

# MICROS Y LABORATORIO

## INTRODUCCION

PROFESOR: ROBINSON JIMENEZ MORENO  
LUISA FERNANDA GARCÍA VARGAS



NOMBRE DE LA ASIGNATURA	MICROS Y LABORATORIO
CÓDIGO	18203
SEMESTRE	V

### OBJETIVO GENERAL

*Diseñar e implementar sistemas embebidos basados en Microcontroladores, integrando herramientas de hardware y software, orientando su aplicación a sistemas mecatrónicos.*

### COMPETENCIA GLOBAL

- *El estudiante generará habilidad en el análisis de los sistemas Microprocesados, Microcontrolados y sistemas embebidos basados en estos.*
- *Podrá seleccionar la tecnología de procesador más adecuada para aplicar en los diferentes procesos de control industrial.*
- *Desarrollará habilidades en la programación de un Sistema embebidos basado en microcontrolador*



## CONTENIDO

Semana	Tema o actividad presencial	Actividades de trabajo independiente
1	Introducción al curso, metodología de trabajo. Introducción a la arquitectura de microcontroladores / Lenguaje C orientado a Microcontroladores	Estudiar e identificar las diferentes arquitecturas microcontroladas. Realizar conversiones entre sistemas de codificación numérica binario-hexa y decimal. Repasar uso de sentencias condicionales en C.
2	Puertos de entrada/salida. Modulo GPIO. Ejemplos y aplicaciones GPIO	Identificar los registros de propósito general para configuración de entradas salidas digitales. Desarrollar aplicaciones básicas microcontroladas.
3	Interrupciones. Ejemplo y aplicaciones.	Resolver problemas de control microcontrolado basado en interrupciones.
4	Aplicaciones: Visualización dinámica, Display n-segmentos, Matrices de LEDs. Aplicaciones: Teclado Matricial (Interrupción)	Desarrollar programas de manejo de dispositivos de entrada y salida digital.
5	Ejercicios de refuerzo/ Taller pre-parcial Primer Parcial	Replicar los programas vistos en clase bajo la solución de problemas.
6	Display LCD – alfanumérico. Display LCD –gráfico.	Desarrollar programas de manejo de dispositivos de visualización

## BIBLIOGRAFÍA

1. Índice con referencias de páginas y citas bibliográficas
2. Libros textos
  - \*Tammy Noergaard, Embedded Systems Architecture, Second Edition: A Comprehensive Guide for Engineers and Programmers, 2012.
  - \*Jason D. Bakos Embedded Systems: ARM Programming and Optimization, 2015
  - \*Título: Manual de usuario de la tarjeta STM32F407
  - (datasheet <http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf>)
3. Libros electrónicos
  - <http://www.st.com/web/catalog/mmo/FM141/SC1169/SS1577/LN11>

## MATERIAL COMPLEMENTARIO DE APRENDIZAJE PARA ESTUDIANTES

1. Glosario
  - CI Circuito integrado: un tipo de circuito en el cual todos los componentes se integran sobre un solo chip semiconductor de tamaño muy pequeño.
  - Ciclo de operación La razón de la anchura de pulso y el periodo, expresada como un porcentaje.
  - Código Un conjunto de bits organizados en un patrón único y utilizados para representar información tal como números, letras y otros símbolos.
  - Datos: Información en forma numérica, alfabética u otra.
  - Reloj. La señal básica de sincronización en un sistema digital.

		digital, integrar interfaces de usuario.
7	Modulo USART (Comunicación Serial) Registros y contadores. Taller: Aplicaciones UART	Diseñar sistemas básicos de transferencia de información serial.
8	Manejo de motores paso a paso. Manejo de servomotores. Aplicaciones serial – inalámbrico.	Desarrollar aplicaciones microcontroladas basadas en el uso de motores.
9	Modulo ADC Taller: Aplicaciones modulo ADC	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos ADC.
10	Modulo DAC Taller: Aplicaciones modulo DAC	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos DAC.
11	Ejercicios de refuerzo/ Taller pre-parcial Segundo Parcial	
12	Modulo TIMER: PWM Taller: Aplicaciones Timer PWM	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos TIMER - PWM
13	Modulo TIMER: OC (Output Capture) – IC (Input Capture) Taller: Aplicaciones Timer (OC)	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos TIMER-OC-IC.
14	Modulo I2C Taller: Aplicaciones I2C	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos I2C.
15	PLL Taller: Aplicaciones PLL	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos PLL.
16	Ejemplos de aplicación Taller integrador	

## SISTEMA DE EVALUACIÓN

Derivado de cada uno de los cortes de notas establecidos en el calendario académico se evaluarán los diferentes temas según el avance del contenido programático mediante las siguientes herramientas:

- Parciales teórico prácticos.
- Talleres y quices en clase.
- Presentación de laboratorios.

Las evaluaciones teórico-prácticas equivalen al 100% de la nota total, se realizan por medio de dos notas cada uno correspondiente al 21%, de la nota final del 28% y la nota de las prácticas desarrolladas en el laboratorio del 30 %.

# EVALUACIÓN DEL CURSO

3 CORTES	CONCEPTO	VALOR (por corte)
	EXAMEN ESCRITO	50%
	TALLERES TAREAS Y QUICES 50%	

El examen escrito tiene carácter individual.

Los talleres tienen carácter grupal.

Las tareas tienen carácter individual, y se solicitarán en la clase. La no presentación de una tarea solo se justificará con excusa escrita validada por la dirección de mecatrónica. Las tareas entregadas en la plataforma deben subirse en el horario y fechas establecidos

Contacto: [luisa.garciav@unimilitar.edu.co](mailto:luisa.garciav@unimilitar.edu.co)

Se llamará a lista una única vez al inicio de cada clase para registro de fallas.



## Fechas importantes:

**Primer parcial 25 – 26 Febrero**

**Segundo parcial 15 – 16 Abril**

**Examen final conjunto según las fechas  
y horarios que indique la dirección de  
programa**



# Componentes para el semestre

- Tarjeta de desarrollo STM32F746
- 8 LEDS.
- 5 displays de 7 segmentos.
- 1 LCD de 16x2.
- 1 motoreductor con encoder.
- 1 puente H
- 8 Pulsadores
- 1 Teclado matricial
- Conversor USB a TTL
- 3 sensores análogos
- resistencias varias para polarización de los componentes anteriores
- 1 potenciómetro 1K
- 1 potenciómetro 10K
- Cable mini USB
- Sensor de ultrasonido
- Motor paso a paso

## Bibliografía de soporte:

**Microcontrolador Stm32 Programación y Desarrollo.**  
Jesús María Pestano Herrera, ed: Grupo Editorial RA-  
MA. ISBN 9788499647555

**Programación De Microcontroladores De Gama Alta  
En Lenguaje De Alto Nivel.** Robinson Jiménez  
Moreno, En: Colombia 2009. ed: Fondo Editorial  
Universidad Antonio Nariño ISBN: 978-958-9423-97-  
4 v. 1.

