Programando en C
Desde la práctica:
Problemas resueltos

José Enrique Guerra Salazar
Marco Vinicio Ramos Valencia
Geovanny Estuardo Vallejo Vallejo

PUERTO MADERO
EDITORIAL

1era Edición
2023

puertomaderoeditorial.com.ar
La Plata - Argentina

Programando en C desde la práctica: Problemas resueltos

José Enrique Guerra Salazar, Marco Vinicio Ramos Valencia
Geovanny Estuardo Vallejo Vallejo
ISBN: 978-987-82912-8-4

Programando en C desde la práctica: Problemas resueltos

Programming in C from Practice:
Solved Problems

Autores:
José Enrique Guerra Salazar
Marco Vinicio Ramos Valencia
Geovanny Estuardo Vallejo Vallejo

AUTORES:

José Enrique Guerra Salazar

Escuela Superior Politécnica de Chimborazo. Facultad de Informática y Electrónica. Panamericana Sur Km 1 1/2, EC060155, Riobamba, Chimborazo. Ecuador

j_guerra@espoch.edu.ec

https://orcid.org/0000-0003-2535-7682

Marco Vinicio Ramos Valencia

Escuela Superior Politécnica de Chimborazo. Facultad de Informática y Electrónica. Panamericana Sur Km 1 1/2, EC060155, Riobamba, Chimborazo. Ecuador

vi_ramos@espoch.edu.ec

https://orcid.org/0000-0003-3033-2404

Geovanny Estuardo Vallejo Vallejo

Escuela Superior Politécnica de Chimborazo. Facultad de Informática y Electrónica. Panamericana Sur Km 1 1/2, EC060155, Riobamba, Chimborazo. Ecuador

g_vallejo@espoch.edu.ec

https://orcid.org/0009-0002-6506-9837

Guerra Salazar, José Enrique
Programando en C desde la práctica : problemas resueltos / José Enrique Guerra Salazar ; Marco Vinicio Ramos Valencia ; Geovanny Estuardo Vallejo Vallejo ; Editado por Juan Carlos Santillán Lima ; Fernando Tiverio Molina Granja. - 1a ed - La Plata : Puerto Madero Editorial Académica, 2023.
Libro digital, PDF/A

Archivo Digital: descarga y online
ISBN 978-987-82912-8-4

1. Lenguajes de Programación. I. Santillán Lima, Juan Carlos, ed. II. Molina Granja, Fernando Tiverio, ed. III. Título.
CDD 005.1

Primera Edición, Abril 2023

Programando en C desde la práctica:
Problemas resueltos
ISBN: 978-987-82912-8-4

Editado por:
Sello editorial: ©Puerto Madero Editorial Académica
Nº de Alta: 933832

Editorial: © Puerto Madero Editorial Académica
CUIL: 20630333971
Calle 45 N491 entre 4 y 5
Dirección de Publicaciones Científicas Puerto Madero Editorial Académica
La Plata, Buenos Aires, Argentina
Teléfono: +54 9 221 314 5902
+54 9 221 531 5142
Código Postal: AR1900

Este libro se sometió a arbitraje bajo el sistema de doble ciego (peer review)

Corrección y diseño:
Puerto Madero Editorial Académica
Diseñador Gráfico: José Luis Santillán Lima

Diseño, Montaje y Producción Editorial:
Puerto Madero Editorial Académica
Diseñador Gráfico: Santillán Lima, José Luis

Director del equipo editorial: Santillán Lima, Juan Carlos

Editor: Santillán Lima, Juan Carlos
Molina Granja, Fernando Tiverio

Hecho en Argentina
Made in Argentina

.

ÍNDICE

RESUMEN

En la actualidad, los procesos manuales se pueden simplificar y automatizar, ahorrando tiempo, dinero y esfuerzo, gracias al uso de los lenguajes de programación. La mayoría de los libros de texto se centran en el conocimiento teórico, semántica y sintaxis, del lenguaje de programación en C y muy pocos libros de texto contienen suficientes ejercicios resueltos y, lo que es peor, soluciones de programación. Esto puede afectar el desempeño de los estudiantes, ya que la falta de información suficiente que ayude a gestionar el conocimiento puede conducir a una programación inadecuada y temas erróneos, por lo que se considera un texto central con un conjunto importante de problemas de programación resueltos en C para ser desarrollado como una herramienta efectiva que complemente el proceso de enseñanza del lenguaje de programación.

Este libro está dirigido a todos los estudiantes de primer año de las diferentes carreras de ingeniería que enseñan los conceptos básicos de programación y que reciben una formación básica en esta área. Además, por el nivel del libro, se puede utilizar perfectamente en la educación secundaria o universitaria en los que se trabaje con el lenguaje C, muy utilizado en el mundo profesional. El propósito de este libro no es armar un conjunto de ejercicios al azar, sino dividir los ejercicios en funciones para diferentes aspectos del idioma, que sean adecuados para los estudiantes, de modo que en cada capítulo practiquen un cierto aspecto del Lenguaje C.

Palabras clave: Programación, lenguaje C, ejercicios resueltos, arreglos, funciones.

SUMARY

Currently, manual processes can be simplified and automated, saving time, money and effort, thanks to the use of programming languages. Most of the textbooks focus on the theoretical knowledge, semantics and syntax, of the C programming language and very few textbooks contain enough solved exercises and, what is worse, programming solutions. This can affect the performance of students, since the lack of sufficient information to help manage knowledge can lead to inappropriate programming and wrong topics, which is why it is considered a central text with an important set of programming problems solved in C to be developed as an effective tool that complements the teaching process of the programming language.

This book is intended for all first-year students of different engineering majors who are taught the basics of programming and who receive basic training in this area. In addition, due to the level of the book, it can be used perfectly in secondary or university education in which they work with the C language, widely used in the professional world. The purpose of this book is not to put together a set of exercises at random, but to break the exercises into functions for different aspects of the language, which are suitable for students, so that in each chapter they practice a certain aspect of Language C.

Keywords: Programming, language C, solved exercises, arrays , functions.

INTRODUCCIÓN

Hoy en día, los procesos manuales se pueden simplificar y automatizar, ahorrando tiempo, dinero y esfuerzo, gracias al uso de lenguajes de programación, que pueden ser técnicas de bajo, medio y alto nivel. Administrar un lenguaje de programación le permite codificar y resolver problemas específicos, lo que significa que los estudiantes aplican la lógica de programación y los conocimientos fundamentales necesarios para diferentes tipos de datos, estructuras de control, iteración, anidamiento, diseño, funciones y más.

Es muy importante indicar que existe una amplia gama de textos que tratan el tema de la programación en lenguaje C, pero en su gran mayoría enfocados al conocimiento teórico del mismo, siendo muy escasos los textos que contienen una cantidad considerable de programas resueltos y peor aún solucionarios sobre programación en este lenguaje. Esto afecta el rendimiento de los estudiantes porque la falta de información adecuada que les ayude a gestionar sus conocimientos conduce a una programación inadecuada y errores de las asignaturas, por lo que se ha considerado elaborar un texto básico que contenga un conjunto significativo de ejercicios resueltos sobre programación en C, para que sea un complemento válido al proceso de enseñanza aprendizaje del lenguaje.

El libro "*Programando en C desde la Práctica, ejercicios resueltos*" es adecuado para todos los estudiantes de primer año de ingeniería de diferentes carreras en donde se dicte la asignatura de Fundamentos de Programación que están recibiendo una formación básica en este aspecto. Además, por el nivel del libro, se puede utilizar perfectamente en la educación secundaria o universitaria en los que se trabaje con el lenguaje C, muy utilizado en el mundo profesional. El objetivo de este libro no es crear una colección aleatoria de ejercicios, sino dividirlos en funciones para diferentes aspectos del lenguaje, haciéndolo adecuado para que los estudiantes puedan practicar principalmente un aspecto particular del lenguaje en cada capítulo.

Los ejercicios resueltos que este texto presenta han sido agrupados y desarrollados de tal manera que concuerden con los siguientes temas: tipos de datos, identificadores y operadores aritméticos, lectura y escritura de datos, estructuras de control alternativas: If-Then (If), operadores relacionales y lógicos, estructuras alternativas múltiples, estructuras de control de bucle: Mientras (while), Para (for) y Hacer...Mientras (do while), funciones con y sin retorno, arreglos uni-dimensionales y arreglos

bidimensionales. Todos los ejercicios son originales del autor, por lo que los lectores obtienen ayuda adicional de este libro en español para complementar todas las extensas referencias y ejercicios disponibles en otros medios, como Internet.

Se ha intentado desarrollar los problemas que no sean dependientes del Sistema Operativo , como la versión o fabricante del mismo (Windows, Linux, MacOS, etc).Como se sabe existen diferentes versiones de lenguajes C, las mismas que presentan en algunos casos ligeras variaciones en las funciones y/o sus formatos, pero manteniendo la esencia principal del lenguaje, por lo que algunos de los ejercicios resueltos en el presente texto, pueden requerir unas ligeras modificaciones en su escritura para que pueda ser ejecutado en otras versiones del lenguaje C, diferentes a la cual fue utilizada para el desarrollo del presente texto.

A tener en cuenta:

Para mayor efectividad, las prácticas deben realizarse en el orden sugerido en este documento:

1. Para responder a las preguntas se requiere comprensión lectora, por lo que antes de los ejercicios se proporciona un resumen de los fundamentos teóricos del curso.
2. Al menos para el primer problema presentado en cada ejercicio, los estudiantes pueden encontrar un pequeño subconjunto de casos difíciles tratando de resolver un problema presentado donde el mismo material está disponible para el estudiante.
3. Si hay un problema al compilar el código en el archivo fuente, es una buena idea consultar el programa incluso si lo ha ejecutado con éxito, ya que al ejecutar otro problema podrá encontrar una solución a un problema que pueda ocurrir.
4. Se recomienda la omisión de los caracteres tildados, aun cuando en este texto no se lo ha hecho por cuestiones de respetar los lineamientos ortográficos.

ISBN: 978-987-82912-8-4

CAPÍTULO I

CONCEPTOS BÁSICOS

1.1 Antes de programar

Cuando hablamos de "Programación en un lenguaje, programación de computadoras" nos referimos al proceso de planificación y desarrollo de soluciones a problemas usando un equipo de cómputo. En su sentido más simple, es el diseño y codificación de un programa que, al ser ejecutado en una computadora, resuelve un problema. En general, un programa es un conjunto de instrucciones para que una computadora haga algo, le dice a la computadora cómo realizar una determinada tarea. Los programas se construyen y escriben en un lenguaje de programación definido por un programador que es el desarrollador y autor del programa (Joyanes; 2005).

Un lenguaje de programación es un conjunto de símbolos y signos que se combinan entre sí según una serie de reglas de sintaxis predefinidas del lenguaje. Existen diferentes clasificaciones de lenguajes de programación: según su desarrollo histórico, propósito, paradigma de programación, nivel de abstracción, etc. Nos centramos en la clasificación por nivel de abstracción., un lenguaje de alto nivel. Estos son los lenguajes de programación que más se asemejan a los lenguajes humanos en cuanto a sus características.(Moreno, 2015)

Ahora el lenguaje de bajo nivel, son específicos de arquitectura, es decir. Los programas escritos en lenguajes de bajo nivel no son portátiles, por lo que solo pueden ejecutarse en la computadora para la que están diseñados. Esta categoría incluye tanto el lenguaje máquina como el lenguaje ensamblador.

Mientras que un lenguaje de programación es una secuencia de caracteres, es decir. letras, números, símbolos, retornos de carro y espacios. Un lenguaje de programación se puede definir como un lenguaje reconocido por una computadora a través de una serie de instrucciones que permiten la resolución de problemas, con un significado concreto y "comprensible". Estas instrucciones son: palabras reservadas (palabras clave), identificadores y símbolos, y la sintaxis del idioma determinará cómo se

ISBN: 978-987-82912-8-4

agruparán para crear un código ejecutable en la computadora. (Martín, Urquía y Rubio, 2021)

Independientemente del lenguaje de programación que nosotros utilicemos todos los programas siguen una línea bien definida para el ciclo de desarrollo del algoritmo o del problema (Juganaru,2015), estas etapas son:

- Edición. - Se codifica el algoritmo según la semántica que sintaxis del lenguaje a utilizar para nuestro caso Lenguaje C.
- Compilación. - Una vez escrito el algoritmo, Lenguaje C, se procede a compilar, es decir, buscar errores sintácticos del algoritmo escrito. Esta compilación es hecha mediante un núcleo, programa propio del lenguaje.
- Enlace. - En el proceso de compilación va a dar como resultado uno o varios archivos. Es aquí donde el compilador también enlaza todos estos archivos para que pueda ejecutarse el programa. Por lo general es un archivo como: .out o .exe el cual enlaza todos los archivos resultantes del proceso anteriormente dicho.
- Ejecución. - Permite validar los errores semánticos que tenga el algoritmo y si realmente puede resolver el problema y lo está realizando adecuadamente. Comúnmente esta etapa es conocida como la puesta a prueba.
- Depuración. - en esta etapa se corrigen los errores que pudieron haberse dado mostrado en etapa de ejecución el depurador es una herramienta que me permite validar el programa algoritmo paso a paso. Con esta Tierra cuenta podemos detectar puntualmente en donde ha sucedido un error y cómo corregirlo.

1.2 Conceptos y definiciones

Un identificador es un nombre asociado a un objeto de programa, puede ser una variable, una función, una constante, un tipo de dato. El nombre de cada identificador debe identificar lo más claramente posible el objeto que identifica, a pesar de la duplicación. Los identificadores generalmente deben comenzar con una letra, no contener espacios (o símbolos extraños) y, por lo general, tener una longitud máxima variable, pero no deben exceder los 20 caracteres para evitar una gran cantidad de lectura.(Oviedo, 2015)

ISBN: 978-987-82912-8-4

Algunos ejemplos de identificadores no válidos y la justificación respectiva:

- 2id — El primer carácter siempre debe ser una letra.
- Estudiante# — El carácter "#" no es válido.
- Codigo estudiante — El espacio en blanco no es válido.
- Codigo-estudiante — El carácter - no es válido.
- "estudiante" — El carácter no es válido.

En un programa siempre se procesa datos, que pueden ser de diferente naturaleza. Dependiendo de su tipo, los datos se representan y almacenan en la memoria de la computadora de cierta manera; es decir, el tipo de datos determina la cantidad de memoria para almacenar. Cabe señalar que estos tamaños son los más comunes, ya que pueden variar entre compiladores, y los rangos de algunos de estos tipos de datos primitivos pueden variar cuando se usan modificadores de tipo, como short long unsigned.(Cevallos, 2018)

Los tipos de datos básicos o primitivos son: char, int, float, double

Tabla 1.1 Tipos Datos

Tipo	Cantidad de Memoria	Rango
Unsigned char	1 byte	0 a 255
Char	1 byte	-128 a 127
Unsigned int	2 bytes	0 a 65,535
Short int	2 bytes	-32,768 a 32,767
Int	2 bytes	-32,768 a 32,767
Unsigned long	4 bytes	0 a 4,294,967,295
Long	4 bytes	-2,147,483,648 a 2,147,483,647
Float	4 bytes	3.4 * (10**-38) a 3.4 * (10**+38)
Double	8 bytes	1.7 * (10**-308) a 1.7 * (10**+308)
Long double	10 bytes	3.4 * (10**-4932) a 1.1 * (10**+4932)

Las palabras reservadas son palabras proporcionadas por un lenguaje de programación para realizar determinadas operaciones, por lo que no pueden utilizarse como nombres identificadores (variables, funciones, constantes, etc.). Estas palabras reservadas pueden ser diferentes, dependerá del lenguaje de programación utilizado,

ISBN: 978-987-82912-8-4

en este caso estamos usando C y algunas de las palabras reservadas son: main, break, while, for, switch, continue, printf, scanf, define, etc. (Joyanes, 2013)

Las Variables

Es un identificador que guarda un único valor, que puede ser como resultado y modificado durante la ejecución del programa. Las variables se declaran al inicio del programa, y antes de que se utilicen en las operaciones. (Jiménez y Otero. 2015). Recordemos que una variable debe poseer: un nombre, un tipo de dato y un valor inicial que este último puede ser opcional.

La sintaxis de una variable es:

```
tipo_dato  nomb_variable;                /*Declaración de una variable */
tipo_daro nom_var1, nom_var2;            /*Declaración de dos variables del mismo tipo de dato */
tipo_dato nom_var = valor_inicial;              /*Declaración e inicialización de una variable */
```

Además, una variable es un repositorio que puede almacenar datos, pero cuyo contenido se puede cambiar, cada variable debe declararse como uno de tipos primitivos, tipos estructurados incorporados o tipos definidos por el usuario como variables estructuradas (Moreno, 2014). Así por ejemplo la definición de una variable basada en un tipo primitivo es la que a continuación se detalla:

- Int a, b;
- Int edad, cont=0;
- float acumulador=0;
- int metros_largo, metros_lancho;
- char respuesta, opcion='S';

Las Constantes

Ahora las constantes Constantes: son valores que no van a cambiar durante la ejecución del programa (Joyanes; 2006). Ejemplos de definición de constantes son:

```
const int edad = 21;
const float PI = 3.14159 7
const char *Universidad = "ESPOCH";
```

ISBN: 978-987-82912-8-4

Las constantes pueden declararse globalmente, es decir, fuera de las funciones que componen el programa, o localmente, es decir, dentro de una determinada función.

Una asignación es una operación que coloca un valor en una dirección RAM. El lenguaje C utiliza el signo igual (=) como símbolo de asignación. El valor de la expresión en el lado derecho debe ser de un tipo de datos compatible con la única variable en el lado izquierdo, o se producirán resultados inesperados, incluido un bloqueo (Rodríguez y Galindo, 2014). veamos algunos ejemplos:

- a = a + 5;
- sueldo=450;
- Estado_civil='D';

1.3 Ejercicios Resueltos

Ejercicio 1.1. Especifique cuál de los siguientes identificadores listados a continuación son correctos:

a) valor_un	**CORRECTO**
b) porcentaje%	**INCORRECTO POR EL %**
c) razon.social	**INCORRECTO POR EL PUNTO**
d) 1ernombre	**INCORRECTO POR INICIAR CON NÚMERO**
e) precio de fábrica	**INCORRECTO POR LA TILDE Y LOS BLANCOS**
f) sueldounificado	**CORRECTO**
g) entero	**CORRECTO**
h) _sueldo	**CORRECTO**

Ejercicio 1.2. Especifique cuál de las siguientes constantes están escritas correctamente:

a) 62,2	**INCORRECTA POR LA COMA**
b) 3.1421	**CORRECTA**
c) 10 5	**INCORRECTA POR EL ESPACIO EN BLANCO**
d) 010	**CORRECTA**
e) "país"	**CORRECTA**
f) 0X4FT	**INCORRECTA POR LA T**
g) 0XAB	**CORRECTA**

ISBN: 978-987-82912-8-4

Ejercicio 1.3. Especifique el tipo de dato al que pertenece las constantes listadas a continuación:

a) '\\'	**CARACTER **
b) .8	**COMA FLOTANTE**
c) 053	**ENTERA OCTAL**
d) 0XAF	**ENTERA HEXADECIMAL**
e) "país"	**CADENA**
f) 50020U	**ENTERA SIN SIGNO**
g) 0XFUL	**ENTERA HEXADECIMAL LARGA SIN SIGNO**

Ejercicio 1.4. Especifique tres tipos de representación que lenguaje c considera válida para las siguientes constantes numéricas:

a) 12.34	**0.1234e3**	**Ó**	**.1234E+3**	**Ó**	**1.234E2**
b) 0.22	**22.E-2**	**Ó**	**2.2e-1**	**Ó**	**0.022E+1**

Ejercicio 1.5. Especifique si existe o no diferencia en lenguaje c al escribir una constante caracter entre comillas simples ('t') o comillas dobles ("t").

'T' .- Representa un constante de tipo caracter (un solo caracter)

"T".- Representa un constante de tipo cadena (dos caracteres la letra T y el caracter nulo de fin de cadena \0)

Ejercicio 1.6. Cuál es la longitud de la cadena de caracteres "Sueldo Básico " justifique su respuesta.

La cadena **tiene 15 caracteres** (12 alfabéticos, 2 espacios en blanco y el caracter de fin de cadena \0, que lenguaje C lo considera como uno solo y lo agrega en las cadenas).

Ejercicio 1.7. De las declaraciones siguientes identifique cuales son correctas y de no serlas realice las correcciones respectivas:

a) int letra = 'A';	**CORRECTA**
b) int letra2="B";	**INCORRECTA** **SERÍA: char letra2='B' ó char letra2[]="B";** **ó int letra2='B';**
c) int valor =67;	**CORRECTA**
d) short letra =='A';	**INCORRECTA**

ISBN: 978-987-82912-8-4

SERÍA: short letra =’A’;

e) float letra=’A’; **CORRECTA**

f) char letra=’\n’; **CORRECTA**

g) int valor = int (3.45); **CORRECTA**

h) include letra = “A”; **INCORRECTA**

SERÍA: #define letra “A” ó #define letra ‘A’

Ejercicio 1.8. Simplifique si es posible el número de instrucciones que tiene el siguiente segmento de programa sin alterar su objetivo.

```
int a, b ,c;
char letra;
char cadena;
a=1;
b=1;
c=1;
letra=’a’;
cadena=” lista”;
```

El segmento simplificado es:

```
int (a,b,c) =1;
char letra =’a’ , cadena[]=”lista”;
```

Ejercicio 1.9. Realice la declaración apropiada para cada una de las siguientes variables:

a) VARIABLE NÚMERO DE DOBLE PRECISIÓN QUE INICIE EN 34.45
double float numero=34.45;

b) VARIABLE ENTERA CORTA SIN SIGNO PARA 134
unsigned short int chico=134;

c) VARIABLE ENTERA LARGA PARA 567890
double int numero=567890;

d) VARIABLE ENTERA OCTAL PARA 453
int octal=0453;

ISBN: 978-987-82912-8-4

e) FORMACIÓN UNIDIMENSIONAL DE CARACTERES PARA “ERROR 1”

char mensaje[]=”ERROR 1”;

Ejercicio 1.10. Defina una constantes simbólicas para los valores de:

a) 3.1416	**#define PI 3.1416**
b) VALORES DE VERDAD	**#define TRUE 1**
	#define FALSE 0
c) EL SÉPTIMO DÍA DE LA SEMANA	**#define DOMINGO 7**
d) “DIGITE ENTER”	**#define CONTINUAR “DIGITE ENTER”**
e) FIN DE LÍNEA	**#define FINCADENA ‘\n’**

Ejercicio 1.11. Mostrar la salida resultante del siguiente segmento de programa

```
char letra ='A';
printf( " EST%c  %cIEN ", letra, letra+1 );
printf( "%c%c%c%c%c", letra+11, letra+4, letra+9, letra+14,letra+18 );
```

La salida es:

ESTA BIEN LEJOS

Ejercicio 1.12. Mostrar la salida que genera el siguiente programa:

```
#include <stdio.h>
main()
{
char pausa;
char texto[]= "LAS LÍNEAS DEL FERROCARRIL";
printf("-%s-\n",texto);
printf("-%25s-\n",texto);
printf("-%30s-\n",texto);
printf("-%25.5s-\n",texto);
printf("-%.5s-\n",texto);
pausa=getchar();
}
```

ISBN: 978-987-82912-8-4

La salida que produce el programa es:

```
-LAS LÍNEAS DEL FERROCARRIL-
-LAS LÍNEAS DEL FERROCARRIL-
-     LAS LÍNEAS DEL FERROCARRIL-
-                              LAS L-
-LAS L-
```

Ejercicio 1.13. Mostrar la salida resultante del siguiente programa:

```
#include <stdio.h>
main()
{
int entero=32;
float real= 12.5;
char pausa, letra= 'a';
printf("-%d-%f-%c-\n",entero, real, letra);
printf("-%5d-%-8.2f-%5c-\n",entero, real, letra);
printf("-%-5d-%-8.2f-%-5c-\n",entero, real, letra);
printf("-%5d-%1.1e-%0c-\n",entero, real, letra);
printf("-%usd-%e-%+5d-\n",entero, real, letra);
pausa=getchar();
}
```

la salida que produce el programa es:

```
-32-12.500000-a-
-    32-12.50    -    a-
-32    -12.50    -a    -
-     32-1.2e+01-a-
-32sd-1.250000e+01-   97-
```

Ejercicio 1.14. Cuál es la salida que genera el siguiente programa cuando se desea visualizar constantes cadena.

```
#include <stdio.h>
#define ES "SULTANA DE LOS ANDES"
// objetivo:
```

ISBN: 978-987-82912-8-4

```
main()
{
char cadena[21]= ES;
char pausa;
printf("*%-12s*\n",ES);
printf("*%21.2s*\n",cadena);
printf("*%-21.2s*\n",cadena);
printf("*%21.5s*\n","RIOBAMBA");
printf("*%21.2s*\n",cadena);
printf("*%.2s*\n",cadena);
printf("*%2.1c*\n",cadena[1]);
scanf( " %c", &pausa);
}
```

La salida que produce el programa es:

```
*SULTANA DE LOS ANDES*
*                    SU*
*SU                     *
*                 RIOBA*
* RIOBA                 *
*SU*
*U*
```

Ejercicio 1.15. Mostrar la salida resultante del siguiente programa:

```
#include <stdio.h>
main()
{
short  int valor2 ='A', valor=short(valor-2);
char pausa;
printf("%1c \n",valor-valor2);
printf("%2c\n",valor2+17);
printf("%1c\n",valor2+7);
printf("%2c\n",valor);
printf("%3c\n",valor2+1);
scanf( " %c", &pausa);
}
```

ISBN: 978-987-82912-8-4

La salida resultante es:

```
]
 R
H
  x
    B
```

Ejercicio 1.16. Cuál es la salida que genera el siguiente programa

```
#include <stdio.h>
#define DIA "DOMINGO"
#define NUMERO 19
// objetivo:

main()
{
int valor = NUMERO;
int numero=18;
char cadena[]= DIA;
char pausa;

printf("EL");
printf(" DÍA %s",cadena);
printf(" NO ES %d", numero);
printf(" SINO ES %d", NUMERO);
printf(" DE  %s","MARZO");
scanf( " %c", &pausa);
}
```

La salida que produce es:

EL DÍA DOMINGO NO ES 18 SINO ES 19 DE MARZO

Ejercicio 1.17. Cuál es la salida de la última instrucción printf del programa listado a continuación si consideramos que se ingresa por teclado: 123 456 789 listo.

