra.

A continuación, se muestra mediante un ejemplo, el procedimiento para crear la tabla de frecuencias de un conjunto de datos no agrupados con Excel:

De una población de 500 alumnos de informática de sexto año, se toma una muestra aleatoria de 50 promedios obtenidos durante los últimos 5 años (10 por año). En la Tabla 2.13 se observan estos valores:

	A	B	C	D	E	F	G	H	I	J	K
1	Año	Calificaciones									
2	1	10	9	9	7	8	9	8	6	6	5
3	2	5	6	6	7	6	8	9	10	7	8
4	3	10	9	9	8	7	5	5	6	6	7
5	4	9	9	8	7	8	7	6	6	5	10
6	5	10	10	8	8	8	6	8	7	8	7

Tabla 2.13. Calificaciones1

Para entender la tabla de frecuencias se utilizará la siguiente nomenclatura:

fao: Frecuencia absoluta observada
faa: Frecuencia absoluta acumulada
fro: Frecuencia relativa observada
fra: Frecuencia relativa acumulada

	A	B	C	D	E
1	**Calificaciones**	**fao**	**fro**	**faa**	**fra**
2	10	6	0.12	6	0.12
3	9	8	0.16	14	0.28
4	8	12	0.24	26	0.52
5	7	9	0.18	35	0.7
6	6	10	0.2	45	0.9
7	5	5	0.1	50	1
8		50	1		

Tabla 2.14. Calificaciones2

Describamos los cálculos que se deben realizar por columna:

- **Calificaciones**: muestra el valor de las calificaciones de 10 a 5.
- **fao:** Se obtiene sumando las incidencias de cada calificación.
- **fro:** Se obtiene dividiendo fao entre el número de la muestra.
- **faa:** Se obtiene sumando fao más su valor anterior.
- **fra:** Se obtiene dividiendo faa entre el número de la muestra.

El 50 se obtiene sumando todos los elementos de la columna fao y el 1 se obtiene sumando todos los elementos de la columna fro

La tabla final trabajada en una hoja de cálculo quedaría de la siguiente forma:

	A	B	C	D	E
1	**Calificaciones**	**fao**	**fro**	**faa**	**fra**
2	10	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A2	=B2/$B$8	6	=D2/$B$8
3	9	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A3	=B3/$B$8	=D2+B3	=D3/$B$8
4	8	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A4	=B4/$B$8	=D3+B4	=D4/$B$8
5	7	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A5	=B5/$B$8	=D4+B5	=D5/$B$8
6	6	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A6	=B6/$B$8	=D5+B6	=D6/$B$8
7	5	=FRECUENCIA(Hoja1!$B$2:$K$6,Hoja2!A7	=B7/$B$8	=D6+B7	=D7/$B$8
8		=SUMA(B2:B7)	UMA(C2:C7)		
9					

Tabla 2.15. Calificaciones3

Las frecuencias de la tabla se calculan con las siguientes fórmulas de Excel:

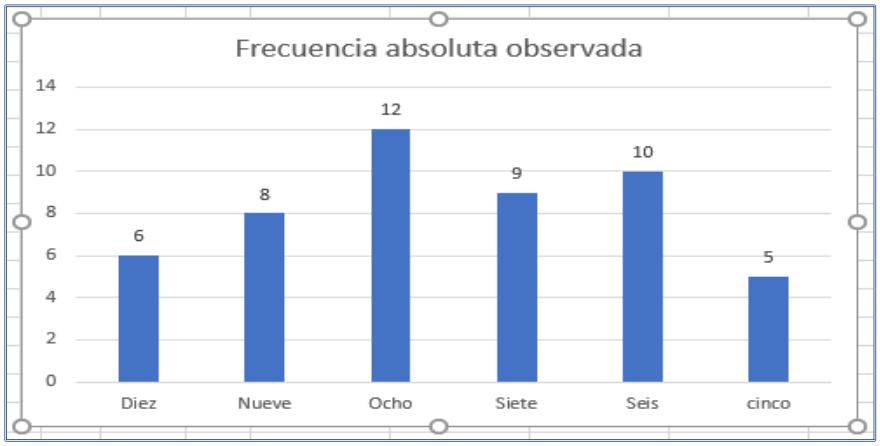
fao. Se obtiene sumando las incidencias de cada calificación. Se puede obtener utilizando la función *contar.si(rango:valor) para nuestra tabla y el valor 10 =CONTAR.SI(Hoja1!$B$2:$K$6,Hoja2!A2), de la misma manera se trabajaría para la celda del valor 9, 8, 7, 6 y 5. Se debe considerar que los datos de la tabla se encuentran en la Hoja 1.*

fro: Se obtiene dividiendo fao entre el número de la muestra, para el primer valor =B2/$B$8. Ahora copia la fórmula al resto de los valores de la columna.

faa: Se obtiene sumando el primer valor de faa más el segundo valor de fao. =D2+B3, ahora copia la fórmula al resto de los valores de la columna

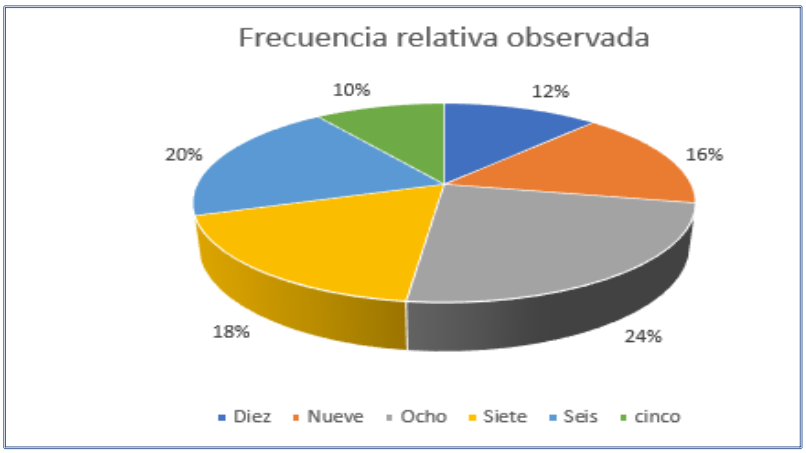
fra: Se obtiene dividiendo faa entre el número de la muestra E14/50. Ahora copia la fórmula al resto de los valores de la columna.

Ahora, grafiquemos la frecuencia absoluta de las calificaciones empleando una gráfica de barras. La gráfica se obtiene seleccionando la columna fao y con la opción insertar gráfica.



Gráfica 2.4. Frecuencia absoluta de calificaciones

Otra gráfica que podemos generar es la de la frecuencia relativa de calificaciones, en este caso, emplearemos una gráfica de pastel, la cual se obtiene seleccionando la columna fro y seleccionando la opción insertar gráfica:



Gráfica 2.5. Frecuencia relativa de calificaciones

Ejercicio 2.10

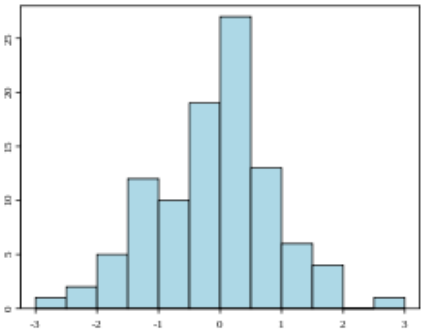
A 15 alumnos de sexto año de bachillerato se les hicieron algunas preguntas acerca de su rendimiento académico: el tiempo que dedican a ejercitarse, a realizar alguna actividad física programada y además si cuentan con un trabajo remunerado fuera del horario de clase. Las respuestas se reportaron como promedio de horas por semana. Los datos quedaron de la siguiente manera:

	Total de materias: 11		*Horas (semanales)*		
	Materias aprobadas	*Promedio anual*	*Estudio extra-clase*	*Actividad física*	*Actividad laboral*
1	5	6	1	0	No
2	9	7	3	2	No
3	7	6	3	6	No
4	9	7	4	4	No
5	11	9	6	2	No
6	11	8	6	3	Sí
7	7	6	3	6	Sí
8	9	8	2	2	No
9	5	7	2	0	No
10	7	6	1	2	No
11	3	5	0	0	Sí
12	11	8	6	4	No
13	9	8	6		No
14	5	5	0	0	No
15	11	10	3	3	Sí

Tabla Rendimiento académico

a) Desarrolla la tabla de frecuencias para el número de materias aprobadas y representa estos datos con una gráfica de barras y de pastel.
b) Desarrolla una tabla de frecuencias para el tiempo que dedica a una actividad física. Representa estos datos con una gráfica de barras.
c) Finalmente desarrolla la tabla de frecuencias que muestre los alumnos que desarrollan una actividad laboral y representa estos datos con una gráfica de pastel.

Gráficas de pastel, barras e histogramas



Un **histograma** es un gráfico que representa la distribución de frecuencias de un conjunto de datos.

El histograma**,** es especialmente útil cuando se tiene un amplio número de datos que es preciso organizar, para analizar más detalladamente o tomar decisiones sobre la base de ellos. También es un medio eficaz para transmitir a otras personas información sobre un proceso de forma precisa e inteligible.

Otra aplicación del histograma es la comparación de los resultados de un proceso con las especificaciones previamente establecidas. En este caso, mediante el histograma, puede determinarse en qué grado el proceso está produciendo buenos resultados y hasta qué punto existen desviaciones respecto a los límites fijados en las especificaciones. En este sentido, el estudio de la distribución de los datos puede ser un excelente punto de partida para establecer hipótesis.

Además, facilitan una representación en la que puede apreciarse si las medidas tienden a estar centradas o a dispersarse. También da respuesta a la cuestión de si el proceso produce buenos resultados; y a si éstos están o no dentro de las especificaciones.

Veamos un ejemplo que incluye la acción de graficar datos:

Pedro trabaja para una empresa de seguros, se le pide que realice varias gráficas de la siguiente tabla de datos agrupados para ver cuál se ve mejor para la presentación de hoy.

Clase	Límite Inferior	Límite Superior	Frecuencia
A	500	699	2
B	700	899	7
C	900	1099	4
D	1100	1299	5
E	1300	1499	3
Total			22

Tabla 2.16. Datos agrupados

Primero generemos una gráfica de pastel:

Recuerda que para calcular los valores de una gráfica circular *se utiliza la siguiente fórmula:*

$$Ángulo\ de\ la\ clase = 360 * \frac{Frecuencia\ de\ la\ Clase}{Total\ de\ Frecuecias}$$

Ángulo Clase A = $360 * \frac{3}{22} = 49.1°$

Ángulo Clase B = $360 * \frac{7}{22} = 114.5°$

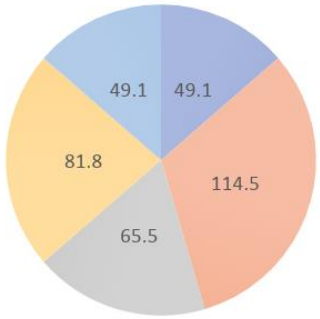
Ángulo Clase C = $360 * \frac{4}{22} = 65.5°$

Ángulo Clase D = $360 * \frac{5}{22} = 81.8°$

Ángulo Clase E = $360 * \frac{3}{22} = 49.1°$

Se genera la tabla y gráfica:

Clase	Grados
A	49.1
B	114.5
C	65.5
D	81.8
E	49.1



Gráfica 2.6. Gráfica de pastel

Nota:
Es la forma matemática para realizar la gráfica, pero la hoja de cálculo de Excel realiza las operaciones de manera automática, tanto en distribución o porcentaje.

Para generar un histograma, puedes usar la columna de la clase o puedes determinar la marca de la clase.

Clase	Límite Inferior	Límite Superior	Marca de clase	Frecuencia
A	500	699	599.5	2
B	700	899	799.5	7
C	900	1099	999.5	4
D	1100	1299	1199.5	5
E	1300	1499	1399.5	3
Total				22

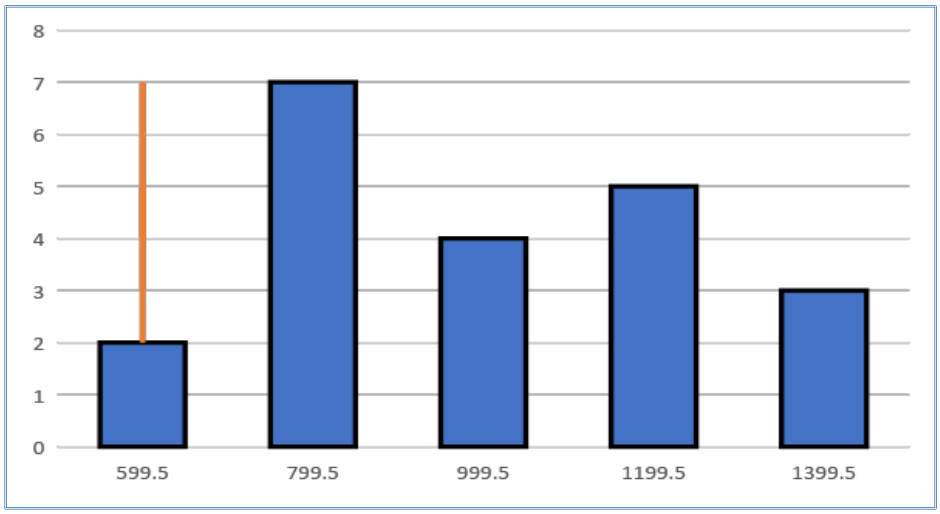
Tabla 2.17. Tabla de distribución de frecuencia con marca de clase.

Recuerda que la marca de la clase es punto intermedio que hay en la clase y para calcular la marca de la clase *se utiliza la siguiente fórmula:*

$$Marca\ de\ la\ clase = \frac{Límite\ Inferior + Límite\ Superior}{2}$$

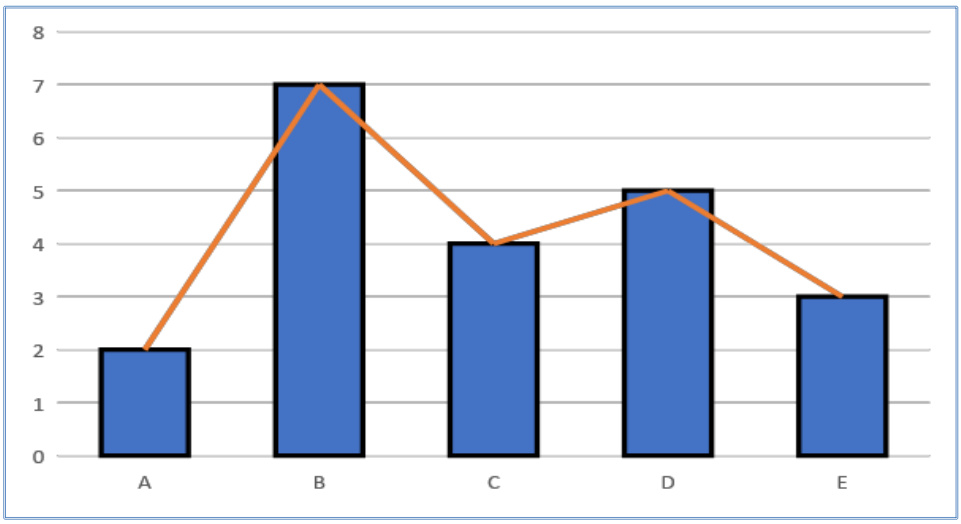
Recuerda que un histograma *es una representación gráfica de una variable por medio de barras verticales que representan la distribución de frecuencias de un conjunto de datos.*

La gráfica queda:



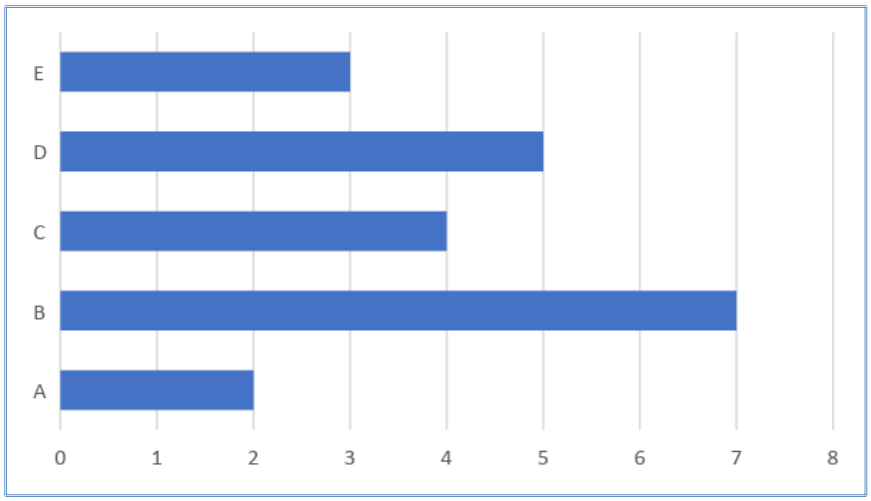
Gráfica 2.7. Histograma que considera la marca de clase contra frecuencia

La gráfica también la podemos presentar con el cruce de clase contra frecuencia, como se muestra a continuación:

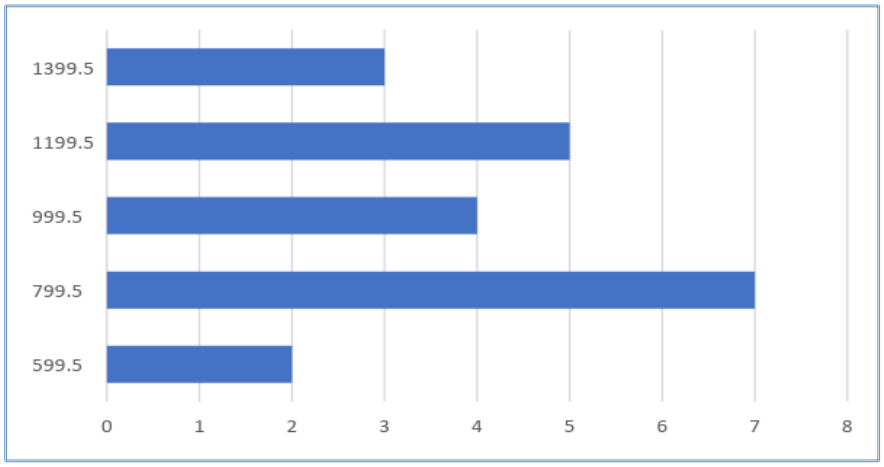


Gráfica 2.8. Histograma que considera la clase contra la frecuencia

De igual forma, podemos presentar los datos con gráficos de barras:



Gráfica 2.9. Gráfica de barras



Gráfica 2.10. Gráfica de barras

Las gráficas e histogramas son muy utilizadas en la prensa, en los medios de comunicación y en los libros para representar los datos de forma vistosa y al mismo tiempo consiguen de un solo vistazo darnos los detalles fundamentales.

Para interpretar una gráfica debes tener muy en claro lo que va a representar cada eje, por lo general en el agrupamiento de los datos, se relaciona la columna de la clase contra cualquiera de las columnas de las frecuencias.

También resulta muy útil, que sepas agrupar y representar por medio de tablas, una gran cantidad de datos, por ejemplo, imagina una fábrica de calzado deportivo que, para hacer una mejor venta de sus productos, necesita conocer el promedio de edad de las personas que habitan en una localidad; para que con esa información decidir qué tipo de modelos y cantidad de pares debe enviar.

Actividad integradora

Describir mediante un análisis de datos:

¿Cuántas horas dedican los alumnos a la semana para el esparcimiento y el uso de redes sociales? Comparar con ¿cuántas horas dedican los alumnos a la semana para el estudio y tareas (fuera del horario escolar)?

Se realizó una encuesta con las siguientes preguntas y los datos se agruparon en las siguientes tablas:

1. ¿Cuántas horas dedicas aproximadamente a la semana al esparcimiento, incluyendo las redes sociales?

2. ¿Cuántas horas dedicas a la semana al estudio y a las tareas fuera del horario escolar?