**Compilador C CCS y simulador PROTEUS para  
Microcontroladores PIC.** Eduardo García Breijo. ed:  
Alfaomega ISBN: 978-84-267-1495-4.





# Sistemas embebidos microcontrolados

- Un sistema embebido es un sistema electrónico diseñado para realizar funciones específicas en tiempo real. Al tratarse de un dispositivo microcontrolado significa que posee un circuito integrado que se encarga de controlar los elementos de entrada y salida, además de un procesador y una memoria que permiten guardar el programa y sus variables





# Lenguajes de programación

- Cada lenguaje de programación está diseñado para describir un conjunto de acciones que debe realizar un microcontrolador, entre los cuales se pueden encontrar los siguientes
  - Lenguaje de ensamblador (Assembler)
  - Lenguaje PIC-BASIC PRO (PBP)
  - Lenguaje C
  - Lenguaje Pascal

<https://prezi.com/aowwntefs3gy/lenguajes-de-programacion-para-microcontroladores/>



# Estructura de programación en C

- Este lenguaje de programación permite programar de manera estructurada o no, y tener acceso a memoria de bajo nivel empleando punteros y diferentes tipos de datos como:

Nombre	Peso (Bytes)	Valor Inicial	Valor final
Signed char	1	-128	127
Unsigned char	1	0	255
Signed short	2	-32768	32728
Unsigned short	2	0	65535
Signed int	2	-2147483648	2147483647
Unsigned int	2	0	4294967295
Signed long	4	-32768	32768
Unsigned long	4	0	65535
Float	4	$3,4 \times 10^{-38}$	$3,4 \times 10^{38}$
Double	8	$1,7 \times 10^{-308}$	$1,7 \times 10^{-308}$
Long double	10	$3,4 \times 10^{-4932}$	$3,4 \times 10^{4932}$



# Estructura de programación en C







- El lenguaje C tiene la ventaja de representar las operaciones aritméticas y lógicas con símbolos como:

Símbolo	Descripción
=	Asignación
+	Suma, adición
-	Resta, sustracción
*	Multiplicación, producto
/	Cociente división entera
%	Resto división entera
	División
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
==	Igual
!=	Diferente
&&	And, y, conjunción
	Or, o, disyunción
!	Not, no, negación
&	And bit a bit
	Or bit a bit
~	Complemento a uno o negación bit a bit
^	O-exclusiva bit a bit



# Estructura de programación en C

- **Diagramas de flujo:**
- Los diagramas de flujo permiten representar de forma gráfica el proceso de un algoritmo de programación
  - Usan símbolos para determinar los subprocesos y flechas para unirlos, de inicio a fin

Símbolo	Significado
	Inicio y fin del programa
	Representa la ejecución de una actividad o sub proceso
	Condición, decisión o pregunta
	Enlace entre actividades o sub procesos por medio de otra actividad o procedimiento
	Guarda un documento o proceso de forma permanente
	Guarda un documento o proceso de forma temporal

# Estructura de programación en C

## ➤ Sentencias condicionales:

- Son herramientas de programación que, basadas en una condición lógica, ejecutan una serie de instrucciones. Entre ellas se encuentran:
  - If
  - For
  - While
  - Do While
  - Switch Case



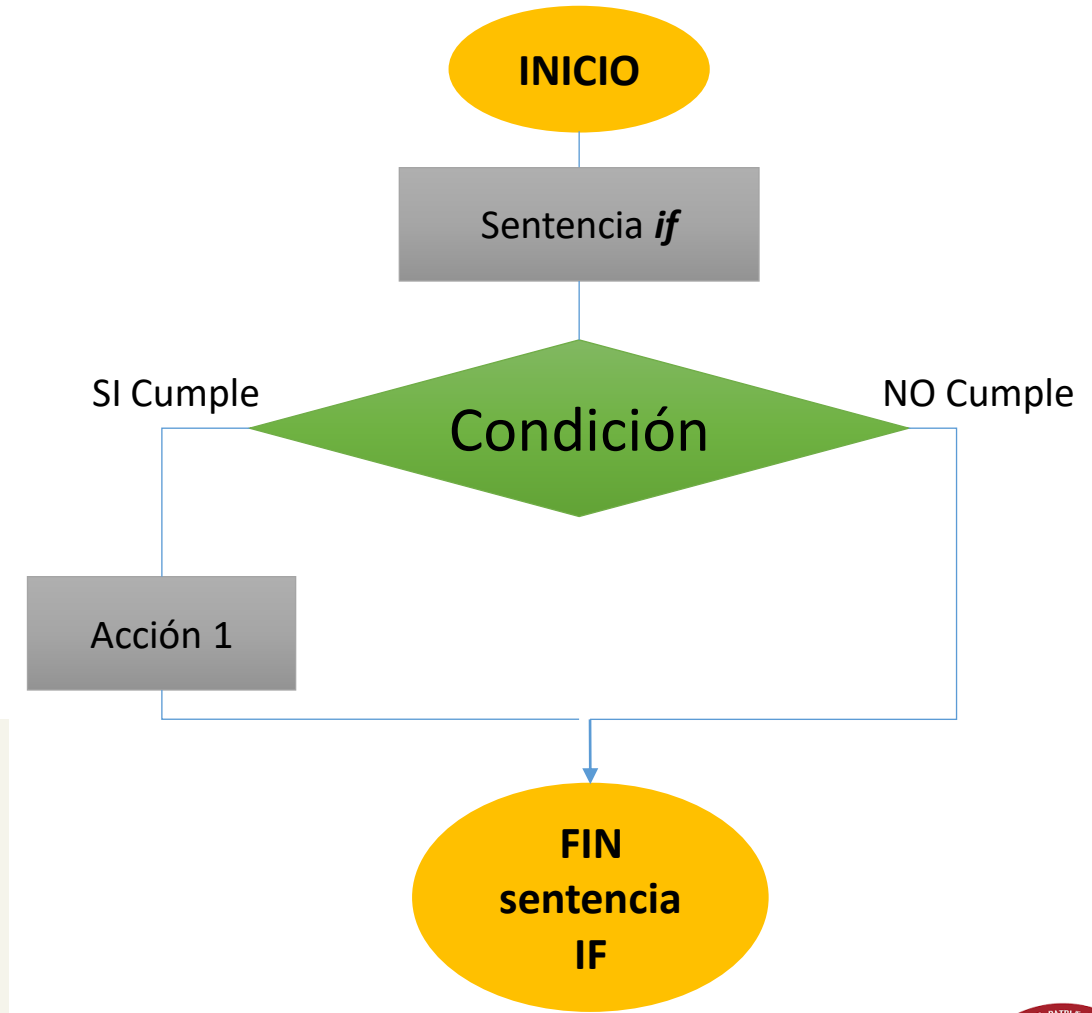
# Estructura de programación en C



## Estructura IF:

Su función es evaluar una condición lógica donde, si la condición se cumple se ejecuta un código, en caso contrario, se ejecuta el código siguiente al IF.

```
int valor;
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10
{
    //Codigo que se ejecuta si se cumple la condición
}
```