ISBN: 978-987-82912-8-4

```
#include <stdio.h>
main()
{
int entero1,entero2, entero3;
char cadena[]="PRUEBA LOS DATOS";
printf("INGRESE LOS DATOS\n");
scanf("%1d %2d %3d %s",&entero1, &entero2, &entero3,cadena);
printf("%d %d %d %s \n",entero1,entero2, entero3,cadena);
}
```

Su salida sería:

1 23 456 789

Pues el valor asignado en el scanf a las variables es:

Entero1=1, entero2=23 entero3=456 y cadena="789".

Ejercicio 1.18. Cuál es la salida que genera el siguiente programa luego de ingresado por teclado el número 78

```
#include <stdio.h>
main()
{
int numero;
char pausa;
printf("INGRESE UN NÚMERO ");
scanf("%d",&numero);
printf(" %d",numero);
printf(" %c",short(numero));
printf(" %f",float(numero));
printf(" %e \n",float(numero));
printf(" %u",unsigned(numero));
printf(" %u",unsigned(-numero));
printf(" %f",(-numero));
scanf(" %c",pausa);
}
```

La salida que produce el programa es:

78 N 78.000000 7.800000e+01

78 4294967218 0.000000

ISBN: 978-987-82912-8-4

Ejercicio 1.19. Determine que función realiza el siguiente programa:

```
#include <stdio.h>
main()
{
char letra;
char pausa;
printf("INGRESE UN CARACTER ");
scanf("%c",&letra);
printf(" %c%c %c%c%c%c%c%c%c %c%c %d",69,76,79,82,68,73,78,65,76,69,83,
letra);
scanf(" %c",&pausa);
}
```

Al ejecutar el programa ingresando desde el teclado el caracter 'a' es:

```
INGRESE UN CARACTER a
EL ORDINAL DE a ES 97
```

Lo que quiere decir que la función que realiza el programa es visualizar el ordinal del caracter ingresado.

Ejercicio 1.20. Escriba un programa que transforme una letra minúscula en mayúscula considerando:

- Que el usuario siempre ingresara unicamente lo solicitado y

- Que no puede utilizar funciones que realicen dicha acción.

```
#include <stdio.h>
// Objetivo: TRANSFORMA UNA LETRA MAYÚSCULA EN MINÚSCULA
main()
{
char letra;
char pausa;
printf("INGRESE UN LETRA MINÚSCULA ");
scanf("%c",&letra);
printf("LA MAYÚSCULA DE %c ES %c",letra,(letra+('A'-'a')));
```

ISBN: 978-987-82912-8-4

```
scanf(" %c",pausa);
}
```

Ejercicio 1.21. Escribir un conjunto de instrucciones para que una variable se le asigne desde el teclado solo valores booleanos. considerando que existen diferentes formas de representar valores de verdad (0,1, verdadero, falso, si, no, true, false).

```
char verdad ;
printf("LECTURA DE FORMATOS DE VALORES BOOLEANOS( V,F, 1,0,S,N,T,F)"
);
scanf("%[10VFvfTFtfSNsn]",&verdad);
printf("EL VALOR DE VERDAD INGRESADO ES %c\n",verdad);
```

Ejercicio 1.22. Escribir un programa con el menor número de instrucciones que sea capaz de distinguir el nombre y la fecha de nacimiento de una persona al ser ingresados por el usuario en una sola línea.

```
#include <stdio.h>
// Objetivo: DIFERENCIA DE UN TEXTO INGRESADO LA FECHA DE
NACIMIENTO DEL NOMBRE.
main()
{
char texto[80], texto1[80];
int pausa;
printf("INGRESE UN TEXTO CON SU NOMBRE Y FECHA DE NACIMIENTO
(mm-dd-aa)\n");
scanf("%[^0123456789-]",texto);
scanf("%[0123456789-]",texto1);
printf("HAS VENIDO A ESTE MUNDO EL %8s,  Y TE LLAMAS %s\n",texto1,
texto);
scanf(" %d", &pausa);
}
```

ISBN: 978-987-82912-8-4

1.4 Ejercicios Propuestos

- Desarrolle un programa en lenguaje C que dado una cantidad (numero entero ingresado por teclado) en dólares, calcule el equivalente en euros y el resultado se muestre en pantalla
- Desarrolle un programa en lenguaje C que dado un nombre (dato cadena texto ingresado por teclado) de una persona, muestre la salida por pantalla el siguiente mensaje: “Hola, <Nombre>, hoy es un día genial”.
- Escriba un programa en lenguaje C, que transforme una letra mayúscula en minúscula considerando que el usuario siempre ingresará lo solicitado y no puede utilizar funciones que realicen dicha acción.
- Cuál será el resultado del siguiente programa:

```
main() {
int a=5, b;
b=a+2;
printf (“%d\n”, b);
b=b+2;
printf (“%d\n”, b);
b++;
printf (“%d\n”, b);
}
```

- Desarrolle un programa en lenguaje C que dado un numero (entero del 1 al 7 ingresado por teclado), valide el ingreso respectivo al día de la semana (lunes, martes, miércoles, jueves, viernes, sábado y domingo) muestre la salida por pantalla el siguiente mensaje: “El día de la semana es <día>”.

ISBN: 978-987-82912-8-4

ISBN: 978-987-82912-8-4

CAPÍTULO II

OPERADORES Y EXPRESIONES

2.1 Entradas y Salidas de datos

Un programa es un conjunto de instrucciones que ejecuta una computadora para lograr un resultado o una solución a un problema específico. Este resultado casi siempre se logra manipulando los datos. Los datos suelen depender del usuario. El usuario necesitará conocer la salida del programa que está utilizando, que tienen instrucciones que permiten al usuario ingresar datos y otras que permiten mostrar la salida generada por estos programas. El lenguaje C cuenta con las funciones scanf() y printf() para la entrada y salida de datos respectivamente (librería , cabecera stdio.h). (Garrido y Valdivia, 2006)

Salida de Datos

La computadora dispone de diversos medios para proporcionar la salida de datos ya sea un archivo una impresora o el monitor. La sintaxis de la función printf() para la salida de los datos es la siguiente:

printf ("texto, cadena_control_tipo", argumentos);

Ejemplo:

```
int x=5;
float y=2.8;
char z='v';

printf("%d", x);          Se muestra 5 qué es el contenido de la variable x
printf("%f", x+y);        Se muestra 7 , que es la suma de la variable
                          X y Y como valor entero
printf("%c", z);          se muestra la letra 'v', qué es el contenido de la
                          variable z
printf("%c %d %f", c, a, b);      Se muestra la letra s, los números  5 y 2.8
                                  qué son los valores de las variables X, Y y
                                  Z respectivamente
```

ISBN: 978-987-82912-8-4

Entrada de datos

La entrada de datos es una operación de escritura que se puede hacer a través de diferentes dispositivos en el equipo de cómputo, como lo es un teclado o desde un archivo. Sin embargo si se usa la función scanf() se trata de una entrada de datos desde el teclado (Eslava, 2016). La sintaxis de scanf es la siguiente:

scanf ("cadena_control_tipo", &variable);

Ejemplo:

```
scanf("%d",&ed);
scanf("%d %d %d",&a,&b,&c);
scanf("%f",&sal);
scanf("%s",&nom);
```

Códigos y formatos más utilizados y su significado, según presentan Joyanes y Zahonero (2003; p.108):

- %d El dato se convierte en entero decimal.
- %o El dato entero se convierte en octal.
- %x El dato entero se convierte en hexadecimal.
- %u El dato entero se convierte en entero sin signo.
- %c El dato se considera de tipo carácter.
- %f El dato se considera de tipo float. Se convierte a notación decimal, con parte entera y los dígitos de precisión.
- %g El datos se considera de tipo float. Se convierte según en código %e o
- %f dependiendo de cuál sea la presentación más corta.
- %s El dato ha de ser una cadena de caracteres.
- %lf El dato se considera de tipo double

2.2 Operadores aritméticos y de asignación

Los operadores aritméticos son los que solemos usar para realizar operaciones aritméticas básicas que son suma, resta, multiplicación, división y resto o módulo. A menudo escuchamos que estos operadores aritméticos se les denominan operadores binarios, lo que significa que siempre usan 2 operandos (datos- variables) para que funcionen (Eslava, 2016).

ISBN: 978-987-82912-8-4

Se utilizan para realizar operaciones aritméticas básicas. Los operadores aritméticos en la letra "C" siguen las típicas reglas algebraicas de jerarquía o precedencia en matemáticas. Estas reglas determinan el orden en que se realizan las operaciones aritméticas, primero se evalúa la expresión entre paréntesis, luego los operadores unarios; esto se trata con más detalle en otras prácticas, luego los operadores unarios. operadores de división, suma y resta. (Martín, Urquía y Rubio, 2021)

Tabla 2.1 Operadores aritméticos

Símbolo	Descripción	Ejemplo
+	Suma	x + y
-	Resta	x – y
*****	Multiplicación	x * y
/	División	x / y
%	Módulo	x % y

También podemos combinar otros operadores de asignación muy particulares que en el lenguaje se nos permite como, por ejemplo:

- a += b; Equivale : a = a + b;
- a -= b; Equivale : a = a – b;
- a *= b; Equivale : a = a * b;
- a /= b; Equivale : a = a / b;
- /a %= b; Equivale : a = a % b;

Los operadores ++ y -- se pueden usar como prefijos o sufijos, es decir, se pueden usar antes o después de una variable y aunque en ambos casos estas incrementan o decrementa en 1 existe una innegable diferencia en cómo utilizarlos como se puede ver en el ejemplo:

```
Caso 1:
b=16;
y=++b;
```

```
Caso 2:
b=16;
y=b++;
```

ISBN: 978-987-82912-8-4

En el Caso 1: La variable **'y'** recibe un 17 porque **'b'** primero se incrementa y luego se asigna.
En el Caso 2: La variable **'y'** recibe un 10 ya que primero se asigna el valor de **'b'** posteriormente se incrementa.

2.3 Operadores relaciónales

Estos operadores se utilizan para expresar condiciones en los programas y así determinar el orden en que se ejecutarán las instrucciones; la condición en el lenguaje C es una expresión booleana cuyo resultado solo puede ser falso o verdadero

Tabla 2.2 Operadores relacionales

Símbolo	Descripción	Ejemplo
<	menor que	(x < y)
>	mayor que	(x > y)
<=	menor o igual que	(x <= y)
>=	mayor o igual que	(x >= y)
==	Igual que	(x == y)
!=	distinto que	(x != y)

Nota: Un error muy común es confundir el operador de asignación (=) con el operador de igualdad (==).

2.4 Operadores lógicos

Los operadores lógicos se utilizan para conectar varias condiciones en cualquier programa, lo que permite condiciones más complejas que solo pueden devolver verdadero o falso. El uso de estos operadores requiere expresiones lógicas, que generalmente se construyen a partir de operaciones relacionales. (Schildt, 2000).

Tabla 2.3 Operadores lógicos

Símbolo	Descripción	Ejemplo
&&	Y (AND)	(x < y) && (w>z)
||	O(OR)	(x < y) || (w>z)
!	NEGACION (NOT)	!(x > y)

ISBN: 978-987-82912-8-4

La prioridad de los operadores lógicos es la siguiente: NOT, AND y OR. La prioridad de los operadores en general es la siguiente: se evalúan primero los paréntesis. Luego exponente y raíz. Después multiplicación, división, mod, not. Le sigue suma, resta, and. Para finalizar con: >, =, <=, <>, =, or.

2.5 Ejercicios Resueltos

Ejercicio 2.1. Determine el valor resultante de cada una de las siguientes expresiones aritméticas:

a) 2+3+45/23	**ES IGUAL A 6**
b) 24+12*5/6	**ES IGUAL A 34**
c) 16/.5-12	**ES IGUAL A 20.0**
d) 45.8/3	**ES IGUAL A 15.266667**
e) 16+ int(34)	**ES IGUAL A 50**
f) 3.0+6/8*2-4	**ES IGUAL A -1.0**
g) 12%int(2*'A'/2)	**ES IGUAL A 12**
h) 8+'a'-34/2	**ES IGUAL A 88**

Ejercicio 2.2. Determine el tipo de dato resultante de cada una de las siguientes expresiones aritméticas:

a) 3*12/16*12- int(4)	**ES DE TIPO ENTERO (20)**
b) –(13+5.6-8)*-1	**ES DE TIPO REAL (10.6)**
c) int (44.6-16.8+5)/3	**ES DE TIPO ENTERO (10)**
d) +45/5%5%12	**ES DE TIPO ENTERO (4)**
e) sqrt(56-+12+3)	**ES DE TIPO REAL (6.855655)**
f) 3.0+6/8*2-4	**ES DE TIPO REAL (-1.0)**
g) int(12.2*.5)%'C'	**ES DE TIPO ENTERO (6)**
h) 'c'-'a'+'f'	**ES DE TIPO CHAR (h)**

ISBN: 978-987-82912-8-4

Ejercicio 2.3. Considerando las variables a=12, b=4.0 y c='2' obtenga el valor que genera cada una de las siguientes expresiones aritméticas:

a) ++a+--b	**EL RESULTADO ES 16.0**
b) -a+--c	**EL RESULTADO ES 37**
c) a<12 && c>1.5	**EL RESULTADO ES 0**
d) c!=45 || b/3 >= 2.5	**EL RESULTADO ES 1**
e) 2*a+(a==12)*c	**EL RESULTADO ES 73**
f) !(48!=a*int(12/3))	**EL RESULTADO ES 0**
g) c*sqrt(int(b)/2)	**EL RESULTADO ES 69.296**
h) 5*c!=20/a==b/2 >3/.5	**EL RESULTADO ES 0**

Ejercicio 2.4. Determine el resultado y el tipo de dato de la variable en las siguientes operaciones de asignación:

a) a= 4.5 /3-2	**EL RESULTADO ES -0.5 (FLOAT)**
b) b=34*16/67-4	**EL RESULTADO ES 4.0 (FLOAT)**
c) c=(3.4*56)	**EL RESULTADO ES 190.39994 (FLOAT)**
d) d=(6/3)	**EL RESULTADO ES 2 (INT)**
e) h='c'%2	**EL RESULTADO ES 1 (INT)**
f) g=24*sqr(36)-17.5	**EL RESULTADO ES 126.5 (FLOAT)**
g) w=12*islower('c')%fmod(4,2)	**EL RESULTADO ES 24 (DOUBLE)**
h) t=4>12 && 36%5 <=8.99	**EL RESULTADO ES 0 (INT)**

Ejercicio 2.5. Determine los valores de las variables en cada una de las siguientes operaciones de asignación; consideré que las variables a,b,c se inicializan siempre con 4.5, 6.7 y 12.0 respectivamente.

a) a*= 4.5 /--c	**a=1.841,**	**b=6.7,**	**c=11.0**
b) b*=34+a	**a=4.5,**	**b=257.950,**	**c=12.0**
c) c/=pow(a,2)	**a=4.5,**	**b=6.7,**	**c=0.593**
d) b*=--c	**a=4.5,**	**b=73.7,**	**c=11.0**
e) a+=1-a+c++	**a=13.0,**	**b=6.7,**	**c=13.0**
f) c=12.0&&a	**a=4.5,**	**b=6.7,**	**c=1.0**
g) a*=2+c-- *++b	**a=424.8,**	**b=7.7,**	**c=11.0**
h) b-=--a*5/c-1	**a=3.5,**	**b=6.242,**	**c=12.0**

ISBN: 978-987-82912-8-4

Ejercicio 2.6. Determine el resultado que entregan las siguientes funciones de biblioteca:

a) floor(x) **REDONDEA AL ENTERO MAS CERCANO ES DE TIPO DOUBLE.**
b) exp(x) **ELEVA e (2.71828…) A LA POTENCIA x ES DE TIPO DOUBLE.**
c) fmod(x,y) **ENTREGA EL MÓDULO DE X/Y ES DE TIPO DOUBLE.**
d) ceil(x) **REDONDEA x POR EXCESO ES DE TIPO DOUBLE.**
e) isalpha(x) **ENTREGA 1 SI C ES ALFABÉTICO ES DE TIPO ENTERO.**
f) strlen(x) **DA EL NÚMERO DE CARACTERES DE X; ES DE TIPO ENTERO.**
g) log(x) **DA EL LOGARITMO NATURAL DE X; ES DE TIPO DOUBLE.**

Ejercicio 2.7. Determine el resultado de las siguientes expresiones que combinan funciones de biblioteca:

a) abs(45.5)- abs(-45.7) **EL RESULTADO ES 0**
b) sqrt(4/78.9/4) **EL RESULTADO ES 0.112580**
c) isdigit('0') **EL RESULTADO ES 4**
d) pow(34%4, 45/8-4) **EL RESULTADO ES 2.0**
e) floor((4.9*5)/(45%13)) **EL RESULTADO ES 4.0**
f) toascii(65+5/4) **EL RESULTADO ES B**
g) ceil(abs(8-12.16)+7%2) **EL RESULTADO ES 5.0**
h) exp(short(6*('4'%'5')/'9')) **EL RESULTADO ES 148.413159**

Ejercicio 2.8. Considerando que las variables c=4, d='2', e=12.4, determine el resultado de las siguientes expresiones:

a) d<'3'+abs(e) **EL RESULTADO ES 1**
b) d+c*abs('e') **EL RESULTADO ES 454**
c) e+'d'/4 **EL RESULTADO ES 37.4**
d) e*3.4*floor(e) **EL RESULTADO ES 505.919984**
e) labs(sqrt(d++)) **EL RESULTADO ES 7**
f) pow(abs(d-45),2) **EL RESULTADO ES 36.0**
g) –-c+(--d)*2 **EL RESULTADO ES 103**
h) 4.5/--12.4+atoi("23") **EL RESULTADO ES 23.362903**

ISBN: 978-987-82912-8-4

Ejercicio 2.9. Determine el resultado y el tipo de dato de cada variable en cada una de las siguientes expresiones de asignación:

a) 4<6?3:9	**EL RESULTADO ES 3**
b) 'A'=='65' ? "SI":"NO"	**EL RESULTADO ES NO**
c) ceil(4.8)<floor(4.8)? "VERDADERO":"FALSO"	**EL RESULTADO ES FALSO**
d) (atoi("EIET")>atoi("EIETC"))?"M":"N"	**EL RESULTADO ES N**
e) 4.2<abs(4.5)?12:7	**EL RESULTADO ES 7**
f) abs(12.5) == labs(12.5)? "v": "f"	**EL RESULTADO ES v**
g) abs(2.5) !=2? 12+45/abs(4.5): '8'+'5'/2	**EL RESULTADO ES 82**

Ejercicio 2.10. Realice un programa que determine si un número entero ingresado por el usuario es par o impar.

```
#include <stdio.h>
#define I "IMPAR"
#define P "PAR"
// Objetivo: DETERMINA SI UN NÚMERO ES PAR O IMPAR
int numero,par;
char pausa;
main()
{
  printf("INDICA SI UN NÚMERO ES PAR O IMPAR\n");
  printf("INGRESE UN NÚMERO ENTERO:");
  scanf("%d",&numero);
  par=int(fmod(numero,2));
  printf("EL NÚMERO %d ES %s",numero,(par==0?P:I));
  scanf( " %c", &pausa);
 }
```

Ejercicio 2.11. Demuestre si se puede sustituir la instrucción gets y puts por instrucciones leer y escribir sin alterar el sentido de la función.

```
#include <stdio.h>
#define  cadenas  char
```

ISBN: 978-987-82912-8-4

```
#define LEER(cadena)  gets(cadena)
#define ESCRIBIR(cadena)  puts(cadena)
// Objetivo: RENOMBRAR LAS INSTRUCCIONES DE ENTRADA/SALIDA GETS Y
PUTS
cadenas cadena[20];
cadenas pausa;
main()
{
printf("RENOMBRAR INSTRUCCIONES GETS Y PUTS\n");
printf("INGRESE UNA CADENA (MAX 20):");
LEER(cadena);
printf("LA CADENA INGRESADA ES: ");
ESCRIBIR(cadena);
printf("PARA CONTINUAR DIGITE UN CARACTER.. ");
scanf( " %c", &pausa);
}
```

Ejercicio 2.12. Realice un programa que multiplique dos números enteros ayudado con definiciones del preprocesador

```
#include <stdio.h>
#include <math.h>
#define MULTIPLICAR(numero1,numero2)  numero1*numero2
// Objetivo: MULTIPLICA DOS NÚMEROS ENTEROS
int numero1,numero2;
char pausa;
main()
{
printf("MULTIPLICA DOS NÚMEROS ENTEROS\n");
printf("INGRESE EL PRIMER NÚMERO (ENTERO):");
scanf("%d",&numero1);
printf("INGRESE EL SEGUNDO NÚMERO (ENTERO):");
scanf("%d",&numero2);
printf("LA MULTIPLICACIÓN ES %d ",MULTIPLICAR(numero1,numero2));
scanf( " %c", &pausa);
}
```

ISBN: 978-987-82912-8-4

Ejercicio 2.13. Realice un programa que transforme de grados farenheit a celcius

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
// Objetivo: TRANSFORMAR GRADOS FAHRENHEIT A CELSIUS Y  KELVIN
float temperaturaf;
float temperaturac;
float temperaturaK;
main()
{
printf("TRANSFORMAR GRADOS FAHRENHEIT A CELSIUS Y  KELVIN\n\n");
printf("INGRESE LA TEMPERATURA EN GRADOS FARENHEIT = ");
scanf("%f",&temperaturaf);
temperaturac=(temperaturaf-32)* 5/9;
printf("LA TEMPERATURA EN GRADOS CELSIUS ES = %3.2f\n", temperaturac);
temperaturac=(temperaturac + 273.15);
printf("LA TEMPERATURA EN GRADOS KELVIN ES = %3.2f\n", temperaturac);
printf("\n\nUNA TECLA PARA CONTINUAR... ");
getch();
}
```

Ejercicio 2.14. Realice un programa que descomponga un número ingresado por el usuario en su parte entera y decimal.

