



UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Córdoba

Informática I

Guía de Trabajos Prácticos

Departamento de Ingeniería Electrónica

Córdoba, 2023

Informática I

Profesor Titular: Luis E. Toledo
Profesor Asociado: Claudio J. Paz
Profesora Adjunta: Silvia J. Arias

JTP: Norma L. Mascietti
JTP: Silvia A. Carrera
Aux: Martín Nievas

Estructura de una computadora.

1 Ejercicios:

1.1 Clasifique los siguientes artículos como hardware o software:

- Procesador
- RAM
- Zinjai
- Preprocesador
- Impresora
- Explorador de Internet

1.2 Responda las siguientes preguntas

- a) Los programas en C normalmente son escritos en un:
- b) ¿Que programa es el encargado de combinar la salida del compilador, con las funciones de bibliotecas para producir el archivo ejecutable?
- c) ¿Cual es el programa que carga en memoria el programa ejecutable, desde el disco?
- d) ¿Como se llama el programa encargado de convertir los programas escritos en lenguajes de alto nivel, al lenguaje de máquina?
- e) ¿Quien es el encargado de colocar un programa en memoria para que pueda ser ejecutado?
- f) ¿Cual es la unidad mínima de información en una computadora?

1.3 Raspberry pi

La Raspberry Pi puede considerarse como una computadora portátil, un ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Raspberry Pi Foundation, con el objetivo de estimular la enseñanza de informática en las escuelas. Busque en internet la información necesaria correspondiente al código de colores de una resistencia, y describa en no mas de una hoja:

- a) Cuales son las diferentes versiones de la raspberry pi?
- b) Cuales son sus dimensiones y cuales son sus periféricos?
- c) Que sistema operativos puede utilizar?

d) Mencionar un proyecto que pueda ser elaborado con esta placa.



Figura 1: Consola retropie realizada por “I Like To Make Stuff”

Sistemas de numeración y representación numérica. Aritmética binaria.

2.1 Completar los espacios en blanco:

Decimal	Binario	Hexadecimal
0	00000000	0x00
158		_____
145	_____	_____
_____	1010 1110	_____
_____	0011 1100	_____
_____	1111 0001	_____

2.2 Suma de números (2.4)

Realizar las siguientes operaciones, considerando números signados. Expresar el resultado en decimal.

$$\begin{aligned}0x3D + 0x40 &= ______ \\0x3D - 0x20 &= ______ \\01110110 - 00101101 &= ______ \\00011110 + 01101101 &= ______\end{aligned}$$

2.3 Suma de números (2.4)

Sin convertir a binario, sumar los siguientes números.

$$\begin{aligned}0x605C + 0x5 &= ______ \\0x605C - 0x20 &= ______ \\0x605C + 32 &= ______ \\0x60fA - 0x605C &= ______\end{aligned}$$

2.4 Conversiones

Realizar las siguientes conversiones:

- Convierta el binario 110011010010 en octal y en hexadecimal.
- Convierta el número 0xFACE hexadecimal a binaria.
- Convierte octal 3016 a binario.
- Convierta 0xCAFE hexadecimales a octal.
- Convertir binario 1001010 a decimal.

- Convertir octal 513 a decimal.
- Convierta hexadecimal AFD4 a decimal.
- Convierta el decimal 177 a binario, a octal y a hexadecimal.
- Calcule la representación binaria del decimal 417. Luego el complemento de 417 y el complemento de dos de 417.
- ¿Cuál es el resultado cuando un número y su complemento de dos se suman?

2.5 Operaciones Booleanas (2.8)

Completar los espacios, evaluando las operaciones Booleanas:

Operación	Resultado
a	[01101010]
b	[11000101]
$\sim a$	_____
$\sim b$	_____
$a \& b$	_____
$a b$	_____
$a \wedge b$	_____

2.6 Operadores binarios (2.14)

Realizar las siguientes operaciones binarias, teniendo en cuenta los números anteriormente utilizados.

Expresión	Valor	Expresión	Valor
$a \& b$	_____	$a \&\& b$	_____
$b b$	_____	$a b$	_____
$\sim a \sim b$	_____	$!a !b$	_____

2.7 Desplazamiento de bits (2.16)

Completar con las representaciones correspondientes para cada desplazamiento.

a		Lógico a<<2		Lógico a>>3	
Hex	Binario	Hex	Binario	Hex	Binario
0xD4	_____	_____	_____	_____	_____
0x64	_____	_____	_____	_____	_____
0xD4	_____	_____	_____	_____	_____
0x44	_____	_____	_____	_____	_____

2.8 Truncado

En la siguiente tabla escribir las equivalencias de los números suponiendo que sólo se tienen en cuenta 3 bits

Hex		No signado		Complemento a 2	
Original	Truncado	Original	Truncado	Original	Truncado
0x1	1	1	_____	1	_____
0x3	3	3	_____	3	_____
0x5	_____	5	_____	5	_____
0xC	_____	12	_____	12	-4
0xE	6	14	_____	14	-2

2.9 Suma

Realizar las siguientes sumas utilizando números signados de 4 bits:

- $-8 - 5 =$
- $-8 + 5$
- $2 + 5$
- $5 + 5$

2.10 Interpretación

Realizar las siguientes sumas e interpretar el resultado como números signados o no signados:

x	y	bin	x+y (4 bits)	x+y (5bits)
11000	11000	_____	_____	_____
10111	01000	_____	_____	_____
00010	00101	_____	_____	_____
01100	00100	_____	_____	_____

Introducción al lenguaje C.

3.1 Completar los espacios en blanco

1. Todo programa en C comienza con la ejecución de la función _____ .
2. Todos los cuerpos de las funciones comienzan con _____ y terminan con _____
3. Todas las declaraciones terminan con _____ .
4. La función _____ de la biblioteca estándar permite mostrar información en la pantalla.
5. La función _____ de la biblioteca estándar permite ingresar información desde el teclado.

3.2 Programas

Escribir un programa en C que resuelva los siguientes problemas (uno por cada enunciado)

- Definir las variables, “unavARIABLE”, “p345”, “numero”.
- Escribir un mensaje en pantalla, solicitando al usuario ingresar un número entero. Se debe imprimir la solicitud, en la cual al final debe incluir dos puntos y dejarse un espacio.
- Solicitar al usuario ingresar un número entero y almacenarlo en una variable llamada “num”.
- Imprimir en pantalla el mensaje “Buen día”.

3.3 Ejercicio

Realizar, para cada enunciado, un programa en C.

- Asignar la suma de las variables **a** y **b** en **c**.
- Leer 3 números enteros desde el teclado, y almacenarlos en las variables **p**, **q** y **r**.

3.4 Verdadero o falso

Indicar si las siguientes afirmaciones son verdaderas o falsas. Si son falsas, indicar porqué:

- La función **printf** siempre imprime al comienzo de una nueva línea.

- Los comentarios son mostrados en la pantalla cuando el programa se está ejecutando.
- La secuencia de escape `\n` en la función **printf**, provoca un salto de línea.
- Todas las variables deben definirse antes de ser utilizadas.
- El operador de resto (`%`) solamente puede ser utilizado en números enteros.
- Las variables **numero** y **NUmero** son idénticas.
- Los operadores aritméticos `/,*,+,-,%` tienen todos el mismo nivel de precedencia.