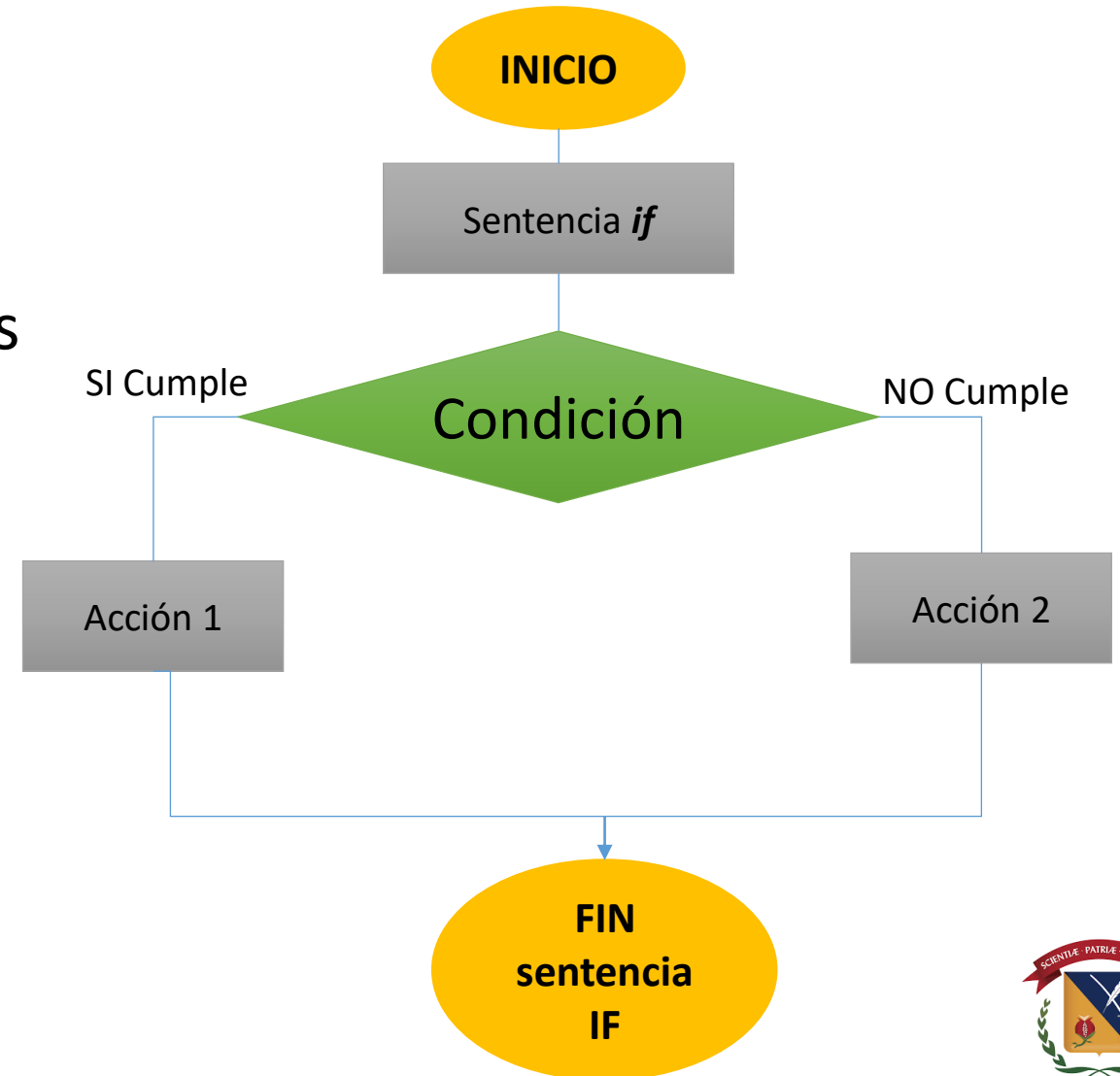




# Estructura de programación en C

- **Estructura IF-ELSE:**
- Se utiliza para casos en que se requiera ejecutar un código tanto cuando la condición es verdadera como cuando es falsa.