```
#include <stdio.h>
#include <math.h>
// Objetivo: SEPARA UN NÚMERO LA  PARTE ENTERA DE LA DECIMAL
float numero,parentera, pardecimal;
char pausa;

main()
{
printf("DESCOMPONE A UN NÚMERO \n");
```

ISBN: 978-987-82912-8-4

```
printf("INGRESE UN NÚMERO REAL:");
scanf("%f",&numero);
parentera=floor(numero);
pardecimal=numero-parentera;
printf("%5.0f ES  LA PARTE ENTERA  DE %8.4f\n",parentera,numero);
printf("%4.3f ES  LA PARTE DECIMAL DE %8.4f\n",pardecimal,numero);
scanf( " %c", &pausa);
}
```

Ejercicio 2.15. Realice un programa que permita determinar el mayor de tres números

```
#include <stdio.h>
// Objetivo: Encuentra el mayor de tres números

main()
{
float numero1, numero2, numero3, mayor;
char pausa;
printf( "INGRESE TRES NÚMEROS:\n");
scanf("%f %f %f",&numero1, &numero2, &numero3);
mayor=numero1>numero2? numero1:numero2;
mayor=mayor>numero3? mayor:numero3;
printf("EL MAYOR DE %4.1f, %4.1f, %4.1f ES
%4.1f\n",numero1,numero2,numero3, mayor);
scanf( " %c", &pausa);
}
```

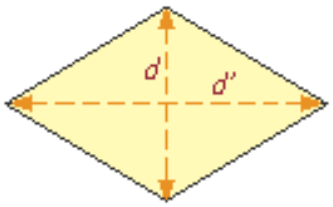
Ejercicio 2.16. Realice un programa que permita calcular el área y perímetro de un cuadrado al ser ingresado un lado.

```
#include <stdio.h>
#include <math.h>
// Objetivo: CALCULA EL ÁREA Y PERÍMETRO DE UN
CUADRADO
```

ISBN: 978-987-82912-8-4

```
main()
{
float lado, area, perimetro;
char pausa;
printf("INGRESE EL LADO DEL CUADRADO:\n");
scanf("%f",&lado);
area=pow(lado,2);
perimetro= lado*4;
printf("EL ÁREA ES     : %4.2f\n", area);
printf("EL PERÍMETRO ES: %4.2f\n", perimetro);
scanf( " %c", &pausa);
}
```

Ejercicio 2.17. Realice un programa que solicite al usuario los datos necesarios para calcular el área de un rombo. A=d*d'/2

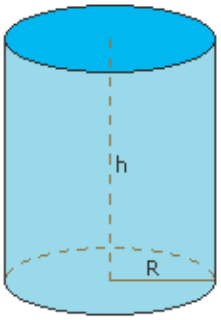



```
#include <stdio.h>
#include <math.h>
// Objetivo: CALCULA EL ÁREA DE UN ROMBO

main()
{
float diagonal1, diagonal2;
char pausa;
printf("CALCULA EL ÁREA DE UN ROMBO\n");
printf("INGRESE LA DIAGONAL PRINCIPAL:\n");
scanf("%f",&diagonal1);
printf("INGRESE LA DIAGONAL SECUNDARIA:\n");
scanf("%f",&diagonal2);
printf("EL ÁREA ES: %4.2f\n",diagonal1*diagonal2/2);
printf("<UNA LETRA> Y <ENTER> PARA CONTINUAR ");
scanf( " %c", &pausa);
}
```

ISBN: 978-987-82912-8-4

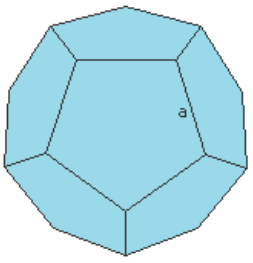
Ejercicio 2.18. Realice un programa que calcule el volumen de un cilindro recto considerando que $V = \pi r^2 \mathrm{h}$.




```
#include <stdio.h>
#include <math.h>
#define PI 3.141516
// Objetivo: CALCULA EL VOLUMEN DE UN CILINDRO RECTO

main()
{
float radio,altura,volumen;
char pausa;
printf("CALCULA EL VOLUMEN DEL UN CILINDRO RECTO\n");
printf("INGRESE EL RADIO DE LA BASE:");
scanf("%f",&radio);
printf("INGRESE LA ALTURA:");
scanf("%f",&altura);
volumen=PI*pow(radio,2)*altura;
printf("EL VOLUMEN DEL CILINDRO DE RADIO %5.2f Y ALTURA %5.2f ES IGUAL A %5.2f\n",radio,altura,volumen);
scanf( " %c", &pausa);
}
```

Ejercicio 2.19. Realice un programa que calcule el volumen de un dodecaedro considerando que su fórmula es $v = a^3 \frac{15+7\sqrt{5}}{4}$




```
#include <stdio.h>
#include <math.h>
#define FACTOR (15+7*sqrt(5))/4
// objetivo: CALCULA EL VOLUMEN DE UN DODECAEDRO
main()
{
float arista,volumen;
char pausa;
printf("CALCULA EL VOLUMEN DEL DODECAEDRO\n");
```

ISBN: 978-987-82912-8-4

```
printf("INGRESE EL VALOR DE LA ARISTA:");
scanf("%f",&arista);
printf("EL VOLUMEN DEL DODECAEDRO ES IGUAL A
%5.2f\n",pow(arista,3)*FACTOR);
scanf( " %c", &pausa);
}
```

Ejercicio 2.20. Realice un programa que calcule el área y perímetro de una circunferencia conociendo su diámetro.

ÁREA	PERÍMETRO
$A = \pi \cdot r^2$ **DONDE:** A **ES EL ÁREA, Y** π **ES LA CONSTANTE MATEMÁTICA PI** r **EL RADIO DEL CÍRCULO**	$P = d \cdot \pi$ **DONDE:** P **ES EL PERÍMETRO ,** π **ES LA CONSTANTE MATEMÁTICA PI,** d **ES EL DIÁMETRO DEL CÍRCULO**

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
#define PI 3.14159265
// Objetivo: CALCULA EL ÁREA Y PERÍMETRO DE UNA CIRCUNFERENCIA
//           CONOCIENDO SU DIÁMETRO

main()
{
float diametro, area, perimetro, radio;
char pausa;
printf("CALCULA EL ÁREA Y PERÍMETRO DE UNA CIRCUNFERENCIA\n");
printf("\nINGRESE EL DIÁMETRO DE LA CIRCUNFERENCIA:");
scanf("%f",&diametro);
radio=diametro/2;
perimetro= PI*diametro;
printf("\nLA CIRCUNFERENCIA DE DIÁMETRO %5.2f TIENE:\n\n",diametro);
```

ISBN: 978-987-82912-8-4

```
printf("UN PERÍMETRO DE: %5.2f\n",perimetro);
area= PI*pow(radio,2);
printf("UNA ÁREA DE    : %5.2f\n",area);
printf("\nUNA TECLA PARA CONTINUAR");
getch();
}
```

Ejercicio 2.21. Realice un programa que tomado una cantidad expresada en metros lineales lo transforme a su equivalente en kilómetros, centímetros y milímetros.

```
#include <stdio.h>
#include <math.h>
#define FACTOR 100.0
// Objetivo: REALIZA LA CONVERSIÓN DE METROS A CM, MM y KM.
int a,b,c;
float longitud,metros,kilometros,centimetros;
double milimetros;
char pausa;
main()
{
printf("TRANSFORMA METROS A KILÓMETROS, CENTÍMETROS Y MILÍMETROS\n");
printf("INGRESE LA LONGITUD EN METROS:");
scanf("%f",&longitud);
kilometros= longitud/(FACTOR*10);
centimetros= longitud*FACTOR;
milimetros= centimetros*FACTOR;
printf("%.2f METROS EQUIVALE A %.4f KILÓMETROS\n",longitud,kilometros);
printf("%.2f METROS EQUIVALE A %.2f CENTÍMETROS\n",longitud,centimetros);
printf("%.2f METROS EQUIVALE A %.2f MILÍMETROS\n",longitud,milimetros);
scanf( " %c", &pausa);
}
```

ISBN: 978-987-82912-8-4

Ejercicio 2.22. Realice un programa que permita calcular la aceleración que tiene un cuerpo al conocer su velocidad inicial y final en un instante de tiempo.

```
#include <stdio.h>
#include <math.h>
#define FACTOR 100.0
// Objetivo: CALCULA LA ACELERACIÓN QUE TIENE UN CUERPO CUANDO VARÍA SU
//              VELOCIDAD EN UN     INSTANTE DE TIEMPO
float Vo,Vf,a;
int t;
double milimetros;
char pausa;
main()
{
printf("CALCULA LA ACELERACIÓN DE UN CUERPO\n");
printf("INGRESE LA VELOCIDAD INICIAL (m/s):");
scanf("%f",&Vo);
printf("INGRESE LA VELOCIDAD FINAL (m/s):");
scanf("%f",&Vf);
printf("INGRESE EL TIEMPO TRANSCURRIDO (s):");
scanf("%d",&t);
a=(Vf-Vo)/t;
printf("EL CUERPO TIENE UNA ACELERACIÓN DE %.2f m2/s2\n",a);
scanf( " %c", &pausa); }
```

Ejercicio 2.23. Realice un programa que indique el total de caracteres que tiene una cadena ingresada

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
// Objetivo: CUENTA LOS CARACTERES DE UNA CADENA
```

ISBN: 978-987-82912-8-4

```
char cadena[80];
char pausa;
main()
{
printf("CUENTA LOS CARACTERES INGRESADOS\n");
printf("INGRESE UN TEXTO (80 CARACTERES MAX)\n ");
printf("EL TEXTO INGRESADO TIENE %d CARACTERES \n
",strlen(gets(cadena)));
scanf( " %c", &pausa);
}
```

Ejercicio 2.24. Realice un programa que transforme un texto a mayúsculas.

```
#include <stdio.h>
#include <string.h>
// Objetivo: TRANSFORMA UN TEXTO LEÍDO A MAYÚSCULAS
char cadena[200];
char pausa;
main()
{
printf("PONE EN MAYÚSCULAS  UN TEXTO \n");
printf("INGRESE UN TEXTO ( MAX 200 CARACTERES)\n ");
puts(strupr(gets(cadena)));
scanf( " %c", &pausa);
}
```

Ejercicio 2.25. Realice un programa que lea un texto y lo convierta a mayúsculas o minúsculas según una opción escogida por el usuario.

```
#include <stdio.h>
#include <string.h>
#define COMENTARIO "PONE EN MAYÚSCULAS/MINÚSCULAS  UN TEXTO
\n\n"
#define MIN  "EL TEXTO EN MINÚSCULAS ES: "
#define MAX  "EL TEXTO EN MAYÚSCULAS ES: "
#define ERROR "ERROR: !!OPCIÓN FUERA DE RANGO!!"
// Objetivo: TRANSFORMA UN TEXTO LEÍDO A MAYÚSCULAS/MINÚSCULAS
```

ISBN: 978-987-82912-8-4

```
char cadena[200], minusculas[250], mayusculas[250];
int opcion;
char pausa;
main()
{
printf("%s ", COMENTARIO);

printf("INGRESE EL TEXTO (MAX 200 CARACTERES)\n");
gets(cadena);
printf("\n 1. MAYÚSCULAS\n 2. MINÚSCULAS \n \n INGRESE LA OPCIÓN: ");
scanf(" %d",&opcion);
strcpy(minusculas,MIN);
strcpy(mayusculas,MAX);
strcat(minusculas,strlwr(cadena));
((opcion==1)||(opcion==2))?opcion==1?puts(strcat(mayusculas,strupr(cadena))):
puts(strcat(minusculas,strlwr(cadena))): puts(ERROR);
scanf( " %c", &pausa);
}
```

Ejercicio 2.26. Realice un programa que demuestre que si el contenido de una variable cadena son solo conjunto de dígitos estas pueden ser utilizadas como variables enteras para realizar operaciones aritméticas con ellos, ejemplo: la cadena "23" sumada más la cadena"40" sea igual al número 43.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* Objetivo: REALIZA OPERACIONES ARITMÉTICAS CON NÚMEROS
INGRESADOS COMO CADENA DE CARACTERES */
char cadena1[4];
char cadena2[4];
char pausa;
main()
{
```

ISBN: 978-987-82912-8-4

```
printf("OPERACIONES ARITMÉTICAS CON NÚMEROS LEÍDOS COMO TEXTO \n");
puts("INGRESE EL PRIMER OPERANDO :");
gets(cadena1);
puts("INGRESE EL SEGUNDO OPERANDO :");
gets(cadena2);
printf("SUMA : %s + %s = %d \n", cadena1, cadena2, atoi(cadena1)+atoi(cadena2));
printf("RESTA : %s - %s = %d \n", cadena1, cadena2, atoi(cadena1)-atoi(cadena2));
printf("MULTIPLICACIÓN: %s * %s = %d \n", cadena1, cadena2, atoi(cadena1)*atoi(cadena2));
printf("DIVISIÓN ENTERA: %s / %s = %d \n", cadena1, cadena2, atoi(cadena1)/atoi(cadena2));
printf("RESIDUO : MOD(%s,%s) = %d \n", cadena1, cadena2, atoi(cadena1)%atoi(cadena2));
scanf( " %c", &pausa);
}
```

Ejercicio 2.27. Realice un programa que determine si son o no números triángulos los ingresados por el usuario. sabiendo que los números triángulos son 3 números diferentes, de los cuales la suma de los cuadrados de los números menores es igual al cuadrado del número mayor. Ejemplo 5, 12 y 13.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
// Objetivo:DETERMINAR SI TRES NÚMEROS SON TRIÁNGULOS
int suma, numero1,numero2,numero3;
char resultado[3];
main()

{
```

ISBN: 978-987-82912-8-4

```
 printf("DETERMINA SI TRES NÚMEROS ENTEROS SON TRIÁNGULOS\n\n");
 printf("INGRESE EL PRIMER NÚMERO = ");
 scanf("%d",&numero1);
 printf("INGRESE EL SEGUNDO NÚMERO = ");
 scanf("%d",&numero2);
 printf("INGRESE EL TERCER NÚMERO = ");
 scanf("%d",&numero3);
 suma=pow(numero1,2)+pow(numero2,2);

 (suma==pow(numero3,2))? strcpy(resultado,"SI"):strcpy(resultado,"NO");

 printf("\n!!EL %d, %d Y %d %s SON NÚMEROS
TRIÁNGULOS!!\n",numero1,numero2,numero3,resultado);

 printf("\nUNA TECLA PARA SALIR....");
 getch();
}
```

Ejercicio 2.28. Realice un programa que calcule la distancia d entre los puntos p1(x1,y1) y p2(x2,y2) considerando que d= $((X_2-X_1)^2+(Y_2-Y_1)^2)^{1/2}$

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

// Objetivo: DETERMINAR LA DISTANCIA ENTRE DOS PUNTOS P1(X1,Y1) Y
P2(X2,Y2)

float distancia;
int puntox1,puntoy1,puntox2,puntoy2;
char continuar;

main()
{
```

ISBN: 978-987-82912-8-4

```
printf("DISTANCIA ENTRE P1(X1,Y1) Y P2(X2,Y2) \n");
printf("INGRESE EL PUNTO P1(X1,Y1) : ");
scanf("%d %d",&puntox1,&puntoy1);
printf("INGRESE EL PUNTO P2(X2,Y2): ");
scanf("%d %d",&puntox2,&puntoy2);

distancia = sqrt(pow(puntox2-puntox1,2)+pow(puntoy2-puntoy1,2));
printf("\nP1(%d,%d)Y P2(%d,%d) DISTAN %5:2f UNIDADES\n",

puntox1,puntoy1,puntox2,puntoy2,distancia);

printf("\n UNA LETRA PARA SEGUIR…");
scanf(" %c",&continuar);
 }
```

Ejercicio 2.29. Realice un programa que calcule la distancia que usted tiene que recorrer para llegar al horizonte.

D = (R+N) * ARCCOS(R/(R + H))

D = DISTANCIA HASTA EL HORIZONTE
R = RADIO DE LA TIERRA
N= ALTURA AL NIVEL DEL MAR
H = ALTURA HASTA LOS OJOS.

NOTA: R se incrementa un 20% para compensar la distorsión de refracción de los rayos de luz y llegar a una medición más exacta.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#define  RETIERRA  6378.0
#define  RPTIERRA  6357.0
#define  PERROR  0.20
#define  H_OJOS  0.15
```

ISBN: 978-987-82912-8-4

```
// Objetivo: CALCULAR LA DISTANCIA QUE  TIENE QUE RECORRER PARA LLEGAR AL
//              HORIZONTE       D = (R) * ARCCOS(R/(R + H))

float distancia,h;
char continuar;
 main()
{
printf("DISTANCIA PARA LLEGAR AL HORIZONTE\n");
printf("\nCONSIDERANDO EL RADIO DE LA TIERRA %d KILÓMETROS\n ", RETIERRA);
printf("INGRESE LA ALTURA QUE USTED TIENE: (EN METROS) ");
scanf("%f",&h);
h=(h-H_OJOS)*1000.0;
distancia = (RETIERRA+(RETIERRA*.20))*acos(RETIERRA/(RETIERRA+h));
printf("\nLA DISTANCIA A RECORRES ES:\n" );
printf("\nCON RADIO ECUATORIAL: %6.2f KILOMETROS\n", distancia );
distancia = (RPTIERRA+(RPTIERRA*.20))*acos(RPTIERRA/(RPTIERRA+h));
printf("CON RADIO POLAR     : %6.2f KILÓMETROS\n", distancia );
printf("\n PARA CONTINUAR (X ? ");
scanf(" %c",&continuar);
 }
```

ISBN: 978-987-82912-8-4

2.6 Ejercicios Propuestos

- Escriba un programa en Lenguaje C que permita calcular la masa de aire con la siguiente fórmula: masa = (presión * volumen) / (0.37 * (temperatura + 460)). El ingreso de la masa, presión y volumen son cantidades enteras ingresadas por el usuario.
- En una concesionaria de vehículos se realizaron tres ventas de vehiculos de alta gama a 3 clientes. Cada vehículo cuesta 30000, 29000 y 33000 usd. El gerente desea saber cuál es porcentaje (comisión) que cada vendedor se llevaría, lo que le pagará cada uno de ellos (considerando el 4% por cada vendedor) y lo que le pagarán en conjunto (Total).
- Un entrenador de aeróbicos desea calcular el número de latidos pulsaciones que una persona debe tener por cada minuto de deporte, si la fórmula es: num._pulsaciones = (220 - edad)/60. Desarrolle un programa en lenguaje C que permita realizar esta tarea.
- Un empleado de la empresa “Mi tienda es tuya” recibe un sueldo mensual, por su trabajo ha recibido un incremento del 25% sobre su sueldo anterior. Desarrolle un programa en Lenguaje C que permita calcular el nuevo sueldo del empleado.
- Un grupo de 8 amigos están preparando un viaje a la ciudad de Baños y desean saber el total que pagaran por esta aventura. Tres del grupo realizarán su viaje en taxi ejecutivo. Otros dos en una buseta turística y los tres restantes en bus que cobra 20% menos que el taxi.
- Realice un programa en C que lea un número entero (entrada estándar) positivo y escriba un mensaje (salida estándar) indicando si el número es par o impar.
- Que resultado mostrarán los siguientes programas en C:

```
#include <stdio.h>

Main() {
    int x=3, w=4;
    x -= w++ *3;
    printf(“El valor de x %d\tvalor de w %d\n” , x, w)
    system (“pause”);
    }
```

ISBN: 978-987-82912-8-4

- Que resultado mostrarán los siguientes programas en C:

```
#include <stdio.h>

Main() {
    int x=3, y;
    x = (x * 2 – (y =4, --y);
    printf("El valor de x %d\t valor de y %d\n" , x, y)
    getch ()
    return 0;
    }
```

- Que resultado mostrarán las siguientes expresiones:

Con a=3, b=2, c=5, d=4

a+b*c/d _____

a % 2 % 5 + c _____

(a+c)*d % b _____

(c<65)&&(3==c) _____

a % d % c _____

5+3*c % 5 _____

c/d % 5 _____

((b>d) && (d<a)) _____

ISBN: 978-987-82912-8-4

CAPÍTULO III

INSTRUCCIONES DE CONTROL

3.1. Sentencias Control Selectivas.

Una declaración If (), evalúa la información que le da una expresión booleana (condición). Esta evaluación devuelve uno de dos posibles resultados: verdadero o falso. Si la evaluación es verdadera, el algoritmo continúa en el bloque de instrucciones posterior; si el resultado es falso, la secuencia del algoritmo salta el bloque de instrucciones. (Gil, 2020)

En C, un conjunto de instrucciones debe ejecutarse después de activarse y desactivarse, respectivamente, mediante una condición booleana entre corchetes { }. Si el código que se ejecutará después de la condición consiste en una sola declaración, no se requiere ningún separador: {}. Tenemos tres tipos de estructuras de selección: simple, doble y múltiple.

Selección simple. - Es aquella que después de evaluar una condición booleana determina su valor y ejecuta las instrucciones cuando la condición es verdadera. Cuando el valor es falso saldrá de la estructura de selección. La sintaxis se esta estructura se puede observar en la tabla 3.1.

Tabla 3.1 Selección If

Sintaxis	Ejemplo

If(Condición_boolena)
Instrucción_1;
	prom =(n1+n2+n3) / 3;
	If (prom >=30)
	printf (“Exonerado”);

If(Condición_boolena)
{
Instrucción_1;
Instrucción_2;
Instrucción_3;
}
	Int n, c;
	printf (“Ingres eel numero: \n”);
	c=0;
	scanf(“%i”, &n);
	if(n==2)
	{
	c=c+n;
	printf (“El numero acumulado es: ” ,c);
	}

ISBN: 978-987-82912-8-4

Selección doble. - Es aquella que permite evaluar una condición y elegir entre 2 opciones (verdadera y Falsa). Si la condición booleana es verdadera ejecutará la instrucción que se encuentra a continuación de la estructura if pero si la condición booleana es falsa se ejecutará la instrucción que se encuentra a continuación de la estructura else. La sintaxis se esta estructura se puede observar en la tabla 3.2.

Tabla 3.2 Selección If-else

Sintaxis	Ejemplo

If(Condición_boolena)
Instrucción_1;
else
Instrucción_2;
	prom =(n1+n2+n3) / 3;
	If (prom >=30)
	printf (“Exonerado”);
	else
	printf (“Reprobado”);

If(Condición_boolena)
{
Instrucción_1;
Instrucción_2;
}
else {
Instrucción_3;
Instrucción_4;
}
	Int n, c;
	printf (“Ingres eel numero: \n”);
	c=0;
	scanf(“%i”, &n);
	if(n==2)
	{
	c=c+n;
	printf (“El numero acumulado es:” ,c);
	}
	else {
	c=n-1;
	printf (“El número reducido es:” ,c);
	}

Nota: Debemos tener en cuenta que podemos combinar tanto una condición if simple como una condición doble If-else y anidarla dependiendo de las necesidades que se requiera en cada uno de los problemas. Es decir, puede haber un if dentro de un else y dentro de este if otro else y así indefinidamente. Ejemplo:

```
If(Condición_boolena)
  {
    Instrucción_1;
    Instrucción_2;
  }
  else {
          If(Condición_boolena) {
              Instrucción_3;
                else
              Instrucción_4;
              }
    Instrucción_5;
    Instrucción_6;
  }
```

ISBN: 978-987-82912-8-4

Selección multiple. - Es aquella que permite escoger entre dos o más opciones. La estructura switch valida la opción que se encuentra dentro del paréntesis y el resultado ejecuta el conjunto de instrucciones seleccionada por la opción antes mencionada. (Gil, 2020)

Una vez que se encuentra la igualdad de la expresión, con la expresión en el switch, se realizarán las instrucciones que la estructura lo indique hasta finalizar las instrucciones un break si no encuentra ningún valor en expresión validada, dentro de la estructura switch, se realizará la instrucción asignada por default si éste existe. La sintaxis se esta estructura se puede observar en la tabla 3.3.

Tabla 3.3 Selección Switch – Case

Sintaxis	Ejemplo

Switch (expresión)
{
case 1: instrucción_1;
break;
 	
case 2: instrucción_2;
break;
 	
.
.
case n: instrucción_n;
break;
 	
default:
instrucción n;
 	
}
	Char d;
	printf (“Elige un día de la semana:\n Lunes = L\n Martes = M\n Jueves = J\n”);
	
	scanf(“%c” , &d);
	switch (d) {
	
	case ‘L’ : printf(“dia 1”);
	break
	
	case ‘M’ : printf(“dia 2”);
	break
	
	case ‘J’ : printf(“dia 4”);
	break
	
	default : printf(“dia no existe”);
	}

Se deberá utilizar la palabra reservada break al término de cada caso para interrumpir la estructura y no seguir revisando las demás opciones. La instrucción ***default*** se ejecutará cuando ninguna de las opciones (case) sean escojidas.

3.2. Sentencias Control Repetición While – Do While

En algunos casos, nos encontramos con que hemos escrito demasiadas líneas de código para resolver un problema, y es en estos casos cuando necesitamos usar lo que se llama ciclos repetitivos o bucles que nos permiten ejecutar repetidamente una

ISBN: 978-987-82912-8-4

sección de instrucciones, mejorando el código reduciendo programas excesivamente grandes. (Schildt, 2000)

Hay tres formas de resolver los ciclos o bucles : While, Do-while y For.

Sentencia While.- Un conjunto de sentencias dentro del while se ejecuta mientras la expresión booleana que controla el ciclo while es verdadera. Mientras que la instrucción while se evalúa como una expresión booleana que comprueba si el ciclo es falso, saldrá del ciclo while y continuará la ejecución del programa de manera secuencialmente (Schildt, 2000). La sintaxis se esta estructura se puede observar en la tabla 3.4.

Tabla 3.4 Repetición While

Sintaxis	Ejemplo

contador;
While(expresión_1)
{
instrucción_1;
instrucción_2;
expresión_2;
}
	int I, S;
	I=0; S=0;
	while (I<=5)
	{
	S=S+I;
	I=I+1;
	}
	printf(“La suma es: %d” , S);

Hay que tener en cuenta la variable de control es un contador, donde:

- Contador, valor e inicio de la variable de control de la expresión booleana.
- Expresión_1, es la condición booleana
- Expresión_2, aquí se incrementa o decrementa la variable de control (contador) para la condición booleana.

Sentencia While.- Esta estructura tiene un comportamiento muy particular similar al while pero de distinta forma de ejecutar; la diferencia radica en que realiza una evaluación al inicio y otra al final del ciclo , el uso de las llaves es imprescindible para esta estructura.

Es importante indicar que en la estructura do-while realiza un conjunto de instrucciones y después se evalúa la condición (muy diferente al while que primero evalúa la condición booleana y luego hace el conjunto de instrucciones en su ciclo o bucle repetitivo) esto quiere decir que, si la condicional resulta ser falsa el conjunto de instrucciones se ejecutaron al menos 1 vez (Schildt, 2000). La sintaxis se esta estructura se puede observar en la tabla 3.5.

ISBN: 978-987-82912-8-4

Tabla 3.5 Repetición do-while

Sintaxis	Ejemplo

contador;
do
{
instrucción_1;
instrucción_2;
expresión_2;
} while(expresión_1)
	int I, S;
	I=0; S=0;
	do
	{
	S=S+I;
	I=I+1;
	} while (I<5)
	printf("La suma es: %d" , S);

Hay que tener en cuenta la variable de control es un contador, donde:

- Contador, valor e inicio de la variable de control de la expresión booleana.
- Expresión_1, es la condición booleana
- Expresión_2, aquí se incrementa o decrementa la variable de control (contador) para la condición booleana.

3.3. Sentencias Control Repetición FOR

El bucle for se utiliza mejor para bucles controlados por contadores (repeticiones automaticas), donde un conjunto de declaraciones se ejecuta una vez para cada valor en un rango específico, aunque esta última nota no significa que no pueda incrementarse en 2 para realizar otro incremento o decremento (Schildt, 2000). La sintaxis se esta estructura se puede observar en la tabla 3.6.

Tabla 3.6 Repetición for

Sintaxis	Ejemplo

For (expresión_1, expresión_2, expresión_3)
{
instrucción_1;
instrucción_2;
}
	int i;
	for(i=0;i>5; i++)
	{
	printf("%d" , i);
	}

Hay que tener en cuenta la variable de control es un contador, donde:

- Expresión_1 Nombre e inicio de la variable de control de la expresión booleana.
- Expresión_2, es la condición booleana
- Expresión_3, aquí se incrementa o decrementa la variable de control (contador) para la condición booleana.

ISBN: 978-987-82912-8-4

Las estructuras repetitivas o estructuras de control de repetición pueden ser equivalentes para la solución de muchos problemas, es decir, podemos implementar tanto con while, do-while y for. Una buena práctica es utilizar for cuando la repetición es definida y las otras dos estructuras cuando la repetición es indefinida.

3.4. Ejercicios Resueltos

Ejercicio 3.1. Realice un programa que permita visualizar los caracteres ASCII y su ordinal.

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
/* Objetivo: VISUALIZAR LA TABLA ASCII */
main()
{
int x,y,i,contador, caracter;
gotoxy(30,1);
printf (" VISUALIZA LA TABLA ASCII\n\n");
x=y=3;
for (contador=1; contador<255; contador++)
{
    gotoxy(x,y);
    printf("%3d.%c",contador, caracter);
    Sleep(100);
    caracter++;
    x=x+7;
    if (contador%11==0)
    {
            x=3;
            y++;
     }
 }
printf("\nCUALQUIER TECLA CONTINUAR...");
getch();
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.2. Realizar un programa que efectúe la división entre dos números enteros por medio de restas sucesivas.

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
/* Objetivo: DIVIDIR DOS NÚMEROS ENTEROS POR MEDIO DE RESTAS SUCESIVAS */

main()
{
char continuar='S';
int divisor,dividendo,dividendo1,resultado;
while (continuar=='S')
{
    clrscr();
    printf ("\nDIVISIÓN DE NÚMEROS ENTEROS POR RESTAS SUCESIVAS\n\n");
    printf("\nINGRESE EL DIVIDENDO:");
    resultado=0;
    scanf("%d",&dividendo1);
    dividendo=dividendo1;
    printf("\nINGRESE EL DIVISOR:");
    scanf("%d",&divisor);
     while (dividendo>=divisor)
    {
            dividendo-=divisor;
            resultado++;
     }
    printf("%d / %d = %d\n",dividendo1, divisor,resultado);
    printf("%d MÓDULO %d = %d\n",dividendo1, divisor,dividendo);
    printf("\nREALIZAR OTRA OPERACIÓN (S/N)?");
    continuar=toupper(getch());
  }
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.3. Realice un programa que busque y separe los múltiplos de un número n mientras cuenta del uno al cien.

```
#include <stdio.h>
#include <windows.h>
#include <ctype.h>
#include <conio.h>
#include <stdlib.h>
#define LIMITE 100
/* Objetivo: BUSCAR LOS MÚLTIPLOS DE N EN LOS NÚMEROS LISTADOS DEL
1 AL 100 */
char continuar='S';
int i,y,x,x1,y1,cuenta, multiplo;
main()
{
clrscr();
while (toupper(continuar)=='S')


{
      x1=x=y=5;
      y1=15;
      gotoxy(5,2);
      printf("BUSCA MÚLTIPLOS DE N EN LOS NÚMEROS DEL 1 AL 100\n");
      gotoxy(8,4);

      printf("INGRESE EL VALOR DE N (N>0)=");
      gotoxy(37,4);
      scanf("%d",&multiplo);
      gotoxy(8,y1++);

    printf("_____
    _______");
      gotoxy(3,y1++);
      printf("MÚLTIPLOS DE %d",multiplo);
      gotoxy(8,y1++);
```

ISBN: 978-987-82912-8-4

```
printf("_____
_____");
 for (i=1; i<=LIMITE; i++)
 {
        gotoxy(x=(x+5),y);
        printf("%3d",i);
         if ((i%multiplo)==0)
        {
                gotoxy(x,y);
                gotoxy(x,y);
                printf("=>");
                Sleep(2500);
                gotoxy(x,y);
                printf("  ");
                gotoxy(x1=x1+5,y1);
                printf("%3d",i);
                gotoxy(x1,y1);
                printf("=>");
                Sleep(2500);
                gotoxy(x1,y1);
                printf("  ");
                gotoxy(x,y);
                printf("  ");
                if (x1>70)
                {
                        y1++;
                        x1=5;
                 }
         }
        else
                Sleep(200);
        if(i%10 == 0)
        {
                y++;
                 x=5;
```

ISBN: 978-987-82912-8-4

```
                }
        }
        gotoxy(8,24);

        printf("PARA REPETIR LA EJECUCIÓN (tecla S)=>");
        gotoxy(40,24);
        continuar=getch();
 }
}
```

Ejercicio 3.4. Realice un programa que permita visualizar las tablas de multiplicar del uno al nueve.

```
#include <stdio.h>
#include <windows.h>
#include <ctype.h>
#include <stdlib.h>
#include <conio.h>
#define LIMITE 9
#define LIMITE1 10
// Objetivo: VISUALIZAR LAS TABLAS DE MULTIPLICAR DEL 1 AL 9
char continuar='S';
int i,j,y,resultado;
main()
{
while (toupper(continuar)=='S')
{
 i=1;
 y=5;
 while (i<=LIMITE)
  {
      clrscr();
     gotoxy(2,2);printf("GENERA LAS TABLAS DE MULTIPLICAR DEL 1 AL 9\n");
     gotoxy(8,y);
     printf("TABLA DEL  %d\n",i);
     gotoxy(8,++y);
```

ISBN: 978-987-82912-8-4

```
        printf("_________________\n");
        for (j=1; j<=(LIMITE1);j++)
        {
                resultado=i*j;
                y++;
                gotoxy(8,y);
                printf("%2d * %2d = %2d \n",i,j,resultado);
                Sleep(300);
        }
        i++;
        y=5;
        gotoxy(8,22);
        printf("CONTINUAR (tecla S) ? ");
        gotoxy(35,22);
        continuar=getch();

        if (toupper(continuar)!='S')
                break;
}
gotoxy(8,22);
printf("NUEVA EJECUCIÓN (tecla S)?");
gotoxy(35,22);
continuar=getch();
 }
}
```

Ejercicio 3.5. Realice un programa que determine si un número entero n es o no perfecto. un número perfecto es un entero que es igual a la suma de los divisores positivos menores que él mismo. ejemplo 6 = 1+2+3.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>  /* Objetivo: DETERMINAR SI UN NÚMERO ES PERFECTO */
char continuar='S';
int i,suma,numero;
main()
```

ISBN: 978-987-82912-8-4

```
{
printf("DETERMINA SI UN  NÚMERO ES PERFECTO\n");
while (toupper(continuar)=='S')
{
    suma=0;
    printf("\nINGRESE EL NÚMERO:");
    scanf("%d",&numero);
    printf("\nDIVISOR DEL [%d]\n", numero);
    i=1;
    while (i<numero)
    {
            if(numero%i==0)
            {
                    suma+=i;
                    printf("+%d ",i);
            }
            i++;
    }
    printf( "\nTOTAL = %d", suma);
    if (suma==numero)
            printf("\n\nEL %d SI ES UN NÚMERO PERFECTO", numero);
     else
            printf("\n\nEL %d NO ES UN NÚMERO PERFECTO", numero);
    printf("\n\n UNA NUEVA EJECUCIÓN (S/N) ? ");
    scanf(" %c",&continuar);
 } }
```

Ejercicio 3.6. Calcular el tiempo equivalente en horas minutos y segundos a un número de segundos leído

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
// Objetivo: CALCULA EL EQUIVALENTE EN HORAS MINUTOS Y SEGUNDOS.
```

ISBN: 978-987-82912-8-4

```
 main()
{
char continuar='S';
int segundos, minutos, horas, datoseg;


do
{
    printf("CALCULA EL EQUIVALENTE EN SEGUNDOS, MINUTOS Y HORAS
\n");
    do
    {
            printf("\n INGRESE LOS SEGUNDOS ( entero positivo) = ");
            scanf("%d", &datoseg);
    }
    while(datoseg<0);
    segundos=datoseg;
    horas= segundos/3600;
    segundos=segundos%3600;
    minutos= segundos/60;
    segundos=segundos%60;
    printf("\n AL CONVERTIR LOS %d  SEGUNDOS TENEMOS QUE HAY: \n ",
datoseg);
    printf("\n %d HORAS, %d MINUTOS , %d SEGUNDOS \n", horas, minutos,
segundos);
    printf("\n\ REALIZAR OTRA OPERACIÓN (S/N) ? ");
    scanf(" %c",&continuar);
 }
while(toupper(continuar)=='S');
}
```

Ejercicio 3.7. Realice un programa que visualice los números impares comprendidos entre un intervalo dado por el usuario.

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

ISBN: 978-987-82912-8-4

```
// Objetivo: VISUALIZA LOS NÚMEROS IMPARES COMPRENDIDOS ENTRE UN
INTERVALO
int limite1, limite2;
int i;
char pausa;
main()
{
printf("PRESENTA LOS NÚMEROS IMPARES QUE ESTÉN DENTRO DE UN
RANGO\n");
printf("INGRESE EL LÍMITE INFERIOR: ");
scanf("%d",&limite1);
do
{
    printf("INGRESE EL LÍMITE SUPERIOR: ");
    scanf("%d",&limite2);
    if(limite2<limite1)
            printf("EL LÍMITE SUPERIOR DEBE SER MAYOR QUE %d
        !!!!\n",limite1);
 }
while (limite2<limite1);
for(i=limite1;i<limite2; i++)
    if (fmod(i,2)!=0)
            printf(" %d ",i);
printf("\nUN CARACTER PARA CONTINUAR.. ");
scanf( " %c",&pausa);
}
```

Ejercicio 3.8. Realice un programa que calcule el tanto por ciento de una cantidad ingresado por el usuario.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: CALCULAR EL TANTO POR CIENTO DEL UNA CANTIDAD
float canpor;
float cantidad;
float porcentaje;
```

ISBN: 978-987-82912-8-4

```
char continuar='s';

main()
{
printf("CALCULA EL TANTO PORCIENTO DE UNA CANTIDAD\n");
printf("INGRESE LA CANTIDAD : ");
scanf("%f",&cantidad);
do
    {
    printf("INGRESE EL TANTO PORCIENTO (0..100) : ");
    scanf("%f",&porcentaje);
    }
while (!((porcentaje>=0) && (porcentaje<=100)));
canpor =cantidad*porcentaje/100.0;
printf("\n DE %5.2f EL   %2.2f %% ES %5.2f\n",cantidad,porcentaje,canpor);
printf("\n UNA LETRA PARA CONTINUAR ");
scanf(" %c",&continuar);
}
```

Ejercicio 3.9. Realice un programa que dibuje el contorno de un cuadrado de n caracteres de longitud.

```
#include <ctype.h>
#define MENOR(n)  (n<0)
#define CAR  '='
#define BLANCO ' '
// Objetivo: DIBUJA CUADRADOS DE N CARACTERES POR LADO

int numero, i,j,h=1;
char continuar='s';
main()
{
printf("DIBUJA CUADRADOS DE N CARACTERES\n");
while (toupper(continuar)=='S')
{
```

ISBN: 978-987-82912-8-4

```
    do
    {
            printf("INGRESE EL LADO  (L > 0): ");
            scanf("%d",&numero);
    }
    while (MENOR(numero));

    for(i=1;i<=numero; i++)
    {
            for(j=1;j<=numero; j++)
            {
                    if (((j==1)||(j==numero))||((i==1)||(i==numero)))
                    printf(" %c",CAR);
                    else
                    printf(" %c",BLANCO);
            }
            printf("\n");
      }
    printf("\n\n EJECUTAR        DENUEVO (S/N) ? ");
    scanf(" %c",&continuar);
}
}
```

Ejercicio 3.10. Realice un programa que por medio de menú permita calcular las operaciones aritméticas fundamentales (suma, resta, multiplicación, división y raíz cuadrada).

```
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#define suma(a,b) a+b
#define resta(a,b)a-b
#define multiplicacion(a,b) a*b
#define division(a,b) a/b
#define raiz2(a) sqrt(a)
```

ISBN: 978-987-82912-8-4

```
//objetivo: REALIZAR POR MEDIO DE MENÚ  OPERACIONES ARITMÉTICAS.
main()
{
float a,b, resultado;
char c;
int opcion;
while (opcion!=0)
 {
      clrscr();

     gotoxy(0,0);
     printf("\n\nOPERACIONES ARITMÉTICAS\n");
     printf("_____\n");
     printf(" 1.- SUMA\n");
     printf(" 2.- RESTA\n");
     printf(" 3.- MULTPLICACIÓN\n");
     printf(" 4.- DIVISIÓN\n");
     printf(" 5.- RAÍZ CUADRADA\n");
     printf(" 0.- SALIR\n");
     printf("_____\n");
     printf("   INGRESE LA OPCIÓN:");
     scanf("%d",&opcion);

     if (opcion!=0)
     {
               if (opcion!= 5)
               {
                         gotoxy(40,5);
                         printf("INGRESE EL VALOR DE A =");
                         gotoxy(65,5);
                         scanf ("%f", &a);
                         gotoxy(40,6);
                         printf("INGRESE EL VALOR DE B =");
                         gotoxy(65,6);
                         scanf ("%f", &b);
               }
```

ISBN: 978-987-82912-8-4

```
        else
        {
                gotoxy(40,5);
                printf("INGRESE EL VALOR DE A =");
                gotoxy(65,5);
                scanf ("%f",&a);
        }
}
 switch (opcion)
{
case 1:
        {
                 gotoxy(40,8);
                printf("SU SUMA ES: %6.2f",suma(a,b));
                 break;
        }
case 2:
        {
        gotoxy(40,8);
                printf("SU RESTA ES: %6.2f",resta(a,b));
        break;
        }
case 3:
        {
                gotoxy(40,8);
        printf("SU MULTIPLICACIÓN ES: %6.2f",multiplicacion(a,b));

        break;
        }
case 4:
 {
        gotoxy(40,8);
        printf("SU DIVISIÓN ES: %6.2f",division(a,b));
        break;
        }
        case 5:
```

ISBN: 978-987-82912-8-4

```
                {
                gotoxy(40,8);

                printf("SU RAÍZ CUADRADA ES: %6.2f",raiz2(a));
                break;
                }

        }
        if (opcion!=0)
        {
                 gotoxy(40,12);
                printf("UNA TECLA PARA CONTINUAR....");
                gotoxy(70,12);c=getch();
        }
 }
}
```

Ejercicio 3.11. Realice un programa que muestre cuantos billetes (20, 10, 5 y 1) y monedas (50, 25 10 y 5 centavos) se deben entregar al recibir n centavos de dólar.

```
#include <stdio.h>
#include <ctype.h>
#define B  "BILLETES DE"
#define M  "MONEDAS DE "
#define C  "CENTAVOS DE DÓLAR"
#define D  "DÓLARES "
// Objetivo: CAMBIA N CENTAVOS EN BILLETES (20,10,5,1) Y MONEDAS
(50,25,10,5)

unsigned long int centavos,centavos1;
int b20,b10,b5,b1,m50,m25,m10,m5;
char continuar='s';
main()
{
do
{
```

ISBN: 978-987-82912-8-4

```
printf("CAMBIA CENTAVOS DE DÓLAR A BILLETES Y MONEDAS\n");
printf("\nCUANTOS CENTAVOS DE DÓLAR QUIERE CAMBIAR: ");
scanf("%d",&centavos);
centavos1=centavos;
b20=centavos/(2000);
centavos%=(2000);
b10=centavos/(1000);
centavos%=(1000);
b5=centavos/(500);
centavos%=(500);
b1=centavos/(100);

centavos%=(100);
m50=centavos/(50);
centavos%=(50);
m25=centavos/(25);
centavos%=(25);
m10=centavos/(10);
centavos%=(10);
m5=centavos/(5);
centavos%=(5);
printf("\nDE %d %s SE OBTIENEN:\n\n", centavos1,C);


if (b20!=0)
    printf("%d %s 20 %s\n",b20,B,D);
if (b10!=0)
    printf("%d %s 10 %s\n",b10,B,D);
if (b5!=0)
    printf("%d %s 05 %s\n",b5,B,D);
if (b1!=0)
    printf("%d %s 01 %s\n",b1,B,D);
if (m50!=0)
    printf("%d %s 50 %s\n",m50,M,C);
if (m25!=0)
    printf("%d %s 25 %s\n",m25,M,C);
```

ISBN: 978-987-82912-8-4

```
        if (m10!=0)
            printf("%d %s 10 %s\n",m10,M,C);
        if (m5!=0)
            printf("%d %s  5 %s\n",m5,M,C);
        if (centavos!=0)
            printf("%d %s  1 %s\n",centavos,M,C);
        printf("\n DESEA EJECUTAR NUEVAMENTE (S/N)? ");
        scanf(" %c",&continuar);
    }
    while (tolower(continuar)=='s');
    }
```

Ejercicio 3.12. Realice un programa que encuentre los números menores a n que sean iguales a la suma de los cubos de cada uno de sus dígitos. Ejemplo: 153 = 1^3+5^3+3^3.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <windows.h>
// Objetivo: ENCONTARA LOS NÚMEROS MENORES QUE N QUE LA SUMA DE
LOS
//              CUBOS DE SUS DÍGITOS SEA IGUAL AL NÚMERO
main()
{
char continuar='S';
int j,i,digito,numero,suma;
while (continuar=='S')
{
     clrscr();
     printf ("\nENCUENTRA LOS NÚMEROS MENORES A N TAL QUE: “);
     printf(" \n LA SUMA DE LOS CUBOS DE SUS DÍGITOS DA EL
    NÚMERO\n\n");
     printf("INGRESE EL NÚMERO N =");
     scanf("%d",&numero);
     for (i=1 ; i<numero; i++)
```

ISBN: 978-987-82912-8-4

```
        {
                suma=0;
                 j=i;
                while(j>0)
                {
                        digito=j%10;

                        j=j/10;
                        suma=suma+pow(digito,3);
                 }
                if (suma==i)
                printf("%d ",i);
         }
       printf("\n\nREALIZAR OTRA OPERACIÓN (S/N)?");
       continuar=toupper(getch());
}}
```

Ejercicio 3.13. Realice un programa que una vez leído un número entero positivo calcule el factorial.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: CALCULAR EL FACTORIAL */
int contador,numero;
long int factorial;
char continuar='S';
main()
{

printf("CALCULA EL FACTORIAL DE UN NÚMERO \n");
while (toupper(continuar)=='S')
{
    factorial=1;
    contador=1;
    do
    {
```

ISBN: 978-987-82912-8-4

```
            printf("INGRESE EL NÚMERO (entero positivo) : ");
            scanf("%d",&numero);
         }
        while (numero<0);
       do
       {
                factorial*=contador++;
       }
       while ( contador<=numero);
       printf("EL FACTORIAL DE %d ES %d\n\n",numero,factorial);
       printf("DESEA REPETIR EL CÁLCULO (S) ? ");
       scanf(" %c",&continuar);
}}
```

Ejercicio 3.14. Realice un programa que genere la suma de los n elementos de la serie 2!+ 4! + 6!+ 8! ……

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: CALCULAR LA SUMA LOS N ELEMENTOS DE LA SERIE 2!+
4!+6!+8!... */
int i,contador,numero;
long int factorial,suma;
char continuar='S';
main()
{
printf("SUMA LOS N ELEMENTOS DE LA SERIE 2!+ 4!+6!+8!... \n\n");
while (toupper(continuar)=='S')
{
     contador=1;
     factorial=1;
     suma=0;
     do
      {
               printf("INGRESE EL NÚMERO DE ELEMENTOS DE LA SERIE (entero
              positivo) : ");
```

ISBN: 978-987-82912-8-4

```
                scanf("%d",&numero);
        }
        while (numero<0);
        for( i=0 ;i<=numero; i++)
        {
                do
                {
                        factorial*=contador;
                        if (contador%2==0)
                        {
                                suma=suma+factorial;
                                printf("%4d! = %10d\n",contador, factorial);
                        }
                        contador++;
                }
                while (contador<=numero);
        }
        printf("      ----------\n");
        printf(" TOTAL= %10d",suma);
        printf("\nDESEA REPETIR EL CÁLCULO (S) ? ");
        scanf(" %c",&continuar);
}}
```

Ejercicio 3.15. Realice un programa que genere los n primeros términos de la serie 1,5,3,7,5,9,7 ……

```
#include <stdio.h>
#include <ctype.h>
#define intervalo1  4
#define intervalo2  -2

/* Objetivo:GENERA LOS N TÉRMINOS DE LA SERIE 1,5,3,7,5,9,7 . */

int i,numero;
int termino;
char continuar='S';
```

ISBN: 978-987-82912-8-4

```
main()
{
printf("GENERA LOS N TÉRMINOS DE LA SERIE 1,5,3,7,5,9,7 \n\n");
while (toupper(continuar)=='S')
{
    termino=1;
    do
    {
            printf("INGRESE EL NÚMERO DE ELEMENTOS DE LA SERIE (entero
positivo) : ");
            scanf("%d",&numero);
     }
    while (numero<0);
    printf("_____\n");
    printf(" No.   VALOR\n");
    printf("----------\n");
    for( i=1 ;i<=numero; i++)
     {
              printf(" %3d %4d\n",i,termino);
              if (i%2==0)
                      termino+=intervalo2;
               else
                      termino+=intervalo1;
               }
    printf("\nDESEA REPETIR EL CÁLCULO (S) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 3.16. Realice un programa que genere los n términos de la serie 1,5,3,10,6,16,9,22,13,29,17,…...