3.5 Ejercicio 3

Escribir un algoritmo para calcular la distancia recorrida (m) por un móvil que se desplaza con velocidad constante (m/s) durante un tiempo (s). La velocidad y el tiempo serán ingresadas por el usuario.

3.6 Ejercicio 4

Escribir un algoritmo para obtener el promedio simple de un estudiante a partir de las tres notas parciales. Las notas serán introducidas una a una por el usuario.

3.7 Ejercicio 5

En un local se hace un descuento del `%20` cuando la compra supera los \$ 1000. Escribir un algoritmo que calcule el precio a pagar por el cliente teniendo como dato el valor de la compra.

3.8 Ejercicio 6

Escribir un algoritmo que determine si un número n tiene tres cifras. El usuario debe ingresar el número n .

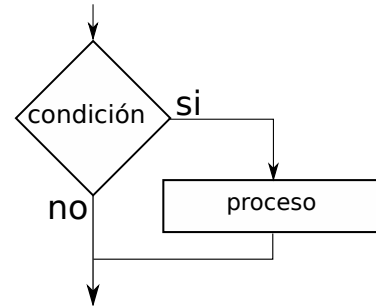
3.9 Ejercicio 7

Escribir un algoritmo que solicite ingresar dos números $n1$ y $n2$. Si el primero es mayor que el segundo mostrar la suma de ambos, por otro lado si el segundo es mayor al primero, mostrar el producto entre los números. En caso de que sean iguales imprimir “Los números son iguales”.

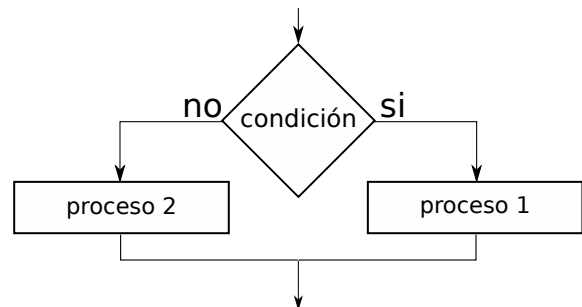
Introducción a la programación estructurada.

4 Repaso

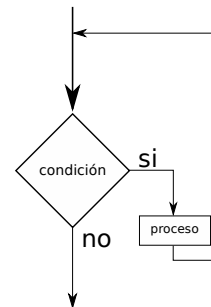
si *condición entonces*
 proceso
fin si



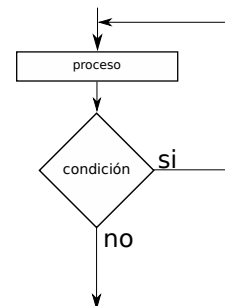
si *condición entonces*
 proceso 1
si no
 proceso 2
fin si



mientras *condición*
 proceso
fin mientras



hacer
 proceso
mientras *condición*

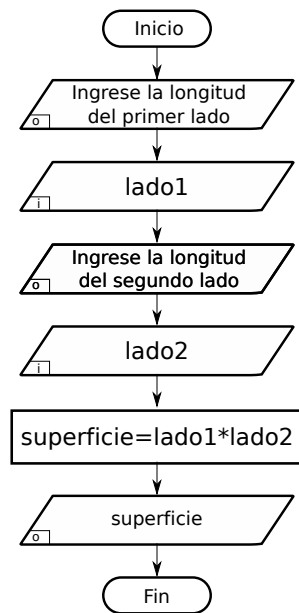


4.1 Ejercicio

Realizar un programa que solicite la dimensión en cm de lados de un rectángulo y muestre la superficie del mismo

Solución

```
imprimir: Ingrese la longitud del primer lado  
leer:lado1  
imprimir: Ingrese la longitud del segundo lado  
leer: lado2  
superficie = lado1 * lado2  
imprimir: superficie
```

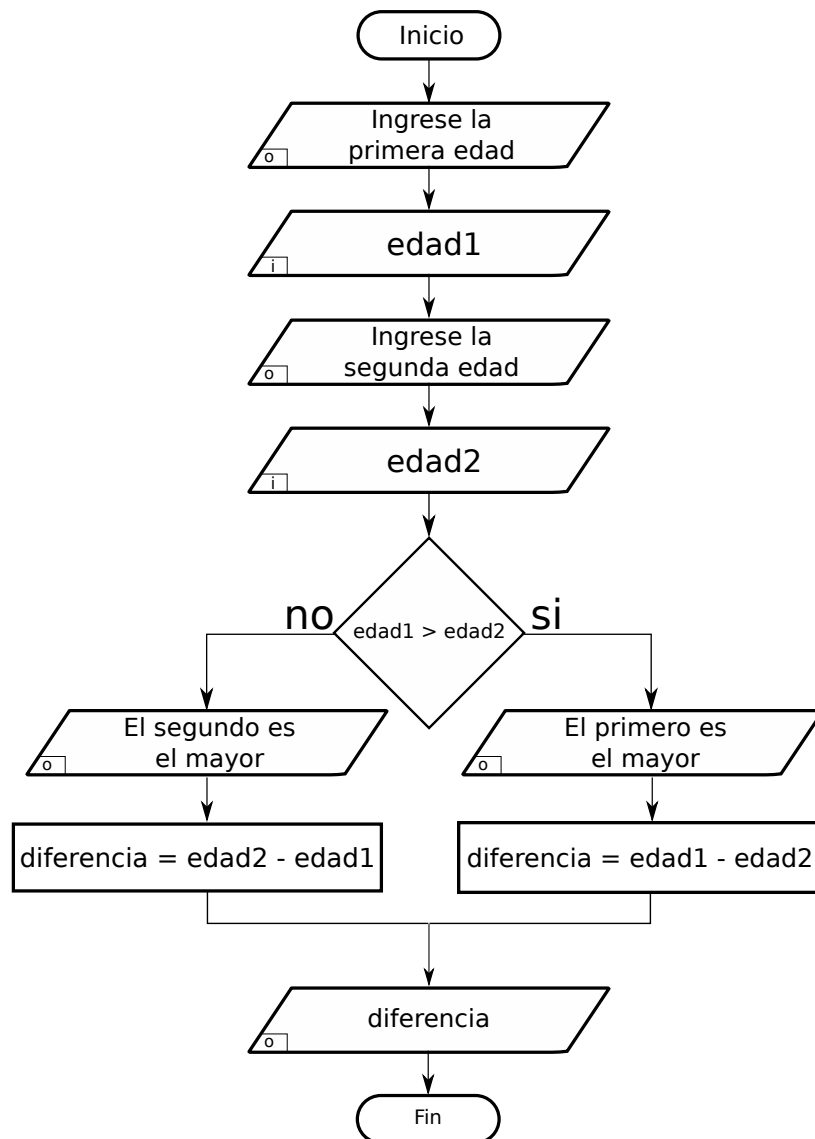


4.2 Ejercicio

Escribir un algoritmo que solicite la edad de dos hermanos, muestre un mensaje indicando el mayor y su diferencia.