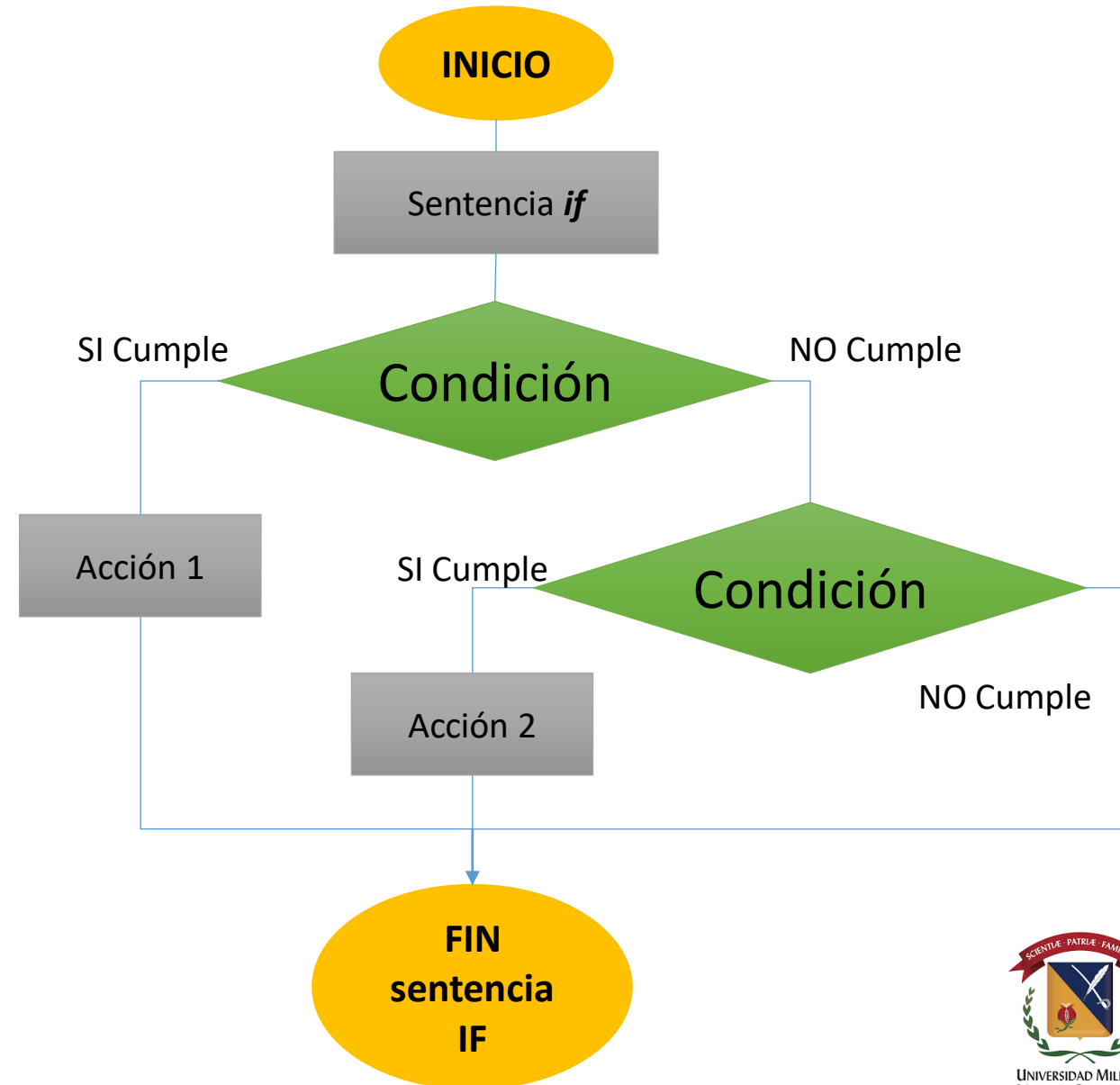
```
int valor;  
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10  
{  
    //Codigo que se ejecuta si se cumple esta condición  
}  
else  
{  
    //Codigo que se ejecuta si la condicion del if no se cumple  
}
```



# Estructura de programación en C

- **Estructura ELSE-IF:**  
Se utiliza para programar condiciones anidadas, dependientes o secundarias

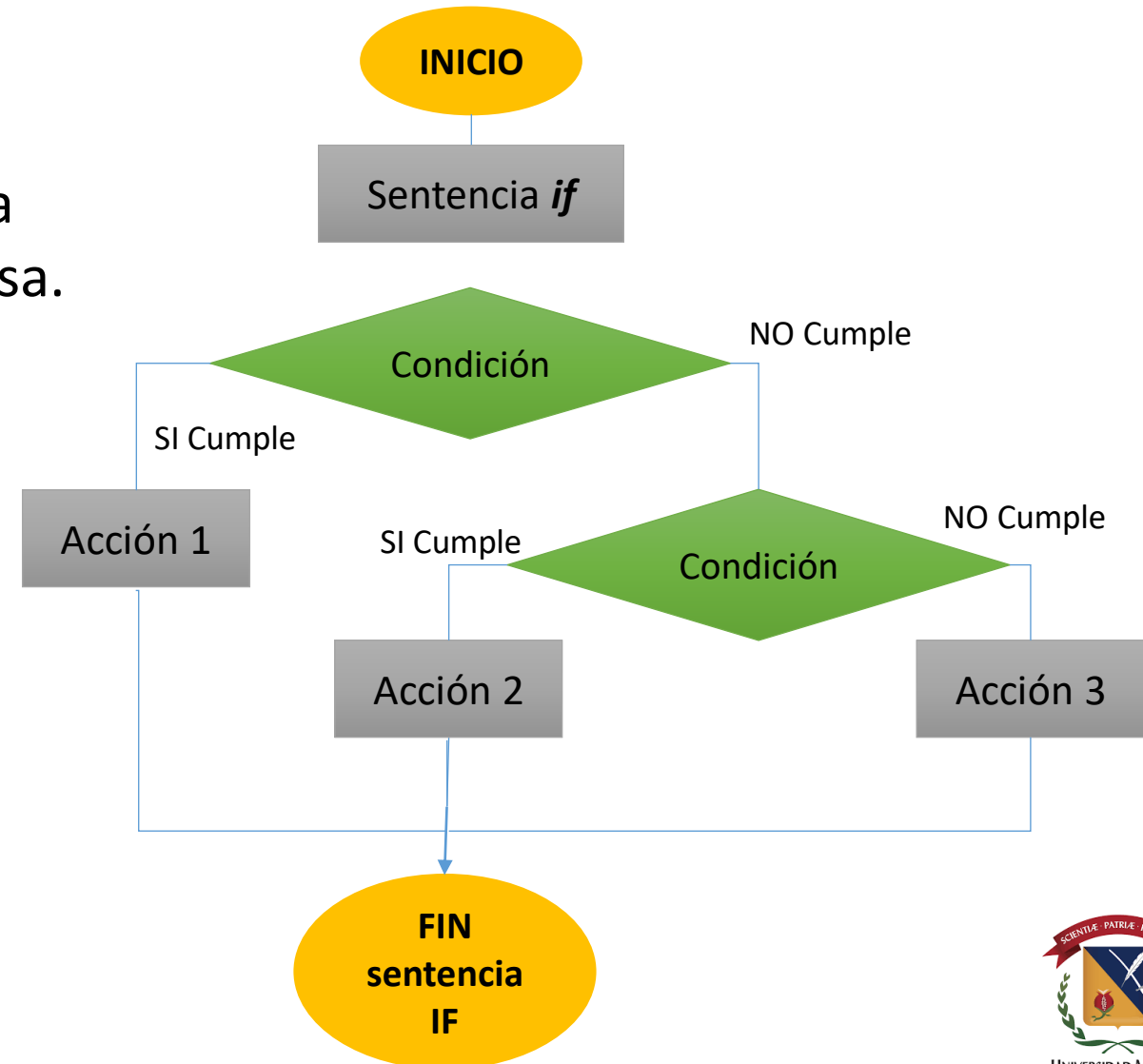
```
int valor;  
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10  
{  
    //Codigo que se ejecuta si se cumple esta condición  
}  
else if (valor == 20) // Condicion que se cumple si la variable valor es igual a 20  
{  
    //Codigo que se ejecuta si se cumple esta condición  
}  
else if (valor == 30) // Condicion que se cumple si la variable valor es igual a 30  
{  
    //Codigo que se ejecuta si se cumple esta condición  
}
```



# Estructura de programación en C

- **Estructura ELSE-IF:**  
También se utiliza para casos en que se requiera ejecutar un código tanto cuando la condición es verdadera como cuando es falsa.