¿Cuántas horas dedicas aproximadamente a la semana al esparcimiento, incluyendo las redes sociales?			
Clase	Límite Inferior	Límite Superior	Frecuencia
A	0	5	25
B	6	10	20
C	11	15	15
D	16	20	17
E	21	------	23
Total			100

Tabla 2.18. Horas dedicadas al esparcimiento

¿Cuántas horas dedicas a la semana al estudio y a las tareas fuera del horario escolar?			
Clase	Límite Inferior	Límite Superior	Frecuencia
A	0	5	40
B	6	10	20
C	11	15	25
D	16	20	10
E	21	------	5
Total			100

Tabla 2.19. Horas dedicadas al estudio y tareas

Determinamos las frecuencias acumulada y complementaria, además elaboramos las gráficas para representar la información, los datos se muestran en las siguientes tablas:

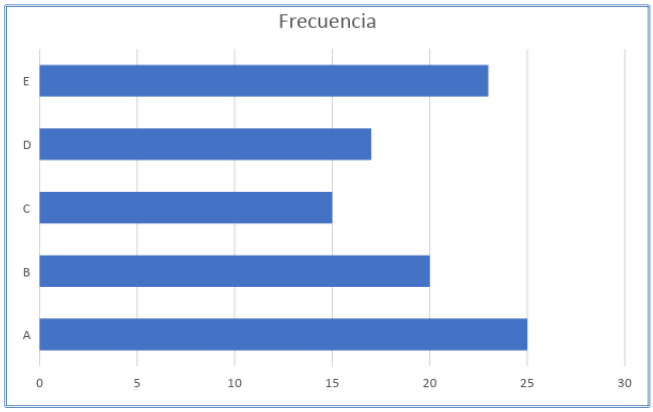
Para la pregunta 1:

Clase	Límite Inferior	Límite Superior	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
A	0	5	25	25	75
B	6	10	20	45	55
C	11	15	15	60	40
D	16	20	17	77	23
E	21	------	23	100	0
TOTAL			100		

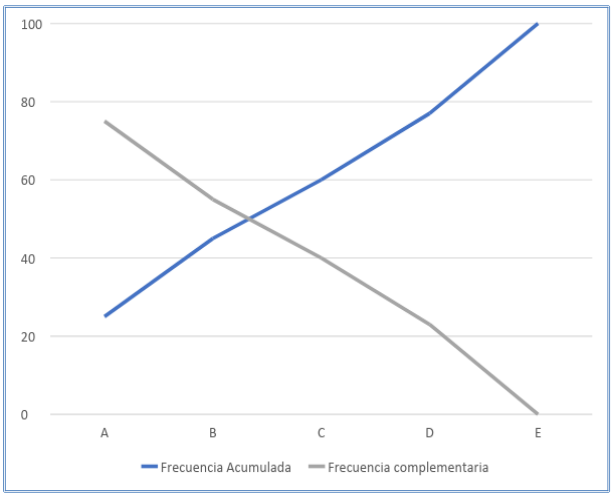
Para la pregunta 2:

Clase	Límite Inferior	Límite Superior	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
A	0	5	40	40	60
B	6	10	20	60	40
C	11	15	25	85	15
D	16	20	10	95	5
E	21	------	5	100	0
TOTAL			100		

Gráficas de la pregunta 1:

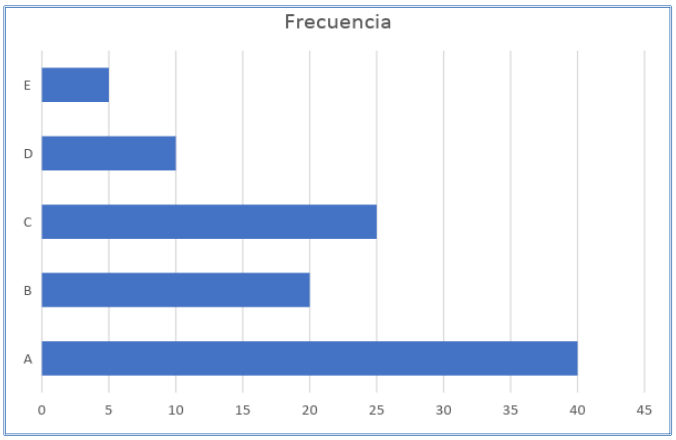


Gráfica 2.11. Gráfica resultante

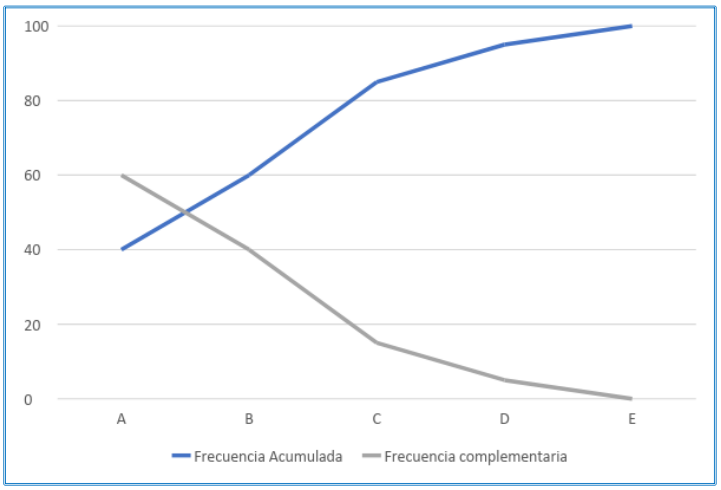


Gráfica 2.12. Gráfica resultante

Gráficas de la pregunta 2:



Gráfica 2.13. Gráfica resultante



Gráfica 2.14. Gráfica resultante

Apóyate realizando graficas circulares en Excel para análizar y discutir lo siguiente:

- ¿Qué porcentaje les corresponde a las situaciones planteadas en las peguntas 1 y 2?

- Explica y reflexiona individual y colectivamente qué relación existe entre ambas preguntas, así como en sus posibles soluciones.

- Según tu criterio personal, ¿los resultados obtenidos al procesar los datos se relacionan con tu vida diaria?, ¿cómo? y ¿por qué? Además, ¿qué propondrías para mejorar tu situación académica y la del grupo?

Ejercicio 2.11

a) Elabora las gráficas para representar los siguientes datos relativos al color de los automóviles vendidos recientemente por una agencia; azul, blanco, verde, azul, verde, rojo, blanco, blanco, verde, blanco.

b) Realiza las gráficas de la siguiente tabla:

Límite Inferior Exacto	Límite Superior Exacto	Frecuencia
503.5	595.5	21
595.5	687.5	19
687.5	779.5	22
779.5	871.5	27
871.5	963.5	15

Ejercicio 2.12

Se desea desarrollar un nuevo negocio escolar de renta de casilleros. Para saber si sería rentable este negocio se les pregunta a 150 estudiantes de diferentes años qué opinan sobre esta idea, y la respuesta es que 14 de cada 15 contestó que sí, para ello, ahora se hace una segunda encuesta con 20 de estos estudiantes en la cual la pregunta es ¿cuánto sería lo que está dispuesto a pagar por este servicio de forma mensual? Se obtuvieron los siguientes datos:

Tabla de datos (pago mensual locker en pesos)

150	200	225	100	90	100	90	100	150	110
100	225	90	200	150	90	200	100	150	125

La tabla de frecuencias quedaría como sigue:

Tabla de frecuencias pago por locker

Costo mensual	Fao	Fro	Fea	Fra
90	4	0.2	4	0.2
100	5	0.25	9	0.45
110	1	0.05	10	0.5
125	1	0.05	11	0.55
150	4	0.2	15	0.75
200	3	0.15	18	0.9
225	2	0.1	20	1
Total	20	1		

De acuerdo con estos datos, obtén tus resultados y las gráficas que consideres necesarias, compáralas con las de tus compañeros y discutan entre todos, durante la clase, el análisis de los resultados, pueden guiarse respondiendo lo siguiente:

- ¿La información presentada es la suficiente para llegar a una solución viable o qué hace falta?
- ¿Será rentable o no tomar la decisión de desarrollar o implementar el nuevo negocio, por qué?

- En el caso de que el negocio sea rentable, ¿cuál es el precio que el mayor porcentaje de los estudiantes están dispuestos a pagar?, ¿qué precio probablemente no dé una ganancia?

Preguntas de reflexión

La hoja de cálculo puede ser en una poderosa herramienta didáctica, que se puede incorporar en el aula para crear entornos de aprendizaje que enriquezcan la representación, el modelado y la comprensión, con el fin de encontrar posibles soluciones a problemas escolares o de la vida cotidiana que, en algún momento, tú integrarás en tu quehacer personal y/o profesional.

Además, ¿consideras que tu capacidad intelectual puede ir más allá de calcular, aplicar fórmulas y hacer gráficas en una hoja de cálculo? Si, al poder emplear adecuadamente el análisis y procesamiento de los datos de manera automatizada en cualquier situación de tu vida para convertirla en información, la cual podrás visualizar, interpretar y al mismo tiempo te permitirá construir y emitir juicios de opinión, para las posibles tomas de decisiones que tendrás que realizar en muchos momentos de tu vida, para ti, para tu comunidad, y de las cuales tendrán que ser de una manera ética, crítica y razonada.

Autoevaluación	

1) Relacione las columnas: en el paréntesis coloque la respuesta correcta de la columna derecha.

1.1 Longitud de 150 tornillos producidos en una fábrica	( )	a) Cuantitativa discreta
1.2 Bebida favorita de alumnos de la ENP	( )	b) Cualitativa ordinal
1.3 Likes que tiene un video en youtube	( )	c) Cuantitativa continua
1.4. Talla (s,m,l,xl) de playeras del equipo de basquetbol	( )	d) Cualitativa nominal

Subraye la respuesta correcta.

2) El conjunto completo de todos los elementos que se someten a estudio se conoce como:

a) Muestra b) Moda

c) Datos d) Población

3) Al subconjunto de miembros seleccionados que se someten a un estudio estadístico se le conoce como:

a) Muestra b) Moda

c) Datos d) Población

4) Los datos numéricos que puede tomar una cantidad incontable de valores entre dos números se denominan como:

a) Ordinales b) Continuos

c) Discretos d) Nominales

5) El valor estadístico de promedio es llamado también

a) Media b) Mediana c) Moda d) Varianza

6) Nos indica qué tan confiables son los datos que estamos investigando en la muestra obteniendo a partir de ésta el error de las mediciones.

a) Variabilidad b) Varianza c) Correlación d) Normalidad

7) _____ es un factor que nos indica que tan simétricos son los datos

8) La _____ es un valor estadístico el cual se utiliza como referencia en varias medidas como en la varianza, coeficiente de correlación, desviación estándar, etc.

9) La tabla de frecuencias se obtiene:

a) dando valores arbitrarios de la muestra
b) por medio de la media de los valores de la muestra
c) por medio de la varianza de los valores de la muestra
d) con los números obtenidos de la muestra

10). Los valores de la frecuencia relativa se obtienen con el _____ de la _____ _____ entre total de _____ _____

11) El porcentaje frecuencia absoluta es:

a) dividir el valor de la frecuencia relativa entre la suma de la frecuencia absoluta
b) multiplicar el valor de la frecuencia relativa por 100
c) dividir el valor de la frecuencia relativa entre 100
d) dividir el valor de la frecuencia absoluta entre 100

12) La diferencia que existe entre la frecuencia relativa y la absoluta es que la primera tiene _____ y la segunda _____

13) En una investigación de mercado, es necesario conocer qué sabor de producto se vende con más frecuencia durante la temporada navideña. ¿Qué medida de tendencia central utilizarías?

_____

14). Durante el trayecto diario a la escuela tú y tus compañeros utilizan diferentes tipos de transporte público. Los datos son los siguientes 1, 1, 2, 2, 2, 2, 2,3 3, 3. Si tomas los valores centrales de esta tabla para saber qué tipo de transporte se utiliza más ¿Qué medida de tendencia central estás utilizando?

---

15) Calcular la media, la mediana y la moda de la siguiente serie de números: 5, 3, 6, 5, 4, 5, 2, 8, 6, 5, 4, 8, 3, 4, 5, 4, 8, 2, 5, 4.

a) Moda = 4	Mediana = 5	Media = 4.8
b) Moda = 5	Mediana = 4	Media = 4.8
c) Moda = 5	Mediana = 5	Media = 4.9
d) Moda = 4	Mediana = 5	Media = 4.9

16) Los 40 alumnos de una clase han obtenido las siguientes puntuaciones, sobre 50, en un examen de Física. 3, 35, 30, 37, 27, 31, 41, 20, 16, 26, 45, 37, 9, 41, 28, 21, 31, 35, 10, 26, 11, 34, 36, 12, 22, 17, 33, 43, 19, 48, 38, 25, 36, 32, 38, 28, 30, 36, 39, 40. ¿Construir la tabla de frecuencias?

a)

Clase	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
0-5	1	1	39
6-10	1	2	38
11-15	3	5	35
16-20	3	8	32
21-25	3	11	29
26-30	6	17	23
31-35	7	24	16
36-40	10	34	6
41-45	4	38	2
46-50	2	40	0

b)

Clase	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
0-5	1	1	39
5-10	1	2	38
10-15	3	5	35
16-20	3	8	32
20-25	3	11	29
25-30	6	17	23
30-35	7	24	16
35-40	10	34	6
40-45	4	38	2
45-50	2	40	0

c)

Clase	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
0-5	1	39	1
5-10	1	38	2
10-15	3	35	5
16-20	3	32	8
20-25	3	29	11
25-30	6	23	17
30-35	7	16	24
35-40	10	6	34
40-45	4	2	38
45-50	2	0	40

d)

Clase	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
0-5	1	1	39
6-10	1	2	39
11-15	3	5	34
16-20	3	8	34
21-25	3	11	34
26-30	6	17	33
31-35	7	24	30
36-40	10	34	36
41-45	4	38	38
46-50	2	40	0

17) Se ha aplicado una prueba de desempeño a los empleados de una fábrica, agrupando los datos por el número de aciertos, obteniéndose la siguiente tabla:

Clase	Frecuencia
38 – 43	7
44 – 50	8
51 – 56	15
57 – 62	25
63 – 68	18
69 – 74	9
75 – 80	6

Determina la frecuencia acumulada

a)

Clase	Frecuencia Acumulada
38 – 43	81
44 – 50	80
51 – 56	73
57 – 62	63
63 – 68	70
69 – 74	79
75 – 80	88

b)

Clase	Frecuencia Acumulada
38 – 43	7
44 – 50	15
51 – 56	30
57 – 62	55

63 – 68	73
69 – 74	82
75 – 80	88

c)

Clase	Frecuencia Acumulada
38 – 43	81
44 – 50	73
51 – 56	58
57 – 62	33
63 – 68	15
69 – 74	3
75 – 80	0

d)

Clase	Frecuencia Acumulada
38 – 43	1
44 – 50	1
51 – 56	2
57 – 62	3
63 – 68	3
69 – 74	1
75 – 80	1

18) El promedio de peso de 6 toros seleccionados al azar en un rancho ganadero debe de ser de por lo menos 425 kg. Ya se han seleccionado 5 toros y sus pesos han sido 408, 441, 418, 429 y 422 kg. ¿Cuánto debe pesar el último toro para que el preso promedio de los 6 toros sea el que se quiere?

a) 432 kg b) 425 kg c) 435 kg d) 425 kg

19) En un centro comercial, se consultó la edad a todas las personas que entraban entre las 12:00 h y 12:30 h. Los resultados obtenidos fueron los siguientes:

15	73	1	65	16	3	42
36	42	3	61	19	36	47
30	45	29	73	69	34	23
22	21	33	27	55	58	17
4	17	48	25	26	11	4
54	70	51	3	34	26	10

Si agrupamos en 8 clases, ¿Cuántas personas tienen 60 o menos años?

a) 27 b) 34 c) 40 d) 36

Unidad 3

Automatización y control de procesos

UNIDAD 3
AUTOMATIZACIÓN Y CONTROL DE PROCESOS

Objetivo

El alumno construirá un sistema de automatización usando software de control para componentes electrónicos, programas computables, simuladores para la solución de problemas aplicados a la ciencia y la industria, desarrollando habilidades de investigación, diseño y gestión de proyectos, trabajo colaborativo y respeto al medio ambiente, fomentando el uso de energías renovables y del reciclaje.

Introducción

En esta unidad se pretende que el alumno construya un sistema de automatización básico usando software de control para componentes electrónicos, programas computables, simuladores y pueda solucionar problemas aplicados a la ciencia y la industria.

Las actividades cuentan con ejercicios que retoman los conceptos y procedimientos de las unidades 1 y 2.

Por lo que los contenidos 3.1, 3.2 y 3.3 van enfocados a la compresión de los conceptos básicos y funcionamiento de los elementos de un sistema automatizado, la adquisición de datos y la importancia de estos en la ciencia y la industria que tiene en el mundo actual. Los contenidos 3.4, 3.5 y 3.6 van encaminados a la construcción y aplicación de un sistema automatizado para la resolución de problemas básicos en la vida cotidiana.

Los temas 3.7, 3.8 y 3.9 y 3.10 contribuyen a generar habilidades para diseñar y gestionar proyectos, apertura para trabajar en equipo desarrollando habilidades colaborativas, interés para realizar investigaciones y respeto al medio ambiente.

También se proporciona al alumno información referente a tutoriales donde puede fortalecer y seguir investigando con más profundidad los contenidos que componen esta unidad.

Contenidos conceptuales	Contenidos procedimentales
3.1 Elementos de un Sistema de Automatización: sensores, controladores, actuadores.  Industrial Internet (Autracen, 2019). Recuperado de: https://goo.gl/CaY4sV	3.4 Construcción de un sistema automatizado para la obtención de valores con sensores. 3.5 Uso de los elementos de un sistema automatizado para obtener y analizar datos que permitan emitir conclusiones 3.6 Aplicación de los elementos de un sistema automatizado para resolver problemas de la vida cotidiana relacionados con la ciencia y la industria. **Contenidos actitudinales** 3.7 Disposición para diseñar y gestionar proyectos. 3.8 Apertura para trabajar en equipo desarrollando habilidades colaborativas. 3.9 Interés para realizar investigaciones.

Tomando como referencia el objetivo general, este primer tema va encaminado a que el alumno comprenda el funcionamiento de los elementos de un sistema automatizado.

Se inicia con una explicación básica del funcionamiento de cada uno de los elementos de un sistema automatizado y las actividades van encaminadas a la comprensión del uso de cada elemento. Desarrollando habilidades en la compresión y lectura de textos.

Elementos de un sistema de automatización

La automatización es un proceso donde las acciones se desarrollan sin la participación directa de un individuo. En dicho sistema la tecnología es indispensable para que las tareas se lleven a cabo de manera automática, de ahí proviene el nombre de automatización.

La siguiente imagen muestra los elementos básicos de un sistema de automatización: sensores, controladores y actuadores.

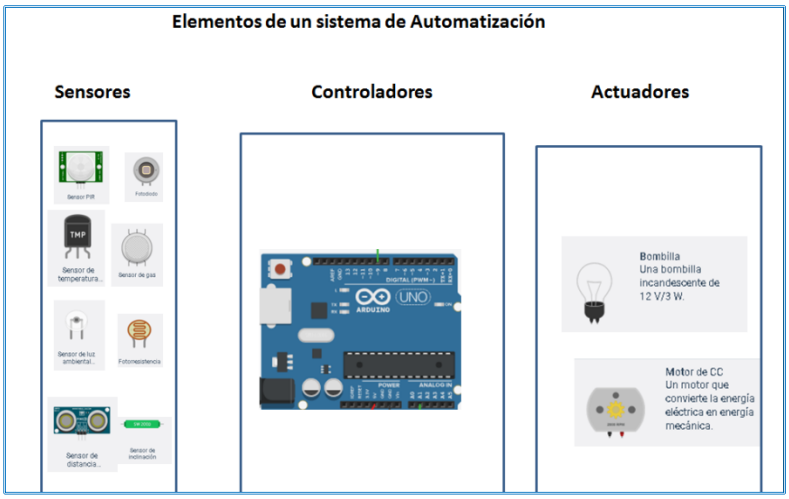


Figura 3.1. Elementos de un sistema de control (Tinkercad, 2018). Recuperado de: https://www.tinkercad.com

Sensores: Son aquellos dispositivos que son sensibles a una magnitud física de la variable que se quiera controlar. Los transductores tienen la función de convertir cualquier señal a otra, por ejemplo, un termopar que convierte una temperatura en resistencia (Enríquez Harper, 2011).

Existen sensores de diferentes tipos en función de la variable que se tenga que medir o detectar.

- **Sensor de contacto.** Se emplea para detectar la posición donde se encuentra el objeto. Estos sensores funcionan con un interruptor que consta de una pieza móvil y una pieza fija, llamados normalmente abierto (NA) o normalmente cerrado (NC) de manera que al accionar la pieza móvil pueden impedir o dar paso a una corriente por el circuito de control. (Carbonell et al., 2011).
- **Sensor óptico.** Conocido como sensor fotoeléctrico, o infrarrojo, es capaz de detectar un objeto o persona al cambiar la intensidad de la luz. Un ejemplo es una fotorresistencia de manera que cuando reciben un haz de luz permiten el paso de la corriente eléctrica. (Carbonell et al., 2011).
- **Sensor de temperatura.** Sirve para medir la temperatura y el transductor convierte la temperatura en una señal de tipo eléctrico. Dos de los más conocidos son el termistor y el detector de temperatura por resistencia (RTD) cuya resistencia varía de acuerdo a la temperatura. El termopar, genera un voltaje en relación a los cambios de temperatura (Enríquez Harper, 2011).
- **Sensor de humedad.** Detecta la variación de conductividad de un material en presencia de agua. (Vilaseca and Rodríguez Arenas, 2015) Éste se puede aplicar para automatizar el riego de un jardín pues puede detectar cuando las plantas necesitan riego o ya tiene el agua suficiente evitando que se dañe el jardín.

A continuación, se presenta una tabla de los sensores más comunes y el fenómeno que miden, tomada de la página de National Instruments.

Sensor	Fenómeno
Termopar, RTD, termistor	Temperatura
Fotosensor	Luz
Sensores pasivos	Movimiento
Micrófono	Sonido
Galga extensiométrica, transductor piezoeléctrico	Fuerza y presión
Potenciómetro, LVDT, codificador óptico	Posición y desplazamiento
Acelerómetro	Aceleración
Electrodo pH	pH

Tabla 3.1. Tipos de sensores según su funcionamiento (National Instruments, 2018). Recuperado de: https://www.ni.com/data-acquisition/what-is/esa/

Controlador de dispositivo. Es un dispositivo encargado del control de la señal proporcionada por el sensor y compara el valor de entrada con el valor deseado en la salida para poder comunicarse con dispositivos actuadores. (Enríquez Harper, 2011). El controlador puede estar compuesto por amplificadores, filtros circuitos digitales y micro controladores. Estos pueden ser eléctricos o neumáticos. Los más usados son los controladores lógicos programables (PLC), los micro controladores y las computadoras con tarjetas de adquisición de datos.

Actuador. Convierte una señal de control eléctrica en una acción física (Schmitt and Farwell, 1988). Hay diferentes tipos de actuadores por mencionar algunos están los:

- **Electrónicos.** Funcionan por medio de corrientes eléctricas. Algunos tipos de actuadores son: motores de corriente directa, motores de pasos, electro válvulas, relevadores, etc.
- **Hidráulicos.** Se accionan con líquidos a presión como el aceite o el agua y generan una fuerza mecánica.
- **Neumáticos.** Convierten la energía del aire comprimido a una fuerza mecánica.

Ejercicio 3.1

Responde las siguientes preguntas.

1. Es un sistema donde las acciones se desarrollan sin la participación directa de un individuo.
2. Los elementos de un sistema de automatización son:
3. Los elementos que contiene un sensor óptico son:
4. Se genera una fuerza mecánica a partir de diferentes tipos de energía, se refiere a:

Ejercicio 3.2

Mi profesor desea conocer la población que ingresa diariamente al plantel.

a) ¿Qué elementos debe usar en este sistema automatizado?
b) ¿Qué sensor debe elegir?
c) ¿Qué actuador debe elegir?
d) ¿Qué controlador debe elegir?
e) ¿Crees que esta solución sea la más adecuada para saber el número de alumnos que ingresan al plantel? Escribe tu reflexión.