Solución

```
imprimir: Ingrese la primera edad
leer: edad1
imprimir: Ingrese la segunda edad
leer: edad2
si edad1 > edad2 entonces
    imprimir: El primero es el mayor
si no
    imprimir: El segundo es el mayor
imprimir: diferencia
```



4.3 Ejercicio

Escribir un algoritmo que imprima los números enteros desde el 0 hasta N . Donde el número N es ingresado por el usuario.

```

imprimir: Ingrese N
leer: cantidad
para contador desde 0 hasta cantidad hacer
imprimir: contador
fin para
  
```

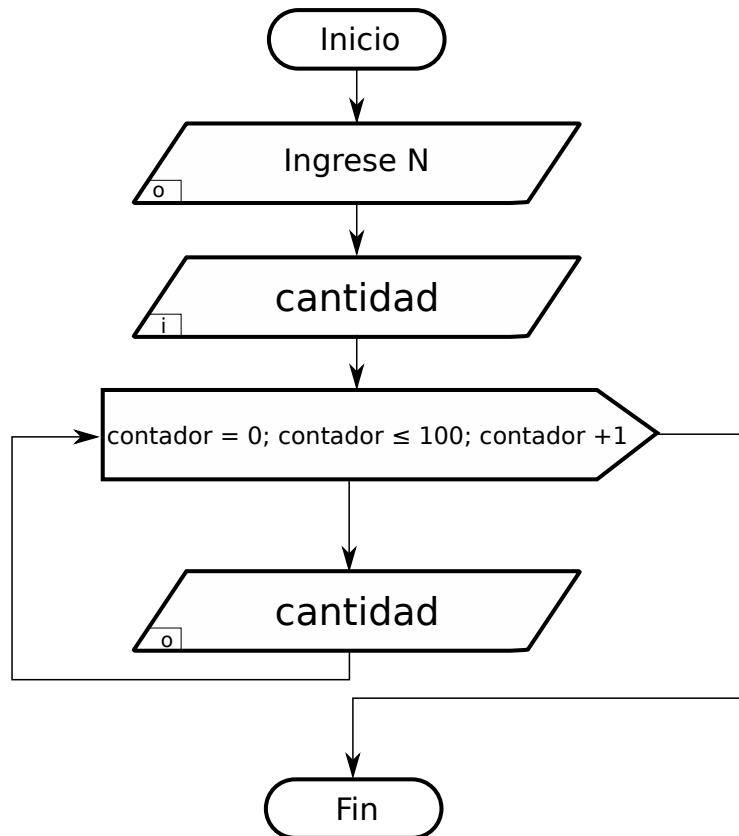


Figura 2: ejercicio 2 versión con bloque **para**

4.4 Preguntas:

Completar los espacios en blanco:

- Todos los programas pueden ser escritos en términos de 3 secuencias de control: _____ , _____ , _____ .
- La secuencia de control _____ es utilizada para ejecutar una acción cuando es verdadera y otra acción cuando es falsa.
- La secuencia _____ especifica que una o varias sentencias se ejecutarán repetidamente, mientras la condición sea verdadera.

4.5 Ejercicio

Escribir un algoritmo para calcular la distancia recorrida (m) por un móvil que se desplaza con velocidad constante (m/s) durante un tiempo (s). La velocidad y el tiempo serán ingresadas por el usuario.

4.6 Ejercicio

Escribir un algoritmo para obtener el promedio simple de un estudiante a partir de las tres notas parciales. Las notas serán introducidas una a una por el usuario.

4.7 Ejercicio

En un local se hace un descuento del %20 cuando la compra supera los \$ 1000. Escribir un algoritmo que calcule el precio a pagar por el cliente teniendo como dato el valor de la compra.

4.8 Ejercicio

Escribir un algoritmo que determine si un número n tiene tres cifras. El usuario debe ingresar el número n .

4.9 Ejercicio 7

Escribir un algoritmo que solicite ingresar dos números $n1$ y $n2$. Si el primero es mayor que el segundo mostrar la suma de ambos, por otro lado si el segundo es mayor al primero, mostrar el producto entre los números. En caso de que sean iguales imprimir “Los números son iguales”.

4.10 Ejercicio 8

Escribir un programa que calcule la potencia $x = num1^{num2}$, donde $num1$ y $num2$ son números enteros positivos ingresados por el usuario.

4.11 Ejercicio 9

Escribir un programa que calcule la suma de los primeros n números. El número n es un entero positivo, ingresado por el usuario.

4.12 Ejercicio 10

Escribir un algoritmo que imprima los número impares desde 0 hasta N . Donde N es ingresado por el usuario.

4.13 Ejercicio 11

Escribir un algoritmo que determine la temperatura promedio de N mediciones de temperatura. El usuario debe ingresar la cantidad N y las N mediciones.

4.14 Ejercicio 12

Escribir un algoritmo que determine el mayor de 10 números ingresados. El usuario debe ingresar cada uno de los 10 números.

4.15 Ejercicio 13

Escribir un algoritmo que determine el mayor de N números positivos ingresados. El usuario debe ingresar cada uno de los N números. Para terminar se debe ingresar un -1.

4.16 Ejercicio 14

Escribir un algoritmo que solicite ingresar N calificaciones de alumnos y determine la cantidad de aprobados, desaprobados y promocionados. El usuario debe ingresar el número N y las N calificaciones.

Según:

- Desaprobado: nota menor a 6
- Aprobado: nota mayor o igual a 6
- Promocionado: nota mayor o igual a 8

4.17 Ejercicio 15

Escribir de cuatro formas diferentes, sentencias en C que incrementen en 1 una variable X.

Control de flujo en lenguaje C.

5 if, if else

5.1 Solución 0

```
#include <stdio.h>

int main(void)
{
    int num;

    printf("Ingrese un número entero: ");
    scanf("%d", &num);

    if (num % 5 == 0)
    {
        printf("El número %d es múltiplo de 5\n", num);
    }
    else
    {
        printf("El número %d no es múltiplo de 5\n", num);
    }
    return 0;
}
```

```
Ingrese un número entero: 95
El número 95 es múltiplo de 5

Ingrese un número entero: 74
El número 74 no es múltiplo de 5
```

5.2 Ejercicio 1

Realizar un programa que determine el mayor entre dos números ingresados por el usuario

```
Ingrese el primer número: 1
Ingrese el segundo número: 4
El mayor es el segundo
```

5.3 Ejercicio 2

Realizar un programa que determine el mayor entre tres números ingresados por el usuario. En caso de que tres o dos números sean iguales y los mayores, es indistinta la elección del mayor.

```
Ingrese el primer número: 12
Ingrese el segundo número: 3
Ingrese el tercer número: 4
El mayor es el primero

Ingrese el primer número: 12
Ingrese el segundo número: 2
Ingrese el tercer número: 12
El mayor es el tercero
```

```
Ingrese el primer número: 23
Ingrese el segundo número: 23
Ingrese el tercer número: 23
El mayor es el tercero
```