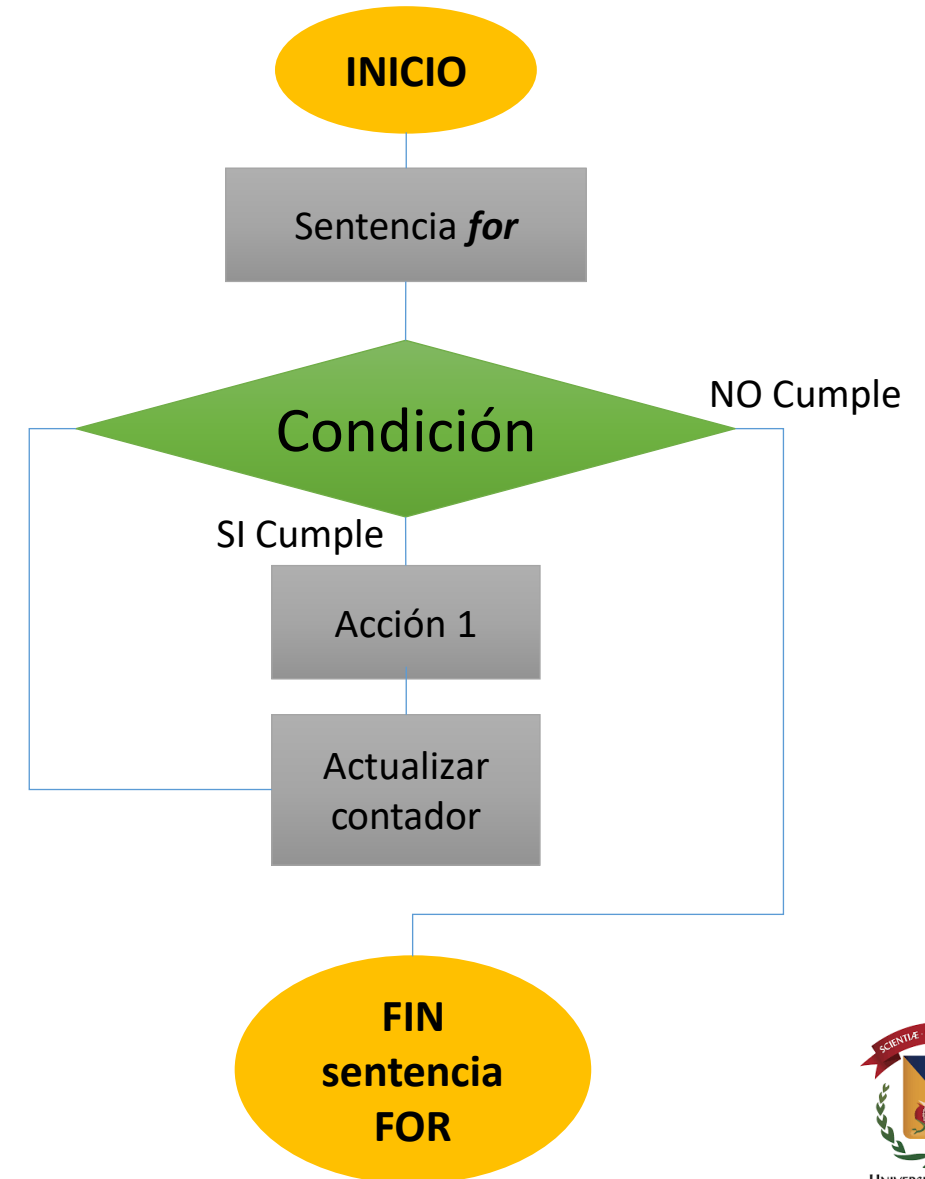
```
int valor;
if(valor > 10) // Condicion que se cumple si la variable valor es mayor a 10
{
    //Codigo que se ejecuta si se cumple la condición del primer if
    if(valor == 15) // Condicion que se cumple si la variable valor es igual a 15
    {
        //Codigo que se ejecuta si se cumple esta condición
    }
    else if (valor == 20) // Condicion que se cumple si la variable valor es igual a 20
    {
        //Codigo que se ejecuta si se cumple esta condición
    }
    else
    {
        //Codigo que se ejecuta si la condicion del segundo if y la del else if no se cumplen
    }
}
```



# Estructura de programación en C

- **Estructura FOR:**
- Se utiliza para ejecutar una tarea iterativamente. Consta de tres parámetros:
- Inicialización de variables
  - Condición que debe cumplir
  - Incremento o cambio de variables condicionales

```
//Declaracion de Variables
int numero1;
int numero2;
//Estructura del ciclo for
for( numero1=0, numero2=10; (numero1<10 && numero2>0) ; A++, B-- )
{
    //Codigo que se ejecuta cuando la condicion se cumple
}
```