Contenidos conceptuales	Contenidos procedimentales
3.2 Adquisición y comunicación de datos.  Measurement (HBM, 2018).Recuperado de: https://goo.gl/xR3qan	3.4 Construcción de un sistema automatizado para la obtención de valores con sensores 3.5 Uso de los elementos de un sistema automatizado para obtener y analizar datos que permitan emitir conclusiones 3.6 Aplicación de los elementos de un sistema automatizado para resolver problemas de la vida cotidiana relacionados con la ciencia y la industria **Contenidos actitudinales** 3.7 Disposición para diseñar y gestionar proyectos 3.8 Apertura para trabajar en equipo desarrollando habilidades colaborativas 3.9 Interés para realizar investigaciones

Este tema tiene como objetivo que el alumno comprenda el concepto y el proceso de la adquisición y comunicación de datos de un sistema automatizado.

Se inicia con una indagación de conocimiento, después se apoya al alumno con información básica sobre el proceso y concepto de la adquisición y comunicación de datos de un sistema automatizado.

Ejercicio 3.3
Indagación y Activación de conocimiento

Contesta las siguientes preguntas

1. ¿Sabes qué es un sistema de adquisición de datos?
   Mucho
   Poco
   Regular
   Nada
2. ¿Sabes cómo funciona un sensor?
   Mucho
   Poco
   Regular
   Nada

3. ¿Sabes a qué se le llama acondicionamiento de la señal?
   Mucho
   Poco
   Regular
   Nada
4. ¿Sabes qué componentes tiene un sistema de adquisición de datos?
   Mucho
   Poco
   Regular
   Nada

En este tema se pretende que el alumno comprenda el funcionamiento de un sistema de adquisición de datos.

Sistema de adquisición de datos

Conocido como Data Acquisition (DAQ). En la página de National Instruments se define como: "*La adquisición de datos es el proceso de medir con una PC un fenómeno eléctrico o físico como voltaje, corriente, temperatura, presión o sonido".* Según (Gonzalez Tamayo, 2018), es un sistema que puede considerarse como un conjunto de hardware y software que conectan al usuario con el mundo físico.

El sistema DAQ está compuesto de sensores, hardware y un software programable.

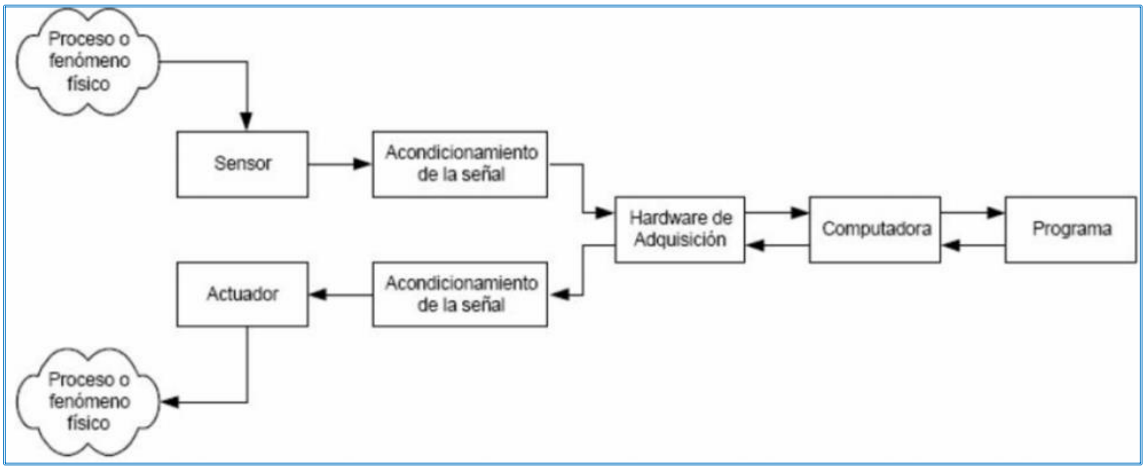


Figura 3.2. Esquema del proceso de adquisición de datos (González,2018). Recuperado de: https://www.redalyc.org/html/849/84950585001/

Funcionamiento

En la Figura 3.2 se muestra el diagrama de un sistema de adquisición de datos. Donde el sensor detecta una señal y es enviada para su acondicionamiento, convirtiéndola en bits para que la computadora la pueda interpretar y analizar en un software y poder extraer información y emitir conclusiones significativas.

Los datos digitales provenientes de la computadora son acondicionados en una señal analógica y enviada a un actuador.

El siguiente esquema muestra las partes que intervienen en la adquisición de datos.

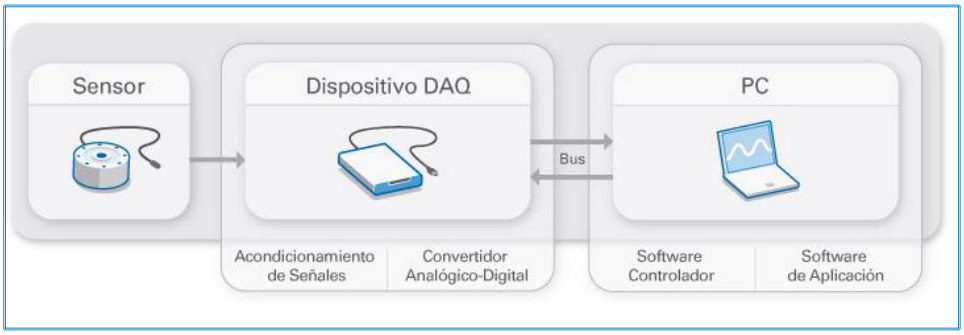


Figura 3.3. Partes de un sistema DAQ (National Instruments, 2018). Recuperado de: https://www.ni.com/data-acquisition/what-is/esa/

Sensores

Como ya se mencionó en el tema anterior, los sensores son los encargados de detectar un fenómeno físico y los transductores son los encargados de convertir este fenómeno en otra señal, dependiendo del tipo de sensor la salida de la señal puede ser un voltaje, una corriente o una resistencia, por mencionar algunos. Los sensores pueden requerir de compones adicionales para entregar una señal con precisión y de manera correcta.

Acondicionamiento de señales

Las señales de los sensores pueden tener ruido o ser peligrosas para medirse directamente. Un circuito de acondicionamiento de señales puede manipular y acondicionar la señal de entrada. Este circuito puede incluir amplificación, atenuación, filtrado y aislamiento de una señal. Algunos dispositivos de adquisición de datos ya tienen integrado el acondicionamiento para señales específicas de los diferentes tipos de sensores.

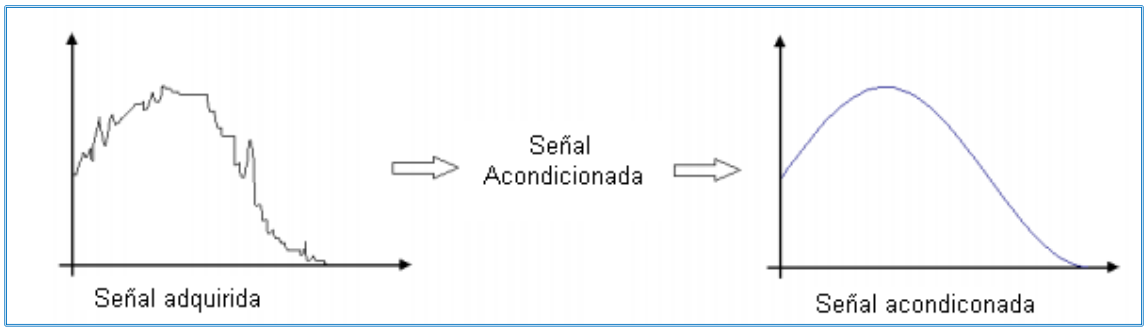


Figura 3.4. Señal acondicionada

Convertidor analógico digital (ADC) y convertidor digital analógico (DAC)

Los fenómenos físicos que los sensores detectan deben ser convertidos en digitales antes de ser manipulados por una PC. Un ADC convierte una señal física a una señal digital. Estas muestras son transferidas a una PC a través de un bus, donde la señal original es reconstruida desde las muestras en software. El convertidor digital analógico (DAC) convierte un código generalmente binario, compuesto de ceros a una señal analógica (corriente, voltaje).

Bus de la PC

Los dispositivos DAQ se conectan a una PC a través de una ranura o puerto. El bus de la PC puede ser USB, Ethernet, Wi-Fi para comunicación inalámbrica por mencionar algunos. Estos sirven como interfaz de comunicación entre el dispositivo DAQ y la PC para pasar instrucciones y datos medidos.

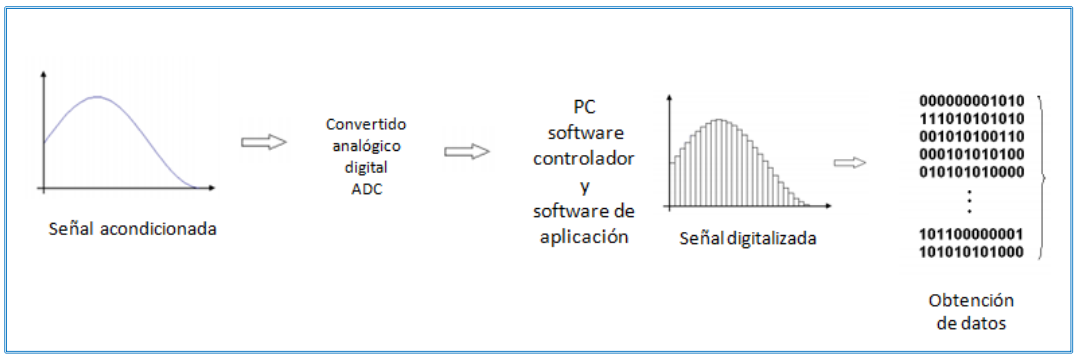


Figura 3.5. Proceso de conversión y digitalización de una señal

Software controlador

El software controlador abstrae comandos de hardware de bajo nivel y programación a nivel de registro. El software controlador DAQ tiene una interfaz de programación de aplicaciones para construir software de aplicación.

Software de aplicación

El software de aplicación facilita la interacción entre la PC y el usuario para adquirir, analizar y presentar datos. Las aplicaciones personalizadas son usadas para automatizar múltiples funciones de un dispositivo DAQ, realizar algoritmos de procesamiento de señales y mostrar interfaces de usuario personalizadas.

En el siguiente ejemplo identificaremos los elementos del proceso de adquisición de datos para realizar la lectura de un sonido y representarlo digitalmente en una computadora.

Ejemplo

Sistema de adquisición y representación del sonido.

En este sistema se requiere un dispositivo transductor que convierta el sonido en una señal de tipo eléctrica, dicho dispositivo puede ser un micrófono el cual convierte las ondas sonoras en una señal eléctrica.

Cuando se adquiere la señal eléctrica ésta se digitaliza con la ayuda de una tarjeta de adquisición de datos la cual convierte la señal eléctrica en una señal digital (bits), los cuales pueden ser leídos por una computadora.

Para poder hacer la representación gráfica de las ondas sonoras se requiere de un programa de cómputo que permita organizar y analizar los datos adquiridos. Este proceso se muestra en la Figura 3.6.

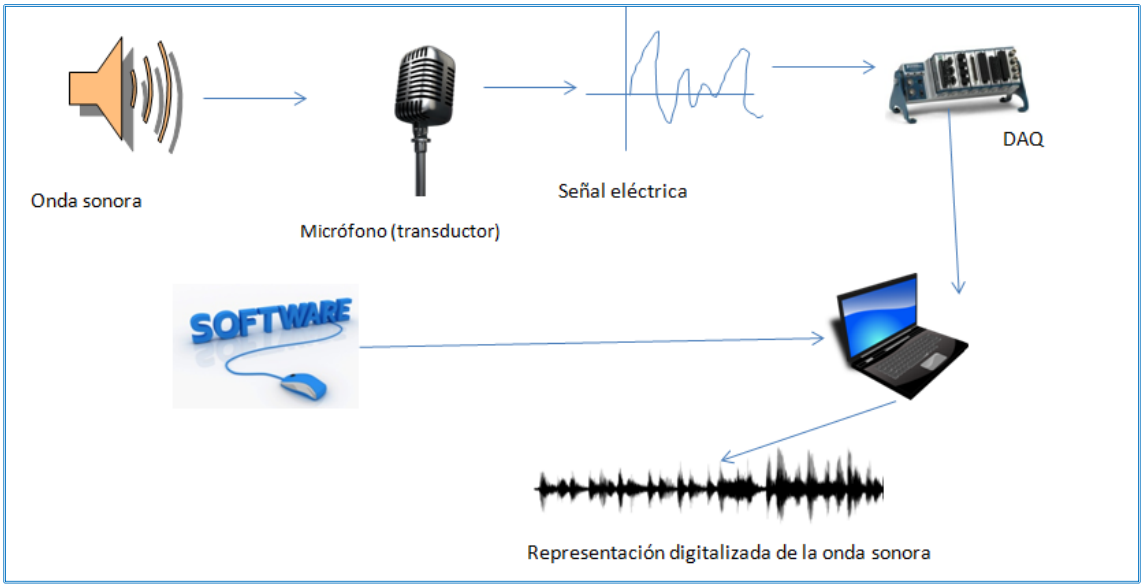


Figura 3.6. Sistema de adquisición de una señal de sonido (Microsoft PowerPoint, 2018)

Ejercicio 3.4

Completa el siguiente texto.

a) _____es un proceso de medición que se hace por medio de una computadora, con la finalidad de medir y analizar fenómenos o procesos físicos.

b) La señal de tipo_____ se obtiene por un _____, luego el _____, la convierte en bits que serán interpretados para extraer información significativa.

Ejercicio 3.5

1) ¿Qué componentes se requieren tener en un sistema de adquisición de datos para detectar el pH de una sustancia?
   a) Sensor, actuador, software
   b) Sensor, DAQ, computadora
   c) Sensor, DAQ, ADC, computadora
   d) Sensor, convertidor, software
2) ¿Qué sensor se requiere para detectar el pH de una sustancia?
   a) Termopar
   b) Fotosensor
   c) Sensores pasivos
   d) Electrodo
3) Cuando se obtiene la información del pH de una sustancia, ¿qué función tiene el software de aplicación?
   a) Procesar, almacenar, detectar información
   b) Procesar, almacenar, controlar actuadores
   c) Procesar, almacenar, analizar información
   d) Procesar, almacenar, controlar focos

Ejercicio 3.6

En los últimos años los científicos afirman que existe un calentamiento global, para comprobar experimentalmente esa teoría,

1) ¿Qué dispositivos utilizarías para realizar un experimento y comprobar esta teoría?

   a) Sensores de temperatura, una tarjeta de adquisición de datos y una computadora.
   b) Electrodo pH, un dispositivo registrador y una computadora.
   c) Sensores de calor, una tarjeta de adquisición y software para registrar datos.
   d) Multímetro para medir temperatura, Excel y computadora.

2) ¿Cómo podría determinar la temperatura promedio y la temperatura máxima de los datos adquiridos en todo un año?

   a) Utilizando calculadora y regla de cálculo.
   b) Usando un programa computable que adquiera y analice los datos
   c) Usando un software que emita conclusiones.
   d) Sensores de calor, una tarjeta de adquisición, software para registrar datos.
   e) Multímetro para medir temperatura, Excel, computadora.

Ejercicio 3.7

Dibuja una representación de un sistema de adquisición de datos de una señal de temperatura. Guíate por el ejemplo presentado en este tema.

Contenidos conceptuales 3.3 Automatización de Procesos en la Ciencia e Industria  (Industria4.0, 2016).Recuperado de: https://goo.gl/ytQXsJ	**Contenidos procedimentales** 3.4 Construcción de un sistema automatizado para la obtención de valores con sensores 3.5 Uso de los elementos de un sistema automatizado para obtener y analizar datos que permitan emitir conclusiones 3.6 Aplicación de los elementos de un sistema automatizado para resolver problemas de la vida cotidiana relacionados con la ciencia y la industria **Contenidos actitudinales** 3.7 Disposición para diseñar y gestionar proyectos 3.8 Apertura para trabajar en equipo desarrollando habilidades colaborativas 3.9 Interés para realizar investigaciones 3.10 Respeto al medio ambiente.

Este tema tiene como objetivo que el alumno comprenda la importancia de la Automatización de Procesos en la Ciencia e Industria y reflexione sobre las ventajas y desventajas que se dan con la necesidad de automatizar procesos.

Se inicia con una indagación de conocimiento, después se apoya al alumno con información básica referente a la Automatización de Procesos en la Ciencia e Industria.

Ejercicio 3.8

Contesta las siguientes preguntas

1. ¿Sabes qué es la automatización para la industria?
   Mucho
   Poco
   Regular
   Nada
2. ¿Sabes por qué es importante la automatización para la industria?
   Mucho
   Poco
   Regular
   Nada
3. ¿Sabes qué es la automatización para la ciencia?
   Mucho

Poco Regular Nada 4. ¿Sabes por qué es importante la automatización para la ciencia? Mucho Poco Regular Nada	

Automatización de Procesos en la Ciencia e Industria

La automatización de procesos en la industria es muy importante ya que se usa como herramienta para poder reducir costos, aumentar la productividad y la calidad de los procesos industriales.

La Informática aplicada a la ciencia y la industria se ocupa principalmente para optimizar la adquisición, almacenamiento, procesamiento y análisis de la información usada en salud, biomedicina, química o física, por mencionar algunas áreas.

A continuación, se describen algunas áreas donde se aplica la automatización de procesos según (Duran Moyano, Martínez García & Gámiz Caro, 2012, p. 8).

En la agricultura, ganadería y pesca. Para sistema de control de invernaderos de riego, clasificación de productos, control climático de viveros, alimentación para reses y aves.

En los servicios básicos. En los sistemas de agua y canalización, estaciones de alimentación eléctrica y monitoreo de emergencia.

En la comunicación. Sistemas de telefonía y satélite.

En la domótica. Sistema de seguridad, calefacción y ventilación, iluminación, etcétera.

En la administración. Para el manejo de la información como en finanzas y contabilidad.

En la educación. Para automatizar procesos en la consulta de información y procesamiento de información.

En la navegación. Se controlan y fijan posiciones geográficas mediante satélites.

En la aeronáutica. Se puede controlar el tráfico aéreo, la posición y altura de los aviones en el radar o simulación de vuelos.

En la ciencia. Es de gran ayuda para analizar los datos, almacenar y recuperar información, controlar experimentos, llevando la información estadística de procesos, etcétera.

Ejercicio 3.9
Elabora un mapa mental sobre cada uno de los siguientes temas:

1. ¿Qué importancia tiene la Informática aplicada a la industria?

2. ¿Qué importancia tiene la Informática aplicada a la ciencia?

Ejercicio 3.10
Analiza la siguiente lectura y dibuja un mapa mental donde expliques la problemática de los cambios en el ámbito laboral a raíz de la automatización de la producción.

Tomado como referencia (Henríquez, 2018)

Una Reflexión sobre la Automatización de la Producción y los Cambios en el Ámbito Laboral

14 de julio de 2017

Dina Henríquez; La Chorrera, Panamá.
IBERCIENCIA. Comunidad de Educadores para la Cultura Científica.
La revolución industrial hizo que la humanidad alcanzara un grado de desarrollo cuya evidencia más palpable es el aumento de la población mundial lo que implica el agotamiento de los recursos, la degradación ambiental y problemas sociales entre los que se encuentran el acceso a educación, salud y a un empleo.

Los niveles de desempleo de la población menor de 35 años generalmente se atribuyen al aumento de la esperanza de vida de la población que actualmente sigue activa en el campo laboral a edades avanzadas, impidiendo la entrada de los jóvenes recién graduados; pero no es común considerar que la formación que está recibiendo esta nueva generación no es acorde al mundo al que vive.
El desplazamiento de la mano de obra humana por la automatización de la producción no es una realidad futura, ya está ocurriendo. Ciertamente debemos hacer una introspección y generar un debate global que involucre a todos los actores de los ámbitos: social, político educativo y empresarial que busque un consenso hacia la evolución del empleo, las economías y la sociedad.

Pasada la primera década del siglo XXI, el tema de la robotización de la producción ha pasado de ser una realidad futura a ser el presente. La revolución industrial hizo que la humanidad alcanzara un grado de desarrollo cuya evidencia más palpable es el aumento de la población mundial.

El 30 de octubre de 2011 marcó un hito en la historia de la humanidad, pero marcó también las alarmas: la población mundial llegó a los 7 mil millones. El periodo entre 1750 y el 2010, es el periodo en la historia en que la población humana ha crecido de forma dramática haciendo que nos planteemos una serie de interrogantes pues el crecimiento de las poblaciones contribuyen al agotamiento de los recursos, la degradación ambiental y problemas sociales entre los que se encuentran el acceso a educación, salud y a un empleo que le permita mantener un nivel de vida adecuado.

La paradoja de la automatización de la producción es esta: a pesar del crecimiento económico debido al aumento de la productividad, la plantilla de trabajadores se ve reducida pues las empresas son capaces de producir más artículos con mucho menos personal. Sugiriendo esto las tecnologías baratas, eficientes y métodos de organización de trabajo pueden ser capaces de sustituir a la mano de obra humana con su consecuente afectación económica, pues sin empleos: ¿quién va a consumir?

La llamada cuarta revolución industrial, que incluye los procesos de automatización robotizada de la producción, implica un cambio en los paradigmas educativos pues es necesario formar a los ciudadanos que serán parte del mercado laboral en estas condiciones. Especialmente pues nos encontramos a las puertas de la transición entre la robótica industrial y la inteligencia artificial.

El advenimiento de la inteligencia artificial al ámbito industrial supone la desaparición de puestos de empleos y la aparición de nuevos lo que nos lleva a la necesidad de evaluar los sistemas educativos nacionales para verificar las ofertas de formación que ofrecen y los perfiles de los profesionales que buscan formar.