5.4 Ejercicio 3

Realizar un programa que controle el encendido de un ventilador en base a la temperatura de encendido y la temperatura ambiente proporcionada por el usuario. Se debe imprimir en pantalla el estado del ventilador luego de ingresar los datos.

```
Ingrese la temperatura de encendido: 23
Ingrese la temperatura ambiente: 24
Ventilador encendido

Ingrese la temperatura de encendido: 30
Ingrese la temperatura ambiente: 10
Ventilador apagado
```

5.5 Ejercicio 4

Realizar un programa que solicite ingresar las notas de dos parciales. Si desaprobó un parcial el programa debe solicitar ingresar la nota del recuperatorio. En base a las notas obtenidas calcular el promedio y determinar la condición académica (Promoción mayor 8, desaprobado menor a 6, lo demás aprobado). Solo se permite recuperar un solo parcial. La nota del recuperatorio se promedia con las notas del parcial y está permitido promocionar si recuperó.

```
Ingrese la primera nota: 9
Ingrese la segunda nota: 6
Aprobado

Ingrese la primera nota: 8
Ingrese la segunda nota: 8
Promocionado

Ingrese la primera nota: 5
Ingrese la segunda nota: 10
Ingrese la nota del recuperatorio: 10
Promocionado
```

5.6 Ejercicio 5

Realizar un programa que dada la duracion en minutos de una llamada, permita calcular el costo, considerando:
-Hasta tres minutos el costo es 0.50 por minuto -Por encima de tres minutos es 1.5 fijo más 0.2 por cada minuto adicional a los tres primeros

```
Ingrese los minutos de la llamada: 3
El costo de la llamada es: 1.500

Ingrese los minutos de la llamada: 4
El costo de la llamada es: 1.700
```

Operadores II, &&

Realizar un programa que determine si 3 números ingresados son distintos entre ellos.

5.7 Solución 0

```
#include <stdio.h>

int main(void)
{
    int num;

    printf("Ingrese un número entero: ");
    scanf("%d", &num);

    if (num % 5 == 0)
    {
        printf("El número %d es múltiplo de 5\n", num);
    }
    else
    {
        printf("El número %d no es múltiplo de 5\n", num);
    }
    return 0;
}
```

5.8 Ejercicio 1

Realizar un programa que determine si un número ingresado por el usuario está en el rango 10-100. En caso de no estar en el rango, el programa debe informarlo y terminar. Si el número está dentro del rango, el programa debe determinar si el número es par.

```
Ingrese un número: 30
El número 30 es par

Ingrese un número: 200
El número ingresado no está dentro del rango
```

5.9 Ejercicio 2

Escribir un programa que determine el mayor de 3 números ingresados. Si los números son iguales, se debe imprimir un mensaje que lo indique.

```
Ingrese el primer número: 42
Ingrese el segundo número: 25
Ingrese el tercer número: 342
El mayor es el tercero

Ingrese el primer número: 2
Ingrese el segundo número: 2
Ingrese el tercer número: 2
Los números ingresados son iguales
```

5.10 Ejercicio 3

Realizar un programa que solicite ingresar el valor nominal de resistencia (en ohms) y la tolerancia (valor entero en porcentaje). Una vez cargados estos valores, el programa debe solicitar al usuario que ingrese un valor de resistencia real y determine si la misma está dentro de los márgenes de tolerancia.

```
Ingrese el valor de la tolerancia: 10
Ingrese el valor nominal: 100
Ingrese el valor medido de la resistencia: 102
El valor se encuentra dentro de la tolerancia
```

```
ngrese el valor de la tolerancia: 10
Ingrese el valor nominal: 100
Ingrese el valor medido de la resistencia: 120
El valor se encuentra fuera de la tolerancia
```

5.11 Ejercicio 4

Realizar un programa que determine si el caracter ingresado es un número o una letra. Ayuda: buscar “tabla ASCII”.

```
Ingrese el caracter: 2
El caracter ingresado es un número
```

```
Ingrese el caracter: m
El caracter ingresado no es un número
```

5.12 Ejercicio 5

Modificar el programa anterior para que ahora solicite ingresar el color de las bandas de colores y la medición real de la resistencia. Con estos valores el programa debe determinar si se encuentra dentro de los valores de tolerancia o no. Considerar el caso de resistencias de 4 bandas de colores, donde las primeras tres indican el valor de resistencia y el cuarto la tolerancia (dorado ± 5 , plateado ± 10 , rojo ± 2 y marrón ± 1). Los colores de las bandas serán ingresados en forma de caracteres. Cada caracter representa un color.

- Negro
- Marrón
- Rojo
- naranJa
- Amarillo
- Verde
- aZul
- vioLeta

- Gris
- Blanco
- Dorado
- Plateado

```
Color primera banda: N
Color segunda banda: M
Color tercera banda: R
Color cuarta banda: D
Valor medido(en ohms): 1002
El valor está dentro de la tolerancia

Color primera banda: N
Color segunda banda: M
Color tercera banda: R
Color cuarta banda: D
Valor medido(en ohms): 1302
El valor está fuera de la tolerancia
```

Estructura repetitiva for

Todos los ejercicios deben utilizar al menos una estructura **for**. Algunos ejercicios pueden requerir utilizar estructuras de condición.

5.13 Ejercicio 0

Realizar un programa que imprima los números desde el 5 hasta el 0 y luego vuelva hasta el 5 como en el siguiente ejemplo (se debe utilizar al menos una estructura for)

5.14 Solución 0

```
#include <stdio.h>

int main(void)
{
    int num;

    printf("Ingrese un número entero: ");
    scanf("%d", &num);

    if (num % 5 == 0)
    {
        printf("El número %d es múltiplo de 5\n", num);
    }
    else
    {
        printf("El número %d no es múltiplo de 5\n", num);
    }
}
```

```

    }
    return 0;
}

```

```

Ingrese un número entero: 95
El número 95 es múltiplo de 5

Ingrese un número entero: 74
El número 74 no es múltiplo de 5

```

5.15 Solución 0 (con un solo for)

```

#include <stdio.h>

int main(void)
{
    int i;

    for (i = 5; i > -6; i--)
    {
        if (i > 0)
            printf("%d ", i);
        else
            printf("%d ", -i);
    }

    return 0;
}

```

```

Ingrese un número entero: 95
El número 95 es múltiplo de 5

Ingrese un número entero: 74
El número 74 no es múltiplo de 5

```

5.16 Preguntas

- La iteración controlada por contador también se conoce como iteración _____ porque se sabe de antemano cuántas veces se ejecutará el bucle.
- La iteración controlada por centinela también se conoce como iteración _____ porque no se sabe de antemano cuántas veces se ejecutará el bucle.
- En la iteración controlada por contador, se usa un(una) _____ para contar el número de veces que se debe repetir un grupo de instrucciones.