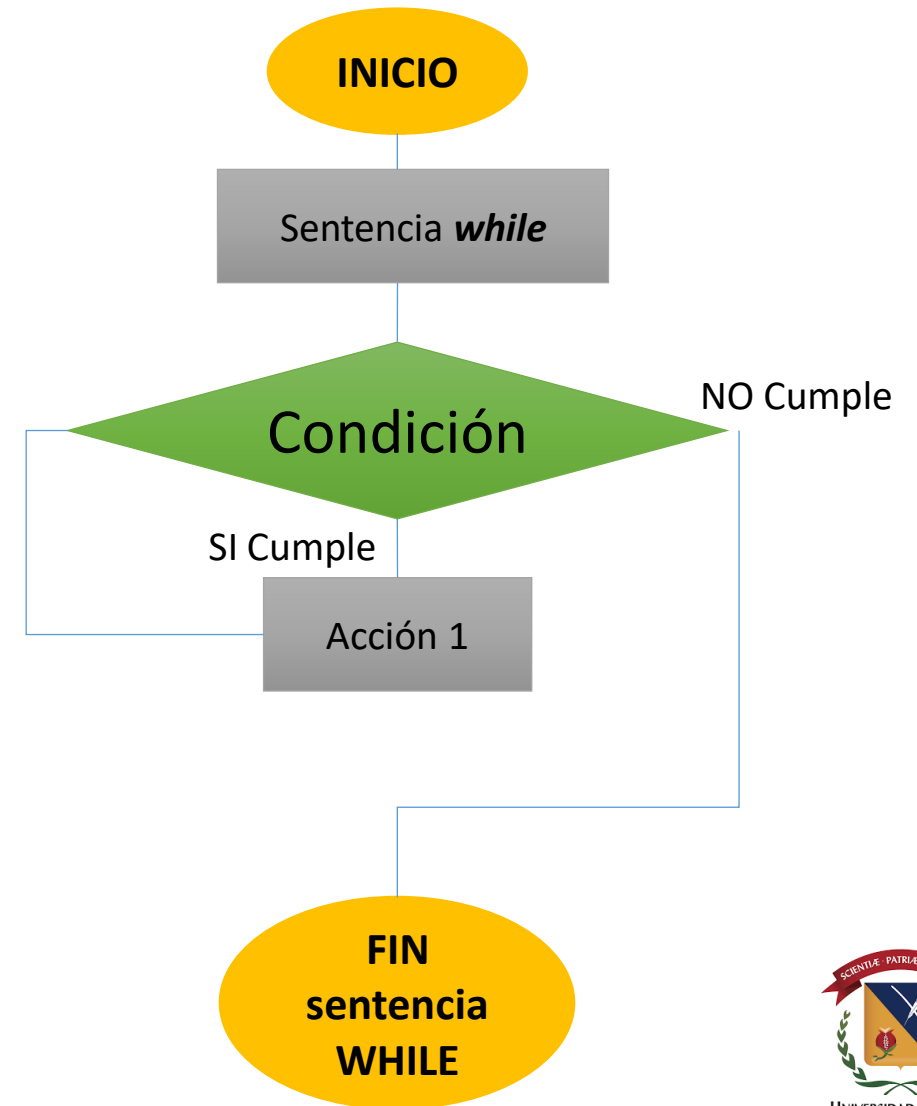


# Estructura de programación en C

## ➤ Estructura WHILE y DO WHILE:

Se utiliza para ejecutar una tarea iterativa de la cual no se conoce el número exacto de iteraciones requeridas antes de que la condición deje de ser verdadera

```
int valor=0; //Declaracion de la Variable
while(valor<10)// Declaracion del Ciclo que evaluara si valor es menor a 10
{
    valor++; // Mientras valor sea menor a 10, se incrementa en 1 esta variable
}
```

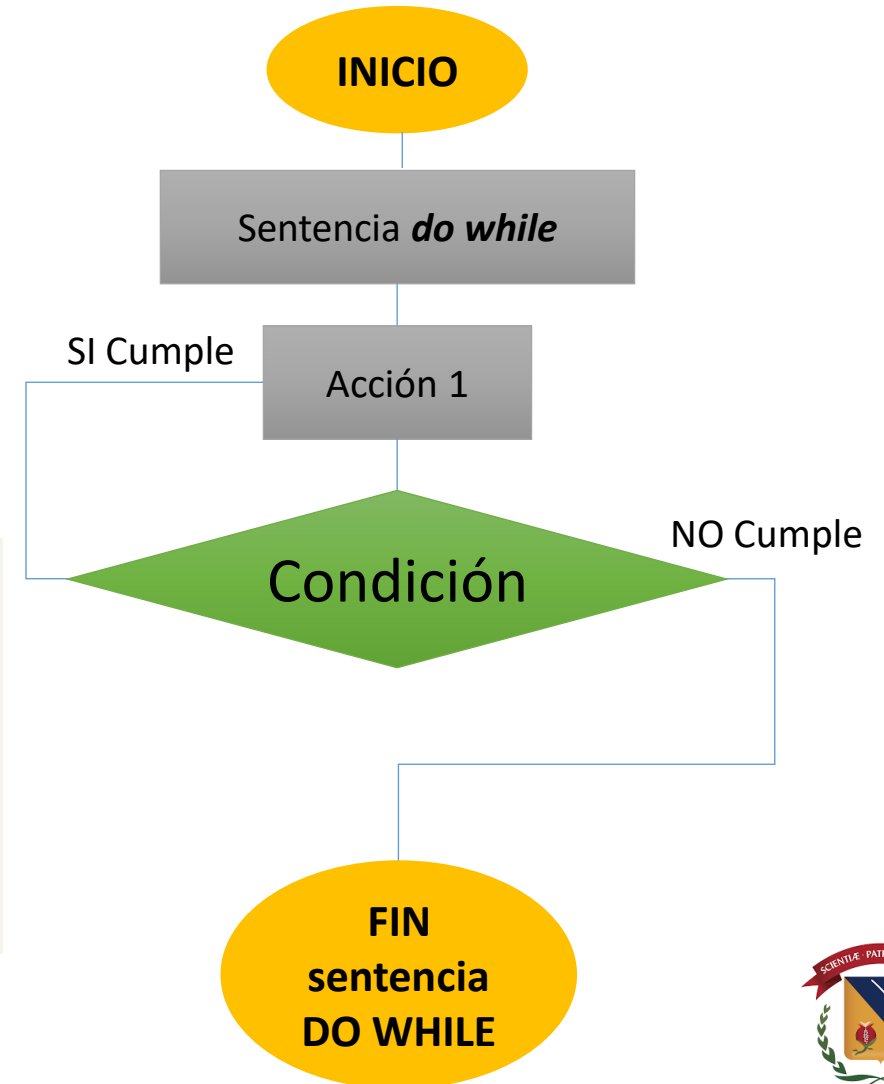


# Estructura de programación en C

## ➤ Estructura WHILE y DO WHILE:

En el caso del DO WHILE, se ejecuta primero el código y luego se evalúa si se cumple la condición.

```
int valor=0; //Declaracion de la Variable
do // Declaracion del Ciclo DO WHILE
{
    valor++; // Mientras valor sea menor a 10, se incrementa en 1 esta variable
}while(valor<10)
```