Los niveles de desempleo de la población menor de 35 años generalmente se atribuyen al aumento de la esperanza de vida de la población que actualmente sigue activa en el campo laboral a edades avanzadas, impidiendo la entrada de los jóvenes recién graduados; pero no es común considerar que la formación que está recibiendo esta nueva generación no es acorde al mundo al que vive.
La educación implica formar un profesional capaz de desenvolverse en su ámbito laboral, pero actualmente vivimos en una aldea global. El teletrabajo, los emprendimientos y los empleos freelance son las nuevas formas en que los jóvenes buscan sortear el desempleo pero ¿qué tanto de lo que necesitan, más allá de los conocimientos propios de su profesión, es impartido en las universidades o institutos técnicos?

El desplazamiento de la mano de obra humana por la automatización de la producción no es una realidad futura, ya está ocurriendo. Ciertamente debemos hacer una introspección y generar un debate global que involucre a todos los actores de los ámbitos: social, político educativo y empresarial que busque un consenso hacia la evolución del empleo, las economías y la sociedad.

Mapa mental de la lectura

Ejercicio 3.11
Responde las siguientes preguntas:

1. Automatizar el proceso industrial para la mejora de la calidad es necesario, aunque por ello no puedan conservar el puesto de muchas personas.
   Mucho
   Poco
   Regular
   Nada
2. La automatización de un proceso en la ciencia puede apoyar a salvar vidas, aunque se contamine el medio ambiente.
   Mucho
   Poco
   Regular
   Nada
3. El uso de la computadora solo debe ser para actividades escolares.
   Mucho
   Poco
   Regular
   Nada

Construcción de un sistema automatizado para la obtención de valores con sensores

Hoy en día la automatización de procesos en la vida cotidiana ha ayudado mucho a que actividades que requerían mucho tiempo para realizarse ahora se lleven a cabo más rápido.

La construcción de sistemas automatizados para la obtención de valores con sensores de temperatura, humedad, sonido, luz, etc. nos permiten tomar decisiones en varias actividades de la ciencia y la industria para mejorar su productividad.

Antes de construir un sistema automatizado, identificaremos los diferentes dispositivos electrónicos y su función usando un simulador en línea.

Existen actualmente simuladores y tarjetas de desarrollo que permiten construir sistemas automatizados para obtener dichos valores. Mencionaremos algunos ya que el uso de ellos depende de cada usuario y sus necesidades.

Simuladores:

- Arduino en línea
  https://create.arduino.cc/
- Tinkercad
  https://www.tinkercad.com
- Virtual Breadboard,
  http://www.virtualbreadboard.com/
- Kicad EDA
  http://kicad-pcb.org/
- Fritzing
  http://fritzing.org/home/

Una tarjeta de desarrollo cuenta con su lenguaje de programación o un sistema operativo, microcontrolador o un procesador. Por ejemplo, Arduino, PCBWay, Raspberry, MSP430 de Texas Instruments, entre otras. (Quispe, 2018).

Para introducir al alumno en la construcción de un sistema automatizado y que conozca los dispositivos electrónicos básicos, se propone usar simuladores.

Cabe mencionar que hay diferentes simuladores en línea que son gratuitos y lo único que se debe de hacer es crear un usuario y contraseña en la página de los simuladores antes mencionados.

Ejercicio 3.12

Identificar los dispositivos electrónicos básicos y su funcionamiento.

1. Revisa los siguientes videos de la Universidad Politécnica de Valencia para que comprendas el funcionamiento de los dispositivos electrónicos básicos:

    - Componentes electrónicos activos (Martínez José, 2011). Conocimiento de componentes activos (diodos, transistores, amplificadores) en el laboratorio de electrónica. https://www.youtube.com/watch?v=WjKiq2ud958

    - Componentes electrónicos pasivos (Martínez José, 2011). Manejo de componentes pasivos (resistencias, condensadores) en el laboratorio de electrónica. https://www.youtube.com/watch?v=JIjsl1Wdrb0

Con lo visto en los videos contesta la siguiente actividad.

2. Relaciona las columnas e identifica el nombre de cada componente electrónico.

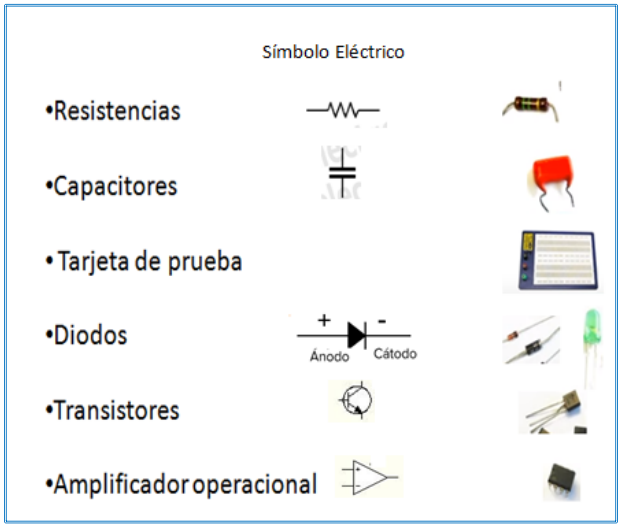


Figura 3.7. Dispositivos electrónicos básicos. (Martínez José, 2011) Recuperado de: https://www.youtube.com/watch?v=WjKiq2ud958 https://www.youtube.com/watch?v=JIjsl1Wdrb0

Nota: Si quieres saber con más detalle el uso de cada dispositivo consulta la siguiente liga.

Curso de electrónica básica desde cero (Editronikx, 2015). En este curso aprenderás a analizar circuitos, planos, soldar, hacer impresos, simulación, testeo de componentes, reparación básica, uso de multímetro, manejo de estación de calor, análisis básico de circuitos, etcétera. https://www.youtube.com/watch?v=LjYClvMPRdE&list=PLNipMBg3MF-by_2uhpnmGbGTBPMfqo7HV

3. Revisa los siguientes videos.

   - De los microcontroladores a la tarjeta Arduino (UNAM, 2018). https://es.coursera.org/lecture/arduino-aplicaciones/de-los-microcontroladores-a-la-tarjeta-arduino-LOTX7

   - ¿Qué es Arduino? Introducción (codigofacilito, 2023). https://www.youtube.com/watch?v=Kgz0vD1vSxY

4. Dibuja un mapa mental del concepto de una tarjeta desarrolladora.
5. Dibuja un mapa mental del funcionamiento de una tarjeta desarrolladora.

Ejercicio 3.13

En esta actividad el alumno aplicará el uso de un simulador para comprender el funcionamiento de los circuitos electrónicos.

Simulación de un circuito básico de encendido y apagado de un led.

1.- Ingresar al simulador de tu preferencia. Para este ejemplo usaremos http://tinkercad.com, en este simulador puedes seleccionar los elementos para poder construir un circuito básico de encendido de un diodo emisor de luz (led). Para identificar la polaridad de tu diodo observa la figura 3.10.

Al crear la cuenta, la primera pantalla que visualizas será la siguiente:

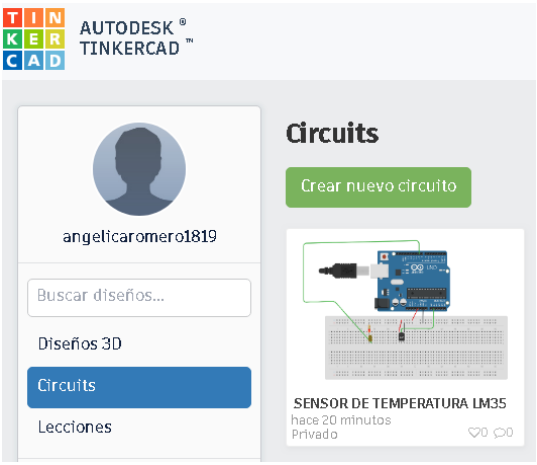


Figura 3.8. Inicio del programa para crea un circuito nuevo

Después debes elegir la opción de circuitos, y a continuación dar clic en la opción crear un nuevo circuito como se muestra en la Figura 3.8.

Se visualizará el área de trabajo de Tinkercard.

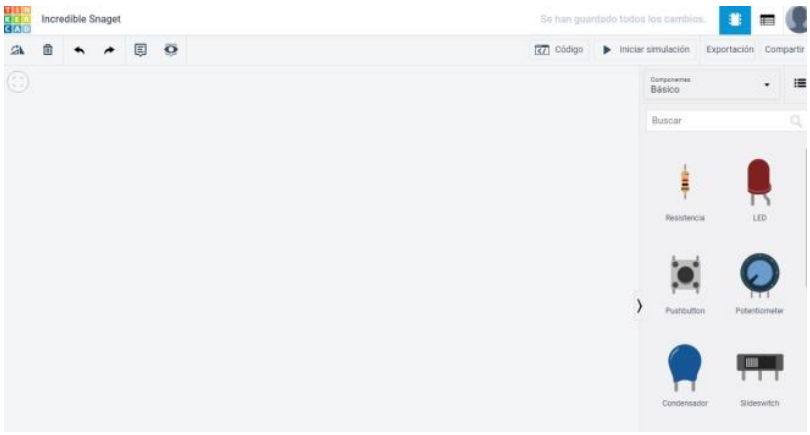


Figura 3.9. Imagen del área de trabajo Tinkercard.

Arrastra los siguientes elementos:

Figura 3.10. Dispositivos electrónicos de Tinkercard

Realiza el circuito que se muestra a continuación:

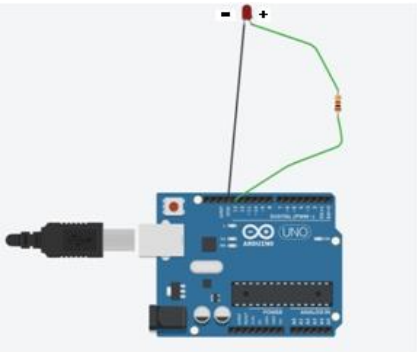


Figura 3.11. Conexión de dispositivos electrónicos con tarjeta de adquisición de datos

Procedimiento de simulación para el prendido del led.

Descripción:

- Dar clic izquierdo con el mouse en la polaridad negativa del led y soltar el curso del mouse dirigirlo al puerto GND del arduino Uno R3.
- Dar clic izquierdo con el mouse la polaridad positiva del led y sin soltar el curso del mouse dirigirlo a un extremo de la resistencia y a su vez dar clic izquierdo con el mouse en el otro extremo de la resistencia y soltar el curso dirigirlo al puerto 13 del arduino Uno R3.

Una vez construido el circuito presionamos el botón:

y nos muestra lo siguiente:

Este código es para que un led prenda y se espere un 1 segundo para apagar el led de manera cíclica.

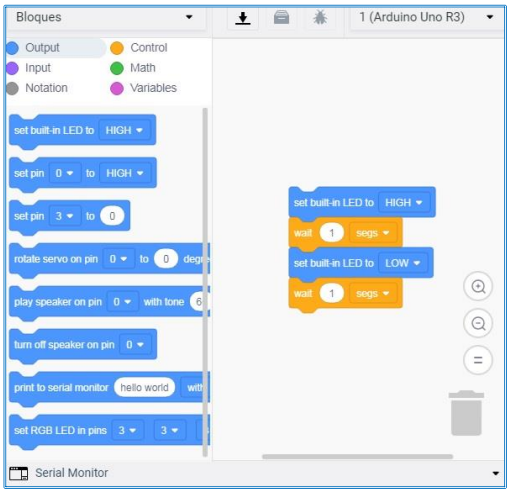
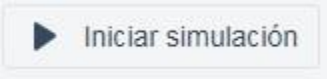


Figura 3.12. Programa utilizando bloques en Tinkercard

Por último, vamos a simular nuestro circuito, para ello vamos a presionar el siguiente botón:



Ejercicio 3.14
Coloca en el siguiente circuito cada uno de los elementos que le corresponde.

a) Sensor de temperatura
b) Resistencia
c) Cables
d) Protoboard
e) Tarjeta controladora

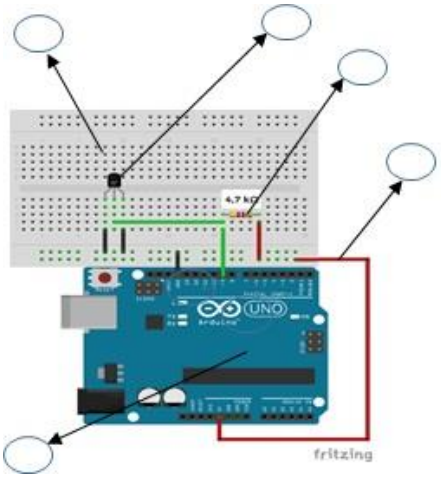


Figura 3.13. Tarjeta desarrolladora de Arduino en Tinkercard

Ejercicio 3.15
Completa el siguiente texto:

Construcción de un sistema automatizado para la obtención de valores de temperatura con un sensor de temperatura. Buscar textos y completar palabras del texto.

Todo comenzó un día por la mañana en la clase de Informática Aplicada a la ciencia y la industria, cuando el profesor, nos comentó en

clase, que deberíamos de llevar a cabo un proyecto de un sistema automatizado para medir la temperatura y formar equipos de trabajo, para realizar dicho proyecto.

Nos reunimos en equipo al finalizar la clase y analizamos el proyecto y los materiales que íbamos a utilizar para llevarlo a cabo como son: un _____, un _____, una _____, una _____ y una _____

Una vez que realizamos el _____ y programamos el _____ para que funcionara nuestro proyecto sobre un sistema automatizado de temperatura, se lo compartimos a nuestro profesor. Realizamos dos pruebas, la primera prueba consistió en pasar una paleta de hielo por el sensor y el _____ encendió en color azul y en la segunda prueba pasamos la llama de una vela y el _____ encendió de color rojo.

Ejercicio 3.16

Compra el siguiente material en una tienda de electrónica y construye el siguiente circuito:

Nombre	Cantidad
Placa de arduino uno	1
Sensor LM35	1
Diodo emisor de luz	1
Resistencia de 220 ohms	1
Tableta de prueba (Protoboard)	1

Figura 3.14. Lista de material Para construir un circuito que mide la temperatura.

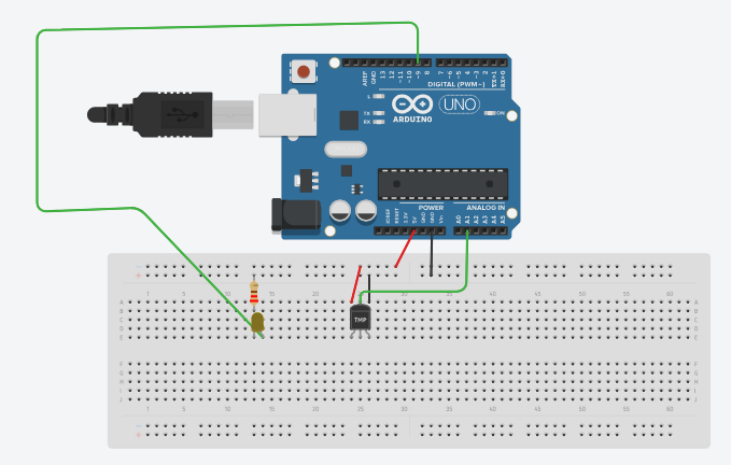


Figura 3.15. Circuito que mide temperatura Tinkercard

Uso de los elementos de un sistema automatizado para obtener y analizar datos que permitan emitir conclusiones

El objetivo de ese tema es que el alumno analice los datos adquiridos y emita conclusiones. Para esto las primeras actividades son ejercicios conceptuales para recordar las estructuras de control que se usan en programación en la primera unidad, así como el uso de la estadística descriptiva de la segunda unidad.

Después de construir un sistema automatizado, pasaremos a la obtención de la información. Este proceso se realizará con el uso de lenguajes de programación y podremos dar la interpretación a través de la estadística descriptiva que nos apoyará en obtener, organizar y presentar la información de una manera ordenada para poder emitir conclusiones.

Ejercicio 3.17
Lee el siguiente caso práctico y contesta las siguientes preguntas:

En un municipio de la costa grande del estado de Guerrero, se lleva a cabo un proyecto por parte de una empresa para automatizar el riego de las cosechas y así evitar perder los productos por falta de esta actividad.

El proyecto consiste en lo siguiente:

a) Por 60 días se debe regar la cosecha por la mañana.
b) En algunos productos de la cosecha se debe verificar, antes de regar, si el suelo está húmedo, si lo está no se debe regar para evitar dañar por el exceso de agua dichos productos.
c) Emita conclusiones si hay una relación por mes en la humedad del suelo con los datos obtenidos de todo el año.

Ejercicio 3.18

1. Para programar el sistema automatizado de riego, qué estructura de control recomendarías, para realizar el regado de las cosechas de manera cíclica hasta cumplir los 60 días.

2. ¿Qué estructura de control aplicamos en el programa del sistema automatizado de riego para determinar si el suelo está seco?

Ejercicio 3.19

Completa los siguientes pseudocódigos para detectar temperatura:

a)

Inicio

Entero X

_____ (X >=38 °C) _____

Encender LED color “Rojo”
ESCRIBIR “Tienes que regar”

_____

Encender LED color “Azul”
ESCRIBIR “No debes regar”

Fin SI

Fin

b)

Inicio

Real temp
_____

ESCRIBIR la temperatura de la ciudad es: “temp”

_____ (temp= 40°C)

Fin

Ejercicio 3.20

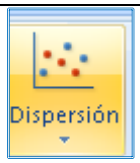
Se tienen los valores de temperatura de los meses de enero y febrero y se quiere saber si hay una correlación para emitir una conclusión.

1. En una hoja de cálculo copia estos valores y selecciona estas dos columnas.

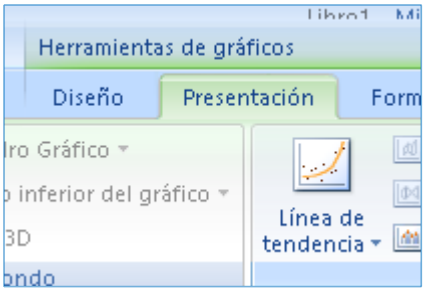
Enero	Febrero
21	18
47	19
29	17
2	9
24	15
11	14
17	39
19	2
42	47
42	33
25	12
29	32
44	32
4	30
23	41
29	40
1	17
12	35
17	14
21	27

Figura 3.16. Tabla de valores capturada en una hoja de cálculo

2. Ve a la opción insertar y elige la opción dispersión



3. En las herramientas de gráficos ve a líneas de tendencia y elige más opciones.



3 Y selecciona presentar ecuación en el gráfico y el valor de r cuadrado en el gráfico (Figura 3.17).

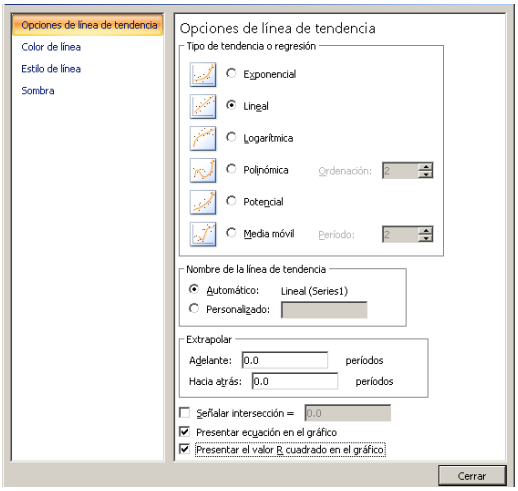


Figura 3.17. Presentar ecuación en el gráfico

Este es el resultado

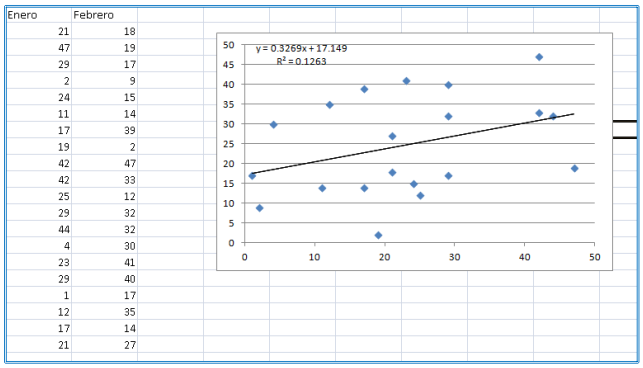


Figura 3.18. Correlación

Según Lind, D., Marchal, W., & Wathen, S. (2015), La correlación estadística nos indica si dos variables están relacionadas o no. El coeficiente de correlación de Pearson se utiliza para medir el grado de relación de dos variables. Su valor varía en el intervalo [-1,1], indicando el signo el sentido de la relación.

Es decir, Cuanto más cerca estén los coeficientes de +1,0 y -1,0, mayor será la fuerza de la relación entre las variables. Por lo que nuestra conclusión de este ejercicio es que la relación entre los meses de enero y febrero es débil ya que los valores no se acercan a 1, por lo que enero y febrero no tienen ninguna relación al comparar su temperatura.

Ejercicio 3.21

1. Arma el circuito detector de temperatura de la siguiente liga: Curso Arduino 7: Temperatura (codigofacilito, 2013) https://www.youtube.com/watch?v=QEIZjCVI2NQ y adquiere los datos de la temperatura ambiente de dos días a las 9 de la mañana.
2. Instala el software de arduino de la siguiente liga https://www.arduino.cc/en/Main/Software
3. Para obtener los datos de la tarjeta desarrolladora (Arduino) ve a la opción de herramientas y elegí la opción serial plotter.
4. Los datos adquiridos cópialos a una hoja de cálculo y genera un gráfico de dispersión.
5. Determina si hay una correlación entre las temperaturas de los dos días a esa hora en la mañana. Emite tu conclusión.

Función de los elementos básicos de un sistema

Para observar cómo funcionan los elementos básicos de un sistema automatizado, ve a la siguiente liga:

Sensor de temperatura y ventilador. Ejemplo sencillo de control de temperatura. https://www.prometec.net/regulacion-simple/

Donde podrás activar un ventilador cuando la temperatura es mayor a la establecida.