5.17 Analizar código

Encontrar el error en las siguientes códigos:

1. .

```
x = 1;
while (x <= 10);
    ++x;
}
```

2. .

```
for (double y = .1; y != 1.0; y += .1) {
    printf("%f\n", y);
}
```

5.18 Ejercicio 1

Modificar el programa anterior para que imprima la progresión de números partiendo de un número n positivo ingresado por el usuario.

```
Ingrese el número "n": 9
9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
```

5.19 Ejercicio 2

Realizar un programa que utilice una estructura **for** e imprima la siguiente salida:

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

5.20 Ejercicio 3

Realizar un programa que utilice una estructura **for** e imprima la siguiente tabla:

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

5.21 Ejercicio 4

Realizar un programa que utilice dos estructuras **for** para imprimir una matriz, donde la cantidad de filas y columnas son ingresados por el usuario como la siguiente:

```
Cantidad de filas: 3
Cantidad de columnas: 5
0  1  2  3  4
5  6  7  8  9
10 11 12 13 14
```


5.22 Ejercicio 5

Escribir un programa que imprima todos los números enteros pares entre el 0 y n , donde n es un número entero ingresado por el usuario.

```
Ingrese el número "n": 23
0 2 4 6 8 10 12 14 16 18 20 22
```

5.23 Ejercicio 6

Escribir un programa que determine el mayor de 10 números enteros ingresados por el usuario.

```
Ingrese el número (1): -2
Ingrese el número (2): -24
Ingrese el número (3): 2
Ingrese el número (4): 13
Ingrese el número (5): 41
Ingrese el número (6): 9
Ingrese el número (7): 20
Ingrese el número (8): 32
Ingrese el número (9): 42
Ingrese el número (10): 4
El mayor número ingresado es: 42
```

5.24 Ejercicio 7

Modificar el programa del ejercicio anterior para que determine también el mínimo número ingresado.

```
Ingrese el número (1): -123
Ingrese el número (2): -2
Ingrese el número (3): 1241
Ingrese el número (4): 1343
Ingrese el número (5): -3253
Ingrese el número (6): -2
Ingrese el número (7): 4
Ingrese el número (8): 0
Ingrese el número (9): 324
Ingrese el número (10): 5
El mayor número ingresado es: 1343
El menor número ingresado es: -3253
```

5.25 Ejercicio 8

Realizar un programa que calcule la tabla de multiplicar de un número n ingresado por el usuario.

```
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
```

5.26 Ejercicio 9

Realizar un programa que calcule el factorial de un número n ingresado por el usuario. Donde $0 < n < 10$.

```
Ingrese el número "n": 5
El factorial de 5 es :120
```

5.27 Ejercicio 10

Realizar un programa que calcule la potencia m de número n , donde m y n son ingresado por el usuario. Operación: n^m .

```
Ingrese el número "n": 2
Ingrese el número "m": 4
2 elevado a la 4 es :16
```

Estructuras repetitivas while y do..while

Todos los ejercicios deben utilizar al menos una estructura **for**. Repetir ahora utilizando estructura **while** y **do..while**.

Arreglos en lenguaje C.

Arreglos unidimensionales

Los ejercicios de ésta guía utilizan **arreglos**, algún tipo de estructura repetitiva (**for**, **while** o **do...while**) y estructuras de control **if...else**

6.1 Ejercicio 1

Completar el siguiente programa, para que solicite al usuario ingresar 10 números enteros, los almacene en un arreglo. Luego, se debe recorrer el arreglo convirtiendo los números negativos en positivos. Por último mostrar en pantalla el arreglo.

```
#include <stdio.h>
#define N 5

int main(void)
{
    int i;
    int a[N];

    /** Completar desde aquí */

    /** hasta aquí, la cantidad de líneas que quiera */

    return 0;
}
```

```
Ingrese el elemento[0]: 1
Ingrese el elemento[1]: -2
Ingrese el elemento[2]: -3
Ingrese el elemento[3]: 141
Ingrese el elemento[4]: -12
Los elementos del arreglo son:
a[0]: 1
a[1]: 2
a[2]: 3
a[3]: 141
a[4]: 12
```

6.2 Ejercicio 2

Escribir un programa que solicite al usuario ingresar N elementos de un arreglo. Luego recorrer el arreglo, buscar el mayor elemento e imprimirlo.

```
Ingrese el elemento[0]: 1
Ingrese el elemento[1]: -12
Ingrese el elemento[2]: 41
Ingrese el elemento[3]: 44
Ingrese el elemento[4]: 5
El mayor elemento del arreglo es: 44
```

6.3 Ejercicio 3

Escribir un programa que solicite ingresar N elementos de un arreglo. Luego, en otro arreglo, almacenar el valor acumulado del primer arreglo. Por ejemplo: el elemento del segundo arreglo $b[0]$ será $a[0]$, el elemento $b[1]$ contendrá la suma $a[0]+a[1]$, el elemento $b[3]$ contendrá la suma $a[0]+a[1]+a[2]+a[3]$

```
Ingrese el elemento a[0]: 1
Ingrese el elemento a[1]: 2
Ingrese el elemento a[2]: 3
Ingrese el elemento a[3]: 4
Ingrese el elemento a[4]: 5
Los elementos del arreglo b son:
b[0]: 1
b[1]: 3
b[2]: 6
b[3]: 10
b[4]: 15
```

6.4 Ejercicio 4

Realizar un programa que solicite al usuario ingresar N elementos de un arreglo. Los valores a ingresar tienen que estar en el rango de (1-100), en caso de ingresar un valor fuera del rango, se debe volver a pedir el valor.

```
Ingrese el elemento a[0]: 1
Ingrese el elemento a[1]: 323
Ingrese el elemento a[1]: 23
Ingrese el elemento a[2]: -12
Ingrese el elemento a[2]: 32
Ingrese el elemento a[3]: 4
Ingrese el elemento a[4]: 9
Los elementos del arreglo son:
a[0]: 1
a[1]: 23
a[2]: 32
a[3]: 4
a[4]: 9
```

6.5 Ejercicio 5

Realizar un programa que solicite ingresar N componentes de dos vectores **a** y **b**. Luego calcular el producto punto entre los mismos. Recordar que el producto punto se puede expresar como:

$$\vec{a} \cdot \vec{b} = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_N \cdot b_N$$

```
Primer vector:
Ingrese el elemento a[0]: 1
Ingrese el elemento a[1]: 4
Ingrese el elemento a[2]: 2
Segundo vector:
Ingrese el elemento b[0]: 5
Ingrese el elemento b[1]: 3
Ingrese el elemento b[2]: 2
El producto punto es: 21
```

6.6 Preguntas

- El número por el cual se refiere a un elemento particular de un arreglo se llama:
- El primer elemento de un arreglo es el número[(completar aquí)].

6.7 Indicar verdadero o falso:

- Un arreglo puede tener elementos de diferentes tipos.
- El índice de un arreglo puede ser de tipo `double`
- En una inicialización de un arreglo, si hay una cantidad menor de elementos que el tamaño del arreglo, C automáticamente inicializa los elementos restantes con el último elemento de la lista de inicialización.
- Es un error si en la inicialización de un arreglo hay mas elementos que el tamaño del arreglo.
- Si hay menos elementos en la lista de inicialización de un arreglo que el tamaño del arreglo, los elementos restantes son inicializados con basura.

Arreglos Bidimensionales

6.8 Ejercicio 1

Escribir un programa que solicite al usuario ingresar un arreglo de dos dimensiones de $N \times M$. N y M son directivas de preprocesador. Al finalizar, el programa debe imprimir la matriz.