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

ISBN: 978-987-82912-8-4

```
/*  Objetivo:GENERA  LOS  N  TÉRMINOS  DE  LA  SERIE
1,5,3,10,6,16,9,22,13,29,17. */
int i,numero,bandera, intervalo1, intervalo2,termino;
char continuar='S';
main()
{
printf("GENERA LOS N TÉRMINOS DE LA SERIE 1,5,3,10,6,16,9,22,13,29,17
\n\n");
while (toupper(continuar)=='S')
{
    intervalo1=4;
    intervalo2=-2;
    termino=1;
    bandera=1;
    do
     {
            printf("CUANTOS ELEMENTOS DE LA SERIE DESEA (N>0) : ");
            scanf("%d",&numero);
     }
    while (numero<=0);
    printf("No.   SERIE \n");
    for( i=1 ;i<=numero; i++)
     {
            printf(" %3d %4d\n",i,termino);
            if (i%2!=0)
            {
                    termino=termino+intervalo1;
                    intervalo1=intervalo1+3;
            }
            else
            {
                    termino=intervalo2+termino;
                    if (bandera)
                     {
                            intervalo2=intervalo2-2;
                            bandera=0;
```

ISBN: 978-987-82912-8-4

```
                    }
                    else
                    {
                            intervalo2=intervalo2-3;
                               bandera=1;
                    }
            }
      }
      printf("\nDESEA REPETIR EL CÁLCULO (S) ? ");
      scanf(" %c",&continuar);
}}
```

Ejercicio 3.17. Realice un programa que genere los n términos de la siguiente serie numérica 0,1,5,4,10,7,15,10,20,13,25 ,16,…

```
#include <stdio.h>
#include <ctype.h>
#define SALTO1 5
#define SALTO2 3

/* Objetivo:GENERA LOS N TÉRMINOS DE LA SERIE
0,1,5,4,10,7,15,10,20,13,25,.. */

int i,numero,bandera,termino, terminor,resultado=0;
char continuar='S';

main()
{
printf("GENERA LOS N TÉRMINOS DE LA SERIE
0,1,5,4,10,7,15,10,20,13,25\n\n");
while (toupper(continuar)=='S')
{
      termino=0;
      terminor=-2;
      bandera=0;
```

ISBN: 978-987-82912-8-4

```
    do
     {
              printf("CUANTOS ELEMENTOS DESEA GENERAR (N mayor que 0):
             ");
              scanf("%d",&numero);
     }
    while (numero<=0);
    printf("SERIE DE %d ELEMENTOS ES: \n", numero);
    for( i=1 ;i<=numero; i++)
     {
              printf(" %3d %4d\n",i,resultado);
              if (bandera)
                {
                        termino=termino+SALTO1;
                        resultado=termino;
                        bandera=0;
                 }
               else
                {
                        terminor=terminor+SALTO2;
                        resultado=terminor;
                        bandera=1;
                }
     }
    printf("\nGENERAR OTROS ELEMENTOS (S/N) ? ");
    scanf(" %c",&continuar);
 }
}
```

Ejercicio 3.18. Realice un programa que encuentre los n primeros términos de la serie de fibonacci.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: GENERAR LOS N PRIMEROS TÉRMINOS DE LA SERIE DE
FIBONACCI */
```

ISBN: 978-987-82912-8-4

```
    char continuar='S';
    int contador,numero, auxiliar;
    int termino1, termino2;
    main()
    {
    printf("GENERA LA SERIE DE FIBONACCI \n\n");
    do
    {
        contador=1;
        termino1=0;
        termino2=1;
        do
        {
            printf("NÚMERO DE TÉRMINOS QUE DESEA CALCULAR (entero
    positivo): ");
            scanf("%d",&numero);

        }
        while (numero<0);
        printf("LOS %d TÉRMINOS SON: ", numero);
        do
        {
            printf("%d ",termino1);
            auxiliar=termino1;
            termino1=termino2;
            termino2=termino2+auxiliar;
         }
        while ( ++contador<=numero);
        printf("\n\n DESEA REPETIR LA EJECUCIÓN (S/N) ? ");
        scanf(" %c",&continuar);
    }
    while (toupper(continuar)=='S');
}
```

Ejercicio 3.19.

ISBN: 978-987-82912-8-4

Ejercicio 3.20. Realice un programa que determine si un número entero ingresado que representa un año es o no bisiesto. (un año es bisiesto si es divisible para 4, con excepción de los años multiplos de 4000).

```
#include <string.h>
#include <ctype.h>
#include <stdio.h>
#define divisible 4
#define multiplo 100
/* Objetivo: DETERMINA SI UN AÑO ES BISIESTO */
char continuar='S';
int anio;
char resultado[3];
main()
{
printf("DETERMINA SI UN AÑO ES BISIESTO \n\n");
while (toupper(continuar)=='S')
{
      do
      {
                printf("INGRESE EL AÑO (NÚMERO +): ");
                scanf("%d",&anio);
      }
      while (anio<0);
      if          ((anio%divisible==0)          &&          (!(anio%multiplo==0))          ||
     (anio%(divisible*multiplo)==0))

                strcpy(resultado,"SI");
      else
                strcpy(resultado,"NO");
                printf("El %d %s ES UN AÑO BISIESTO!!", anio, resultado);
                printf("\n\n DESEA INGRESAR NUEVOS DATOS (S/N) ? ");
                scanf(" %c",&continuar);
     }
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.21. Realice un programa que encuentre nos n primeros términos de la serie 1, 2, 6, 42, 1806…

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: GENERAR LOS N PRIMEROS TÉRMINOS DE LA SERIE 1,2,6,42,1806 */
char continuar='S';
int contador,numero;
float termino;
main()
{
printf("GENERA LA SERIE 1,2,6,42,1806 \n\n");
do
{
    contador=1;
    termino=1;
    do
    {
            printf("NÚMERO DE TÉRMINOS QUE DESEA CALCULAR (entero positivo): ");
            scanf("%d",&numero);
     }
    while (numero<0);
    printf("LOS %d TÉRMINOS SON: ", numero);
    do
    {
            printf("%.0f ",termino);
            termino=termino*(termino+1);
     }
    while ( ++contador<=numero);
    printf("\n\n DESEA REANUDAR LA EJECUCIÓN (S/N) ? ");
    scanf(" %c",&continuar);
}
while (toupper(continuar)=='S');
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.22. Realice un programa que encuentre el promedio de un conjunto de n notas leídas, las mismas que están comprendidas entre 0 y 20.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: ENCUENTRA EL PROMEDIO DE N ELEMENTOS
*/
char continuar='S';
int i,n, total;
int nota;
main()
{
printf("ENCUENTRA EL PROMEDIO DE N NOTAS \n\n");
while (toupper(continuar)=='S')
{
    total=0;
    do
    {
        printf("DE CUANTAS NOTAS DESEA CALCULAR (entero positivo): ");
        scanf("%d",&n);
     }
    while (n<0);
    for (i=0; i<n; i++)
    {
         printf("Ingrese la nota[%d]= ",i+1,nota);
         scanf("%d",&nota);
         total+=nota;
     }
    printf("EL PROMEDIO ES: %.2f", float(total/n));
    printf("\n\nDESEA REANUDAR EL CÁLCULO (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.23. Realice un programa que encuentre el mayor de n números leídos los cuales se encuentran comprendidos entre 100 y 200.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: ENCUENTRA EL MAYOR DE N NÚMEROS LEÍDOS  */
char continuar='S';
int i,n;
float numero, mayor;
main()
{
printf("ENCUENTRA EL MAYOR DE N NÚMEROS LEÍDOS \n\n");
while (toupper(continuar)=='S')
{
    mayor=0;
    do
    {
            printf("CUANTOS NÚMEROS VA A INGRESAR (entero +): ");
            scanf("%d",&n);
     }
    while (n<0);
    for (i=0; i<n; i++)
     {
                do
            {
                printf("El numero[%d] (100=<x<=200) es = ",i+1,numero);
                scanf("%f",&numero);
            }
            while (numero<100 ||NÚMERO >200);
            if (mayor<numero)
                mayor=numero;
    }
    printf("EL MAYOR NÚMERO INGRESADO ES: %.2f", mayor);
    printf("\n\n DESEA INGRESAR NUEVOS DATOS (S/N) ? ");
    scanf(" %c",&continuar);
}
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.24. Realice un programa que calcule la suma de los dígitos de un número entero positivo ingresado por el usuario.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: SUMA LOS DÍGITOS DE UN NÚMERO */
char continuar='S';
int numero, auxiliar,suma;
main()
{
printf("SUMA  LOS DÍGITOS DE UN NÚMERO \n\n");
while (toupper(continuar)=='S')
{
    suma=0;
    do
    {
            printf("INGRESE EL NÚMERO (entero positivo): ");
            scanf("%d",&numero);
     }
    while (numero<0);

    auxiliar=numero;
    while (auxiliar!=0)
    {
            suma+=(auxiliar%10);
            auxiliar/=10;
     }
    printf("DEL NÚMERO %d AL SUMAR SUS DÍGITOS DA %d",numero,suma);
    printf("\n\n REALIZARA NUEVOS CÁLCULOS (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 3.25. Realice un programa que leído un número positivo no mayor a 50000 cuente cuantas veces se repite el dígito x dentro del número.

ISBN: 978-987-82912-8-4

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: DETERMINA CUANTAS VECES UN DÍGITO  SE REPITE EN EL
NÚMERO */
char continuar='S';
int  auxiliar,numero, repite,digito;
main()
{
printf("CUANTAS VECES SE REPITE UN DÍGITO DENTRO DEL NÚMERO \n\n");
while (toupper(continuar)=='S')
{
    repite=0;
    do
    {
            printf("INGRESE UN NÚMERO (0=<X<=50000): ");
            scanf("%d",&numero);
      }
    while (numero<0 || numero>50000);
    do
    {
     printf("INGRESE EL DÍGITO A BUSCAR (0=<D<=9): ");
     scanf("%d",&digito);
    }
    while(digito<0 || digito>9);
    auxiliar=numero;
    while (auxiliar!=0)
     {
            if (auxiliar%10==digito)
                    repite+=1;
            auxiliar/=10;
     }
    printf("EN EL %d EL DÍGITO %1d SE REPITE %d VECES ",numero,digito,
repite);
    printf("\n\n DESEA REALIZAR NUEVOS CÁLCULOS (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.26. Realice un programa que lea un número entero positivo y entregue como resultado cuantas veces se repiten los dígitos 0..9 en el número.

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: DETERMINAR CUANTAS VECES SE REPITEN LOS DÍGITOS EN UN
NÚMERO  */

char continuar='S';
int  auxiliar,numero;
int digito,repite;
main()
{
printf("DETERMINA CUANTAS VECES  LOS DÍGITOS SE REPITEN EN UN
NÚMERO \n\n");
while (toupper(continuar)=='S')
{
    digito=0;
    do
    {
            printf("INGRESE UN NÚMERO (entero positivo): ");
            scanf("%d",&numero);
     }
    while (numero<0);
    while (digito<10)
     {
            repite=0;
            auxiliar=numero;
            while (auxiliar!=0)
            {
                    if (auxiliar%10==digito)
                            repite+=1;
                            auxiliar/=10;
            }
```

ISBN: 978-987-82912-8-4

```
            if (repite!=0)
                printf("EN %d EL %1d SE REPITE %d
        VECES\n",numero,digito,repite);
            digito++;
    }
    printf("\n\n DESEA REALIZAR NUEVOS CÁLCULOS (S/N) ? ");
    scanf(" %c",&continuar);
}
}
```

Ejercicio 3.27. Realice un programa que lea un número entero positivo y lo invierta.

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

/* Objetivo: INVERTIR UN NÚMERO */

char continuar='S';
int  auxiliar,numero;
int invierte;

main()
{
printf("INVIERTE UN NÚMERO \n\n");
while (toupper(continuar)=='S')
{
    invierte=0;
    do
     {
          printf("INGRESE UN NÚMERO (entero positivo): ");
         scanf("%d",&numero);
     }
    while (numero<0);
    auxiliar=numero;
    while (auxiliar!=0)
```

ISBN: 978-987-82912-8-4

```
        {
                invierte+=auxiliar%10;
                auxiliar/=10;
                if (auxiliar!=0)
                        invierte*=10;
         }
        printf("EL NÚMERO INVERTIDO DE %d ES %d \n",numero,invierte);
        printf("\n\n REALIZARA NUEVOS CÁLCULOS (S/N) ? ");
        scanf(" %c",&continuar);
    }
    }
```

Ejercicio 3.28. Realice un programa que lea un texto y lo invierta

```
    #include <stdio.h>
    #include <ctype.h>
    #include <string.h>
    /* Objetivo: INVERTIR UN TEXTO INGRESADO  */
    char continuar='S';
    char  texto[300];
    int  longitud,i=0;
    main()
    {
    printf("INVIERTE UN TEXTO \n\n");
    while (toupper(continuar)=='S')
    {
        printf("INGRESE EL TEXTO : ");
        do
        {
                gets(texto);
        }
        while (texto[0]=='\0');
        i=strlen(texto);
        while (i>=0)
                printf("%c",texto[i--]);
        printf("\n\nDESEA UNA NUEVA EJECUCIÓN (S/N) ? ");
        continuar=getchar();
    }}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.29. Realice un programa que cuente cuantas veces se repiten los caracteres en un texto.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
/* Objetivo: CUENTA EL NÚMERO DE VECES QUE SE REPITE UNA LETRA EN
EL TEXTO */
char continuar='S';
char letra='A';
char  texto[300];
int  cuenta=0,i;
main()
{
printf("NÚMERO DE VECES QUE UNA LETRA SE REPITE\n\n");
while (toupper(continuar)=='S')
{
    letra='A';
    printf("INGRESE EL TEXTO : ");
    do
     {
        gets(texto);
     }
    while (texto[0]=='\0');

    while (letra<='z')
     {
            while (texto[i]!='\0')
           {
                  if (texto[i]==letra)
                  cuenta++;
                  i++;
           }
           if (cuenta!=0)
                  printf("EN EL TEXTO LA  \"%c\" SE REPITE %d VECES\n",
    letra,cuenta);
```

ISBN: 978-987-82912-8-4

```
            letra++;
            cuenta=0;
            i=0;
        }
        printf("\n\n DESEA UNA NUEVA EJECUCIÓN (S/N) ? ");
        continuar=getchar();
    }}
```

Ejercicio 3.30. Realice un programa que calcule el índice de masa corporal (IMC = peso (kg) / talla² (mts)²) de una persona e indique el estado en el que se encuentra. tomando en cuenta que:

- **un IMC de menos de 18,5 se considera por debajo del peso normal.**
- **un IMC situado entre 18,5 y 24,9 corresponde a un peso normal y con un peso dentro de lo que se considera saludable.**
- **un IMC entre 25,0 y 29,9 indica un claro sobrepeso.**
- **un IMC entre 30,0 y 39,9 corresponde a lo que se considera obesidad.**
- **un IMC de 40,0 o mayor es lo propio de una obesidad severa o mórbida.**

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <conio.h>
// Objetivo: CALCULAR EL INDICE DE GRASA CORPORAL DE UNA PERSONA
float estatura;
float peso;
float imc;
char continuar='S';
main()
{
while (toupper(continuar)=='S')
{
        clrscr();
        printf("CALCULA EL INDICE DE MASA CORPORAL DE UNA PERSONA\n\n");
```

ISBN: 978-987-82912-8-4

```
        do
        {
                printf("INGRESE SU ESTATURA (METROS) : ");
                scanf("%f",&estatura);
        }

        while(estatura<=0);
        do
        {
                printf("INGRESE SU PESO (Kg): ");
                scanf("%f",&peso);
        }
        while(estatura<=0);
        imc=peso/pow(estatura,2);
        printf("\nSU INDICE DE MASA CORPORA ES %5.3f\n", imc);
        printf("\nUSTED SE ENCUENTRA CON ");
        if (imc<18.5)
                printf("BAJO EN PESO");
        if (imc>=18.5 && imc<25.0)
                printf("UN PESO ADECUADO");
        if (imc>=25.0 && imc<30.0)
                printf("SOBRE PESO");
        if (imc>=30.0 && imc<40.0)
                printf("OBESIDAD");
        if (imc>=40.0)
                printf("OBESIDAD SEVERA");
        printf("\n\nINGRESAR NUEVOS DATOS(S/N)?");
        continuar=getch();
}
}
```

Ejercicio 3.31. Realice un programa que calcule el número de calorías diarias mínimas requeridas por una persona conociendo que:

Si es mujer: [655+(9.6*peso-kg)+(1.8*altura-cm)-(4.7 * edad)] * f. de actividad.

ISBN: 978-987-82912-8-4

Si es hombre: [66+(13.7*peso-kg)+(5*altura-cm) - (6.8 * edad)] * f. de actividad

el factor de actividad es:

- **Si es una persona sedentaria es 1.2**
- **Si hace deporte de 1 a 3 veces por semana es 1.375.**
- **Si hace deporte de 3 a 5 veces por semana es 1.55**
- **Si hace deporte de 6 a 7 veces por semana es 1.725**
- **Si entrena varias horas casi todos los días es 1.9**

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

// Objetivo: CALCULAR EL REQUERIMIENTO DIARIO DE CALORIAS DE
UNA PERSONA

int edad,opción, estatura;
float peso, calorias, factividad;
char continuar='S',sexo;

main()
{
while (toupper(continuar)=='S')
{
          clrscr();
          printf("CALCULAR EL REQUERIMIENTO DIARIO DE CALORIAS DE
         UNA PERSONA\n\n");
          printf("INGRESE SU EDAD (AÑOS): ");
          scanf("%d",&edad);
          printf("INGRESE SU ESTATURA (EN CM) : ");
          scanf("%d",&estatura);
          printf("INGRESE SU PESO (KILOGRAMOS): ");
          scanf("%f",&peso);
          do
```

ISBN: 978-987-82912-8-4

```
{
        printf("INGRESE SU SEXO (M/F): ");
        scanf(" %c",&sexo);
 }
 while (toupper(sexo)!='M' && toupper(sexo)!='F');

 do
 {
        printf(" A LA SEMANA USTED HACE DEPORTE : \n\n");
        printf(" 1. 0 VECES\n");
        printf(" 2. DE 1 A 3 VECES\n");
        printf(" 3. DE 3 A 5 VECES\n");
        printf(" 4. DE 6 A 7 VECES\n");
        printf(" 5. VARIAS HORAS CASI TODOS LOS DÍAS\n\n");
        printf("    ESCOJA SU OPCIÓN:");
        scanf("%d",&opcion);
 }
 while (!(opcion>=1 && opcion<=5));
 switch (opcion)
  {
  case 1: {factividad=1.2; break; }
  case 2: {factividad=1.375;break;}
  case 3: {factividad=1.55;break; }
  case 4: {factividad=1.725;break;}
  case 5: {factividad=1.9; break; }
  }
 switch (toupper(sexo))
  {
  case 'M':{calorias=(66+(13.7*peso)+(5*estatura)-(6.8*edad))*factividad;
break; }
  case 'F':{calorias=(655 +
(9.6*peso)+(1.8*estatura)+(4.7*edad))*factividad; break; }
  }
 printf("\nUSTED REQUIERE CONSUMIR COMO MÍNIMO %5.2f
CALORIAS DIARIAS\n\n",calorias);
 printf("\nINGRESAR NUEVOS DATOS(S/N)?");
```

ISBN: 978-987-82912-8-4

```
            continuar=getch();
    }}
```

Ejercicio 3.32. Realice un programa que permita determinar si una palabra es un palíndromo (se designa a aquellas palabras que pueden leerse tanto de izquierda a derecha como de derecha a izquierda). ej: oso, ana, ojo.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

/* Objetivo: DETERMINA SI UNA PALABRA ES UN PALÍNDROMO */

char continuar='S';
char  palabra[20], palabra1[20],resultado[3];
int  longitud,i=0,j=0;

main()
{
printf("DETERMINA SI UNA PALABRA ES UN PALÍNDROMO \n\n");

while (toupper(continuar)=='S')
{
    j=0;
    printf("INGRESE LA PALABRA : ");

    do
     {
            gets(palabra);
     }
    while (palabra[0]=='\0');

    i=strlen(palabra);

    while (i>=0)
```

ISBN: 978-987-82912-8-4

```
        {
                palabra1[j]=palabra[--i];
                j++;
        }

        palabra1[--j]='\0';

        if (strcmp(palabra,palabra1)==0)
                strcpy(resultado,"SI");
        else
                strcpy(resultado,"NO");

        printf("\n%s ES UN PALÍNDROMO %s == %s",resultado, palabra,palabra1);
        printf("\n\n REALIZAR UNA NUEVA EJECUCIÓN (S/N) ? ");
        continuar=getchar();
}
}
```

Ejercicio 3.33. Realice un programa que ingresado un número comprendido entre 1 y 7 entregue como resultado el día de la semana al cual representa.

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

/* Objetivo: ESCRIBIR EL DÍA DE LA SEMANA

char continuar='S';
char  pdia[13], orden[]="   ";
int   dia;

main()
{
printf("ESCRIBE EL DÍA DE LA SEMANA \n\n");
```

ISBN: 978-987-82912-8-4

```
while (toupper(continuar)=='S')
{
    printf("INGRESE EL NÚMERO DEL DÍA (1<=X<=7) : ");
    scanf("%d",&dia);

    switch (dia)
    {
    case 1:
            {strcpy(pdia," ES LUNES"); strcpy(orden,"er."); break; }
    case 2:
            {strcpy(pdia,"ES MARTES");strcpy(orden,"do."); break; }
    case 3:
            {strcpy(pdia,"ES MIÉRCOLES");strcpy(orden,"er."); break; }
    case 4:
            {strcpy(pdia,"ES JUEVES"); strcpy(orden,"to.");break; }
    case 5:
            {strcpy(pdia,"ES VIERNES");strcpy(orden,"to."); break; }
    case 6:
            {strcpy(pdia,"ES SÁBADO"); strcpy(orden,"to.");break; }
    case 7:
            {strcpy(pdia,"ES DOMINGO");strcpy(orden,"mo."); break; }
    default:{strcpy(pdia,"NO !!EXISTE!!"); break; }
     }

    printf("\nEL %d%s DÍA DE LA SEMANA %s",dia,orden,pdia);
    printf("\n\nDESEA INGRESAR NUEVOS DATOS (S/N) ? ");
    scanf(" %c",continuar);
    strcpy(orden,"  ");
}
}
```

Ejercicio 3.34. Realice un programa que indique en palabras cuantas unidades, decenas, centenas, etc. que tiene un número ingresado por el usuario.