Realiza el ejemplo que se te pide y con los datos obtenidos compara haciendo una correlación de los datos obtenidos en tu sistema. Emite conclusiones sobre los resultados encontrados. Determina si este sistema no daña el medio ambiente.

Autoevaluación	

Nota: Puedes utilizar la placa desarrolladora, simulador o lenguaje de programación de tu preferencia. Aquí se propone la placa desarrolladora de arduino por la variedad de ejemplos y aplicaciones que maneja.

Se necesita construir un sistema automatizado para medir la temperatura de dos días consecutivos para saber cómo varia la temperatura en el periodo de 9 de la mañana a las 13 horas. Con el fin de investigar si hay una relación entre la temperatura de dos días a la misma hora. Por lo que primero debemos de saber

1. ¿Qué elementos debe usar en este sistema automatizado?

a) automatización, control, actuadores
b) sensores, controladores y actuadores
c) emisor, receptor, controlador
d) actuador, sensor, control

2. ¿Qué sensor debe elegir?

a) temperatura
b) calor
c) magnitud física
d) incremento

3. ¿Qué actuador debe elegir?

a) motor
b) valvula
c) sensor
d) ninguno

4. ¿Qué controlador debe elegir?

a) tarjeta desarrolladora
b) tarjeta de envío de datos
c) tarjeta procesadora
d) tarjeta de automatización

5. ¿Qué componentes se requieren tener en tu sistema para la adquisición de datos para medir la temperatura del periodo ya mencionado?

a) sensor, actuador, software
b) sensor, DAQ, computadora
c) sensor, DAQ, ADC, computadora
d) sensor, convertidor, software

6. Cuando se obtiene la información de la temperatura, ¿qué función tiene el software de aplicación?

a) procesar, almacenar, detectar información
b) procesar, almacenar, controlar actuadores
c) procesar, almacenar, analizar información
d) procesar, almacenar, controlar focos

7. De los dispositivos electrónicos básicos ¿Cuáles vas a usar para este sistema automatizado?

a) resistencia, led
b) capacitor, diodos
c) transistor, capacitor
d) diodo, led

8. Para el análisis de la información qué fórmulas te servirían para saber la correlación de los datos obtenidos.

a) estadística descriptiva
b) correlación
c) relación de variables
d) comparación

9. Si usa una hoja de cálculo que gráfico te servirá para representar tus datos

a) dispersión
b) correlación
c) pastel
d) barra

10. Si al automatizar un proceso industrial para la mejora de la calidad es necesario que se despida a muchas personas. ¿Automatizarías el proceso?

a) si
b) no
c) ¿por qué?

11. Si al automatizar un proceso industrial en la ciencia puedes apoyar a salvar vidas, pero con ello puedes contaminar el medio ambiente. ¿Automatizarías el proceso?

a) si
b) no
c) ¿por qué?

GLOSARIO

ADC. Convertidor de señales analógicas a digitales. siglas en inglés *Analog to Digital Converter*.

Actuador. Dispositivo que realiza una acción en respuesta a una señal eléctrica.

Algoritmo. Conjunto de pasos finitos bien definidos y con un orden lógico que permiten realizar una tarea.

Amplificador operacional. Es un dispositivo amplificador electrónico de alta ganancia.

Análisis de datos. Es una técnica y por medio de ésta se inspeccionan, purifican y transforman datos, con la finalidad de destacar toda la información que sea de gran utilidad, a fin de poder elaborar conclusiones que sirvan de apoyo en la toma de decisiones.

Argumento. Es el valor que es pasado a una subrutina cuando ésta es invocada.

Arreglo. Colección finita de celdas que sirve para almacenar datos homogéneos de forma temporal y ordenada.

Automatización. Proceso donde las acciones se desarrollan sin la participación directa de un individuo.

Automatización industrial. Aplicación de tecnologías tele-informáticas a las actividades de control de producción, minimizando la intervención humana. Sistemas que sean capaces de cerrar un lazo con la mínima intervención del operador. Implica medir el proceso, determinar su estado, tomar una decisión en base a un objetivo pautado y actuar sobre el proceso para llevarlo a su objetivo.

Biblioteca. Es un archivo que contiene funciones hechas previamente para realizar alguna tarea específica.

Bus. Medio de comunicación entre los diferentes elementos de una computadora o entre varias computadoras.

Capacitor o condensador eléctrico. Almacena energía en la forma de un campo eléctrico y se llama capacitancia o capacidad a la cantidad de cargas eléctricas que es capaz de almacenar.

Caracter. Una letra, un signo de puntuación, un símbolo especial (@, ^, etc.) o un número sobre el que no se pueden realizar operaciones aritméticas.

Celda. Representa un espacio de memoria que almacenar el valor de un elemento de un arreglo.

Coeficiente de Correlación. Se denomina correlación al vínculo recíproco o correspondiente que existe entre dos o más elementos. El concepto se emplea de diferentes maneras de acuerdo al contexto. En el ámbito de las matemáticas y las estadísticas, la correlación alude a la proporcionalidad y la relación lineal que existe entre distintas variables.

Controlador. Es un dispositivo encargado del control de la señal proporcionada por el sensor. Compara el valor de entrada con el valor deseado en la salida para poder comunicarse con dispositivos actuadores.

Conjunto. Agrupación de elementos del mismo tipo.

Constante. Es un dato que no puede cambiar su valor durante la ejecución de un programa.

Conteo. Es el proceso de contar y obtener un número.

Cualitativa. El símbolo que aparece en proposiciones, algoritmos, fórmulas y funciones y que adopta distintos valores recibe el nombre de variable. De acuerdo a sus características, se puede distinguir entre diferentes clases de variables.

Cuantitativa. Son aquellas que adoptan valores numéricos (es decir, cifras). De este modo se diferencian de las variables cualitativas, que el símbolo que aparece en proposiciones, algoritmos, fórmulas y funciones y que adopta distintos valores recibe el nombre de variable. De acuerdo a sus características, se puede distinguir entre diferentes clases de variables.

DAC. Convertidor de señales digitales a analógicas. Por sus siglas en inglés *Digital to Analog Converter*.

DAQ. Sistema de adquisición de datos. Por sus siglas en inglés *Data Acquisition*.

Dato. Es una cifra, letra, símbolo, palabra, frase u objeto que la computadora almacena en un formato determinado para poder procesarlo.

Dato estadístico. Con origen en el latín datum, refiere a la información que brinda acceso a un conocimiento preciso y concreto. Estadístico, por su parte, es aquello vinculado a la estadística: la especialidad de la matemática que apela a cifras para generar inferencias o para reflejar cuantitativamente un fenómeno.

Datos de entrada. Son aquellos que se van a procesar para dar un resultado.

Datos de salida. Son el resultado del procesamiento de los datos de entrada.

Diagrama de flujo. Representación gráfica de un algoritmo.

Digitalizar. Proceso en el que datos analógicos se transforman en digitales.

Diodo. Son componentes de dos terminales, cátodo y ánodo que permiten la circulación de la corriente en un solo sentido**.**

Dispositivo. Aparato o mecanismo que desarrolla determinadas acciones.

Dispositivo electrónico. Componente electrónico utilizado en los circuitos electrónicos.

Entidad. Es aquello que representa un objeto único, es exclusivo. Por ejemplo: funciones, variables, etc.

E/S. Entrada-Salida

Estadística. Significación, del latín significativo, es la acción y efecto de significar. Este verbo refiere a manifestar o hacer saber algo. Dicho de una palabra o expresión, está vinculado a ser el signo de un pensamiento o de algo material mientras que, aplicado a una cosa, refiere a ser, por naturaleza o convención, una representación de otra cosa distinta.

Estructuras anidadas. Se utiliza este término para referirse al uso de una estructura de control dentro del conjunto de instrucciones que forman parte de otra estructura de control.

Estructura de control. Permite definir el flujo de ejecución de las instrucciones de un programa.

Estructura de datos. Conjunto de variables asociadas bajo un mismo nombre.

Estructura de iteración. También se le conoce como ciclo o repetición. Permite la ejecución de una o más instrucciones varias veces.

Estructura secuencial. Es aquella en la que la ejecución de una instrucción sigue a la otra en secuencia.

Estructura selectiva. También conocida como decisión o comparación. Permite la ejecución de diferentes bloques de código dependiendo de la evaluación de una expresión lógica.

Expresión. Es una combinación de uno o más valores con uno o más operadores, que al evaluarse nos entregan un valor.

Flujo de datos. Se refiere al movimiento que presentan los datos dentro de la ejecución de un programa.

Función. Secuencia finita de instrucciones que realizan una tarea específica y retorna un valor.

Hardware. Son los elementos físicos que componen un equipo de cómputo.

Identificador. Es el nombre que se le asigna a cada espacio de memoria utilizado en un programa.

Informática aplicada. Conjunto de conocimientos científicos y técnicos que hacen posible el tratamiento automático de la información por medio de computadoras.

LED. Es una fuente luz construida con un material semiconductor que posee dos terminales.

Lenguaje de programación. Es un lenguaje formal que especifica una serie de instrucciones para que la computadora realice determinadas acciones.

Matriz. Es un arreglo de dos dimensiones, también conocido como arreglo bidimensional.

Microprocesador. Unidad de control de procesos.

Operador. Es un símbolo que representa una operación que se debe aplicar sobre uno o más valores.

Palabra reservada. Es un identificador predefinido por el compilador que tiene un significado especial y realiza una función específica dentro de la ejecución del programa.

Parámetro. Es una variable definida en una subrutina que recibe un valor cuando la subrutina es invocada.

PC. *Personal Computer*, Computadora Personal.

Periférico. Cualquier dispositivo que se conecta a un equipo de cómputo.

Preprocesador. Es un programa perteneciente al compilador que modifica el código fuente previo al proceso de traducción a código binario.

Procedimiento. Es un tipo de subrutina que no regresa ningún valor.

Programa. Es una secuencia de instrucciones que la computadora ejecuta para realizar una tarea específica.

Protoboard. Es una placa que contiene unos orificios conectados eléctricamente entre sí, siguiendo un patrón vertical y horizontal.

Pseudocódigo. Es el estilo de escritura que se usa en los algoritmos.

Resistencia eléctrica. Es una magnitud que se utiliza para medir la electricidad y se define como: la oposición que se presenta al paso de la corriente.

Sensor. Dispositivo que capta magnitudes físicas u otras alteraciones en su entorno.

Simuladores. Permite reproducir, explorar y manipular situaciones reales de manera virtual

Sintaxis. Es el conjunto de reglas que se deben seguir al escribir el código fuente de un programa y así considerarlo correcto para ese lenguaje de programación.

Software. Es un conjunto de programas cuya ejecución permite al usuario realizar un trabajo en un equipo de cómputo o dispositivo móvil.

Subrutina. Es una secuencia finita de instrucciones, separada del programa principal e identificada bajo un nombre, que resuelve una tarea específica.

Temperatura. Magnitud física

Tipo de dato. Determina el dominio (posibles valores que puede tener el dato) y las operaciones que se pueden realizar sobre el dato.

Transistor. Está formado por materiales semiconductores deja pasar o corta señales eléctricas a partir de una señal de mando (interruptor) y también funciona como un elemento amplificador de señales.

Valor de dato. La medición, recolección, análisis y elaboración de informes a partir de datos que se estén midiendo.

Variable. Es un espacio de memoria donde se puede almacenar un dato cuyo valor puede cambiar mientras se ejecuta un programa.

Variable local. Es aquella que se define dentro de una subrutina y sólo puede ser usada en ese ámbito.

Variable global. Es aquella que se utiliza en cualquier subrutina del programa y conserva su valor almacenado.

UNIDAD 1
Respuestas a los ejercicios

Ejercicio 1.1

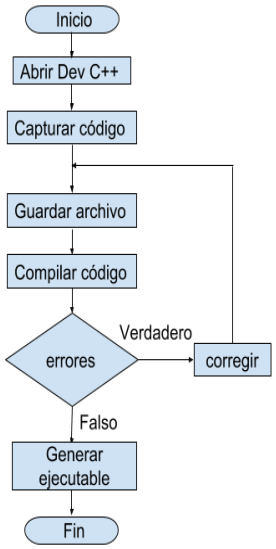
a) Pasos de la metodología de solución de problemas computables.

<table>
<tr><td colspan="3">Planteamiento del problema</td></tr>
<tr><td colspan="3">En esta primera fase se plantea el problema, el cual debe estar bien definido y delimitado, pues de ella depende el desarrollo para llegar a una óptima solución.</td></tr>
<tr><td colspan="3">Análisis de la solución del problema</td></tr>
<tr><td>Entrada: Se deben distinguir y analizar los datos de entrada que se utilizarán para la solución del problema.</td><td>Proceso: Son las operaciones que se realizarán sobre los datos.</td><td>Salida: Es el resultado de los datos de entrada procesados.</td></tr>
<tr><td colspan="3">Planteamiento de alternativas de solución: Generalmente, la solución de un problema puede alcanzarse por varias vías. Es útil plantear diferentes alternativas, pero hay que cuidar no excederse, pues una mayor cantidad puede dificultar la elección de la mejor.</td></tr>
<tr><td>Alternativa 1:</td><td>Alternativa 2:</td><td>Alternativa 3:</td></tr>
<tr><td colspan="3">Diseño de la solución del problema</td></tr>
<tr><td colspan="3">Una vez elegida la mejor alternativa, se realiza el diseño de la solución, esta fase consta de dos herramientas:</td></tr>
<tr><td colspan="3">Algoritmo. Es una secuencia ordenada de pasos o instrucciones a seguir para llegar a la solución de un problema, se escribe en pseudocódigo y es independiente de los lenguajes de programación.

Diagrama de flujo. Es la representación gráfica del algoritmo.</td></tr>
<tr><td colspan="3">Codificación</td></tr>
</table>

Consiste en escribir el algoritmo en un lenguaje de programación.

b) Realiza el algoritmo o diagrama de flujo del proceso para capturar, compilar, depurar y ejecutar un programa en lenguaje C utilizando el entorno de desarrollo integrado Dev-C++.

ALGORITMO	DIAGRAMA DE FLUJO
INICIO 1. Abrir entorno de desarrollo Dev C++ 2. Abrir archivo fuente 3. Capturar código de programación 4. Guardar archivo 5. Compilar código 6. Si hay errores ● corregir, guardar, compilar 7. Si no ● Generar archivo ejecutable FIN	

Ejercicio 1.2. Escribe la declaración en lenguaje C de las siguientes variables:

Descripción del dato	Declaración de la variable
Velocidad inicial de un automóvil.	float vel_inicial;
Velocidad final de un automóvil.	float vel_final;
Codifica en una sola instrucción de C, la declaración de las variables velocidad inicial y final de un automóvil.	float vel_inicial, vel_final;
Posición de la palanca de velocidades de un auto (P, R, N, 1, 2 o 3).	char palanca;
Bolsas de aire que tiene un coche.	int bolsas_aire;

Ejercicio 1.3. Escribe la declaración en lenguaje C de las siguientes constantes:

Descripción del dato	Declaración de la variable
Capacidad máxima del tanque de combustible de un tráiler (1500 l).	#define LITROS_MAX 1500
Mensaje que indique que se ha alcanzado la velocidad máxima	#define MENSAJE “Velocidad máxima alcanzada”
Rendimiento de combustible de un automóvil (5.8 l/100 km).	#define RENDIMIENTO 5.8/100

Ejercicio 1.4. Identifica los datos necesarios para resolver cada problemática y llena la siguiente tabla:

a) Se desea conocer la distancia que recorre un auto que se desplaza en línea recta a una velocidad de 20 km/h durante 45 minutos

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato
distancia recorrida por el auto	distancia	números reales	float	float distancia;
velocidad del auto	velocidad	números reales	float	float velocidad=20;
tiempo del recorrido	tiempo	números enteros	int	int tiempo=45;

b) Desde un edificio se deja caer una pelota, que tarda 8 segundos en llegar al piso. ¿con qué velocidad impacta la pelota contra el piso?

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato
tiempo en caer	tiempo	números enteros	int	int tiempo=8;
velocidad pelota	velocidad	números reales	float	float velocidad;

fuerza de gravedad	GRAVEDAD	9.8	float	#define GRAVEDAD 9.8

c) Un profesor tiene registrados los promedios de cada uno de sus estudiantes de un grupo de Informática de la ENP y desea calcular el promedio del grupo. También desea saber cuántos alumnos exentaron el examen final, considerando que la calificación mínima para exentar es 8.

Dato	Identificador	Dominio	Tipo de dato	Declaración e inicialización del dato
promedio de cada alumno	prom_a	números reales	float	float prom_a;
promedio del grupo	prom_g	números reales	float	float prom_g;
número de alumnos del grupo	num_a	números enteros	int	int num_a;
total de alumnos exentos	num_e	números enteros	int	int num_e;
calificación mínima para exentar	CAL_MIN_E	8	int	#define CAL_MIN_E 8

Ejercicio 1.5. Calcula el resultado de las siguientes expresiones:

Expresión	Resultado

int a = 1;
int b = 2;
int c;
c = a + b * 3 - a * b + 3;
	c = 8

#define FACTOR 10
int x = 5;
int y = 4;
int z, w;
z = (x + y) * FACTOR;
w = x + y * FACTOR;
	z = 90
	w = 45

Ejercicio 1.6. Escribe las expresiones en lenguaje C para representar las siguientes expresiones matemáticas y lógicas:

Expresión matemática	Expresión en lenguaje C
$a = \Pi r^2$	a = 3.1416 * r * r;
$c = \frac{1}{ab}$	c = 1 / (a * b);
$c = \frac{ab}{c}$	c = a * b / c;
$v = \frac{d}{t_f - t_i}$	v = d / (tf - ti)
Expresión relacional	**Expresión en lenguaje C**
a es mayor que b y que c	a>b && a>c
temperatura es mayor que 10 y menor que 20	temperatura > 10 && temperatura < 20
6<=calificación<=10	calificacion >= 6 && calificacion <= 10
a y b son iguales	a == b
a es igual a b, pero diferente de c	a == b && a != c
a es igual a b o b es igual a c	a == b || b == c

Ejercicio 1.7. Analiza la codificación en C del *programa 3 Cálculo de los kilómetros recorridos y litros consumidos* y responde las siguientes preguntas:

¿Qué biblioteca se está utilizando?	stdio.h
¿Con qué identificador se nombra a la constante del programa y cuál es el valor que almacena?	Identificador: RENDIMIENTO Valor: 5.8/100
¿Cuáles son los tipos de datos de las variables del programa?	Todas se declaran de tipo float

¿Cuál es el dominio del tipo de datos cada una de las variables del programa?	Números reales
¿Qué secuencias de escape se utilizan en el programa y qué hace cada una de ellas?	\n salto de línea \t tabulador
En la cadena de control: "\nKm recorridos: %2.2f \tlitros consumidos: %2.2f", qué significado tiene %2.2f?	Que se va a mostrar en la pantalla el valor de una variable tipo float con el formato de dos enteros y dos decimales.

Ejercicio 1.8. Diseña los programas en C que resuelvan los siguientes problemas. Solicita al usuario que capture los datos de entrada que necesites conocer.

a) Distancia que recorre un auto que se desplaza en línea recta a una velocidad constante durante un periodo de tiempo.

```
#include <stdio.h>
int main()
{
	float distancia, velocidad, tiempo;
	printf("\tPrograma que calcula la distancia que recorre un auto a una velocidad constante");
	printf("\nCapture la velocidad del auto en km/h: ");
	scanf("%f", &velocidad);
	printf("\nCapture el tiempo en horas: ");
	scanf("%f", &tiempo);
	distancia=velocidad*tiempo;
	printf("\n\n \t\t La distancia recorrida es: %0.2f km", distancia);
	return 0;
}
```

b) La velocidad con que impacta un objeto en caída libre.

```
#include <stdio.h>
#define GRAVEDAD 9.8
int main()
{
	float velocidad;
	int tiempo;
	printf("Programa que calcula la velocidad con que impacta un objeto en caida libre");
```

```
        printf("\nCapture el tiempo que tarda en caer el objeto: ");
        scanf("%d", &tiempo);
        velocidad=tiempo*GRAVEDAD;
        printf("\nLa velocidad con que impacta el objeto es: %f",velocidad);
        return 0;
}
```

Ejercicio 1.9. Captura, compila y ejecuta el programa anterior probando los casos descritos en la prueba de escritorio. ¿Qué resultado arrojará el programa si la temperatura exterior e interior del auto son iguales?

El programa no muestra ningún mensaje, ya que el resultado de la expresión temp_int > temp_ext es falso (en el lenguaje C, la expresión es igual a 0).

Ejercicio 1.10. Utilizando el método de solución de problemas, diseña los programas en lenguaje C que resuelvan cada una de las situaciones planteadas:

a) Encender el testigo de aire acondicionado de un automóvil si la temperatura interior del auto excede los 20°C.

```
/* Programa
Encender el testigo de aire acondicionado de un automóvil
si la temperatura interior del auto excede los 20°C.
*/
#include <stdio.h>
#define TEMP_MAX 20
#define MENSAJE "Testigo de aire acondicionado encendido"
int main() {
        float temp_int;
        printf("Programa que enciende el testigo de aire acondicionado");
        printf("\nDigite la temperatura interior del auto: ");
        scanf("%float", &temp_int);
        if (temp_int > TEMP_MAX) {
                printf("%s",MENSAJE);
        }
        return 0;
}
```

b) Enviar una advertencia al conductor si la presión de una llanta está fuera del intervalo: [29, 41] psi.