6.9 Ejercicio 2

Escribir un programa que solicite ingresar al usuario los elementos de un arreglo bidimensional de **n** filas por **m** columnas. Los valores de **n** y **m** son ingresados por el usuario, los mismos deben ser menores que N y M (directivas de preprocesador) y mayores que 0. Imprimir el arreglo al finalizar.

6.10 Ejercicio 3

Escribir un programa que solicite al usuario ingresar un arreglo de dos dimensiones de $N \times M$. N y M son directivas de preprocesador. Luego se deben copiar los elementos del arreglo se deben copiar a otro arreglo, de tal manera de obtener la matriz transpuesta. Al finalizar, el programa debe imprimir las dos matrices.

6.11 Ejercicio 4

Modificar el programa anterior para obtener el producto de una matriz por su transpuesta. El producto entre las matrices $A * B$ es una matriz C donde:

$$C_{ij} = \sum_{k=1}^m A_{ik} B_{kj}$$

Funciones en lenguaje C.

Funciones 1^{era} Parte

7.1 Ejercicio 1

Completar el siguiente programa:

```
#include <stdio.h>

void intro(void)
{
    printf("Bienvenidos! Comenzando en ");
}

int main(void)
{
    int i;

    /** Completar a partir de aquí */

    /** Hasta aquí */

    return 0;
}
```

```
Bienvenidos! Comenzando en 10...
Bienvenidos! Comenzando en 9...
Bienvenidos! Comenzando en 8...
Bienvenidos! Comenzando en 7...
Bienvenidos! Comenzando en 6...
Bienvenidos! Comenzando en 5...
Bienvenidos! Comenzando en 4...
Bienvenidos! Comenzando en 3...
Bienvenidos! Comenzando en 2...
Bienvenidos! Comenzando en 1...
```

7.2 Ejercicio 2

Completar el siguiente programa:

```
#include <stdio.h>

float ingreso_numero(void)
{
    float num;
    /** Completar desde aquí */

    /** Hasta aquí */
    return num;
}

int main(void)
```

```

{

    printf("Usted ingresó el número %f\n", ingreso_numero());

    return 0;
}

```

Para obtener la siguiente salida:

```

Ingrese un número: 3.14
Usted ingresó el número: 3.140000

```

7.3 Ejercicio 3

Completar el siguiente programa:

```

#include <stdio.h>

void imprime_mayor(int x, int y)
{
    /** Completar desde aquí */

    /** Hasta aquí */
}

int main(void)
{
    int num_1;
    int num_2;

    printf("Ingrese dos números diferentes: ");
    scanf("%d %d", &num_1, &num_2);

    imprime_mayor(num_1, num_2);

    return 0;
}

```

Para obtener la siguiente salida:

```

Ingrese dos números diferentes: 15 92
El número 92 es el mayor

```

7.4 Ejercicio 4

Completar el siguiente programa:

```

#include <stdio.h>

int ret_max(int a, int b)
{
    int max;

```



```

    if (a > b )
        max = a;
    else
        max = b;

    return max;
}

int main(void)
{
    int num_1;
    int num_2;

    /** Completar desde aquí */

    /** Hasta aquí */
    return 0;
}

```

Para obtener la siguiente salida:

```

Ingrese dos números diferentes: 65 35
El número 65 es el máximo

```

7.5 Ejercicio 5

Completar el siguiente programa:

```

#include <stdio.h>

int ret_maximo(int a, int b)
{
    int max;

    if (a > b)
        max = a;
    else
        max = b;

    return max;
}

int main(void)
{
    int num_1, num_2, num_3, num_4;

    printf("Ingrese cuatro números diferentes: ");
    scanf("%d %d %d %d", &num_1, &num_2, &num_3, &num_4);

    /** Completar desde aquí */

    /** Hasta aquí */

```

```
    return 0;
}
```

Para obtener la siguiente salida:

```
Ingrese cuatro números diferentes: 89 79 32 38
El número 89 es el máximo
```

7.6 Ejercicio 6

Completar el siguiente programa:

```
#include <stdio.h>

/** Completar desde aquí */

/** Hasta aquí */

int main(void)
{
    int base, exp;

    printf("Ingrese base y exponente: "); scanf("%d %d", &base
        , &exp);
    printf("El número %d elevado a la %d es: %d\n", base, exp,
        potencia(base,exp));
    return 0;
}
```

Para obtener la siguiente salida:

```
Ingrese base y exponente: 3 2
El número 3 elevado a la 2 es: 9
```

7.7 Ejercicio 7

Completar el siguiente programa:

```
#include <stdio.h>

/** Completar desde aquí */

/** Hasta aquí */

int main(void)
{
    int i, num;

    printf("Ingrese un número: ");
    scanf("%d", &num);

    /** Completar desde aquí */
}
```

```

    /** Hasta aquí */

    return 0;
}

```

Para obtener la siguiente salida:

```

Ingrese un número : 15
Los números primos menores que 15 son :
1
2
3
5
7
11
13

```

7.8 Ejercicio 8

Crear los prototipos para las diferentes funciones que resuelvan:

- Función **hipotenusa**, toma dos argumentos float de doble precisión, **lado1** y **lado2**. Devuelve el resultado en float de doble precisión.
- Función **menor**, toma tres enteros, **x**, **y**, **z**. Devuelve un entero.
- Función **intToFloat**, que toma un argumento entero llamado **numero**, y devuelve el resultado en float.

Funciones 2^{da} Parte

7.9 Ejercicio 1

Implementar la función **fibonacci** en forma recursiva.

```
int fibonacci(int);
```

7.10 Ejercicio 2

El rompecabezas de la Torre de Hanoi fue inventado por el matemático francés Edouard Lucas en 1883. Se inspiró en una leyenda acerca de un templo hindú donde el rompecabezas fue presentado a los jóvenes sacerdotes. Al principio de los tiempos, a los sacerdotes se les dieron tres postes y una pila de 64 discos de oro, cada disco un poco más pequeño que el de debajo.

Su misión era transferir los 64 discos de uno de los tres postes a otro, con dos limitaciones importantes. Sólo podían mover un disco a la vez, y nunca podían colocar un disco más grande encima de uno más pequeño. Los sacerdotes trabajaban muy eficientemente, día y noche, moviendo un disco

cada segundo. Cuando terminaran su trabajo, dice la leyenda, el templo se desmenuzará en polvo y el mundo se desvanecerá.

En la Figura Se puede ver la representación gráfica del juego.

Supongamos que los sacerdotes intentan mover los discos de la clavija 1 a la clavija 3. Deseamos desarrollar un algoritmo que imprima la secuencia precisa de transferencias de clavija de disco a disco. Si tuviéramos que abordar este problema con métodos convencionales, nos encontraríamos rápidamente anudados en la gestión de los discos. En cambio, si atacamos el problema con la recursión en mente, inmediatamente se vuelve manejable. Mover n discos se puede ver en términos de mover solo $n - 1$ discos (y, por lo tanto, la recursividad) de la siguiente manera:

1. Mueva $n - 1$ discos de la clavija 1 a la clavija 2, utilizando la clavija 3 como área de retención temporal.
2. Mueva el último disco (el más grande) de la clavija 1 a la clavija 3.
3. Mueva los discos $n - 1$ de la clavija 2 a la clavija 3, utilizando la clavija 1 como área de retención temporal.