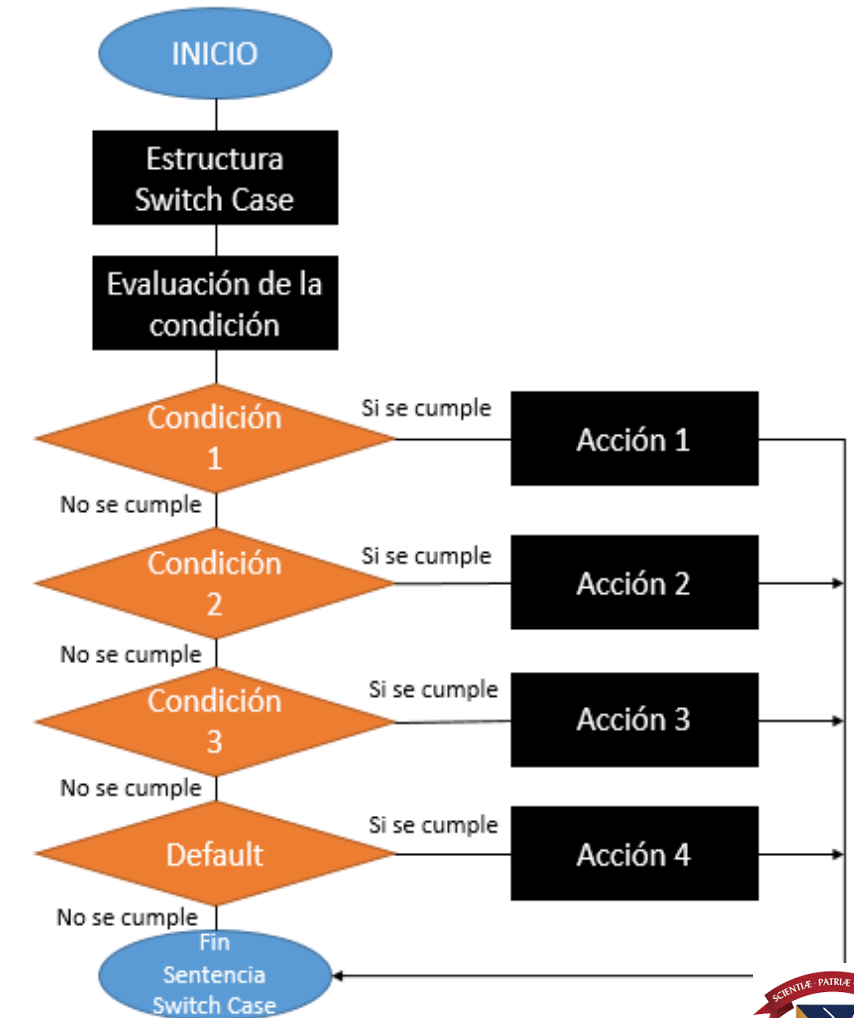


# Estructura de programación en C

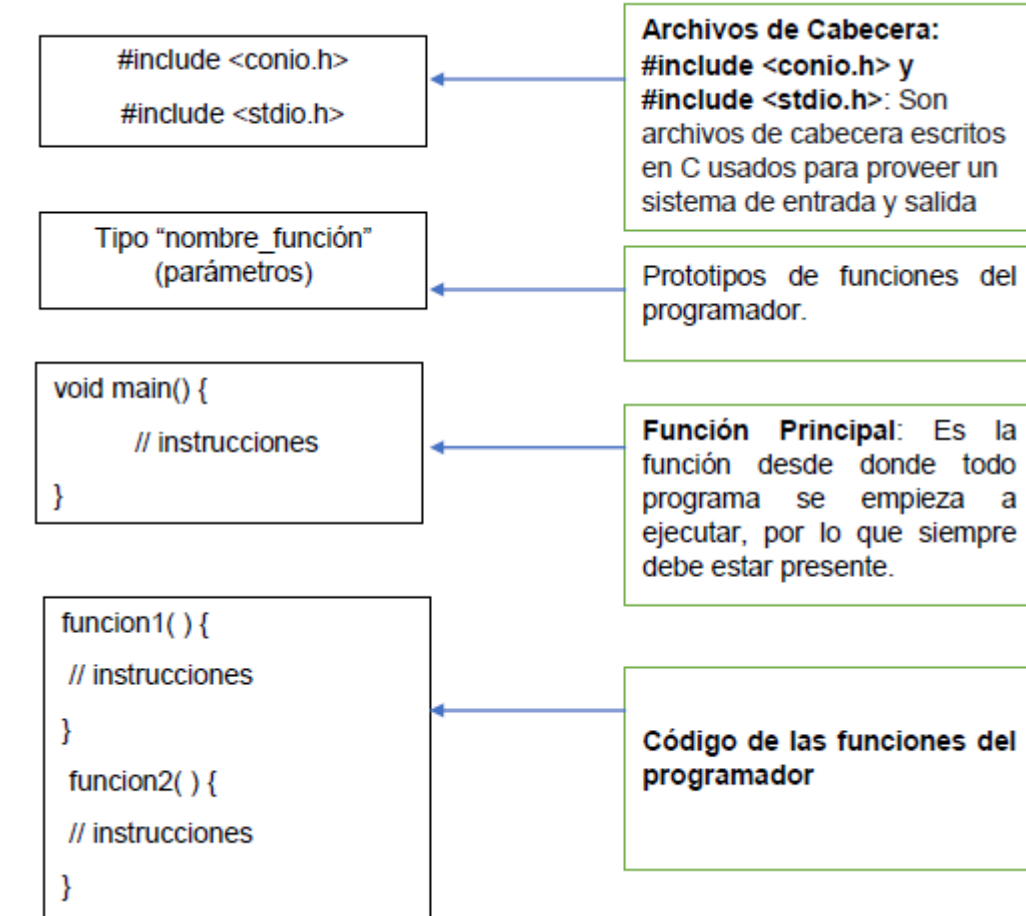
## ➤ Estructura SWITCH CASE:

Se utiliza para evaluar una variable y ejecutar un código según su valor. Cada condición del Case termina en un *break* y las condiciones irrelevantes se ponen en *default*.

```
int valor; //Declaracion de la Variable
switch(valor)
{
    case 0:      //Codigo que se ejecuta si valor es igual a 0
        break; //Ruptura del codigo para case 0
    case 10:     //Codigo que se ejecuta si valor es igual a 10
        break; //Ruptura del codigo para case 10
    case 20:     //Codigo que se ejecuta si valor es igual a 20
        break; //Ruptura del codigo para case 20
    case 30:     //Codigo que se ejecuta si valor es igual a 30
        break; //Ruptura del codigo para case 30
    default:    //Codigo que se ejecuta si valor es otro valor
        break;
}
```



# Partes de un programa en C



4



# Printf y Scanf

- Impresión de un mensaje: `printf("Como te llamas?:");`
- `printf("El resultado es: %f" , r);`
- Impresión de una variable carácter: `printf("El resultado es: %c", letra);`
- Impresión de una expresión (operación): `printf("El resultado es: %d", 5*3);`
- Impresión de una expresión: `printf("El resultado es: %d", 17);`