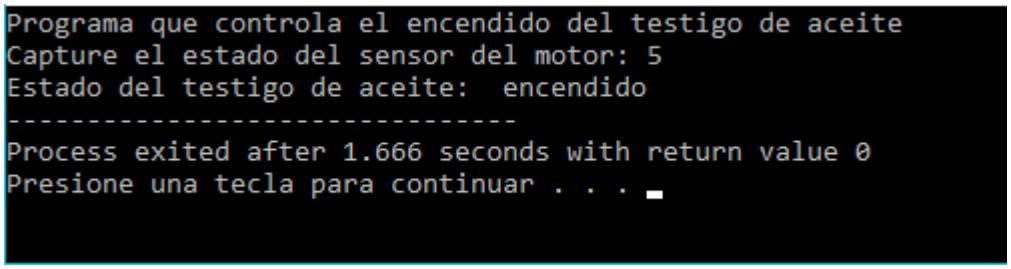
```
/* Programa
Enviar una advertencia al conductor si la presión de una llanta
está fuera del intervalo: [29, 41] psi.
*/
#include <stdio.h>
#define PRESION_MIN 29
#define PRESION_MAX 41
#define MENSAJE "Presion de la llanta fuera del intervalo"
int main() {
	float presion;
	printf("Programa que advierte si una llanta tiene la presion fuera del
intervalo [29, 41]");
	printf("\nDigite la presion de la llanta: ");
	scanf("%float", &presion);
	if (presion < PRESION_MIN || presion>PRESION_MAX) {
		printf("%s",MENSAJE);
	}
	return 0;
}
```

c) Encender la luz de advertencia de gasolina si el tanque de gasolina contiene entre 0 y 5 litros.

```
/* Programa
Encender la luz de advertencia de gasolina si el tanque de gasolina contiene entre
0 y 5 litros.
*/
#include <stdio.h>
#define L_MIN 0
#define L_MAX 5
#define MENSAJE "Testigo de gasolina encendido"
int main() {
	float litros;
	printf("Programa que enciende el testigo de gasolina si se encuentra en la
reserva");
	printf("\nDigite los litros de gasolina que contiene el tanque: ");
	scanf("%float", &litros);
	if (litros >= L_MIN && litros <= L_MAX) {
		printf("%s", MENSAJE);
	}
	return 0;
}
```

Ejercicio 1.11. Captura, compila y ejecuta el programa 5 con los dos casos de prueba. ¿Qué resultado se obtiene si capturas un número distinto de 0 y 1 cuando la computadora te solicita el estado del sensor del motor?, ¿por qué?

El resultado de la ejecución del programa es:



Muestra el mensaje Estado del testigo de aceite: encendido porque al evaluarse la expresión *estado_sensor* el valor que devuelve es 5, que es diferente de 0, por lo que la expresión es verdadera, por lo tanto se ejecutan las instrucciones de dicha rama del *if*.

Ejercicio 1.12. Utiliza la metodología de solución de problemas para diseñar los siguientes programas en C:

a) Modifica el programa 4 de esta guía para que también avise al conductor si el sistema desempañador está desactivado.

```
/*      Programa que avisa al conductor si el sistema desempañador
está activado o desactivado
*/
#include <stdio.h>
#define MENSAJE_A "Sistema desempañador activado"
#define MENSAJE_D "Sistema desempañador desactivado"
int main() {
        float temp_ext;
        float temp_int;
        printf("Programa de activacion del sistema desempañador");
        printf("\nCapture la temperatura interior del auto: ");
        scanf("%f", &temp_int);
        printf("Capture la temperatura exterior del auto: ");
        scanf("%f", &temp_ext);
        if (temp_int > temp_ext) {
                printf("%s", MENSAJE_A);
        }
        else {
                printf("%s", MENSAJE_D);
```

```
        }
        return 0;
}
```

b) Simular la activación de la alerta sísmica de la Ciudad de México, considerando que solamente se debe activar en caso de que el temblor tenga una magnitud superior a 6 grados, en caso contrario, no se activará. Solicita al usuario la magnitud del temblor y con base en ésta, envía el mensaje de si la alerta sísmica se activa o no se activa.

```
/*      Programa que simula la activación de la alerta sísmica
        en caso de que el temblor tenga una magnitud mayor que 6
*/
#include <stdio.h>
#define MAGNITUD_ALERTA 6
#define MENSAJE_A "Alerta sismica activada"
#define MENSAJE_D "Alerta sismica desactivada"
int main() {
        float magnitud;
        printf("Programa de activacion de la alerta sismica");
        printf("\nCapture magnitud del temblor: ");
        scanf("%f", &magnitud);
        if (magnitud > MAGNITUD_ALERTA) {
                printf("%s", MENSAJE_A);
        }
        else {
                printf("%s", MENSAJE_D);
        }
        return 0;
}
```

c) Calcular el importe a pagar en la compra de un artículo, considerando que si el monto de compra es superior a $2,000.00, se realiza un descuento del 10%.

```
/* Programa que calcula el importe a pagar
        en la compra de un artículo, considerando
        que si el monto de compra es superior a $2,000.00,
        se realiza un descuento del 10%.
*/
#include <stdio.h>
#define DESCUENTO 10
#define MONTO_MIN 2000
int main() {
        float precio=500;
        float total_pagar;
```

```
        printf("Programa que calcula el importe a pagar por la compra de un articulo");
        printf("\nSi la compra es mayor a $2,000.00, se hace un descuento del 10%");
        printf("\nDigite el precio del articulo: ");
        scanf("%f", &precio);
        if (precio > MONTO_MIN) {
                total_pagar = precio - precio * DESCUENTO / 100;
        }
        else {
                total_pagar = precio;
        }
        printf("\nTotal a pagar: %5.2f", total_pagar);
        return 0;
}
```

Ejercicio 1.13. Realiza la captura, compilación y ejecución del programa 6. Haz la prueba de escritorio para todos los valores posibles de día y además, prueba capturar la clave del día con letra minúscula. ¿Qué mensaje se muestra al usuario?, ¿por qué?

El mensaje que muestra al usuario es: "*Error al capturar la clave del dia. Utilice mayusculas*" porque al teclear la clave del día en letras minúsculas, este valor no es igual a ninguno de los definidos en los casos del switch y por lo tanto se ejecuta el código correspondiente al default.

Ejercicio 1.14. Diseña los programas en lenguaje C, utilizando el método de solución de problemas, que resuelva las situaciones planteadas:

a) En un automóvil de transmisión automática existen diferentes velocidades (P, R, N, D, 1, 2 y 3). Elabora un programa que encienda el testigo correspondiente a la velocidad seleccionada por el conductor.

```
// Programa que enciende el testigo del estado de la palanca de velocidades
#include <stdio.h>
int main() {
  char dia;
  printf("Programa que enciende el testigo del estado de la palanca de velocidades");
  printf("\nEstados");
  printf("\n\tP");
  printf("\n\tR");
  printf("\n\tN");
  printf("\n\tD");
  printf("\n\t1");
  printf("\n\t2");
```

```
   printf("\n\t3");
   printf("\nCapture el estado de la palanca de velocidad: ");
   scanf("%c",&dia);
   switch (dia) {
     case 'P': printf("Testigo P encendido");
                 break;
     case 'R': printf("Testigo R encendido");
                 break;
     case 'N': printf("Testigo N encendido");
                 break;
     case 'D': printf("Testigo D encendido");
                 break;
     case '1': printf("Testigo 1 encendido");
                 break;
     case '2': printf("Testigo 2 encendido");
                 break;
     case '3': printf("Testigo 3 encendido");
                 break;
   }
return 0;
}
```

b) Se necesita calcular el importe a pagar de un boleto de cine, considerando los siguientes costos y promociones:

Costo del boleto: $75.00	
Día de la semana	Promoción
Lunes y martes	50% de descuento
Miércoles	2x1
Jueves	20% descuento
Viernes, sábado y domingo	Ninguna

```
/*      Programa que calcula el precio a pagar por un boleto de cine
        considerando las promociones:
        L y M 50% de descuento, Mi 2x1, J 20% de descuento
        V, S y D No aplica ninguna promoción
*/
#include <stdio.h>
#define COSTO_BOLETO 75
#define DESC_LM 0.50
#define DESC_J 0.80
```

```
int main() {
        float precio;
        char dia;
        printf("Programa para calcular el precio de un boleto de cine");
        printf("\nClaves del dia");
        printf("\n\tL - lunes");
        printf("\n\tM - martes");
        printf("\n\tI - miercoles");
        printf("\n\tJ - jueves");
        printf("\n\tV - viernes");
        printf("\n\tS - sabado");
        printf("\n\tD - domingo\n");
        printf("\nCapture la clave del dia: ");
        scanf("%c",&dia);
        switch (dia) {
    case 'L':   precio = COSTO_BOLETO * DESC_LM;
                break;
    case 'M': precio = COSTO_BOLETO * DESC_LM;
                break;
    case 'I': precio = COSTO_BOLETO;
                break;
    case 'J':   precio = COSTO_BOLETO * DESC_J;
                break;
    case 'V': precio = COSTO_BOLETO;
                break;
    case 'S':  precio = COSTO_BOLETO;
                break;
    case 'D': precio = COSTO_BOLETO;
                break;
    default: printf("Error al capturar la clave del dia. Utilice mayusculas");
                break;
        }
        printf("El precio del boleto es: %f", precio);
        return 0;
}
```

Ejercicio 1.15. Captura, compila y ejecuta el programa 7 para diferentes casos. Prueba qué pasa si el número de multas es 0. En este caso, ¿cuántas veces se ejecutó el ciclo? ¿por qué?

Ninguna vez, ya que el valor de las variables *cont_multas* y *total_multas* es cero y al evaluarse la expresión cont_multas < total_multas (0 < 0) el resultado es 0 (falso) por lo que el control del programa se va al final de la estructura de control *while*.

Ejercicio 1.16. Diseña un programa en C que resuelva las siguientes problemáticas utilizando la metodología de solución de problemas:

a) Modifica el *programa 4 Activación automática del sistema desempañador del vidrio trasero de un automóvil* de esta guía para que el usuario lo ejecute el número de veces que él desee.

```
#include <stdio.h>
#define MENSAJE "Sistema desempañador activado"
int main() {
	float temp_ext;
	float temp_int;
	char continuar='S';
	while (continuar != 'N') {
		printf("\nPrograma de activacion del sistema desempañador");
		printf("\nCapture la temperatura interior del auto: ");
		scanf("%f",&temp_int);
		printf("Capture la temperatura exterior del auto: ");
		scanf("%f",&temp_ext);
		if (temp_int>temp_ext) {
			printf("%s",MENSAJE);
		}
	printf("\nDeseas volver a ejecutar el programa: ");
	scanf(" %c", &continuar);
	}
	return 0;
}
```

b) Modifica el programa del ejercicio 14, inciso a) de esta guía, para que se pueda capturar el estado de la palanca de velocidades de un auto tantas veces desee el usuario.

```
// Programa que enciende el testigo del estado de la palanca de velocidades
#include <stdio.h>
int main() {
	char dia;
	char continuar='S';
	while (continuar != 'N') {
		printf("\nPrograma que enciende el testigo del estado de la palanca de
velocidades");
		printf("\nEstados");
```

```
                printf("\n\tP");
                printf("\n\tR");
                printf("\n\tN");
                printf("\n\tD");
                printf("\n\t1");
                printf("\n\t2");
                printf("\n\t3");
                printf("\nCapture el estado de la palanca de velocidad: ");
                scanf(" %c",&dia);
                switch (dia) {
                case 'P': printf("Testigo P encendido");
                                break;
                case 'R': printf("Testigo R encendido");
                                break;
                case 'N': printf("Testigo N encendido");
                                break;
                case 'D': printf("Testigo D encendido");
                                break;
                case '1': printf("Testigo 1 encendido");
                                break;
                case '2': printf("Testigo 2 encendido");
                                break;
                case '3': printf("Testigo 3 encendido");
                                break;
                }
        printf("\nDeseas volver a ejecutar el programa: ");
        scanf(" %c", &continuar);
        }
        return 0;
}
```

Ejercicio 1.17. Captura, compila y ejecuta el programa 8 para diferentes casos y responde las siguientes preguntas:

a) Si el número de multas es 0, ¿se ejecuta alguna vez el ciclo? ¿por qué?

No se ejecuta ninguna vez el ciclo porque cuando el número de multas es 0, la variable *total_multas* toma el valor de 0, después el *for* asigna a la variable *cont_multas* el valor de 1 y evalúa la expresión *cont_multas* < *total_multas* (0<0), y el resultado es 0 (falso) por lo que el control del programa se va al final de la estructura de control *for*.

b) ¿Por qué en la última instrucción del programa se le resta uno a la variable *cont_multas*?

Porque el valor con el que termina la variable cont_multas al finalizar el ciclo siempre valdrá el número de multas + 1, dado que la estructura *for* finaliza cuando la expresión *cont_multas* < *total_multas* sea falsa, es decir, cuando (n+1<n).

Ejercicio 1.18. El programa “Pago de multas” que utiliza la estructura *for* da el mismo resultado que el que emplea la estructura *while*, sin embargo, al comparar las pruebas de escritorio se observa que no son iguales. Responde las siguientes preguntas:

a) ¿En qué varían y por qué?

En el ciclo *while* la variable *cont_multas* inicia con el valor 0 y termina con 3, mientras que en el ciclo *for* la variable *cont_multas* inicia con el valor de 1 y termina con 4.

b) ¿Podría empezar el ciclo *for* inicializando la variable *cont_multas* con 0 como en la solución que emplea el *while*? Si es así, ¿qué cambios tendrías que hacer al programa?

Sí se puede, el programa sería:

```
#include <stdio.h>
int main() {
	int total_multas;
	int cont_multas;
	float importe_multa;
	float importe_total = 0;
	printf("Programa que calcula el importe a pagar por concepto de multas");
	printf("\nCuantas multas va a pagar: ");
	scanf("%d", &total_multas);
	for(cont_multas = 0; cont_multas < total_multas; cont_multas++) {
		printf("\nImporte de la multa %d: ", cont_multas + 1);
		scanf("%f", &importe_multa);
		importe_total = importe_total + importe_multa;
	}
	printf("\nImporte a pagar de %d multas es: $%5.2f", cont_multas, importe_total);
}
```

Ejercicio 1.19. Diseña los programas en lenguaje C, que resuelvan las problemáticas planteadas utilizando la metodología de solución de problemas:

a) Tu tío es dueño de una pequeña tlapalería. La persona a cargo del mostrador tiene problemas para calcular el importe total que debe pagar un cliente cuando compra varios productos. Por lo anterior, decides ayudarlo haciendo una aplicación que solicita el número total de productos comprados, el importe de cada uno y calcula el total a pagar.

```
#include <stdio.h>
int main() {
	int total_prod;
	int num_prod;
	float precio;
	float total_pagar;
	printf("Programa que calcula el total a pagar de varios productos");
	printf("\nCapture el total de productos: ");
	scanf("%d", &total_prod);
	for (num_prod = 1; num_prod <= total_prod; num_prod++) {
		printf("\nDigite el precio del producto %d: ", num_prod);
		scanf("%f", &precio);
		total_pagar = total_pagar + precio;
	}
	printf("\nEl total a pagar por los %d productos es: %5.2f",num_prod-
1,total_pagar);
	return 0;
}
```

b) Estás realizando una investigación sobre el cambio climático en la CDMX dentro de la Estación Meteorológica de la ENP. Tu profesor de física, quien te asesora en este proyecto, te solicita hacer un programa que determine la temperatura máxima registrada en el día. En tus notas, dispones de 24 lecturas de temperatura diarias.

```
#include <stdio.h>
#define NO_LECTURAS 24
int main() {
	float temp_min;
	float temp_max;
	float temp;
	int lectura;
	printf("Programa que calcula la temperatura maxima del dia");
	for (lectura=1; lectura <= NO_LECTURAS; lectura++) {
		printf("\nDigite la lectura %d de temperatura: ", lectura);
		scanf("%f", &temp);
		if (temp > temp_max) {
```

```
                temp_max = temp;
            }
        }
        printf("\nLa temperatura maxima del dia fue: %5.2f grados",temp_max);
        return 0;
}
```

Ejercicio 1.20. Captura, compila y ejecuta el programa 9 para que observes los resultados. ¿Puede suceder el caso de que el programa no calcule ningún importe?

No, al utilizar la estructura de iteración *do-while*, el programa calculará al menos un importe.

Ejercicio 1.21. Diseña un programa en C que resuelva las siguientes problemáticas utilizando la metodología de solución de problemas:

c) Modifica el *programa 4 Activación automática del sistema desempañador del vidrio trasero de un automóvil* de esta guía para que el usuario lo ejecute el número de veces que él desee. Emplea la estructura *do-while* en la solución.

```
#include <stdio.h>
int main() {
        #define MENSAJE "Sistema desempañador activado"
        float temp_ext;
        float temp_int;
        char continuar;
        do {
            printf("\nPrograma de activacion del sistema desempañador");
            printf("\nCapture la temperatura interior del auto: ");
            scanf("%f",&temp_int);
            printf("Capture la temperatura exterior del auto: ");
            scanf("%f",&temp_ext);
            if (temp_int>temp_ext) {
                printf("%s",MENSAJE);
            }
        printf("\nDeseas volver a ejecutar el programa: ");
        scanf(" %c", &continuar);
        } while (continuar != 'N');
        return 0;
}
```

d) Modifica el programa del ejercicio 14, inciso a) de esta guía, para que se pueda capturar el estado de la palanca de velocidades de un auto tantas veces desee el usuario. Emplea la estructura *do-while* en la solución.

```
// Programa que enciende el testigo del estado de la palanca de velocidades
#include <stdio.h>
int main() {
	char dia;
	char continuar;
	do {
		printf("\nPrograma que enciende el testigo del estado de la palanca de
velocidades");
		printf("\nEstados");
		printf("\n\tP");
		printf("\n\tR");
		printf("\n\tN");
		printf("\n\tD");
		printf("\n\t1");
		printf("\n\t2");
		printf("\n\t3");
		printf("\nCapture el estado de la palanca de velocidad: ");
		scanf(" %c",&dia);
		switch (dia) {
		case 'P': printf("Testigo P encendido");
				break;
		case 'R': printf("Testigo R encendido");
				break;
		case 'N': printf("Testigo N encendido");
				break;
		case 'D': printf("Testigo D encendido");
				break;
		case '1': printf("Testigo 1 encendido");
				break;
		case '2': printf("Testigo 2 encendido");
				break;
		case '3': printf("Testigo 3 encendido");
				break;
		}
	printf("\nDeseas volver a ejecutar el programa: ");
	scanf(" %c", &continuar);
	}	while (continuar != 'N');
	return 0;
}
```

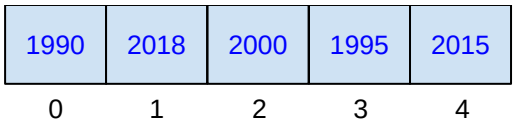
Ejercicio 1.22. Escribe la declaración de los siguientes arreglos en lenguaje C:

Descripción de los datos	Declaración del arreglo en C
El kilometraje de 10 viajes	float kilometraje [10];

15 números de motor de vehículos	int motor [15];
7 modelos de vehículos (año)	int modelo [7];
Los litros de gasolina gastados en 5 viajes	float gasolina [5];

Ejercicio 1.23. Realiza las referencias a los elementos del arreglo que se te indican:

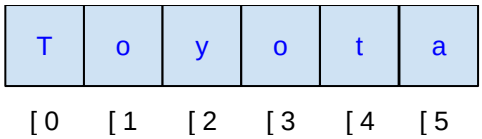
modelo_auto=



Referencia	Respuesta
Al primer elemento	modelo_auto[0]
Al tercer elemento	modelo_auto[2]
Al último elemento	modelo_auto[4]

Ejercicio 1.24. Asigna a los elementos de un arreglo los valores que se te indican:

marca_auto



Asignación	Respuesta
En la declaración haciendo referencia a sus elementos	char marca_auto [6]; marca_auto[0]=’T’; marca_auto[1]=’o’; marca_auto[2]=’y’; marca_auto[3]=’o’; marca_auto[4]=’t’; marca_auto[5]=’a’;
En la declaración	char marca_auto[6]={“T”,”o”,”y”,”o”,”t”,”a”};

Ejercicio 1.25. Programa Registro de viajes

Un automóvil realiza en un día 10 viajes, el kilometraje de cada viaje es guardado y al final del día, la computadora del automóvil despliega el total de kilómetros recorridos y pregunta al usuario si desea el listado de los kilómetros de cada viaje. Si el usuario selecciona la opción de listado, despliega en la pantalla cada viaje y los kilómetros recorridos. Diseña el programa que resuelva este problema.

```
#include<stdio.h>
#include<conio.h>
int  main ()
{
  float kilometros[10];
  int i;
  int despliega=0;
  float Total_Km=0.0;
  printf (" Programa kilometraje de viaje \n");
  for (i=0; i<=9; i++)
  {
       printf (" \nProporciona el kilometraje del viaje No. %d:",i+1);
       scanf("%f", &kilometros[i]);
  }

  for (i=0; i<=9; i++)
  {
       Total_Km=Total_Km+kilometros [i];
  }
  printf("\n El total de kilómetros  recorridos es: %.2f \n", Total_Km);
  printf("Desea el listado de los viajes [1-Sí, 2-No] ?");
  scanf ("%d",&despliega);

  if (despliega==1)
  {
     printf ("Los viajes realizados son:\n");
     for (i=0; i<=9; i++)
     {   printf ("Viaje [%d]: %.2f kms \n", i+1 , kilometros[i]); }
     despliega =0;
  }
getch ();
}
```

Ejercicio 1.26. Realiza la declaración de los arreglos bidimensionales requeridos para solucionar las siguientes problemáticas:

a) En una fábrica de refacciones, los productos que ya están listos para su distribución se encuentran almacenadas en una bodega. La bodega tiene 6 anaqueles con 3 repisas cada uno. Para identificar las refacciones en la bodega se les asigna un número de inventario (un número de 3 dígitos). Para

tener guardados los números de las refacciones, ¿cómo se debe declarar el arreglo?

b) Una empresa automotriz lleva con un programa el orden y el registro de los montos de ventas por mes de 5 sucursales, ¿cómo implemento en un programa de C la estructura para almacenar los montos de ventas?

Declaración del arreglo	
a) int refacciones [6][3] ;	Donde las filas corresponden a los anaqueles y las columnas a los entrepaños.
b) float ventas [12][5] ;	Donde las filas corresponden a los meses y las columnas a las sucursales.