El proceso finaliza cuando la última tarea implica mover $n = 1$ disco, es decir, el caso base. Esto se logra moviendo trivialmente el disco sin la necesidad de un área de retención temporal. Escribe un programa para resolver el problema de Towers of Hanoi. Use una función recursiva con cuatro parámetros:

1. El número de discos a mover
2. La clavija en la que se enroscan estos discos inicialmente
3. La clavija a la que se debe mover esta pila de discos
4. La clavija que se utilizará como área de retención temporal.

Su programa debe imprimir las instrucciones precisas que necesitará para mover los discos desde la clavija de inicio a la clavija de destino. Por ejemplo, para mover una pila de tres discos de la clavija 1 a la clavija 3, su programa debe imprimir la siguiente serie de movimientos:

```
1 -> 3
1 -> 2
3 -> 2
1 -> 3
2 -> 1
2 -> 3
1 -> 3
```

7.11 Ejercicio 3

Modificar el siguiente programa, para que los valores del arreglo sean ingresados con la función **carga**.

```
#include <stdio.h>

#define TAM 500

int main(void)
{
    int i, n;
    int arreglo[TAM];

    printf("Ingrese la cantidad de elementos: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        printf("Ingrese el elemento [%d]: ", i);
        scanf("%d", &arreglo[i]);
    }

    for (i = 0; i < n; i++)
        printf("%d\n", arreglo[i]);

    return 0;
}
```

Prototipo de la función **carga**

```
void carga(int a[], int n);
```

7.12 Ejercicio 4

Modificar el programa anterior para que la impresión del arreglo se realice con la función con prototipo:

```
void imprime(int a[], int n);
```

7.13 Ejercicio 5

Modificar el programa anterior para que el arreglo ingresado se ordene de mayor a menor con la función con prototipo:

```
void ordenar(int a[], int n);
```

7.14 Ejercicio 6

Modificar el programa del Ejercicio 4, para que se imprima la cantidad de números primos en el arreglo. Utilizar los prototipos:

```
int es_primo(int num);  
int contar_primos(int a[], int n);
```

7.15 Ejercicio 7

Modificar el programa del Ejercicio 4, para que se imprima el mayor y el menor de los números en el arreglo, utilizando los prototipos:

```
int mayor(int a[], int n);  
int menor(int a[], int n);
```

Punteros en lenguaje C.

Paso de parámetros por referencia

Se solicita un programa que cumpla con lo siguiente:

- a) Requisito 1 El usuario debe ingresar **N** valores en un arreglo unidimensional. Los valores ingresados solo pueden ser positivos (incluyendo el cero) menores que 100.

Se debe realizar la validación en una función que reciba el parámetro por referencia.

- a) Requisito 2 Una vez cargados los **N** valores se debe controlar que no haya números primos. Si existiesen deben ser incrementados en una unidad. Implementar esto en una función, que debe recibir el valor que debe ser chequeado por referencia.

- a) Requisito 3 Debe ordenarse el arreglo de mayor a menor utilizando el método de la burbuja. Debe implementarse una función llamada *swap* que reciba por referencia los dos elementos del arreglo que deben intercambiarse para lograr el ordenamiento.

- a) Requisito 4 Debe imprimirse el arreglo antes y después del ordenamiento.

El siguiente programa de ejemplo puede servir de ayuda:

```
#include <stdio.h>

#define N 10

void swap ( int * a , int * b);
void validar_positividad ( int * a);
void validar_no_primo ( int * a);

int main ( void) {

    int arreglo [ N ];
    /* completar */

    return 0;
}

/* completar */
```

Estructuras y uniones en C. Campos de bit.

Estructuras

9.1 Ejercicio 1

Crear las estructuras según los siguientes requerimientos:

- Una estructura `inventario` que contenga un arreglo de 30 char llamado `nombre`, un entero `numero_parte`, flotante `precio`, y un entero `stock`
- Una estructura que contenga los siguientes arreglos: un arreglo de 20 elementos para el nombre de la calle `direccion`, un arreglo `ciudad` de 25 elementos para el nombre de la ciudad, un arreglo `codpost` de 6 elementos para el código postal.

Dentro de `main` se tienen inicializar las estructuras anteriores.

9.2 Ejercicio 2

Dada la siguiente estructura:

```
struct cliente {  
    char apellido[20];  
    char nombre[20];  
    int numero;  
    int dni;  
    char calle[20];  
    char ciudad[20];  
};
```

Escribir un programa que cumpla con:

- a) Solicitar al usuario ingresar cada dato de la estructura y almacenarlo.
- b) Imprimir los datos cargados por el usuario.

9.3 Ejercicio 3

Completar el siguiente programa para ingresar los miembros de cada elemento de tipo `alumno` en el arreglo `alumnos`.

```
#include <stdio.h>  
#define CANT_ALU 3  
  
struct alumno {  
    int legajo;  
    char nombre[20];  
    char apellido[20];  
    char curso[3];  
    int notas[4];  
};
```



```

    float promedio;
    char estado;
};

/** Completar */

int main(void)
{
    struct alumno alumnos[CANT_ALU];

    /** Completar */

    return 0;
}

```

El programa debe cumplir:

- Se debe validar que el legajo y DNI sea un número positivo
- El usuario debe cargar las 3 notas. Luego con esta información, el programa calcula y almacena el promedio de cada alumno en el miembro **promedio**.
- El programa debe determinar el **estado** académico del alumno según:
 - Promoción: promedio mayor o igual a 8.
 - Regular: promedio mayor o igual a 6 y menor que 8.
 - Libre: promedio menor a 6.

Uniones y Enumeraciones

9.4 Ejercicio 1

En los siguientes fragmentos de programa, determinar si son correctos o no. En caso de no serlo justificar.

a) –

```

union valores {
    char a;
    float b;
    double c;
};
union valores p = { 1.27 };

```

b) –

```

struct person {
    char lastName[15];
    char firstName[15];
    unsigned int age;
}

```

- c) Suponga que estructura `carta` se ha definido con dos punteros para char (número y palo). Además, la variable `c` se ha definido para ser de tipo `struct carta` y la variable `cPtr` se ha definido para ser de tipo puntero a `struct carta`. A la variable `cPtr` se le ha asignado la dirección de `c`.

```
struct carta c;  
cPtr = &c;  
printf("%s\n", *cPtr->face);
```

9.5 Ejercicio 2

Indique si cada uno de los siguientes es verdadero o falso. Si es falso, explique por qué.

- a) Las estructuras pueden contener variables de un solo tipo de datos.
- b) Se pueden comparar dos uniones (usando `==`) para determinar si son iguales.
- c) El nombre de la etiqueta de una estructura es opcional.
- d) Los miembros de diferentes estructuras deben tener nombres únicos.
- e) La palabra clave `typedef` se usa para definir nuevos tipos de datos.
- f) Las estructuras siempre se pasan a las funciones por referencia.
- g) Las estructuras no pueden compararse utilizando operadores `==` y `!=`.