Especificador	Descripción
<code>%c</code>	Un solo carácter
<code>%d</code>	Un valor entero decimal
<code>%f</code>	Un valor fraccionario
<code>%u</code>	Un entero decimal sin signo
<code>%s</code>	Una cadena de caracteres

Secuencia	Descripción
<code>\n</code>	Salto de línea
<code>\t</code>	Tabulación
<code>\"</code>	Imprime comillas

```

1  #include <stdio.h>
2  #define TAM_MAXIMO 80
3
4  int main(void)
5  {
6      char cadena[TAM_MAXIMO];
7      int entero1, entero2;
8      float decimal;
9
10     printf("Introduce dos enteros separados por un espacio: \n");
11     scanf("%d %d", &entero1, &entero2);
12     printf("Introduce un número decimal:\n");
13     scanf("%f", &decimal);
14     printf("Introduce una cadena:\n");
15     scanf("%s", cadena);
16     printf("Esto es todo lo que has escrito:\n");
17     printf("%d %d %f %s\n", entero1, entero2, decimal, cadena);
18     return 0;
19 }

```

[http://www.it.uc3m.es/pbasanta/asng/course\\_notes/input\\_output\\_function\\_scanf\\_es.html](http://www.it.uc3m.es/pbasanta/asng/course_notes/input_output_function_scanf_es.html)



# Strings

```
char cadena_hola[]="Hola";  
char otro_hola[]={ 'H','o','l','a','\0'}; // Igual al anterior  
char vector[]={ 'H','o','l','a'}; /* Un vector de 4 elementos, con los elementos 'H','o','l' y 'a' */  
char espacio_cadena[1024]="Una cadena en C";  
char cadena_vacia[]="";
```

```
-----  
/* devuelve la cantidad de caracteres en cadena sin contar el '\0' */  
int largo_cadena(char cadena[])  
{  
    int largo=0;  
    while (cadena[largo]!='\0')  
        largo++;  
    return largo;  
}
```

```
-----  
#include <stdio.h>  
main()  
{  
    char nombre[20];  
  
    printf( "Introduzca su nombre (20 letras máximo): " );  
    scanf( "%s", nombre ); //no va el & porque es un vector  
    printf( "\nEl nombre que ha escrito es: %s\n", nombre );  
}
```



# Vectores y matrices

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      //Declaración del vector tipo int de 100 posiciones
6      int i, numero[100];
7      //Ciclo para llenar el vector con numeros del 1 al 100
8      for (i=0; i<100; i++){
9          numero[i]=i+1; //Operación para llenar el vector
10         //imprime el valor de los elementos del vector
11         printf("%d\n", numero[i]);
12     }
13 }
14

```

o `Int arreglo[2][4] = {{1,2,3,4},{1,3,2,4}};`

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      //Declaración de variables y de la matriz 10x10
6      int x,y, numeros[10][10];
7      //Ciclo para mover por filas
8      for (x=0; x<10; x++){
9          //Ciclo para mover por columnas
10         for (y=0; y<10; y++){
11             //operación para generar la tabla de multiplicar
12             numeros[x][y]=(x*10)+1+y;
13             //Imprime en pantalla la operación
14             printf("%d ", numeros[x][y]);
15         }
16         //Imprime un salto de linea cuando el ciclo interno termina
17         printf("\n");
18     }
19 }

```



# Funciones

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  float precio(int base, float impuesto); //función precio con parametros
5  int main()
6  {
7      int base=0;
8      float impuesto=0.16, precio_final;
9      //Ingresa el precio base del producto
10     printf("Digite el precio base del producto: ");
11     scanf("%d",&base);
12     precio_final=precio(base, impuesto); //Se llama la función precio
13     printf("\nEl precio final del producto es: %1.f", precio_final);
14 }
15 //Función precio tipo float para retornar el valor de precio
16 float precio(int base, float impuesto){
17     float precio;
18     //Operación para generar el valor del precio
19     precio=base+(base*impuesto);
20     return precio; //retorna valor de precio
21 }
```





# Ejercicio en clase

Escriba un código en lenguaje c que entregue el promedio de los elementos de una matriz, con los siguientes valores:

5	4	3
1	2	5
3	7	6

## TAREA

Instalar el software de programación keil (microVision 5) de:  
<https://www.keil.com/download/>



### Requisitos del proyecto

- Solo se debe programar en lenguaje C
- Cada subpunto de los Modos Administrador y Usuarios debe estar en funciones
- El desarrollo de las funciones debe incluir parámetros cuando estos correspondan
- Manejo de cadena de caracteres
- **Fecha de entrega 4 - 5 de febrero, Se debe realizar en grupo de 3 o 2 personas, adjuntar el código en C e imágenes del funcionamiento del sistema**

### Título: Sistema de control de visitantes por el COVID 19

Se debe diseñar un sistema que facilite el control de visitantes a un edificio de apartamentos, conformado por 6 pisos y 5 apartamentos por piso.

El sistema debe cumplir con las siguientes características

- Se debe llevar un control de que apartamentos tienen residente con Covid, y en ese caso no se debe permitir el ingreso de las visitas
- Al iniciar el sistema la matriz de apartamentos con Covid es la siguiente. Donde 1: tiene Covid

Apartamento					
Piso	a	b	c	d	e
1	0	0	1	0	0
2	1	0	1	0	0
3	1	1	0	1	1
4	0	1	1	0	1
5	1	0	1	0	1
6	0	0	1	0	0

- Solo pueden tener 3 visitantes máximo al mismo tiempo por apartamento
- Se debe llevar un registro de los visitantes que lleguen sin tapabocas, almacenando el nombre y apellido. Su ingreso no será permitido. Se puede almacenar máximo la información de 10 visitantes.

### El proyecto tiene dos modos

Modo administrador

- Se debe solicitar la clave antes de mostrar el menú de administrador
- Reportar en pantalla el número total de visitantes por apartamento que se encuentran en ese momento en el edificio.
- Reportar en pantalla los apartamentos que tengan residentes con Covid
- Reportar en pantalla el nombre y apellido de los visitantes que llegaron sin tapabocas
- Cambio de clave

La clave inicial es 9876, el administrador tiene la opción de cambiarla

Solo debe permitir números enteros

Modo usuario

- Registro del ingreso de un nuevo visitante y verificación de acceso
- Registro de visitantes sin tapabocas
- Registro del egreso de un visitante

El usuario del sistema es el responsable de mirar si el visitante tiene o no tapabocas, y realizar el registro respectivo en el menú de usuario.

TODO EL PROGRAMA DEBE ESTAR DOCUMENTADO Y DEBE SER CLARO DE ENTENDER