Ejercicio 1.27. Realiza las referencias a los valores del arreglo que se te indican:

autos_ventas=

	[0]	[1]	[2]	[3]
[0]	15	13	12	4
[1]	18	5	8	10
[2]	25	16	11	7

Referencia	Respuesta
Al valor 12	ventas_auto [0][2]
Al valor 18	ventas_auto [1][0]
Al valor 7	ventas_auto [2][3]
Al valor 16	ventas_auto [2][1]
Al valor 10	ventas_auto [1][3]

Ejercicio 1.28.
Tomando como base el arreglo bidimensional del ejercicio Ejercicio 1.27, realiza la asignación de los valores del arreglo bidimensional como se pide:

Asignación en la	Respuesta

declaración	
Describiendo todos los elementos del arreglo en una línea de código	int autos_ventas[3][4]={15,13,12,4,18,5,8,10, 25,16,11,7};
Con filas en renglones diferentes	int autos_ventas=[3] [4]= { {15,13,12, 4} , {18, 5, 8, 10} {25, 16,11,7}};

Ejercicio 1.29. Programa Refacciones

Una agencia guarda sus refacciones en un almacén que está integrado por varios anaqueles. Cuando se localiza una refacción en el almacén, se identifica primero el anaquel y posteriormente el entrepaño en el que se encuentra. ¿Cómo implementarías la organización y localización de las refacciones utilizando arreglos?

Supongamos que el almacén tiene cuatro anaqueles con dos entrepaños cada uno. Por tanto, se tendría que crear un arreglo bidimensional que representara el almacén. Las columnas serían los anaqueles y las filas los entrepaños (Ilustración 1.30).

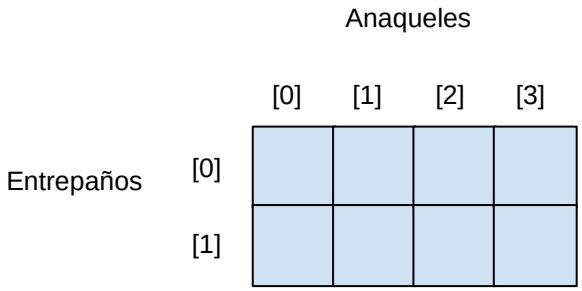


Ilustración 1.30. Representación del almacén en un arreglo bidimensional.

La agencia desea colocar el precio a cada una de las refacciones de su almacén. Diseña un programa en C que solicite al usuario el precio de la refacción según su ubicación y muestre en pantalla un listado con los precios de todas sus refacciones.

```
#include <stdio.h>
#include <conio.h>
int main()
{
  float refacciones  [2][4];
  int fila;
  int col;
  /* lectura de datos*/
  printf("Almacén de refacciones\n\n");
```

```
  for (col=0; col<4; col++)
  {
  for (fila =0; fila < 2; fila++)
    {
        printf("Precio refacción anaquel %d, entrepaño %d: ", col, fila);
        scanf ("%f", &refacciones [fila][col]);
    }
  }
/* Impresión de los datos*/
  printf("\nPrecio de refacciones\n");
  for (col=0; col<4; col++)
  {
          for (fila =0; fila < 2; fila++)
          { printf("Precio refacción anaquel %d, entrepaño %d: $ %.2f \n", col, fila,
refacciones [fila][col]); }
  }
  getch();
  return 0;
}
```

Ejercicio 1.30.

Según las siguientes declaraciones de funciones en lenguaje C, indica el tipo de valor que devuelve, los parámetros (si es que tiene) y de qué tipo son éstos, ya sea paso por valor o paso por referencia.

a) int calculaEdad(int dianacimiento, int mesnaciemiento, int anionacimiento)				
Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?
int (entero)	calculaEdad	dianacimiento, mesnaciemiento y anionacimiento	Ninguno	Función
b) void intercambioDeValores(int *val1, int *val2)				
Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?
void	intercambioDeValores	Ninguno	val1, val2	Procedimiento
c) float promediaCalificaciones(float calif[100], int numvalores)				

Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?
float	promediaCalificaciones	numvalores	calif[100]	Función
d) void punto(float x1, float y1, float x2, float y2, float *pendiente, float *ordenada)				
Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?
void	punto	x1, y1, x2, y2	pendiente, ordenada	Procedimiento
e) void estadisticas(int datos[100], int num, float *promedio, int *max, int *min)				
Tipo de dato que regresa	Identificador de la subrutina	Identificadores de parámetros por valor	Identificadores de parámetros por referencia	¿Es una función o un procedimiento?
void	estadisticas	num	datos[100], promedio, max, min	Procedimiento

UNIDAD 2
Respuestas a los ejercicios

Ejercicio 2.1
Indica cuál es la población y la muestra para cada problema.

a) ¿Cuál es la muestra y la población?
Población: Todos los alumnos de la Escuela Nacional Preparatoria que cursan sexto año.
Muestra: Conjunto de 5000 alumnos de sexto año con los que se realizó el estudio

b) ¿Cuál es la muestra y la población?
Población: Grupo de pacientes de diabetes tipo 1
.Muestra: Conjunto de 1500 personas con diabetes tipo 1 (personas sometidas al estudio).

Ejercicio 2.2

Indica si el dato es cuantitativo o cualitativo marcando con una X la respuesta correcta.

Enunciado	Dato Cuantitativo	Dato Cualitativo

El número de mis seguidores en Twitter.	X	
El color de cabello de mi mejor amigo.		X
La edad de mis padres.	X	
El puesto que conseguiste en una carrera de atletismo.		X
Mi grupo favorito de música.		X

Datos cuantitativos discretos o continuos y cualitativos ordinal o nominal.

Ejercicio 2.3

Indica si el dato es cuantitativo discreto, cuantitativo continuo, cualitativo ordinal, cualitativo nominal marcando con una X la respuesta correcta.

Enunciado	Cuantitativo discreto	Cuantitativo continuo	Cualitativo ordinal	Cualitativo nominal
La clase favorita de los alumnos del grupo de sexto año.				X
Los pares de zapatos que guardo en mi closet.	X			
Estatura de los niños y niñas de una escuela primaria.		X		
Cantidad de clientes atendidos en una tienda durante un día.	X			
Lugar que ocupa un alumno en la olimpiada del conocimiento.			X	
Volumen de agua dentro de tinaco.		X		
El peso de los animales de una granja.		X		

El número de pétalos que tiene una flor.	X			
Color de ojos de los alumnos de quinto año de la ENP.				X
Tiempo que dura una llamada en un call center.		X		

Ejercicio 2.4

Cuando llegas a comprar zapatos, lo primero que te pregunta el vendedor es cuáles zapatos, de qué talla y color, marca, modelo, etc. Registra tu respuesta en la siguiente tabla según consideres que sean datos cualitativos o cuantitativos.

Datos	
Cuantitativos	Cualitativos
Tallas	Colores
Cantidades	Gustos
Peso	Tamaño en categorías

Ejercicio 2.5

Completa con las palabras los siguientes párrafos

Los datos cualitativos se pueden clasificar en dos tipos los cuales son: **ordinales** y **nominales**, los primeros no tienen una característica implícita en su valor, como puede ser lugar de nacimiento, color favorito, música favorita, etc., mientras con los otros tienen valores, ya sea de orden **binominal** o tienen dos opciones de respuesta, por ejemplo, estado civil (soltero, casado, viudo, divorciado u otro) o bien fuma sí o no, género hombre o mujer.

Mientras que los datos cuantitativos se pueden dividir en **discretos** y **continuos**, siendo los primeros valores que no tienen número decimal por ejemplo número de amigos que se tienen, número de hijos. Y el segundo son valores que pueden tener un número decimal dentro de sus mediciones, por ejemplo, estatura, promedio, etc.

Ejercicio 2.6

a) Capítulos de series vistos por día. Se aplicó a un grupo de 10 estudiantes de 6to año de bachillerato un cuestionario, del que se obtuvo el número de capítulos que ven en un día de la serie televisiva de moda. Los capítulos tienen una duración de media hora. Los

resultados obtenidos se muestran en la siguiente tabla. Ordena la muestra y obtén las medidas solicitadas.

Obtén la media, mediana y moda de los datos propuestos.

Muestra	5	3	4	3	5	2	6	4	4	2
Muestra ordenada	2	2	3	3	**4**	**4**	4	5	5	6
Media: **3.8** (la suma de los elementos, entre el número de datos)										
Mediana: **4** (al ser la muestra impar se toman los 3 centrales, y se dividen)										
Moda: el número **4** es el que más se repite(3 veces)										

b) Número de libros leídos al mes. Diez de tus compañeros de clase, desarrollaron una tabla en la que listan el número de libros que han leído durante el mes (incluyendo los que les dejaron de tarea). Ordena la muestra y obtén las medidas solicitadas.

Muestra	6	4	7	5	6	4	1	6	3	2
Muestra ordenada	1	2	3	4	**4**	**5**	6	6	6	7
Media: **4.4** (la suma de los elementos, entre el número de datos)										
Mediana: (4+5)/2= **4.5**										
Moda: el número 6 es el que más se repite (3 veces)										

c) Durante el ciclo escolar se le solicitó al grupo de 6to que realizará varias visitas académicas a museos dentro de la ciudad. Estas fueron el número de visitas realizadas. Obtén las medidas solicitadas.

Muestra	10	12	10	8	9	12	10	6	5	9
Muestra ordenada	5	6	8	9	**9**	**10**	10	10	12	12
Media: **9.1** (la suma de los elementos, entre el número de datos)										

Mediana: (9+10)/2= **9.5**
Moda: el número 10 es el que más se repite (3 veces)

Ejercicio 2.7

a) Indaga dos precios de computadoras de las siguientes marcas Dell, Azuz, Hp, Lenovo, Acer. y realiza la tabla de frecuencia de cuantas se han vendido el fin de semana en un establecimiento dedicado a su venta.

Marcas	fr ad	fr	fa ac	fr ac	%
Dell	8	0.0344828	8	0.03448276	0.08
	18	0.0775862	26	0.11206897	0.18
Azuz	20	0.0862069	46	0.19827586	0.2
	18	0.0775862	64	0.27586207	0.18
Acer	40	0.1724138	104	0.44827586	0.4
	32	0.137931	136	0.5862069	0.32
Hp	39	0.1681035	175	0.75431034	0.39
	12	0.0521724	187	0.80603448	0.12
Lenovo	45	0.1939655	232	1	0.45
	232	1			

Tabla 7. Tabla de frecuencias de computadoras vendidas

b) Se están haciendo encuestas de la frecuencia de visitas a las cafeterías para conocer su popularidad. Se obtuvieron los siguientes datos.

Edad (años)		fr ab	fa ac	fr ac	fa ac	%
límite inferior	límite superior					
10	12	40	0.22727273	40	0.22727273	0.4
13	15	17	0.09659091	57	0.32386364	0.17
16	18	40	0.22727273	97	0.55113636	0.4
19	23	17	0.09659091	114	0.64772727	0.17
24	26	38	0.21590909	152	0.86363636	0.38
27	30	24	0.13636364	176	1	0.24
		176	1			

Ejercicio 2.8

a)

CLASE	FRECUENCIA ACUMULADA
A	23
B	42
C	64
D	85

b)

Clase	Límite Inferior Exacto	Límite Superior Exacto	Frecuencia Relativa	Frecuencia Acumulada	Frecuencia Complementaria
1	$ 360.0	$ 399.0	258	258	1594
2	$ 400.0	$ 439.0	261	519	1333
3	$ 440.0	$ 479.0	162	681	1171
4	$ 480.0	$ 519.0	301	982	870
5	$ 520.0	$ 559.0	321	1303	549
6	$ 560.0	$599.0	341	1644	208
7	$ 600.0	$599.0	208	1852	0

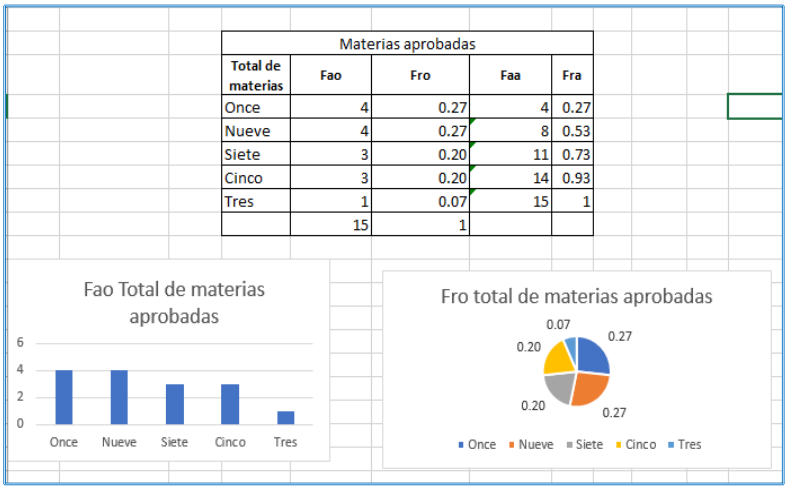
Ejercicio 2.9

a) Moda=64 **Media=50** **Mediana=49.28**

b) Moda=blanco

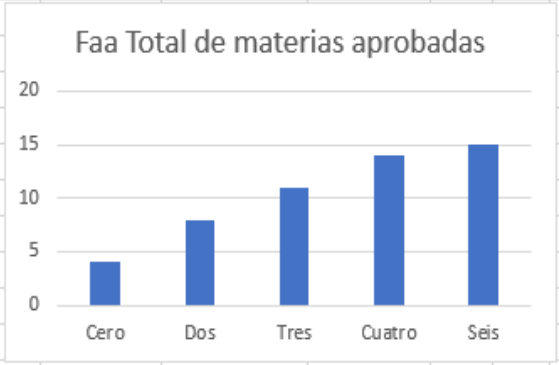
Ejercicio 2.10

a)



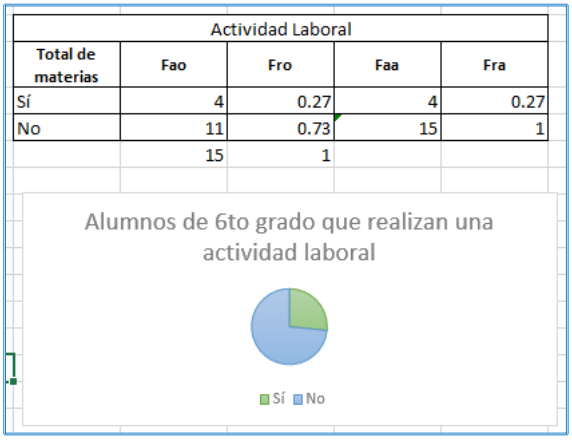
b)

Horas dedicada a una actividad física				
Total de materias	Fao	Fro	Faa	Fra
Cero	4	0.27	4	0.27
Dos	4	0.27	8	0.53
Tres	3	0.20	11	0.73
Cuatro	3	0.20	14	0.93
Seis	1	0.07	15	1
	15	1		



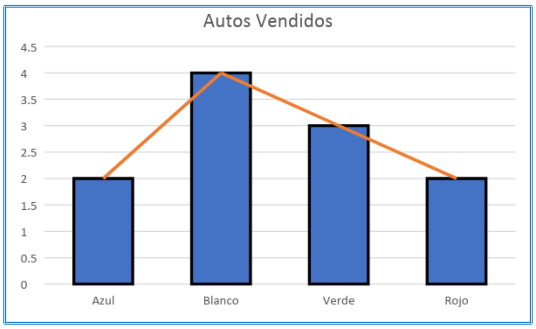
c)

Actividad Laboral				
Total de materias	Fao	Fro	Faa	Fra
Sí	4	0.27	4	0.27
No	11	0.73	15	1
	15	1		

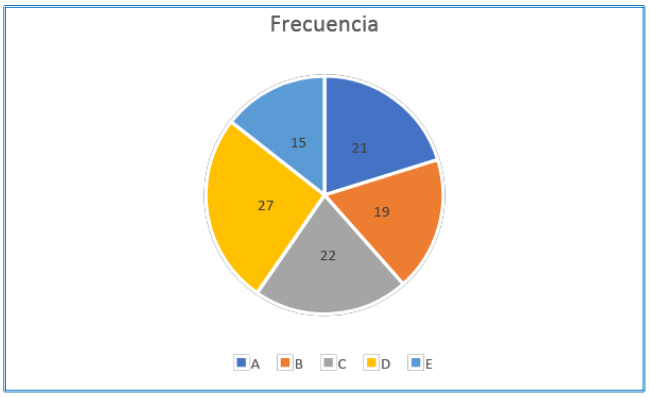


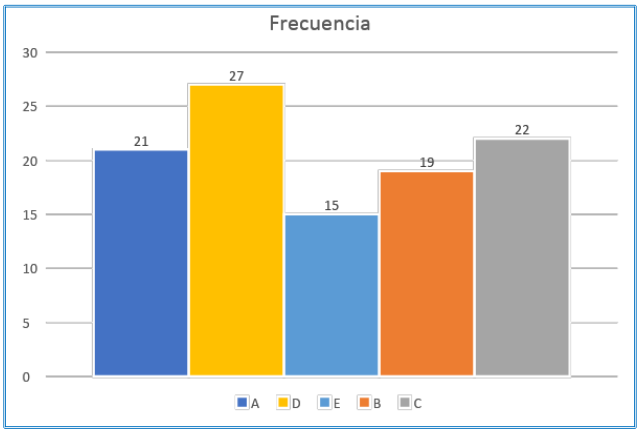
Ejercicio 2.11

a)



b)





Formulario

FÓRMULAS DE ESTADÍSTICA DESCRIPTIVA PARA CUADO SE TIENEN TODOS LOS DATOS DE UNA POBLACIÓN (CENSO)
N = tamaño de la población

PARA DATOS NO AGRUPADOS

Medidas de tendencia central

MEDIA $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

MEDIANA $\tilde{\mu} = \begin{cases} x_{\left(\frac{N+1}{2}\right)} & ;\text{si N es impar} \\ \frac{x_{\left(\frac{N}{2}\right)} + x_{\left(\frac{N}{2}+1\right)}}{2} & ;\text{si N es par} \end{cases}$

MODA **Mo** = Dato que más se repite (puede haber una o más modas)

Medidas de Dispersión

RANGO: $x_{(max)} - x_{(min)}$

VARIANZA $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 = \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right) - \mu^2$

DESV. EST. $\sigma = \sqrt{\sigma^2}$ COEF. DE VARIACION $CV = \frac{\sigma}{\mu}$

Medidas de forma

COEFICIENTE DE SESGO (Tercer momento estandarizado) $\alpha_3 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^3}{\sigma^3}$

COEFICIENTE DE CURTOSIS (Cuarto momento estandarizado) $\alpha_4 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^4}{\sigma^4}$

PARA DATOS AGRUPADOS

Medidas de tendencia central

MEDIA $\mu = \frac{1}{N}\sum_{i=1}^{m} f_i x_i$

donde m=número de clases

MEDIANA $\tilde{\mu} = L_{i\,inf} + \left[\frac{\frac{N}{2} - F_{i-1}}{f_i}\right] c_i$

donde: c_i = longitud de la clase que contiene a la mediana
$L_{i\,inf}$ = límite inferior de la clase que contiene a la mediana

MODA $Mo = L_{Moinf} + \left[\frac{a}{a+b}\right] c_{Mo}$

donde: $a = f_{Mo} - f_{Mo-1}$, $b = f_{Mo} - f_{Mo+1}$
f_{Mo} = frecuencia absoluta de la clase que contiene a la moda
c_{Mo} = longitud de la clase que contiene a la moda
$L_{Mo\,inf}$ = límite inferior de la clase que contiene a la moda

Medidas de Dispersión

RANGO= Lím. Sup. de la última clase - Lím. Inf. de la primera clase

VARIANZA $\sigma^2 = \frac{1}{N}\sum_{i=1}^{m} f_i(x_i-\mu)^2 = \frac{1}{N}\sum_{i=1}^{m} f_i(x_i^2) - \mu^2$

DESV. EST. $\sigma = \sqrt{\sigma^2}$ COEF. DE VARIACIÓN $CV = \frac{\sigma}{\mu}$

Medidas de forma

COEFICIENTE DE SESGO (Tercer momento estandarizado) $\alpha_3 = \frac{\frac{1}{N}\sum_{i=1}^{m} f_i(x_i-\mu)^3}{\sigma^3}$

COEFICIENTE DE CURTOSIS (Cuarto momento estandarizado) $\alpha_4 = \frac{\frac{1}{N}\sum_{i=1}^{m} f_i(x_i-\mu)^4}{\sigma^4}$

Fractiles (ó cuantiles) $FRACTIL = L_{inf} + \left[\frac{Na - F_{i-1}}{f_i}\right] c_i$

donde: L_{inf} = límite inferior de la clase que contiene al fracti
a = Fraccion de interes

por ejemplo, para:

Cuartiles: Q_1, Q_2, Q_3 : a = 1/4, 1/2, 3/4

Deciles: $D_1, D_2, \ldots D_8, D_9$: a = 1/10, 2/10, ..., 8/10, 9/10

Centiles o Percentiles: $C_1, \ldots C_{35}, \ldots C_{99}$: a = 1/100, ..., 35/100, ..., 99/100

Rango intercuartil = Q_3-Q_1

Nota: L se refiere siempre a las fronteras de clase.