9.6 Ejercicio 3

Escriba el código para lograr cada uno de los siguientes:

- a) Defina una estructura llamada `parte` que contenga la variable `int` sin signo `partNumber` y `char array` `partName` con valores que pueden tener hasta 25 caracteres (incluido el carácter nulo de terminación).
- b) Definir `Parte` como sinónimo de la `parte` de tipo estructura.
- c) Use `Part` para declarar que la variable `a` sea de tipo `struct part`, la matriz `b [10]` sea de tipo `struct parte` y la variable `ptr` para que sea de tipo puntero a `struct part`.
- d) Lea un número de `parte` y un nombre de `parte` del teclado en los miembros individuales de la variable `a`.
- e) Asigne los valores de miembro de la variable `a` al elemento 3 de la matriz `b`.

- f) Asigne la dirección de la matriz b a la variable de puntero ptr.
- g) Imprima los valores de miembros del elemento 3 de la matriz b usando la variable ptr y el operador de puntero de estructura para referirse a los miembros.

9.7 Ejercicio 4

Dada la estructura `personaje_t`, realizar un programa que solicite al usuario ingresar los datos necesarios para completar la estructura. Los miembros `escudo` y `sales` deben ser inicializados en 0. El miembro `vida` en 150,0. El miembro `rango` se debe generar en forma aleatoria, con un valor entre 0 y 100.

```
typedef struct {
    char apodo[20];
    int rango;
    float vida;
    float escudo;
    float sales;

    struct {
        char nombre[20];
        char apellido[20];
        int edad;
    } info_personal;
} personaje_t;
```

El programa debe cargar los miembros mediante dos formas: la primera en forma directa con la variable `personaje`, y en forma indirecta mediante el puntero `pPersonaje` según la siguiente definición:

```
personaje_t personaje, *pPersonaje;
pPersonaje = &personaje;
```

Acontinuación se adjunta un ejemplo de salida válido:

```
—————Cargar los elementos directamente—————
Ingrese su apodo: snow
Ingrese su nombre: martin
Ingrese su apellido: nievas
Ingrese su edad: 25
—————Datos—————
El apodo es: snow
Rango: 84
Vida: 150.000000
Escudo 0.000000
Sales: 0.000000
El nombre es: martin
El apellido es: nievas
—————Cargar los elementos mediante puntero—————
Ingrese su apodo: snow
Ingrese su nombre: martin
```

```
Ingrese su apellido: nievas
Ingrese su edad: 25
-----Datos-----
El apodo es: snow
Rango: 87
Vida: 150.000000
Escudo 0.000000
Sales: 0.000000
El nombre es: martin
El apellido es: nievas
```

Campos de bits

Ejercicio 1

Dado el siguiente programa, implementar las funciones según las especificaciones indicadas en los comentarios. El mazo de cartas debe ser inicializado de manera aleatoria, estando permitida la repetición de cartas.

```
#include <stdio.h>
#define CANT 100

typedef struct {
    int palo : 4; // basto, copa, espada, oro
    int num : 12;
} cartasBit_t;

/* Función que recibe un puntero a un mazo de cartas
 * y lo inicializa con valores aleatorios
 * Datos:
 * *mazo: Puntero a un arreglo de tipo cartasBit_t
 * cant: La cantidad de cartas en el mazo
 * */
void crear_baraja(cartasBit_t *mazo, int cant);

/* Función que recibe un puntero a un mazo de cartas
 * e imprime dos columnas correspondientes a las cartas
 * de cada uno de los dos jugadores.
 * Datos:
 * *mazo: Puntero a un arreglo de tipo cartasBit_t
 * cant: La cantidad de cartas en el mazo
 * */
void repartir_baraja(cartasBit_t *mazo, int cant);

int main(void)
{
    cartasBit_t cartas[CANT];

    /** Completar... */
```

```

    return 0;
}

```

A continuación se muestra un ejemplo de salida del programa:

```

Iniciando el mazo...
Repartiendo el mazo...

```

Carta	Palo	Número	Carta	Palo	Número
0	4	11	1	2	8
2	2	8	3	3	1
4	2	2	5	3	8
6	3	8	7	4	11
8	1	7	9	1	5
10	4	9	11	4	10
12	3	3	13	3	8
14	4	8	15	2	7
16	3	7	17	2	4
18	2	5	19	4	11
20	2	10	21	2	12
22	1	2	23	3	9
24	4	7	25	2	11
26	4	9	27	1	10
28	3	3	29	1	2
30	2	10	31	1	4
32	1	6	33	3	10
34	2	6	35	1	4
36	3	10	37	3	4
38	3	9	39	4	3
40	4	1	41	1	3
42	1	1	43	4	12
44	3	8	45	1	1
46	1	1	47	4	5
48	3	7	49	3	12

Manejo de archivos en C.

Preguntas:

a) Completar los espacios en blanco:

1. La función _____ cierra un archivo.
2. La función _____ lee los datos de un archivo de manera similar a cómo scanf lee desde stdin.
3. La función _____ lee un caracter de un archivo especificado.
4. La función _____ reposiciona el puntero de posición del archivo a una ubicación específica en el archivo.

10.1 Ejercicio 1

Indique cuáles de los siguientes son verdaderos y cuáles son falsos.
Si es falso, explique por qué.

- a) La función fscanf no se puede usar para leer datos de la entrada estándar.
- a) Debe usar explícitamente fopen para abrir la entrada estándar, la salida estándar y las secuencias de error estándar.
- a) Un programa debe llamar explícitamente a la función fclose para cerrar un archivo.
- a) Si el puntero de posición del archivo apunta a una ubicación en un archivo secuencial que no sea el comienzo del archivo, el archivo debe cerrarse y volverse a abrir para leerlo desde el principio del archivo.
- a) La función fprintf puede escribir en la salida estándar.
- a) Los datos en archivos de acceso secuencial siempre se actualizan sin sobrescribir otros datos.
- a) No es necesario buscar a través de todos los registros en un archivo de acceso aleatorio para encontrar un registro específico.
- a) Los registros en archivos de acceso aleatorio no tienen una longitud uniforme.
- a) La función fseek puede buscar solo en relación con el comienzo de un archivo.

Ejercicio 2

Encuentre el error en cada uno de los siguientes segmentos del programa y explique cómo corregirlo.

- a) El archivo al que hace referencia fPtr ("pagos.dat") no se ha abierto.

```
printf(fPtr, "%d%s%d\n", account, company, amount);
```

- b)

```
open("receive.dat", "r+");
```

- c) La siguiente declaración debería leer un registro del archivo "pagos.dat". El puntero de archivo payPtr se refiere a este archivo, y el puntero de archivo recPtr se refiere al archivo recibir.dat"

```
scanf(recPtr, "%d%s%d\n", &account, company, &amount);
```

- d) El archivo "tools.dat" debe abrirse para agregar datos al archivo sin descartar los datos actuales.

```
if ((tfPtr = fopen("tools.dat", "w")) != NULL)
```

- e) El archivo "course.dat" debe abrirse para agregarlo sin modificar el contenido actual del archivo.

```
if ((cfPtr = fopen("courses.dat", "w+")) != NULL)
```

Uso del lenguaje C en aplicaciones de bajo nivel.