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

ISBN: 978-987-82912-8-4

```
#include <stdlib.h>
/* Objetivo: DESCRIBE UN NÚMERO DESCOMPONIÉNDOLO EN UNIDADES, DECENAS, CENTENAS, ETC */
char continuar='S';
int numero, auxiliar,digito,i=0;
main()
{
printf("DESCOMPONE UN NÚMERO EN UNIDADES, DECENAS, CENTENAS, ETC. \n\n");
while (toupper(continuar)=='S')
{
    i=0;
    printf("INGRESE UN NÚMERO: ");
    scanf("%d",&numero);
    auxiliar=abs(numero);
    printf("EL NÚMERO %d TIENE: \n",numero);
    while (auxiliar!=0)
    {
            digito= auxiliar%10;
            auxiliar/=10;
            switch (digito)
            {
             case 0: {printf("CERO "); break; }
            case 1: {printf("UNA "); break; }
            case 2: {printf("DOS "); break; }
            case 3: {printf("TRES "); break; }
            case 4: {printf("CUATRO "); break; }
            case 5: {printf("CINCO "); break; }
            case 6: {printf("SEIS "); break; }
              case 7: {printf("SIETE "); break; }
             case 8: {printf("OCHO "); break; }
             case 9: {printf("NUEVE "); break; }
              }
            switch (i%3)
            {
            case 0: {printf("UNIDADES "); break; }
```

ISBN: 978-987-82912-8-4

```
                case 1: {printf("DECENAS "); break; }
                case 2: {printf("CENTENAS "); break; }
                 }
                if ((i<6) &&(i>=3))  printf("MIL ");
                if(i>=6)  printf("MILLON ");
                 i++;
                 printf("\n");
        }
         printf("\n\n REALIZAR UNA NUEVA EJECUCIÓN (S/N) ? ");
         scanf(" %c",&continuar);
}}
```

Ejercicio 3.35. Realice un programa que leído un conjunto de caracteres los ordene en forma alfabético.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

/* Objetivo: ORDENA ALFABÉTICAMENTE UN TEXTO */

char aux,continuar='S';
char texto[200];
int i=0,j=0;

main()

{
printf(" ORDENA ALFABÉTICAMENTE LOS CARACTERES DE UN TEXTO\n\n");
while (toupper(continuar)=='S')
{
     i=0;
     do
      {
                printf("INGRESE EL TEXTO: ");
```

ISBN: 978-987-82912-8-4

```
            gets(texto);
        }
        while (texto[0]=='\0');

        for (i=0; i<=strlen(texto); i++)
                for (j=i; j<strlen(texto); j++)
                        if (texto[i]>texto[j])
                {
                        aux=texto[i];
                        texto[i]=texto[j];
                        texto[j]=aux;
                }

        texto[j]='\0';
        printf("LOS CARACTERES ORDENADOS SON: %s",texto );
        printf("\n\n REALIZAR UNA NUEVA EJECUCIÓN (S/N) ? ");
         scanf(" %c",&continuar);
}
}
```

Ejercicio 3.36. Realice un programa que calcule el máximo común divisor de dos números dados.

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

/* Objetivo: CALCULA EL MÁXIMO COMÚN DIVISOR DE DOS NÚMEROS
*/

char continuar='S';
int auxiliar,menor=0,mayor=0,numero1, numero2,i=0;

main()
{
```

ISBN: 978-987-82912-8-4

```
printf("CALCULA EL MÁXIMO COMÚN DIVISOR DE DOS NÚMEROS\n");

while (toupper(continuar)=='S')
{
    i=0;
    printf("\n INGRESE DOS NÚMEROS: ");
    scanf("%d %d",&numero1, &numero2);
    numero1=abs(numero1);
    numero2=abs(numero2);

    if (numero1>numero2)
    {
            menor=numero2;
            mayor=numero1;
     }
    else
     {
            menor=numero1;
            mayor=numero2;
     }

    while (mayor%menor!=0)
     {
             auxiliar=mayor;
            mayor=menor;
            menor=auxiliar%menor;
     }
    printf("EL MCD(%d,%d) = %d", numero1,numero2,menor);
    printf("\n\n REALIZAR UNA NUEVA EJECUCIÓN (S/N) ? ");
     scanf(" %c",&continuar);
}
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.37. Realice un programa que descomponga en factores un número entero dado por el usuario.

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

/* Objetivo: DESCOMPONE EN FACTORES UN NÚMERO
*/

char continuar='S';
int contador,numero1, numero2,i=0;

main()
{
printf("DESCOMPONE EN FACTORES UN NÚMERO\n");
while (toupper(continuar)=='S')
{
    i=2;
    printf("\n INGRESE EL NÚMERO: ");
    scanf("%d",&numero1);
    printf("\n\n  %d = ", numero1);

    while(numero1>1)
     {
            contador=0;
            while (numero1%i==0)
            {
                    numero1/=i;
                    contador++
            }
            if (contador!=0)
                    printf(" %2d^(%2d) ",i,contador);
                    i++;
     }
    printf("\n\n REANUDAR EJECUCIÓN  (S/N) ? ");
```

ISBN: 978-987-82912-8-4

```
    scanf(" %c",&continuar);
}
}
```

Ejercicio 3.38. Realice un programa que calcule el mínimo común múltiplo de dos números dados.

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

/* Objetivo: CALCULA EL MÍNIMO COMÚN MÚLTIPLO DE DOS NÚMEROS
*/

char continuar='S';
int mcm,numero1, numero2,i=0;

main()

{
printf("CALCULA EL MÍNIMO COMÚN MÚLTIPLO DE  DOS NÚMEROS\n");
while (toupper(continuar)=='S')
{
i=2;
mcm=1;
printf("\n INGRESE DOS NÚMEROS: ");
scanf("%d %d",&numero1, &numero2);

while(numero1>1 || numero2>1)
{
    while (numero1%i==0 || numero2%i==0)
    {
            mcm=mcm*i;
            if (numero1%i==0)
                    numero1/=i;
            if (numero2%i==0)
```

ISBN: 978-987-82912-8-4

```
                    numero2/=i;
        }
        i++;
        }
        printf("EL mcm(%d,%d) = %d", numero1,numero2,mcm);
        printf("\n\n REANUDAR LA EJECUCIÓN (S/N) ? ");
         scanf(" %c",&continuar);
    }
    }
```

Ejercicio 3.39. Realice un programa que determine si dos números son amigos (dos números son amigos cuando cada uno es igual a la suma de los divisores del otro ejemplo 220 y 284)

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

/* Objetivo: DETERMINAR SI DOS NÚMEROS SON AMIGOS
*/

char continuar='S';
int suma1,suma2,numero1, numero2,i=0;

main()
{
printf("DETERMINA SI DOS NÚMEROS SON AMIGOS\n");

while (toupper(continuar)=='S')
{
    suma1=suma2=0;
    printf("\n INGRESE DOS NÚMEROS: ");
    scanf("%d %d",&numero1, &numero2);
    printf( "\n\nLOS DIVISORES DEL [%d] SON :\n\n", numero1);
```

ISBN: 978-987-82912-8-4

```
    for (i=1; i<numero1; i++)
            if (!(numero1%i))
            {
                    suma1+=i;
                    printf( "+%d",i);
            }

    printf( "= %d", suma1);
    printf( "\n\nLOS DIVISORES DEL [%d] SON :\n\n", numero2);

    for (i=1; i<numero2; i++)
            if (!(numero2%i))
            {
                    suma2+=i;
                    printf( "+%d",i);
            }

    printf( "= %d", suma2);

    if ((suma1==numero2) && (suma2==numero1))
             printf("\n\n EL %d Y EL %d SI SON NÚMEROS AMIGOS",
            numero1,numero2);
    else
             printf("\n\n EL %d Y EL %d NO SON NÚMEROS AMIGOS",
            numero1,numero2);
    printf("\n\n\nREANUDAR LA EJECUCIÓN (S/N) ? ");
     scanf(" %c",&continuar);
}}
```

Ejercicio 3.40. Realice un programa que calcule la media armónica y geométrica de n elementos.

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

ISBN: 978-987-82912-8-4

```
/* Objetivo: CALCULAR LA MEDIA GEOMÉTRICA Y ARMÓNICA DE N
ELEMENTOS
*/

char continuar='S';
int i,n;
float numero, mediageo, mediarm;

main()

{
printf("CALCULA LA MEDIA GEOMÉTRICA Y ARMÓNICA \n\n");

while (toupper(continuar)=='S')
{
    mediageo=1.0;
    mediarm=0;
    do
    {
            printf("EL NÚMERO DE ELEMENTOS A INGRESAR SON (entero +):
           ");
            scanf("%d",&n);
     }
    while (n<0);

    for (i=0; i<n; i++)
    {
            printf("INGRESE EL ELEMENTO[%d] = ",i+1,numero);
            scanf("%f",&numero);
            mediageo*=abs(numero);
            mediarm+= (1.0/numero);
     }
    mediageo=pow(mediageo,float(1.0/n));
    mediarm=n/mediarm;
    printf("LA MEDIA GEOMÉTRICA ES %.3f", mediageo);
```

ISBN: 978-987-82912-8-4

```
        printf("\n LA MEDIA ARMÓNICA    ES %.3f", mediarm);
        printf("\n\n DESEA INGRESAR NUEVOS DATOS (S/N) ? ");
        scanf(" %c",&continuar);
    }
    }
```

Ejercicio 3.41. Realice un programa que transforme un número entero decimal a su equivalente en binario.

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <string.h>
#define CERO "0"
#define UNO "1"

/* Objetivo: TRANSFORMAR UN NÚMERO ENTERO POSITIVO EN DECIMAL A
BINARIO. */

char continuar='S';
char cadena[10],cadena1[10];
int i,j,numero,auxiliar,contador,cuenta, cuenta2;
int binario;

main()
{
printf("TRANSFORMA UN NÚMERO DECIMAL A BINARIO \n\n");

while (toupper(continuar)=='S')
{
    strcpy(cadena,"\0");
    strcpy(cadena1,"\0");
    do
    {
            printf("INGRESE UN NÚMERO DECIMAL (entero +)= ");
           scanf(" %d",&numero);
```

ISBN: 978-987-82912-8-4

```
        }

    while (numero<0);
    auxiliar=numero;
    binario=0;

    while (numero!=0)
    {
            binario=(numero%2);
            numero=numero/2;

            if (binario)
                        strcat(cadena,UNO);
            else
                    strcat(cadena,CERO);
    }
    i=strlen(cadena);
    j=0;

    while (i>=0)
    {
            cadena1[j]=cadena[--i];
            j++;
    }
    cadena1[--j]='\0';
    printf("%d (DECIMAL) = %s (BINARIO)\n",auxiliar,cadena1);
    printf("\n\n DESEA TRANSFORMAR OTRO NÚMERO (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 3.42. Realice un programa que codifique en bcd 8421 un número decimal

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

ISBN: 978-987-82912-8-4

```
#define DIGITO(a) (a%10)
// Objetivo: CONVIERTE UN NÚMERO DECIMAL A BCD
int  decimal1,decimal, numero;
char pausa,continuar='S', bcd[80];
main()
{
printf("CONVIERTE UN NÚMERO DECIMAL A BCD\n");
while (toupper(continuar)=='S')
{
     numero=0;
    strcpy(bcd,"\0");
    do
    {
            printf("\n INGRESE El NÚMERO (entero positivo): ");
            scanf("%d",&decimal1);
    }
    while (decimal1<0);
    decimal=decimal1;
     while (decimal!=0)
    {
                    numero=numero+decimal%10;
                    decimal/=10;
                    if (decimal>0)
                            numero=numero*10;
            }
    while (numero>0)
    {
            switch DIGITO(numero)
            {
            case 0: {strcat(bcd,"0000 "); break; }
            case 1: {strcat(bcd,"0001 "); break; }
            case 2: {strcat(bcd,"0010 "); break; }
            case 3: {strcat(bcd,"0011 "); break; }
            case 4: {strcat(bcd,"0100 "); break; }
            case 5: {strcat(bcd,"0101 "); break; }
            case 6: {strcat(bcd,"0110 "); break; }
```

ISBN: 978-987-82912-8-4

```
                case 7: {strcat(bcd,"0111 "); break; }
                case 8: {strcat(bcd,"1000 "); break; }
                case 9: {strcat(bcd,"1001 "); break; }
                        }
                        numero/=10;
        }
        printf("EL %d EN CÓDIGO BCD ES =  %s", decimal1,bcd);
        printf("\n\n REANUDAR LA EJECUCIÓN (S/N) ? ");
        scanf(" %c",&continuar);
}}
```

Ejercicio 3.43. Escriba un programa que tome un número entero, de hasta cuatro cifras y lo tranforme en números romanos.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define LIMITE 1000

// Objetivo:  TRANSFORMA UN NÚMERO A ROMANOS

main()
{
char continuar='S';
int i,n,r,n1,j;
printf("TRANSFORMA UN NÚMERO A ROMANOS \n");
do
{
      do
      {
            printf("\n INGRESE EL NÚMERO ( 0<N>4000)? ");
            scanf("%d",&n);
      }
      while (n<=0 || n>=4000);
      n1=n;
      i=LIMITE;
```

ISBN: 978-987-82912-8-4

```
while (n1!=0)
{
        r=n1/i;
        switch (i)
        {
        case 1000:
        {
                        for (j=1; j<=r; j++)
                        printf("M");
                        break;
                }
        case 100:
                {
                        switch (r)
                {
                case 4:printf("CD");r=0; break;
                case 5:
                    case 6:
                    case 7:
                    case 8:printf("D"); r=abs(r-5); break;
                    case 9:printf("CM"); r=0; break;
                    }
                        for (j=1; j<=r; j++)
                         printf("C");
                        break;
                }

        case 10:
                {
                         switch (r)
                 {
                  case 4:printf("XL");r=0; break;
                      case 5:
                      case 6:
                      case 7:
                      case 8:printf("L"); r=abs(r-5); break;
```

ISBN: 978-987-82912-8-4

```
                    case 9:printf("XC"); r=0; break;

                }
                    for (j=1; j<=r; j++)
                    printf("X");
                    break;
            }
            case 1:
             {

                    switch (r)
                {
                case 4:printf("IV");r=0; break;
                   case 5:
                   case 6:
                   case 7:
                   case 8:printf("V"); r=abs(r-5); break;
                case 9:printf("IX"); r=0; break;
                    }
                    for (j=1; j<=r; j++)
                     printf("I");
                    break;
                }

            }
            n1%=i;
            i/=10;
        }
        printf("\n\nEJECUTAR NUEVAMENTE EL PROGRAMA (S/N)? ");
        scanf(" %c",&continuar);
}
while(toupper(continuar)=='S');
}
```

ISBN: 978-987-82912-8-4

Ejercicio 3.44. Realice un programa que genere el plural de una palabra ingresada por el usuario.

```
#include <stdio.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

/* Objetivo: DETERMINA EL PLURAL DE UNA PALABRA */

char continuar='S',caracter;
char  palabra[20], palabra1[20];
int  longitud,i=0,j=0;

main()
{
printf("DETERMINA EL PLURAL DE UNA PALABRA \n\n");

while (toupper(continuar)=='S')
{
    j=0;
    printf("INGRESE LA PALABRA : ");

    do
    {
            gets(palabra);
     }
    while (palabra[0]=='\0');

    strcpy(palabra1,strlwr(palabra));
    strcpy(palabra1,palabra);
    caracter=(palabra[strlen(palabra)-1]);

    if((caracter=='a')||(caracter=='e')||(caracter=='i')||(caracter=='o')||(caracter=='u'))
```

ISBN: 978-987-82912-8-4

```
                strcat(palabra,"s\0");
        else
        {
                if(caracter=='z')
                        palabra[strlen(palabra)-1]='c';
                strcat(palabra,"es\0");
        }

        printf("\n El plural de %s es %s",palabra1,palabra);
        printf("\n\n REALIZAR UNA NUEVA EJECUCIÓN (S/N) ? ");
        continuar=getchar();
    }
    }
```

Ejercicio 3.45. Realice un programa que imprima un triángulo de la forma:

N=9

4
3 4
2 3 4
1 2 3 4
0 1 2 3 4
1 2 3 4
2 3 4
3 4
4

N=3

1
0 1
1

```
#include <math.h>
#include <ctype.h>
#define PAR(n)  (n%2)

// Objetivo: IMPRIMIR UN TRIÁNGULO DE NÚMEROS DE N LÍNEAS IMPARES
```

ISBN: 978-987-82912-8-4

```
int numero;
int i,j,h=1;
char continuar='s';

main()
{
printf("IMPRIME TRIÁNGULO DE NÚMEROS\n");
while (toupper(continuar)=='S')
{
        do
        {
                printf("INGRESE EL NÚMERO DE LÍNEAS  (IMPAR): ");
                scanf("%d",&numero);
        }
        while (!PAR(numero));
        for(i=0;i<=numero; i++)
        {
                for(j=abs(numero/2-i);j<=(numero/2); j++)
                {
                        if (j==(numero/2))
                                printf(" %d",j);
                        if (j>(numero/2))
                                printf(" %d",abs(numero-j));
                        if (j<(numero/2))
                                printf(" %d",j);
                }
                printf("\n");
          }
        printf("\n\n DESEA UNA NUEVA EJECUCIÓN (S/N) ? ");
        scanf(" %c",&continuar);
}
}
```

Ejercicio 3.46. Realizar un programa que calcule las n primeras líneas del triángulo de pascal

ISBN: 978-987-82912-8-4

```
#include <stdio.h>
#include <ctype.h>
#define MENOR(n)  (n<0)

// Objetivo: OBTIENE LAS N PRIMERAS LÍNEAS DEL TRIÁNGULO DE PASCAL

int numero;
int i,j,f1,f2,f3,h;
char continuar='s';

main()
{
printf("GENERA LAN N LÍNEAS DEL TRIÁNGULO DE PASCAL\n");
while (toupper(continuar)=='S')
{
    do
    {
            printf("INGRESE EL NÚMERO DE LÍNEAS (N > 0): ");
            scanf("%d",&numero);
     }
    while (MENOR(numero));
    for(i=0;i<=numero; i++)
    {
            for(j=0;j<=i; j++)
            {
                    f1=f2=f3=1;
                     for(h=1;h<=i; h++)
                            f1=f1*h;
                    for(h=1;h<=j; h++)
                            f2=f2*h;
                    for(h=1;h<=i-j; h++)
                            f3=f3*h;
                     printf(" %2d",f1/(f2*f3));
            }
            printf(" \n");
     }
```

ISBN: 978-987-82912-8-4

```
        printf("\n\n DESEA EJECUTAR NUEVAMENTE EL PROGRAMA (S/N) ? ");
        scanf(" %c",&continuar);
    }
    }
```

ISBN: 978-987-82912-8-4

3.5. Ejercicios Propuestos

- Un estudiante que cursa una asignatura y ha obtenido su calificación por cada uno de los tres parciales durante el semestre. Escriba un programa en Lenguaje C que permita determinar si el estudiante aprobó el curso.
- Dado un numero (Ingresado por el usuario). Escriba un programa en Lenguaje C que permita determinar si el número es par o impar.
- El año bisiesto es una característica que ocurre cada 4 años en donde el movimiento de traslación de la tierra dura un día mas, teniendo 366 días. Escriba un programa en Lenguaje C que permita determinar si un año es bisiesto o no.
- Escriba un programa en Lenguaje C que ingresando un mes cualesquiera, permita escribir el número de mes que le corresponde, ejemplo Abril …4to mes.
- Escriba un programa en Lenguaje C que permita realizar una de las operaciones básicas de una calculadora: suma resta, multiplicación y división.
- En un servicio de consulta de fecha de nacimiento se le brinda la opción de saber su signo zodiacal. Escriba un programa en Lenguaje C que permita mostrar el signo zodiacal de una persona a partir de la fecha.
- Dado una cantidad (número de elemento, n, de una serie). Escriba un programa en Lenguaje C que permita imprimir los n elementos de la serie de Fibonacci.
- Escriba un programa en Lenguaje C que permita determinar el producto de varios números enteros (ingresados por el usuario). Esta acción se la debe realizar tantas veces lo pida el usuario.
- Escriba un programa en Lenguaje C que permita determinar sumar todos los números impares y los reste de la sumatoria de los impares hasta que la suma sea igual a 75 en los impares y 50 en los pares.
- Escriba un programa en Lenguaje C que permita crear la tabla de multiplicar de un numero cualquiera. La cantidad máxima de números los ingresa el usuario.
- En una gasolinera se despacha dos tipos de productos con un costo por galón de: extra $ 2.40 y super $ 3.90. Escriba un programa en Lenguaje C

ISBN: 978-987-82912-8-4

que permita saber el pago a realizar según el consumo en litros de gasolina despachados.

- En una escuela primaria se enseña el abecedario, para mejorar el aprendizaje la profesora ha solicitarlo aprenderlo de la última letra a la primera. Escriba un programa en Lenguaje C que permita imprimir de forma inversa las letras del abecedario.
- Se desea generar un cuadro y/o marco en la pantalla utilizando el símbolo "@". Escriba un programa en Lenguaje C que permita imprimir ese cuadro utilizando coordenadas ingresadas por el usuario.
- Escriba un programa en Lenguaje C que permita determinar los 20 primeros números primos.
- Escriba un programa en Lenguaje C que permita determinar los primeros 10 elementos de las siguientes series y calcule el resultado:
    - 1+3+5+…+n.
    - 1*3*5*…*n
    - 1,4,9,16,25,36,49
    - 4,7,10,13,…
    - 8, 16,32,64,…
    - 7,13,24,45,…

- Dado un vector x de n elementos reales, donde n es impar, diseñar una función que calcule y devuelva la mediana de ese vector. La mediana es el valor tal que la mitad de los números son mayores que el valor y la otra mitad son menores.

- Se trata de resolver el siguiente problema escolar. Dadas las notas de los alumnos de un colegio en el primer curso de bachillerato, en las diferentes asignaturas (5, por comodidad), se trata de calcular la media de cada alumno, la media de cada asignatura, la media total de la clase y ordenar los alumnos por orden decreciente de notas medias individuales.

- Escribir un algoritmo de consulta de teléfonos. Leer un conjunto de datos de mil nombres y números de teléfono de un archivo que contiene los números en orden aleatorio. Las consultas han de poder realizarse por nombre y por número de teléfono.

ISBN: 978-987-82912-8-4

CAPÍTULO IV

ESTRUCTURAS Y FUNCIONES DE UN PROGRAMA

4.1. Arrays Unidimensionales

Se suele hablar de listas o vectores cuando sólo se usa una dimensión, cuando se usan dos, de matrices. Un Array es una colección de datos del mismo tipo, que se almacena en posiciones consecutivas de memoria y recibe un nombre común. Para referirse a un determinado elemento de un array se deberá utilizar un índice, que especifique su posición relativa en el array. (Seacord, 2020)

Todo identificador debe ser declarado, los Array no son una excepción y lo primero que se debe hacer es crear el tipo, para luego poder declarar datos de dicho tipo, además de agregar el número de elementos que contendrá. La sintaxis se esta estructura se puede observar en la tabla 4.1.

Tabla 4.1 Estructura Array-Vector

Sintaxis	Ejemplo
<tipo dato> <identificador> [<núm_elemen>];	Int prueba [6]; Char lista [30]; int num[5], sueldo[10]; float precio[100], IVA[100]; float cuotas[max]; char nombre[20], apellidos[30]; char letras[26];

Hay que tener en cuenta que en el lenguaje de programación C la numeración de los subíndices inicia en cero pero sus elementos es más uno, es decir, si tenemos un vector por ejemplo de 5 elementos su numeración de los subíndices de irá del 0 hasta el 4.

Elementos ---->	1	2	3	4	5
	5	15	17	25	30
Índices ------>	0	1	2	3	4

ISBN: 978-987-82912-8-4

La cantidad de elementos en el vector se especificará mediante el número entre [ ] al declarar, esto se denomina dimensionamiento que es necesario a menos que el tamaño del vector se especifique implícitamente a través de la inicialización:.