FÓRMULAS DE ESTADÍSTICA DESCRIPTIVA PARA CUANDO SE TIENEN LOS DATOS DE UNA MUESTRA
n = tamaño de la muestra

PARA DATOS NO AGRUPADOS

Medidas de tendencia central

MEDIA $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$

MEDIANA $\tilde{X} = \begin{cases} x_{\left(\frac{n+1}{2}\right)} & ;\text{si n es impar} \\ \frac{x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}}{2} & ;\text{si n es par} \end{cases}$

MODA **Mo** = Dato que más se repite (puede haber una o más modas)

Medidas de Dispersión

RANGO: $x_{(max)} - x_{(min)}$

VARIANZA $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{X})^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\overline{X}^2\right)$

DESV. EST. $S = \sqrt{S^2}$ COEF. DE VARIACION $CV = \frac{S}{\overline{X}}$

Medidas de forma

COEFICIENTE DE SESGO (Tercer momento estandarizado) $a_3 = \frac{n}{(n-1)(n-2)}\frac{\sum_{i=1}^{n}(x_i-\overline{X})^3}{s^3}$

COEFICIENTE DE CURTOSIS (Cuarto momento estandarizado) $a_4 = \frac{n(n+1)}{(n-1)(n-2)(n-3)}\frac{\sum_{i=1}^{n}(x_i-\overline{X})^4}{S^4}$

PARA DATOS AGRUPADOS

Medidas de tendencia central

MEDIA $\overline{X} = \frac{1}{n}\sum_{i=1}^{m} f_i x_i$

donde m = no. de clases

MEDIANA $\tilde{X} = L_{i\,inf} + \left[\frac{\frac{n}{2} - F_{i-1}}{f_i}\right] c_i$

donde: c_i = longitud de la clase que contiene a la mediana
$L_{i\,inf}$ = límite inferior de la clase que contiene a la mediana

MODA $Mo = L_{Moinf} + \left[\frac{a}{a+b}\right] c_{Mo}$

donde: $a = f_{Mo} - f_{Mo-1}$, $b = f_{Mo} - f_{Mo+1}$
f_{Mo} = frecuencia absoluta de la clase que contiene a la moda
c_{Mo} = longitud de la clase que contiene a la moda
$L_{Mo\,inf}$ = límite inferior de la clase que contiene a la moda

Medidas de Dispersión

RANGO= Lím. Sup. de la última clase - Lím. Inf. de la primera clase

VARIANZA $S^2 = \frac{1}{n-1}\sum_{i=1}^{m} f_i(x_i-\overline{X})^2 = \frac{1}{n-1}\left[\sum_{i=1}^{m} f_i(x_i)^2 - n\overline{X}^2\right]$

DESV. EST. $S = \sqrt{S^2}$ COEF. DE VARIACIÓN $CV = \frac{S}{\overline{X}}$

Medidas de forma

COEFICIENTE DE SESGO (Tercer momento estandarizado) $a_3 = \frac{n}{(n-1)(n-2)}\frac{\sum_{i=1}^{m} f_i(x_i-\overline{X})^3}{S^3}$

COEFICIENTE DE CURTOSIS (Cuarto momento estandarizado) $a_4 = \frac{n(n+1)}{(n-1)(n-2)(n-3)}\frac{\sum_{i=1}^{m} f_i(x_i-\overline{X})^4}{S^4}$

Fractiles (ó cuantiles) $FRACTIL = L_{inf} + \left[\frac{na - F_{i-1}}{f_i}\right] c_i$

donde: L_{inf} = límite inferior de la clase que contiene al fracti
a = Fracción de interés

por ejemplo, para:

Cuartiles: Q_1, Q_2, Q_3 : a = 1/4, 1/2, 3/4

Deciles: $D_1, D_2, \ldots D_8, D_9$: a = 1/10, 2/10, ..., 8/10, 9/10

Centiles o Percentiles: $C_1, \ldots C_{35}, \ldots C_{99}$: a = 1/100, ..., 35/100, ..., 99/100

Rango intercuartil = Q_3-Q_1

Nota: L se refiere siempre a las fronteras de clase.

Formulario. Estadística-descriptiva (Patricia Valdez & Antonio Gómez, 2019). Recuperado de: https://goo.gl/VEUCzV

Respuestas de la autoevaluación

1) Relacione las columnas: en el paréntesis coloque la respuesta correcta de la columna derecha.

1.1 Longitud de 150 tornillos producidos en una fábrica	(c)	a) Cuantitativa discreta
1.2 Bebida favorita de alumnos de la ENP	(d)	b) Cualitativa ordinal
1.3 Likes que tiene un video en youtube	(a)	c) Cuantitativa continua
1.4. Talla (s,m,l,xl) de playeras del equipo de basquetbol	(b)	d) Cualitativa nominal

Subraye la respuesta correcta.

2) El conjunto completo de todos los elementos que se someten a estudio se conoce como:

a) Muestra b) Moda

c) Datos **d) Población**

3) Al subconjunto de miembros seleccionados que se someten a un estudio estadístico se le conoce como:

a) Muestra b) Moda

c) Datos d) Población

4) Los datos numéricos que puede tomar una cantidad incontable de valores entre 2 números se denominan como:

a) Ordinales **b) Continuos**
c) Discretos d) Nominales

5) El valor estadístico de promedio es llamado también
b) media b) mediana c) moda d) varianza

6) Nos indica que tan bien son tomados los datos que estamos investigando en la muestra obteniendo a partir de esta el error de las mediciones.
b) variabilidad b) varianza c) correlación d) normalidad

7) Es m_3 es un factor que nos indica que tan simétrica son los datos

8) La media es un valor estadístico el cual se utiliza en varias medidas como en la varianza, coeficiente de correlación, desviación estándar, etc.

9) La tabla de frecuencias se obtiene:

a) dando valores arbitrarios de la muestra
b) por medio de la media de los valores de la muestra
c) por medio de la varianza de los valores de la muestra
d) números obtenidos de la muestra

10) Los valores de la frecuencia relativa se obtiene con el **valor** de la **frecuencia absoluta** entre total de **frecuencia absoluta**

11) El porcentaje frecuencia absoluta es:
a) dividir el valor de la frecuencia relativa entre la suma de la frecuencia absoluta
b) multiplicar el valor de la frecuencia relativa por 100
c) dividir el valor de la frecuencia relativa entre 100
d) dividir el valor de la frecuencia absoluta entre 100

12) Diferencia que existe entre la frecuencia relativa y la absoluta es que la primera tiene **rangos** y la segunda **sus valores no son agrupados**

13) En una investigación de mercado, es necesario conocer qué sabor de producto se vende con más frecuencia durante la temporada navideña. ¿Qué medida de tendencia central utilizarías?
Moda

14) Durante el trayecto diario a la escuela tú y tus compañeros utilizan diferentes tipos de transporte público. Los datos son los siguientes 1,1, 2, 2, 2, 2, 2,3 3, 3. Si tomas los valores centrales de esta tabla para saber qué tipo de transporte se utiliza más ¿Qué medida de tendencia central estás utilizando?
Mediana

15) Moda = 5, Mediana = 5 y Media = 4.8

16)
a)

Clase	Frecuencia	Frecuencia Acumulada	Frecuencia Complementaria
0-5	1	1	39
6-10	1	2	38
11-15	3	5	35
16-20	3	8	32
21-25	3	11	29
26-30	6	17	23
31-35	7	24	16
36-40	10	34	6
41-45	4	38	2
46-50	2	40	0
	40		

17)

b)

Clase	Frecuencia	Frecuencia Acumulada
38 – 44	7	7
44 – 50	8	15
50 – 56	15	30
56 – 62	25	55
62 – 68	18	73
68 – 74	9	82
74 – 80	6	88

18)

a) 432 kg

19)

d) 36

UNIDAD 3
Respuestas a los ejercicios

Ejercicio 3.1

Pregunta	Respuesta
1	Automatización
2	Sensores, controladores y actuadores
3	Emisor y receptor
4	Actuador

Ejercicio 3.2

Pregunta	Respuesta
a)	Sensores, controladores y actuadores
b)	Sensor óptico
c)	Indicador led
d)	Software (lenguaje de programación, simulador)

Ejercicio 3.4

a) La adquisición de datos
b) Analógica
c) Sensor
d) Software de aplicación

Ejercicio 3.5

Pregunta	Respuesta
1	c) Sensor, DAQ, ADC, computadora
2	d) Electrodo
3	c) Procesar, almacenar, analizar información

Ejercicio 3.6

Pregunta	Respuesta
1	a) Sensores de temperatura, una tarjeta de adquisición de datos, una computadora
2	b) Usando un programa computable que adquiera y analice los datos

Ejercicio 3.15

Todo comenzó un día por la mañana en la clase de Informática Aplicada a la Ciencia y la Industria, cuando el profesor, nos comentó en clase, que deberíamos de llevar a cabo un proyecto de un sistema automatizado para medir la temperatura y formar equipos de trabajo, para realizar dicho proyecto.

Nos reunimos en equipo al finalizar la clase y analizamos el proyecto y los materiales que íbamos a utilizar para llevarlo a cabo como son: un sensor de temperatura, un led, una resistencia, una tarjeta de prueba (protoboard) y una tarjeta desarrolladora.

Una vez que realizamos el circuito y programamos el código para que funcionara nuestro proyecto sobre un sistema automatizado de temperatura, se lo compartimos a nuestro profesor. Realizamos dos pruebas, la primera prueba consistió en pasar una paleta de hielo por el sensor y el led encendió en color azul y en la segunda prueba pasamos la llama de una vela y el led encendió de color rojo.

Ejercicio 3.19

a)

Inicio

Entero X

SI (X >=38 °C) **ENTONCES**

Encender LED color “Rojo”
ESCRIBIR “Tienes que regar”
SI NO

Encender LED color “Azul”
ESCRIBIR ““No debes regar”

Fin SI

Fin

b)

Inicio

Real temp

REPETIR

ESCRIBIR la temperatura de la ciudad es: “temp”

HASTA (temp=40°C)

Fin

Respuestas de la autoevaluación

Pregunta	Respuesta
1	b)
2	a)
3	d)
4	a)
5	c)
6	c)
7	a)
8	b)
9	a)

REFERENCIAS BIBLIOGRÁFICAS

Unidad 1

Aho, V., Hopcroft, J. y Ullman, J. (1998). *Estructuras de datos y algoritmos.* México: Pearson.

Cairó, O. (2005). *Metodología de la programación.* México: Alfaomega.

CLion. (2018). *Referencia de C*. Recuperado el 19 de diciembre, de https://es.cppreference.com/w/cpp/language/escape

Euán, J. y Cordero, L. (1991). *Estructuras de datos*. México: Noriega Limusa.

Gobierno del Estado de México (2018). Calendario_HoyNoCircula [imagen]. Recuperado de: http://sma.edomex.gob.mx/sites/sma.edomex.gob.mx/files/images/TramitesServicios/VerificacioVehicular/Calendario_HoyNoCircula.jpg

IBM Knowledge Center. (2017). *Conceptos Clave: entidad, atributo y tipo de entidad.* Recuperado el 28 de Noviembre de 2018, de: https://www.ibm.com/support/knowledgecenter/es/SSWSR9_11.6.0/com.ibm.mdmhs.overview.doc/entityconcepts.html

Joyanes, L. y Zahonero, I. (2010). *Programación C, C++, Java y UML*. España: Mc GrawHill.

Joyanes, L., Fernández, M., Sánchez, L. y Zahonero, I. (2005). *Estructuras de datos en C.* México: Mc GrawHill.

Kernighan, B. y Ritchie, D. (1985). *El lenguaje de Programación C*. México: Prentice Hall.

Mendoza, J. (2018). *Ícono de Dev C++* [imagen]. Recuperado de: https://icon-icons.com/es/icono/dev/22513

Olivares, L. (2008). *Manual de Programación en Lenguaje C++*. México: DGSCA, UNAM. Recuperado el 7 de diciembre de 2018, de https://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf

Pes, C. (2018). *Secuencias de Escape en la Función printf del Lenguaje C.* Recuperado el 17 de diciembre de 2018, de http://www.carlospes.com/curso_de_lenguaje_c/01_09_01_03_secuencias_de_escape.php

Universidad III de Madrid. (s/f). *Funciones E/S para tipos de datos*. Recuperado el 25 de noviembre de 2018, de http://www.it.uc3m.es/pbasanta/asng/course_notes/input_output_function_scanf_es.html

Verificación Vehicular (2018). *Holograma de verificación* [imagen]. Recuperada de: https://www.verificacion-vehicular.com.mx/images/hologramas.png

Wikilibros. (2018). *Programación en C++*. Recuperado el 11 de diciembre de 2018, de https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B

Wirth, N. (1980). *Algoritmos + Estructuras de datos = Programas*. España: Ediciones del Castillo.

Unidad 2

Alonso, M. y Flores, J. (2004). *Estadística descriptiva para el bachillerato*. Facultad de Ciencias. México: UNAM

Altman, D.G. (1991). *Practical Statistics for Medical Research.* London: Chapman and Hall.

Armitage, P., y Berry, G. (1994). *Statistical Methods in Medical Research*. (3rd ed.) Oxford: Blackwell Scientific Publications.

Buján, A. (2017). Coeficiente de correlación [imagen]. Recuperada de: https://www.enciclopediafinanciera.com/images/coeficiente-de-correlacion.jpg

Carrasco, J.C. y Hernán, M.A. (1993). *Estadística multivariante en las ciencias de la vida.* Madrid: Editorial Ciencia 3, S.L.

Camacho Martínez Vara de Rey C. (s.f.) *Coefientes de correlación de Pearson* [Apuntes] recuperado en noviembre de 2018 de: http://personal.us.es/vararey/adatos2/correlacion.pdf

Disfruta las matemáticas. (2011). *Rango* [imagen]. Recuperada de: http://www.disfrutalasmatematicas.com/definiciones/rango-estadistica-.html

Glosario de términos estadísticos. (2018). *Distribuciones* [imagen]. Recuperada de: https://glosarios.servidor-alicante.com/terminos-estadistica/coeficiente-de-curtosis

Johnson, R. (1990). *Estadística Elemental.* México: Trillas.

Johnson, R. (2012) *Estadística elemental*. México: Cengage Learning.

López, A. (s/f). *Curva normal* [imagen]. Recuperada de: https://www.researchgate.net/figure/Figura-16-Curva-normal_fig3_256278076

Martínez, A. (2015). *Tipos de asimetrías de un conjunto de datos* [imagen]. Recuperada de: https://upload.wikimedia.org/wikipedia/commons/c/ca/Posiciones_relativas_de_par%C3%A1metros_centrales.svg?download

Triola, F. M. (2009). *Estadística.* México: Pearson Education.

Orellana, L. (2001). *Estadística Descriptiva*. [Apuntes] Universidad de Buenos Aires. Recuperado en octubre 2018 de: http://www.dm.uba.ar/materias/estadistica_Q/2011/1/modulo%20descriptiva.pdf

Portal Educativo (s.f.) *Tablas de frecuencia*. Recuperado en noviembre 2018 de: https://www.portaleducativo.net/octavo-basico/791/Tablas-de-frecuencias-con-datos-agrupados#

Revista Biomédica. (2011). *Medidas de tendencia central y dispersión*. Recuperado el 21 de febrero, de: https://www.medwave.cl/link.cgi/Medwave/Series/MBE04/4934

Sánchez Corona, O. (1996) *Probabilidad y Estadística.* México: McGraw Hill

Universitat de Valencia. (s.f.). *Estadística Inferencial.* Recuperado de: https://www.uv.es/webgid/Inferencial/1_introduccin1.html

Universidad Autónoma del Estado de México. (s.f.) *Medidas de tendencia central para datos no agrupados*. [Apuntes] Recuperado en noviembre 2018 de: http://www.seduca2.uaemex.mx/ckfinder/uploads/files/2-1__medidas_de_tend.pdf

Universo Fórmulas. (s/f). *Curtosis* [imagen]. Recuperada de: https://www.universoformulas.com/estadistica/descriptiva/curtosis/

Valdez I. y Gómez M. (2017). *Aprende Probabilidad y Estadística*. Recuperado el 21 de febrero de 2019, de http://aprendeprobayestadistica.blogspot.com/p/creditos-y-citas.html

Vicente Villardón, J.L. (2010). *Introducción al análisis de la varianza* Recuperada en noviembre 2018 de: http://biplot.usal.es/problemas/libro/7%20ANOVA.pdf

Unidad 3

Arduino (2018). *What is Arduino Create?* Recuperado de: https://create.arduino.cc/

Barrientos, A., y Gambao, E. (2014). *Sistemas de producción automatizados*. Madrid : Dextra Editorial S.L

Boix, O., Córcoles, F., Sainz, L. y Suelves, F. J. (2009). *Tecnología eléctrica* (2a. ed.). Barcelona: Cano Pina - CEYSA

Carbonell, A., Fernández, C., López, L., Loureiro, M., Poyatos, A., y Ramírez, L. et al. (2018). *Robótica.* Recuperado de: http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena11/index_4quincena11.htm

Caska, J. (2018). VBB - *The rocking embedded development environment.* Recuperado de: http://www.virtualbreadboard.com/

Casa Vilaseca, M. y Rodríguez Arenas, A. (2015). *Instalaciones Domóticas* Barcelona: Marcombo.

Código facilito. (2018). *¿Qué es Arduino?* [Video]. Recuperado de https://www.youtube.com/watch?v=Kgz0vD1vSxY

Código facilito (04/06/13). *Curso Arduino 7: Temperatura* [Video]. Recuperado de https://www.youtube.com/watch?v=QEIZjCVI2NQ

Duran Moyano, J., Martínez García, H., y Gámiz Caro, J. (2012). *Automatismos eléctricos e industriales*. Barcelona: Marcombo.

Editronik (2015). *Curso de electrónica básica desde cero* [Video]. Consultado de https://youtu.be/LjYClvMPRdE

Enríquez Harper, G. (2011). *El ABC de la instrumentación en el control de procesos industriales.* México: Limusa.

Fritzing. (2018). Electronic made easy. Recuperado de: http://fritzing.org/home/

García, A. E. (2015). *Robots*. Recuperado de https://ebookcentral.proquest.com

Hair, J., y Gómez Suárez, M. (2010). *Análisis multivariante.* Madrid: Prentice-Hall.

Henríquez, D. (2017). *Una Reflexión sobre la Automatización de la Producción y los Cambios en el Ámbito Laboral.* [En línea] Iberoamérica Divulga Sección Divulgación Científica. Recuperado de: https://www.oei.es/historico/divulgacioncientifica/?Una-Reflexion-sobre-la-Automatizacion-de-la-Produccion-y-los-Cambios-en-el

KiCad EDA. (2018). *A Cross Platform and Open Source Electronics Design Automation Suite*. Recuperado de http://kicad-pcb.org/

Lind, D. A., Marchal, W. G, y Wathen, S. A. (2015). *Estadística aplicada a los negocios y la economía*. México: McGraw-Hill Interamericana

Martínez, J. (2018). *Componentes electrónicos activos* [Video]. Recuperado de https://www.youtube.com/watch?v=WjKiq2ud958

Martínez, J. (2018). *Componentes electrónicos pasivos* [Video]. Recuperado de https://www.youtube.com/watch?v=JIjsl1Wdrb0

Mora, H. (2011) *Sistemas de adquisición y Procesamiento de datos.* [Apunte] Recuperado de https://rua.ua.es/dspace/handle/10045/19119

National Instruments. (2018). *¿Qué es un sensor?* Recuperado de: https://www.ni.com/data-acquisition/what-is/esa/

National Instruments. (2018). *What Is Data Acquisition?* Recuperado de: https://www.ni.com/data-acquisition/what-is/esa/

Prometec (2018). *Sensor de temperatura y ventilador*. Recuperado de: https://www.prometec.net/regulacion-simple/

Quispe, O. (2018). *Tarjetas Para Desarrollo De Hardware*. Recuperado de: http://www.lightpath.io/tarjetas-de-desarrollo/

Schmitt, N., & Farwell, R. (1988). *Robótica y sistemas automáticos*. Madrid: Anaya Multimedia.

Tamayo, J. (2018). *Diseño e implementación de controladores digitales a través de un sistema de adquisición de datos.* Scientia et Technica, Vol. 21. Num 3. Recuperado de: http://revistas.utp.edu.co/index.php/revistaciencia/article/view/10421/9551

Tinkercad (2018). *From mind to design in minutes*. Recuperado de https://www.tinkercad.com

Universidad Nacional Autónoma de México. (2018). *Arduino y algunas aplicaciones* [Video]. Recuperado de: https://es.coursera.org/lecture/arduino-aplicaciones/de-los-microcontroladores-a-la-tarjeta-arduino-LOTX7

Vacas Martínez, J. A. (12/11/17). *Aprende a usar Arduino desde 0: Simuladores* [Video]. Recuperado de: https://youtu.be/VoWSmP5UpmI

UNAM

Dr. Enrique Graue Wiechers
Rector

Dr. Leonardo Lomelí Vanegas
Secretario General

Dr. Alberto Ken Oyama Nakagawa
Secretario de Desarrollo Institucional

Ing. Leopoldo Silva González
Secretario Administrativo

Dra. Mónica González Contró
Abogada General

Universidad Nacional Autónoma de México

DGENP

Biól. María Dolores Valle Martínez
Directora General

Lic. Jaime Cortés Vite
Secretario General

M. en C. María Josefina Segura Gortares
Secretaria Académica

Lic. José Luis Sánchez Varela
Secretario Administrativo

M. en C. Ana Laura Gallegos y Téllez Rojo
Secretaria de Planeación

Q.F.B. Roberta Ma. del Refugio Orozco Hernández
Secretaria de Difusión Cultural

Mtra. Marcela Cuapio Campos
Jefa del Departamento de Informática

Directores de Planteles

Lic. Enrique Espinosa Terán
Plantel 1 "Gabino Barreda"

Lic. Isabel Jiménez Téllez
Plantel 2 "Erasmo Castellanos Quinto"

Lic. Samuel David Zepeda Landa
Plantel 3 "Justo Sierra"

Mtro. Eduardo Adolfo Delgadillo Cárdenas
Plantel 4 "Vidal Castañeda y Nájera"

Mtra. Velia Carrillo García
Plantel 5 "José Vasconcelos"

Mtro. Isauro Figueroa Rodríguez
Plantel 6 "Antonio Caso"

I.Q. María del Carmen Rodríguez Quilantán
Plantel 7 "Ezequiel A. Chávez"

Arq. Ángel Huitrón Bernal
Plantel 8 "Miguel E. Schulz"

Q.F.B. Gabriela Martínez Miranda
Plantel 9 "Pedro de Alba"