```
char Letras[] = {'a', 'b', 'c', 'd'};
int num_entero[] = {2, 3, 0, 4, 8, -13, 7, 45, 5, -21};
```

Ahora para poder ingresar los elementos de un vector podemos realizarlo de dos formas :

- Por cada elemento del vector:

```
int tabla[5];
scanf ("%f", tabla[3]);
```

			25	
0	1	2	3	4

- Llenar todos los elementos del vector desde una estructura de repetición automática for.

```
Int x, tablas[5];
for (x=1;x<5; x++)
 {
tabla[x]=x;
}
```

3	4	6	25	13
0	1	2	3	4

De igual manera para la impresión de un vector lo realizamos igual de dos formas:

- Por cada elemento del vector:

```
printf ("%d", tabla[3]);
```

- Llenar todos los elementos del vector desde una estructura de repetición automática for.

```
Int i,
tablas[5];
for (i=1;i<=4; i++)
  {
```

ISBN: 978-987-82912-8-4

```
        printf ("%d", tabla[i]);
}
```

4.2. Arrays Bidimensionales

Un Array bidimensional o llamado también matriz (un vector de vectores). Por lo tanto, es un tipo de matriz de elementos donde el orden de los componentes es significativo y se deben especificar dos subíndices para identificar cada elemento de la matriz. (Szuhay, 2022).

El dimensionamiento en lenguaje C se coloca el tipo de dato, luego sigue el identificador proporcionado para el nombre de la variable. Finalmente encerrado entre llaves [ ] se ubicará el número de las filas y el número de las columnas. La sintaxis se esta estructura se puede observar en la tabla 4.2.

Tabla 4.2 Estructura Array-Matriz

Sintaxis	Ejemplo
<tipo dato> <identificador> <[núm_filas]> <[núm_columnas]>	Int prueba [6][5]; Char empleado [30][20]; float precio[10][6], float cuotas[6][6]; char nombre_apellido[20][20]; char sopa_letras[26][20];

De forma general la representación es m[i][j] , donde i es el numero de filas y j número de columnas De la misma forma que en los vectores en el lenguaje de programación C la numeración de los subíndices inicia en cero pero sus elementos es más uno. Ejemplo:

Elementos ---->

	0	1	2	3	4	j
1		3	4	5	6	
2		7	8	9	11	

ISBN: 978-987-82912-8-4

3	8	5	9	7
4	3	2	1	15
i				

En el proceso de asignación de valores a una matriz, en general es muy similar como cuando se trabaja con un vector, cuidando que se haga referencia a los índices de fila y columna.

```
Int matriz [3,3] ={2,4,6,8,1,0} ó
Int matriz [3,3] = {
                        {2,4}
                        {6,8}
                        {1,0}
                   }
```

Ahora para almacenar o imprimir el elemento en particular se lo haría :

- Desde el teclado , ingreso por el usuario.

```
int matriz[3] [3];
scanf (“ %d”, matriz[0][0]);
```

- Desde el teclado , ingreso por el usuario.

```
int matriz[3] [3];
printf(“ %d”, matriz[1][1]);
```

Ahora para poder manipular todos los elementos de una matriz, se lo debe realizar mediante un recorrido de los elementos, y esto se lo lleva a cabo mediante dos estructuras for anidadas, ya que es necesario recorrer todas las filas y columnas de la matriz

- Desde el teclado , ingreso por el usuario.

```
int matriz[3] [3];
for(i=0;i<=3;i++)
```

ISBN: 978-987-82912-8-4

```
for(j=0;j<=3;j++)
        scanf (“ %d”, matriz[i][j]);
```

- Desde el teclado , ingreso por el usuario.

```
int matriz[3] [3];
for(i=0;i<=3;i++)
        for(j=0;j<=3;j++)
                printf(“ %d”, matriz[1][1]);
```

4.3. Funciones

Inicialmente pensemos en que deseamos ejecutar una acción en un vector, pues bien declaramos una estructura for y se ejecutan las instrucciones. Ahora bien, qué tal si deseamos hacer esta operación unas 20 veces, pues tendríamos que volver a escribir la estructura con sus instrucciones 20 veces lo que no sería lo más optimo. (Szuhay, 2022).

Todo lo anterior se resolvería si esa sección de código la colocáramos en una función y/o módulo, y tan solo llamaríamos a la función en el lugar del programa donde se lo necesitara sin la necesidad de volver a escribir la estructura for con sus instrucciones para manejar las acciones del vector. El uso de las funciones tiene ciertas ventajas las mismas que podemos describir a continuación:

- Aplicando la técnica del divide y vencerás reducimos la complejidad del programa en lugar de tener un programa grande complejo lo tenemos dividido en subsecciones pequeñas
- Este eliminar o reducir duplicación de código innecesario
- Controlar y verificar errores en la codificación con mayor eficacia
- Reutilización del código.
- Al tener seleccionado el programa en varias subsecciones me permitirá tener un entendimiento mucho más accesible y menos ambiguo de este.

Todas las secciones, funciones que decidamos crear deben ser verificadas y depuradas, pero no pueden ejecutarse por sí solas, estas deben estar ancladas a un programa principal que es el main () como se puede ver en la figura 4.1. El objetivo de

ISBN: 978-987-82912-8-4

la función main es coordinar y orquestar, por así decirlo, las llamadas y retorno de datos de las otras funciones.

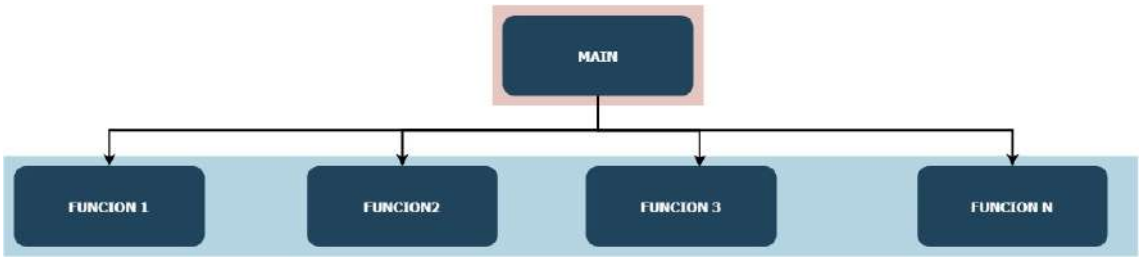


Figura 4.1 Jerarquía Funciones

En el lenguaje se puede determinar dos tipos de funciones las que están predeterminadas por el lenguaje, por ejemplo, librerías #include <stdio.h>, y las que están definidas por el desarrollador. Ahora las funciones solicitadas por el main a su vez pueden llamar a otras funciones (Szuhay, 2022). LA estructura general de una función es:

```
<Tipo_dato> <identificador> (**Lista de parámetros)

    {
      **Variables Locales
      Cuerpo función
      **return dato
    }
```

** Las variables locales y el retorno del dato, así como la lista de parámetros dependen del diseño e implementación de la función.

Las funciones en el lenguaje C pueden ser desarrolladas de las siguientes maneras:

Funciones sin envío parámetros. - Este tipo de funciones no requiere información adicional pues simplemente se ejecutan cada vez que son invocados.

Funciones sin retorno de valor. - Luego de su llamado y de su ejecución no devuelven ningún valor es decir se ejecutan por sí solas ejemplo esto cuando hacemos el llenado de un vector llena y no devuelve ningún valor. A continuación, en la tabla 4.3, podemos revisar la sintaxis del funcionamiento de estos dos tipos de funciones.

ISBN: 978-987-82912-8-4

Tabla 4.4 Estructura Funciones sin retorno de valor ni paso de parámetros

Sintaxis	Ejemplo

Void <identificador> (**Lista de parámetros)
{
**Variables Locales
Cuerpo función
}
	Void main ()
	{
	suma();
	}
	Void suma()
	{
	int a, b;
	int suma =0;
	printf(“ingrese el primer valor”);
	scanf(“%f”, &b);
	printf(“ingrese el segundo valor”);
	scanf(“%f”, &b);
	suma= a+b
	printf(“ La sumatoria es : %f”, suma);
	}

** La lista de parámetros está en paréntesis en blanco “( )” y dependen del diseño e implementación de la función.

Las variables locales sólo pueden ser reconocidas y utilizadas por la función en la que están declaradas por ejemplo Si una variable es definida en cualquiera de las funciones que no sea “main” esta variable sólo va a ser reconocida en cada una de esas funciones donde fue definida. Además, que el uso de la memoria sólo será utilizado en el momento que se ha llamada la función que quiere decir esto cuando la función esté ejecutando su código pues ahí se declarará la variable se la utilizará, pero cuando la función termine su tarea pues obviamente se liberará la memoria. (Szuhay, 2022).

Funciones con retorno de valor. - Este tipo de funciones una vez llamadas y ejecutadas o acción de la función nos devuelve entregan un valor o resultado puntualmente que puede ser un número un tipo de dato entero real un nombre un vector llenado o una matriz, como se puede observar en la tabla 4.5.

ISBN: 978-987-82912-8-4

Tabla 4.5 Estructura Funciones con retorno de valor

Sintaxis	Ejemplo

<Tipo dato> <identificador> ()
{
Variables Locales
Cuerpo de la función
**return (valor)
}
	Void main ()
	{
	int resultado;
	resultado = suma();
	printf(" La sumatoria es : %f", resultado);
	}
	Void suma()
	{
	int a, b;
	int suma =0;
	printf("ingrese el primer valor");
	scanf("%f", &b);
	printf("ingrese el segundo valor");
	scanf("%f", &b);
	suma= a+b;
	return suma;
	}

** Es una palabra reservada que envía el valor de la función, retorna el trabajo que la función realizó para ser usada como el requerimiento en otra función necesite.

Funciones con envío de parámetros. - Además de su de su llamado a estas funciones se requiere que se le pase o se añada información adicional como requerimiento para que la función pueda realizar su trabajo. (Szuhay, 2022).

Es importante indicar que los parámetros pueden enviarse de 2 maneras:

- Un parámetro es enviado por valor, que como su nombre lo indica estamos enviando el valor del contenido de la variable, ejemplo =5, se envía el 5 a la función que lo utilizara en sus instrucciones, pero la variable original no se altera. Ver Tabla 4.6.

ISBN: 978-987-82912-8-4

Tabla 4.6 Estructura Funciones con envío parámetros por valor

Sintaxis	Ejemplo

<Tipo dato> <identificador> (<tipo dato> parámetro)
 	
{
Variables Locales
Cuerpo de la función
return (valor)
}
	Void main ()
	{
	int a, b, c;
	int suma (int a, int b);
	
	printf("ingrese el primer valor");
	scanf("%f", &b);
	printf("ingrese el segundo valor");
	scanf("%f", &b);
	c = suma(a,b);
	printf(" La sumatoria es : %f", c);
	
	}
	
	int suma(int a, int b)
	{
	return a+b;
	}

- La segunda forma de envío de los parámetros tenemos el paso por referencia, aquí no enviamos del valor que contiene la variable si no enviamos la dirección de memoria de la variable es decir si dentro de la función se realiza algún cambio pues obviamente la variable fuera de la función sufrirá este cambio. Ver Tabla 4.7

Tabla 4.7 Estructura Funciones con envío parámetros por referencia

Sintaxis	Ejemplo

<Tipo dato> <identificador> (<tipo dato> parámetro)
 	
{
Variables Locales
Cuerpo de la función
return (valor)
}
	Void Cambio(int x, int y)
	{
	int aux;
	aux = x;
	x = y;
	y = aux;

ISBN: 978-987-82912-8-4

```
    }


Void main ()
  {
        int a, b;

      printf("ingrese los valores de a y b %d ");
      scanf("%d %d", &a, &b);
      Cambio (a,b);
      printf(" Los valores de a =%d y b =%d ", a,
      b);

  }
```

El paso de parámetros por referencia en una función implica utilizar los operadores & y * el primer operador se antecede a una variable con ello el acceso a la dirección de memoria asignada a esa variable. El segundo operador "*", es un apuntador que apunta a la dirección de la variable pasada como argumento, habitualmente este último utilizado en la teoría de punteros. (Martín, Urquía y Rubio, 2021)

4.4. Ejercicios Resueltos

Ejercicio 4.1. Realice un programa que encripte un texto utilizando encriptación poligráfica (cuando n caracteres del mensaje original generan n caracteres del mensaje encriptado) y muestre que no existe alteración o perdida de la información al ser desencritado.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#define INCREMENTO  5

// Objetivo: REALIZA LA ENCRIPTACIÓN POLIGRÁFICA DE UN TEXTO
char cadena[100],cadena1[100];
int i=0;
```

ISBN: 978-987-82912-8-4

```
char continuar;
main()
{
 do
 {
      printf("ENCRIPTA Y DESENCRITA UN MENSAJE POR EL MÉTODO
    POLIGRÁFICO\n\n");
      printf("MENSAJE A ENCRIPTAR (<100): ");
      gets(cadena);
      strcpy(cadena1,cadena);
      while (cadena1[i]!='\0')
      {
               cadena1[i]=cadena1[i]-(i+INCREMENTO);
               i++;
      }
      cadena1[i]='\0';
      puts("\nMENSAJE         : ");
      puts(cadena);
      puts("\nMENSAJE ENCRITADO: ");
      puts(cadena1);
      i=0;
      while (cadena1[i]!='\0')

      {
               cadena1[i]=cadena1[i]+(i+INCREMENTO);
               i++;
      }
      cadena1[i]='\0';
      puts("\nMENSAJE           : ");
      puts(cadena);
      puts("\nMENSAJE DESENCRITADO: ");
      puts(cadena1);
      puts("\n\nUNA TECLA PARA TERMINAR... ");
      continuar=getch();
}
while (toupper(continuar)=='S');
```

ISBN: 978-987-82912-8-4

```
}
```

Ejercicio 4.2. Realice un programa que por medio de un menú se defina y dibuje los triángulos según sus ángulos.

```
#include <stdio.h>
#include <conio.h>
#include <stdio.h>
#define CENTRO 20

char comentario[3][100]={"TRIÁNGULO ACUTÁNGULO: Sus tres ángulos interiores son agudos\n",
                         "TRIÁNGULO RECTÁNGULO: Posee un angulo de 90 grados y dos ángulos agudos",
                        "TRIÁNGULO OBTUSÁNGULO:Posee un ángulo mayor a 90 grados (obtuso) y dos
                          ángulos agudos"};

dibujatriangulo(int altura, int tipo);

int main()
{
 int altura,tipo,opcion;

 do
 {
      do
      {
           clrscr();
           printf("\t\tTIPOS DE TRIÁNGULOS\n");
           printf("\t\tSEGÚN SUS ÁNGULOS\n\n");
           printf("1. ACUTÁNGULO\n");
           printf("2. RECTÁNGULO\n");
           printf("3. OBTUSÁNGULO\n\n");
           printf("0. SALIR\n");
```

ISBN: 978-987-82912-8-4

```
            printf("\n\nINGRESE LA OPCIÓN:");

            scanf("%d", &opcion);
        }
        while(opcion<0 || opcion>3);
        if (opcion!=0)

        {
                clrscr();
                printf("\nDEFINICIÓN ");
                printf("\n_____\n\n%s",comentario[opcion-1]);
                printf("\n\nINGRESE LA ALTURA QUE TENDRA EL TRIÁNGULO = ");
                scanf("%d",&altura);
                dibujatriangulo(altura,(opcion));
        }
 }
 while(opcion!=0);
}

dibujatriangulo(int altura, int tipo)
{
int i,j, final, lugar=0;
printf("\n");
for(i=1;i<=altura;i++)
{
    final=altura-i;
    for(j=1;j<=(final*tipo+CENTRO);j++)
            printf(" ");
    for(j=1; j<=i*2-1 ; j++)
            printf("*");
    printf("\n");
}
puts("\n\n\t\tUNA TECLA PARA CONTINUAR...");
getch();
return 0;
}
```

ISBN: 978-987-82912-8-4

Ejercicio 4.3. Realice un programa que por medio de funciones lea dos fracciones y obtenga la fracción equivalente más simple resultante de su suma.

```
#include <stdio.h>
#include <ctype.h>
#include <windows.h>
#include <conio.h>

// Objetivo: OBTENER LA FRACCIÓN EQUIVALENTE MÁS SIMPLE DEL RESULTADO DE SU SUMA.

int numerador1,denominador1,numerador2,denominador2;
char continuar='s';

suma(int n1, int d1, int n2, int d2, int *rn, int *rd)
{
 *rn=d2*n1+d1*n2;
 *rd=d1*d2;
return 0;
}


reducefraccion(int *rn, int *rd)
{
int i,menor;
menor=*rd<*rn? *rd :*rn;
for (i=2; i<= menor; i++)
 {
      while((*rn%i==0) && (*rd%i==0))
       {
                *rd=*rd/i;
                *rn=*rn/i;
       }
 }
```

ISBN: 978-987-82912-8-4

```
return 0;
}

 main()
{
 int rn,rd;
 while (toupper(continuar)=='S')
  {
        clrscr();
        printf("\n SUMA DOS FRACIONES Y REDUCE SU RESULTADO A LA
     FRACCIÓN MÁS SIMPLE \n\n");
       do
      {
              printf("PRIMERA FRACCIÓN A SUMAR (forma: n1/d1 donde d1<>0): ");
              scanf("%d/%d",&numerador1,&denominador1);
        }
       while (denominador1==0);

       do
      {
              printf("SEGUNDA FRACCIÓN A SUMAR (forma: n2/n2 donde n2<>0):
     ");
              scanf("%d/%d",&numerador2,&denominador2);
        }
       while (denominador2==0);
       suma(numerador1,denominador1,numerador2,denominador2,&rn,&rd);
       printf("\n\n%d/%d+%d/%d                                                =
     ",numerador1,denominador1,numerador2,denominador2);
       Sleep(500);
       printf(" %d/%d", rn,rd);

       Sleep(500);
       printf(" => ");
       Sleep(500);
       reducefraccion(&rn,&rd);
       printf("%d/%d ", rn,rd);
```

ISBN: 978-987-82912-8-4

```
        printf("\n\n DESEA REALIZAR OTRA SUMA (S/N) ? ");
        scanf(" %c",&continuar);
    }
    }
```

Ejercicio 4.4. Implemente un programa que por medio de funciones determine si tres puntos p1(x1,y1), p2(x1,y1) y p3(x2,y2), ingresados desde el teclado, son vértices de un triángulo isósceles.

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

#define  TRIÁNGULO(x,y,z)  (((x==y)||(y==z)||(x==z))? "SON VÉRTICES" : "NO SON VÉRTICES")

// Objetivo: DETERMINAR SI TRES PUNTOS P1(X1,Y1),P2(X1,Y1) Y P3(X2,Y2)
//          SON VÉRTICES DE UN TRIÁNGULO ISÓSCELES.

float d1, d2, d3;
int puntox1,puntoy1,puntox2,puntoy2,puntox3,puntoy3;
char continuar='s';

float distanciapunto(int x1, int y1, int x2, int y2)
{
float distancia;
distancia=sqrt(pow(x2-x1,2)+pow(y2-y1,2));
return(distancia);
}

main()
{
printf("DISTANCIA ENTRE P1(X1,Y1) Y P2(X2,Y2) \n");

while (toupper(continuar)=='S')
 {
```

ISBN: 978-987-82912-8-4

```
        printf("INGRESE EL PUNTO P1(X1,Y1) : ");
        scanf("%d %d",&puntox1,&puntoy1);
        printf("INGRESE EL PUNTO P2(X2,Y2): ");
        scanf("%d %d",&puntox2,&puntoy2);
        printf("INGRESE EL PUNTO P2(X2,Y2): ");
        scanf("%d %d",&puntox3,&puntoy3);
        d1= distanciapunto(puntox1,puntoy1,puntox2,puntoy2);
        d2= distanciapunto(puntox2,puntoy2,puntox3,puntoy3);

        d3= distanciapunto(puntox1,puntoy1,puntox3,puntoy3);
        printf("\nP1(%d,%d), P2(%d,%d) Y P2(%d,%d) %s DE UN TRIÁNGULO
    ISÓSCELES
    \n",puntox1,puntoy1,puntox2,puntoy2,puntox3,puntoy3,TRIÁNGULO(d1,d2,d3))
    ;
        printf("\n REALIZAR UNA NUEVA EJECUCIÓN (S/N)? ");
        scanf(" %c",&continuar);
 }
}
```

Ejercicio 4.5. Implemente un programa que encuentre los ángulos interiores de un triángulo de vértices a(x1,y1), b(x1,y1) y c(x2,y2), ingresados desde el teclado.

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
// Objetivo: DETERMINAR LOS ÁNGULOS DEL TRIÁNGULO DE VÉRTICES
//              A(X1,Y1),B(X1,Y1) Y C(X2,Y2)
char continuar='s';
float angulo1, angulo2, angulo3;
int x1,y1,x2,y2,x3,y3;
float dab,dbc,dca;
float pendiente(float x1, float y1, float x2, float y2);
float angulo(float pen1,float pen2);
main()
```

ISBN: 978-987-82912-8-4

```
{
printf("ENCUENTRA LOS ÁNGULOS DEL TRIÁNGULO DE VÉRTICES
A(X1,Y1),B(X2,Y2) Y C(X3,Y3) \n");
while (toupper(continuar)=='S')
{
     printf("INGRESE EL VÉRTICE A(X1,Y1) : ");
    scanf("%d %d",&x1,&y1);
     printf("INGRESE EL VÉRTICE B(X2,Y2): ");
    scanf("%d %d",&x2,&y2);
    printf("INGRESE EL VÉRTICE C(X3,Y3): ");
    scanf("%d %d",&x3,&y3);
    dab= pendiente(x1,y1,x2,y2);
    dbc= pendiente(x2,y2,x3,y3);
    dca= pendiente(x1,y1,x3,y3);
     printf("EL ÁNGULO DE A(%d,%d) ES %2.2f
    GRADOS\n",x1,y1,angulo(dab,dca));
     printf("EL ÁNGULO DE B(%d,%d) ES %2.2f
    GRADOS\n",x2,y2,angulo(dbc,dab));
     printf("EL ÁNGULO DE C(%d,%d) ES %2.2f
    GRADOS\n",x3,y3,angulo(dca,dbc));
     printf("QUE SUMADOS DAN %3.2f GRADOS\n",
     angulo(dab,dca)+angulo(dbc,dab)+angulo(dca,dbc));
     printf("\n REALIZAR UNA NUEVA EJECUCIÓN (S/N)? ");
    scanf(" %c",&continuar);
}}
float angulo(float pen1,float pen2)
{
#define PI 3.141516
float angulo1;
angulo1=atan((pen1-pen2)/(1+pen1*pen2));
return(angulo1*360/(2*PI));
}
float pendiente(float x1, float y1, float x2, float y2)
{
float resultado;
resultado=((y2-y1)/(x2-x1));
```

ISBN: 978-987-82912-8-4

```
return(resultado);
}
```

Ejercicio 4.6. Realice un programa que simule el juego de obtener siete al lanzar dos dados.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
// Objetivo: JUEGO QUE SIMULA EL LANZAMIENTO DE DOS DADOS - GANA AL
OBTENER 7
int dado()
{
int numero;
numero=(rand()/32768.0)*6+1;
return(numero);
}
main()
{
int x,y,lanzamientos=0;
char desicion='S';
printf("SIMULA EL LANZAMIENTO DE UN DADO\n");
while (toupper(desicion)=='S')
{
    printf("\n LANZAR LOS DADOS S/N? ");
    scanf(" %c",&desicion);
      if (toupper(desicion)=='S')

    {
            printf("\nLOS DADOS DAN: [%d]  [%d]  " , x=dado(), y=dado());
                    lanzamientos++;
                    if (x+y==7)
                            printf("  !!  FELICIDADES  GANÓ  LUEGO  DE    %d
LANZAMIENTO!!!",
                    lanzamientos);
                    else
```

ISBN: 978-987-82912-8-4

```
                    printf(" INTÉNTELO NUEVAMENTE!!!");
            }
    }}
```

Ejercicio 4.7. Escriba un programa que determine si un número es o no múltiplo de un dígito por medio de una función.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: DETERMINAR UN NÚMERO ES MÚLTIPLO DE UN DÍGITO CON FUNCIONES
int numero,digito;
char continuar='s';
char multiplodigito(int numero, int digito)
{
char cadena;
(numero%digito==0)? cadena='S':cadena='N';
return (cadena);
}


main()
{
printf("SI UN NÚMERO ES MÚLTIPLO DE UN DÍGITO USANDO FUNCIONES\n");
while (toupper(continuar)=='S')
{
     printf("INGRESE EL NÚMERO (ENTERO): ");
    scanf("%d",&numero);
    do
     {
            printf("INGRESE EL DÍGITO (ENTERO): ");
           scanf("%d",&digito);
      }
           while (!(digito>=0 && digito<=9));
```

ISBN: 978-987-82912-8-4

```
            if (multiplodigito(numero,digito)=='S')
             printf(" EL %d ES MÚLTIPLO DE %d ",numero,digito);
            else
            printf(" EL %d NO ES MÚLTIPLO DE %d ",numero,digito);

    printf("\n\n EJECUTAR NUEVAMENTE (S/N) ? ");
            scanf(" %c",&continuar);
}}
```

Ejercicio 4.8. Escriba un programa que por medio de una función determine si un número es primo o no.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: DETERMINAR SI UN NÚMERO ES PRIMO USANDO FUNCIONES
int numero;
char continuar='s';

int numeroprimo(int numero)
{
int i,contador=0;
for (i=1; i<=numero; i++)
    if (numero%i==0)
            contador++;
if (contador==2)
     return(1);
else
     return(0);
}

main()
{
printf("DETERMINA SI UN NÚMERO ES PRIMO USANDO FUNCIONES\n");
while (toupper(continuar)=='S')
```

ISBN: 978-987-82912-8-4

```
{
    do
    {
            printf("INGRESE EL NÚMERO (ENTERO POSITIVO): ");
            scanf("%d",&numero);
     }
    while (!numero>0);
    if (numeroprimo(numero))
            printf(" EL %d ES PRIMO !! ",numero);
    else
           printf(" EL %d NO ES PRIMO !!",numero);
    printf("\n\n REALIZAR UN NUEVO CÁLCULO (S/N) ? ");
    scanf(" %c",&continuar);
}
}
```

Ejercicio 4.9. Escriba una función que devuelva como resultado el mayor de tres números.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: DETERMINAR EL MAYOR DE TRES NÚMEROS CON UNA FUNCIÓN
int numer1;
int  numer2;
int numer3;
char continuar='s';
int mayornumero(int numero, int numero1, int numero2)
{
int mayor=numero;
if (mayor< numero1)
    mayor=numero1;
if (mayor< numero2)
    mayor=numero2;
return(mayor);
}
```

ISBN: 978-987-82912-8-4

```
main()
{
printf("MAYOR DE TRES NÚMEROS CON FUNCIONES\n");
while (toupper(continuar)=='S')
{
    printf("INGRESE EL PRIMER NÚMERO (ENTERO): ");
    scanf("%d",&numer1);
    printf("INGRESE EL SEGUNDO NÚMERO (ENTERO): ");
    scanf("%d",&numer2);
    printf("INGRESE EL TERCER NÚMERO (ENTERO): ");
    scanf("%d",&numer3);
    printf("DE %d, %d, %d EL MAYOR ES %d",numer1,numer2,numer3,
     mayornumero(numer1,numer2,numer3));
    printf("\n\n REALIZAR UN NUEVO CÁLCULO (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 4.10. Escriba un función que obtenga el dígito mayor de un número entero.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: DETERMINAR UN DÍGITO MAYOR DE UN NÚMERO CON FUNCIONES
int numero;
char continuar='s';
int digitomayor(int numero)
{
int digito=0;
while (numero>0)
{
    if (digito< numero%10)
            digito=numero%10;
    numero=numero/10;
}
return(digito);
}
```

ISBN: 978-987-82912-8-4

```
main()
{
printf("ENCUENTRA DÍGITO MAYOR DE UN NÚMERO USANDO FUNCIONES\n");
while (toupper(continuar)=='S')
{
     printf("INGRESE EL NÚMERO: ");
    scanf("%d",&numero);
    printf("EL DÍGITO MAYOR DE %d ES %d ",numero,digitomayor(numero));
    printf("\n\n EJECUTAR NUEVAMENTE (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 4.11. Escriba una función que entregué como resultado el complemento a9 de un número entero dado.

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
// Objetivo: OBTENER EL COMPLEMENTO A NUEVE DE UN NÚMERO DECIMAL
int numero;
int complementoa9(int numero)
{
int complemento=1;
while (numero>complemento)
     complemento*=10;
complemento-=1;
return(complemento-numero);
}
main()
{
printf("COMPLEMENTO A9 DE UN NÚMERO CON FUNCIONES\n");
while (toupper(continuar)=='S')
{
     printf("INGRESE EL NÚMERO : ");
```

ISBN: 978-987-82912-8-4

```
    scanf("%d",&numero);
    numero=abs(numero);
    printf("DE %d EL COMPLEMENTO A9 ES %d \n",numero,
complementoa9(numero));
    printf("\n\n EJECUTAR UNA VEZ MÁS  (S/N) ? ");
    scanf(" %c",&continuar);
}}
```

Ejercicio 4.12. **Realice un programa con una función recursiva que permita sumar los dígitos de un número entero dado.**

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: SUMAR EN FORMA RECURSIVAVA LOS DÍGITOS DE UN NÚMERO
ENTERO
int numero;
char continuar='S';
int sumadigitos(int numero)
{
int suma=0;
if (numero!=0)
    suma=numero%10+sumadigitos(numero/10);
return (suma);
}
main()
{
printf("SUMA RECURSIVAMENTE LOS DÍGITOS DE UN NÚMERO\n");
while (toupper(continuar)=='S')
{
    printf("\nINGRESE UN NÚMERO (ENTERO):");
     scanf("%d", &numero);
    printf("\nEL %d SUMANDO LOS DÍGITOS ES = %d\n",numero,
sumadigitos(numero));
    printf("\n\nEJECUTAR NUEVAMENTE EL PROGRAMA (S/N) ? ");
    scanf("  %c",&continuar);
}}
```

ISBN: 978-987-82912-8-4

Ejercicio 4.13. **Implemente un programa con una función recursiva que lea una línea de texto y escriba los caracteres en orden inverso**

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: INVIERTE TEXTO EN FORMA RECURSIVA. */
char continua,continuar='S' , cadena;
invierte (char cadena)
{
if ((cadena=getchar())!='\n')
        invierte(cadena);
putchar(cadena);
}
main()
{
printf("INVIERTE UNA CADENA EN FORMA RECURSIVA \n\n");
while (toupper(continuar)=='S')
 {
        printf("INGRESE LA CADENA: ");
        invierte(cadena);
        printf("\n\n EJECUTAR NUEVAMENTE (s/n) ? ");
        scanf(" %c",&continuar);
        continua=getchar();  // toma caracteres sueltos
 }}
```

Ejercicio 4.14. **Escriba un programa que en forma recursiva transforme a mayúsculas un texto ingresado por el teclado**

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

// Objetivo: TRANSFORMA A MAYÚSCULAS UNA TEXTO EN FORMA RECURSIVA
```

ISBN: 978-987-82912-8-4

```
int numero;
char basura,continuar='S';

char mayusculas ()
{
char c2;
scanf("%c",&c2);
if (c2!='\n')
{
      putchar(toupper(c2));
      mayusculas();
}
return 0;
}

main()
{
printf("TRANSFORMA A MAYÚSCULAS UNA PALABRA RECURSIVAMENTE\n");
while (toupper(continuar)=='S')
{
     printf("\nINGRESE EL TEXTO :\n");
      mayusculas();
     printf("\n\nEJECUTAR NUEVAMENTE EL PROGRAMA (S/N) ? ");
     scanf("  %c",&continuar);
     basura=getchar();
}
}
```

Ejercicio 4.15. **Escriba un programa con una función recursiva que calcule la factorial de un número dado.**

```
#include <stdio.h>
#include <ctype.h>
 /* Objetivo: CALCULA EL FACTORIAL EN FORMA RECURSIVA.*/
char continuar='S';
```

ISBN: 978-987-82912-8-4

```
int numero;
int factorial(int numero)
{
int fac=1;
if (numero>=1)
  fac=numero*factorial(numero-1);
return (fac);
}
main()
{
printf("CALCULA EL FACTORIAL EN FORMA RECURSIVA \n\n");
while (toupper(continuar)=='S')
{
        do
        {
                printf("INGRESE UN NÚMERO  (Positivo): ");
               scanf("%d",&numero);
         }
        while (numero<0);
        printf("EL FACTORIAL DE %d ES = %d \n ", numero, factorial(numero));
        printf("\n\n EJECUTAR NUEVAMENTE (s/n) ? ");
        scanf(" %c",&continuar);
}}
```

Ejercicio 4.16. **Escriba un programa que genere recursivamente el valor de PI por el método de leibnitzen. π = 4(1 – 1/3 + 1/5 – 1/7 + 1/9 - ...)**

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
// Objetivo: CALCULAR EL VALOR DE PI POR EL MÉTODO DE LEIBNITZEN
//pi = 4*(1 – 1/3 + 1/5 – 1/7 + 1/9 - ...)
float valordepi(int numero1, int opcion)
{
int numero;
```

ISBN: 978-987-82912-8-4

```
float valpi=0;
for (numero=0; numero<=numero1; numero++)
 {
      valpi=pow(-1,numero)/(float(2.0*numero)+1.0)+valpi;
      if (opcion==1)
       printf("para n = %d  pi=%f \n",numero,4.0*valpi);
 }

 return (4.0*valpi);
}
main()
{
char continuar='S';
int opcion,numero;
float valpi;
printf("CALCULA EL VALOR DE PI POR EL MÉTODO DE LEIBNITZEN. \n\n");
while (toupper(continuar)=='S')
 {
      do
       {
                printf("INGRESE EL NÚMERO DE TÉRMINOS DE LA SERIE
               (Positivo):");
                scanf("%d",&numero);
       }
      while (numero<0);
      do
       {
                printf("\n     VISUALIZAR:\n\n");
                printf("1. LOS N VALORES DE PI\n");
                printf("2. EL VALOR FINAL DE PI\n\n");
                printf("ESCOJA LA OPCIÓN:");
                scanf("%d",&opcion);
       }
      while (!(opcion==1 || opcion ==2));

      if (opcion==1)
```

ISBN: 978-987-82912-8-4

```
            printf("\n\nTABAL DE VALORES DE PI SEGUN LEIBNITZEN\n\n");
        valpi=valordepi(numero, opcion);
        printf("\n EL VALOR DE PI OBTENIDO ES : %f", valpi);
        printf("\n\n NUEVA EJECUCIÓN DE PROGRAMA (S/N) ? ");
        scanf(" %c",&continuar);
    }}
```

Ejercicio 4.17. **Escriba un programa que genere recursivamente los n primeros términos de la serie de fibonacci.**

```
#include <stdio.h>
#include <ctype.h>
 // Objetivo: GENERAR LA SERIE DE FIBONACCI EN FORMA RECURSIVA.
char continuar='S';
int numero;
fibonacci(int numero,int fibo0, int fibo1)
{
int fibo2;
if (numero>0)
{
        printf("%d ",fibo0);
        fibonacci(numero-1,fibo1,fibo0+fibo1);
}

return 0;
}

main()
{
printf("OBTIENE LOS TÉRMINOS DE LA SERIE DE FIBONACCI EN FORMA
RECURSIVA \n\n");
while (toupper(continuar)=='S')
{
        do
         {
```

ISBN: 978-987-82912-8-4

```
                printf("INGRESE UN NÚMERO  (Positivo): ");
                scanf("%d",&numero);
            }
        while (numero<0);
        printf("\n LOS %d TÉRMINOS DE LA SERIE SON : ", numero);
        fibonacci(numero, 0,1);
        printf("\n\n REALIZAR UN  NUEVO CÁLCULO (s/n) ? ");
        scanf(" %c",&continuar);
}
}
```

Ejercicio 4.18. **Demuestre que se puede obtener los términos de la serie de fibonacci con funciones recursivas que trabajen con un solo parámetro en la llamada.**

```
#include <stdio.h>
#include <ctype.h>
/* Objetivo: CALCULA  LOS TÉRMINOS DE LA SERIE DE FIBONACCI CON
FUNCIONES  RECURSIVAS DE UN  SOLO PARÁMETRO */
char continuar='S';
int numero;
int terminon(int numero)
{
int termino;
if (numero<3)
        termino=1;
else
        termino=terminon(numero-1)+terminon(numero-2);
  return(termino);
}

seriefibonacci(int numero)
{
if (numero>1)
{
```

ISBN: 978-987-82912-8-4

```
        seriefibonacci(numero-1);
        printf("%d ",terminon(numero));
 }
return 0; }

main()
{
printf("OBTIENE LOS TÉRMINOS DE LA SERIE DE FIBONACCI CON
FUNCIONES  RECURSIVAS DE UN  SOLO PARÁMETRO \n\n");
while (toupper(continuar)=='S')
{
        do
         {
                 printf("INGRESE UN NÚMERO  (Positivo): ");
                scanf("%d",&numero);
          }
        while (numero<0);
        printf("\n LOS %d TÉRMINOS LA SERIE SON :%d ",numero);
        seriefibonacci(numero);
        printf("\n\n EJECUTAR NUEVAMENTE (s/n) ? ");
        scanf(" %c",&continuar);
}
}
```

Ejercicio 4.19. **Implemente un programa que invierta un número entero positivo utilice para el efecto funciones recursivas.**

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

 // Objetivo: INVERTIR UN NÚMERO RECURSIVAMENTE.

char continuar='S';
int numero;
```

ISBN: 978-987-82912-8-4

```
int cifras(int numero)
{
int cifra=0;
if (numero/10 ==0)
        cifra=1;
else
        cifra=1+cifras(numero/10);
return(cifra);
}

int invierte(int numero, int pot)
{
int inv;
inv=numero%10;
if (numero>9)
        inv=inv*pow(10,pot)+invierte(numero/10,pot-1);
return(inv);
}

main()
{
printf("INVIERTE UN NÚMERO RECURSIVAMENTE \n\n");
while (toupper(continuar)=='S')
 {
        do
                {
                        printf("INGRESE UN NÚMERO  (Positivo): ");
                        scanf("%d",&numero);
                }
         while (numero<0);
        printf("\n %d INVERTIDO ES %d
",numero,invierte(numero,cifras(numero)-1));
        printf("\n\n REPETIR LA EJECUCIÓN (s/n) ? ");
        scanf(" %c",&continuar);
}
}
```

ISBN: 978-987-82912-8-4

Ejercicio 4.20. Implemente un programa que obtenga en forma recursiva la potencia de un número.

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

// Objetivo: ELEVA UN NÚMERO A UNA POTENCIA EN FORMA RECURSIVA.

char continuar='s';
int a,b;
int pot_recur(int numero, int potencia);

main()
{
do
{
    printf("A^B RECURSIVAMENTE \n\n");
    do
    {
            printf("INGRESE A [>0]: ");
            scanf("%d",&a);
    }
    while (a<=0);
    do
    {
            printf("INGRESE b [>=0]: ");
            scanf("%d",&b);
    }
    while (b<0);
    printf("\n(%d)^%d = %d ",a,b,pot_recur(a,b));
    printf("\n\n REPETIR LA EJECUCIÓN (s/n)? ");
    scanf(" %c",&continuar);
}
while(tolower(continuar)=='s');
```

ISBN: 978-987-82912-8-4

```
}
int pot_recur(int numero, int potencia)
{
int pot=1;
if (potencia!=0)
{
      if (potencia==1)
                        pot=numero;
     else
                        pot=numero* pot_recur(numero,potencia-1);
  }
return(pot);
}
```

Ejercicio 4.21. Implemente un programa que obtenga recursivamente las n líneas del triángulo de pascal.

```
#include <stdio.h>
#include <ctype.h>
// Objetivo: OBTIENE LAS N PRIMERAS LÍNEAS DEL TRIÁNGULO RECURSIVAMENTE

int numero;
char continuar='s';

int factorial (int n)
{
int fac=1;
if (n<1)
      fac=1;
else
     fac=n*factorial(n-1);
return(fac);
}
```

ISBN: 978-987-82912-8-4

```
linea(int i,int j)
{
printf(" %2d",factorial(i)/(factorial(j)*factorial(i-j)));
if (j<i)
     linea(i,j+1);
return 0;
}

triangulo(int n)
{
if (n>0)
{
     triangulo(n-1);
     linea(n-1,0);
     printf(" \n");
}
return 0;
}

main()
{
printf("GENERA EL TRIÁNGULO DE PASCAL RECURSIVAMENTE\n");
while (toupper(continuar)=='S')
{
      do
     {
               printf("INGRESE EL NÚMERO DE LÍNEAS (N > 0): ");
               scanf("%d",&numero);
               }
     while (numero<=0);
     triangulo(numero);
     printf("\n\n  EJECUTAR NUEVAMENTE EL PROGRAMA (S/N) ? ");
     scanf(" %c",&continuar);
}
}
```

ISBN: 978-987-82912-8-4

4.5. Ejercicios Propuestos

- Escriba un programa en Lenguaje C que permita determinar cuál es el mayor elemento en un vector.
- Escriba un programa en Lenguaje C que permita determinar ordenar descendentemente un vector.
- Escriba un programa en Lenguaje C que permita encontrar un elemento en un vector y decir cuantas veces se repite en el arreglo.
- Escriba un programa en Lenguaje C que permita determinar cuál es el mayor elemento en cada una de las filas de una matriz.
- Escriba un programa en Lenguaje C que permita determinar ordenar descendentemente los elementos de cada columna de una matriz.
- Escriba un programa que dada una matriz cuadrática multiplique las filas por las columnas y el resultado se almacene en un vector ordenado descendentemente.

Funciones:

- Resuelva los problemas anteriores e implemente las soluciones mediante funciones.
- Escriba un programa en Lenguaje C que permita determinar la solución de una ecuación cuadrática cuyos datos sean enviados a una función por parámetros.
- Escriba un programa en Lenguaje C que permita leer 10 números enteros por teclado y guardarlos en un array. Calcula y muestra la media de los números que estén en las posiciones pares del array.
- Escriba un programa en Lenguaje C que permita leer por teclado la nota de los alumnos de una clase y calcular la nota media del grupo. Mostar los alumnos con notas superiores a la media.
- Escriba un programa en Lenguaje C que permita ingresar el nombre y sueldo de 20 trabajadores y muestre el nombre con el sueldo del trabajador que más gana.
- Escriba un programa en Lenguaje C que permita generar los N primeros múltiplos de X comprendidos entre los valores A y B
- Escriba un programa en Lenguaje C que permita calcular el rango de los números primos que existen entre varios valores ingresados por el usuario
- Escriba un programa en Lenguaje C que permita determinar los puntos de coordenadas enteras que están sobre la circunferencia X2 + Y2 = R2

ISBN: 978-987-82912-8-4

- Dado un vector x de n elementos reales, donde n es impar, diseñar una función que calcule y devuelva la mediana de ese vector. La mediana es el valor tal que la mitad de los números son mayores que el valor y la otra mitad son menores.

- Se trata de resolver el siguiente problema escolar. Dadas las notas de los alumnos de un colegio en el primer curso de bachillerato, en las diferentes asignaturas (5, por comodidad), se trata de calcular la media de cada alumno, la media de cada asignatura, la media total de la clase y ordenar los alumnos por orden decreciente de notas medias individuales.

- Escribir un programa en lenguaje C para la consulta de teléfonos. Leer un conjunto de datos de mil nombres y números de teléfono de un archivo que contiene los números en orden aleatorio. Las consultas han de poder realizarse por nombre y por número de teléfono.

- Una compañía de turismo de transporte cuenta con diez choferes, de los cuales se conoce: nombre, horas trabajadas cada día de la semana (seis días) y sueldo por hora. Realice un algoritmo que:
    - Calcule el total de horas trabajadas a la semana para cada chofer.
    - Calcule el sueldo semanal para cada uno de ellos.
    - Calcule el total que pagará la empresa de turismo.
    - Indique el nombre del chofer que labora más horas el día lunes.
    - Imprima un reporte de los datos calculados anteriormente.

ISBN: 978-987-82912-8-4

BIBLIOGRAFÍA

- Moreno Pérez, J. C. (2015). Programación. Madrid España, RA-MA Editorial.
- Juganaru Mathieu, M. (2015). Introducción a la programación. Grupo Editorial Patria. Mexico.
- Oviedo Regino, E. M. (2015). Lógica de programación orientada a objetos. Bogotá, Colombia: Ecoe Ediciones.
- Joyanes Aguilar, L. (2013), Fundamentos Generales De programación. McGraw-Hill España.
- Joyanes Aguilar, L. (2005). Programación en C: metodología, algoritmos y estructura de datos (2a. ed.).. McGraw-Hill España.
- Joyanes Aguilar, L. (2006). Programación en C++: un enfoque práctico. McGraw-Hill España.
- Moreno Pérez, J. C. (2014). Programación en lenguajes estructurados. Paracuellos de Jarama, Madrid España, RA-MA Editorial.
- Ceballos Sierra, F. J. (2018). Programación orientada a objetos con C++ (5a. ed.). Paracuellos de Jarama, Madrid, RA-MA Editorial.
- Rodríguez Corral, J. M. y Galindo Gómez, J. (2014). Aprendiendo C (3a. ed.). Cádiz, Spain: Servicio de Publicaciones de la Universidad de Cádiz.
- Cortés, J. L. (2012). El ciclo «C». Barcelona, Spain: Herder Editorial.
- Garrido Carrillo, A. (2005). Fundamentos de programación en C++. Las Rozas (Madrid), Delta Publicaciones.
- Garrido Carrillo, A. y Valdivia Joaquín, F. (2006). Abstracción y estructura de datos en C++. Las Rozas (Madrid), Delta Publicaciones.
- Gil Montero, R. (2020). Aprendiendo a resolver problemas con C++. Córdoba, Jorge Sarmiento Editor - Universitas.
- Eslava Muñoz, V. J. (2016). Aprendiendo a programar paso a paso con C. Madrid, Spain: Bubok Publishing S.L.
- Schildt, H.(2000). The Complete Reference. Cuarta Edición. McGraw Hill.
- Seacord, R. (2020). Effective C: An Introduction to Professional C Programming. No Starch Press.
- Szuhay, J. (2022). Learn C Programming: A beginner's guide to learning the most powerful and general-purpose programming language with ease, Segunda Edición, Packt Publishing.

- Martín Villalba, C. Urquía Moraleda, A. & Rubio González, M. Á. (2021). Lenguajes de programación. UNED - Universidad Nacional de Educación a Distancia. https://elibro.net/es/lc/espoch/titulos/184827

ISBN: 978-987-82912-8-4

ISBN: 978-987-82912-8-4

DE LOS AUTORES

JOSÉ ENRIQUE GUERRA SALAZAR

Tecnólogo en Informática Aplicada, Ingeniero en Electrónica y Computación, Magíster en Docencia Universitaria e Investigación Educativa, Máster en Diseño de Sistemas Electrónicos Mención: Sistemas Electrónicos programables para aplicaciones específicas.

Docente de la Escuela Superior Politécnica de Chimborazo

MARCO VINICIO RAMOS VALENCIA

Ingeniero en Sistemas Informáticos, Magíster en Interconectividad de Redes, Máster Universitario en Ingeniería de Software y Sistemas Informáticos.

Docente de la Escuela Superior Politécnica de Chimborazo

GEOVANNY E. VALLEJO VALLEJO.

Tecnólogo en Informática Aplicada, Licenciado en Ciencias de la Educación Mención informática Educativa, Doctor en Ciencias de la Educación Mención Informática Educativa, Magister en Docencia Universitaria e Investigación Educativa, Máster en Diseño de Sistemas Electrónicos Mención: Sistemas Electrónicos programables para aplicaciones específicas.

Docente de la Escuela Superior Politécnica de Chimborazo.

PUERTO MADERO
EDITORIAL

ISBN 978-987-82912-8-4