

# PROGRAMACIÓN

**Unidad N° 3 – Parte 3.** Estructuras de Control Iterativas.  
Sintaxis y semántica. Diferencias entre las distintas  
construcciones

# Repasemos lo visto



# Estructura Iterativa: MIENTRAS()

Son utilizados para repetir acciones bajo un condición

Su notación es:

```
MIENTRAS(condición)
    acciones
Fin_mientras
```

--> Condición: La condición puede ser simple o compuesta. En este último caso estará "unida" por una conjunción (y) o una disyunción (o).

# Estructura Iterativa: MIENTRAS()

Veamos un ejemplo de su funcionamiento

*Analizando un sector de pseudocódigo*

# Estructura Iterativa: MIENTRAS()

```
...
LEER(edad)
MIENTRAS( edad > 0)
|   LEER(dni)
|   SI (edad >= 18) ENTONCES
|   |   ESCRIBIR(dni)
|   Fin_si
|   LEER(edad)
Fin_mientras
...
```

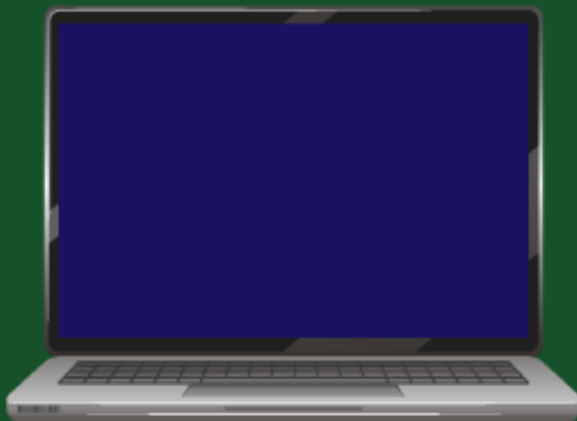
← Lee la edad de la primera persona ingresada

← Controla si edad es mayor a 0. Como edad = 5 por lo tanto ingresa al bucle.

← Lee dni

← Controla si edad es mayor a 18. Como edad = 5 entonces no ingresa a la selección

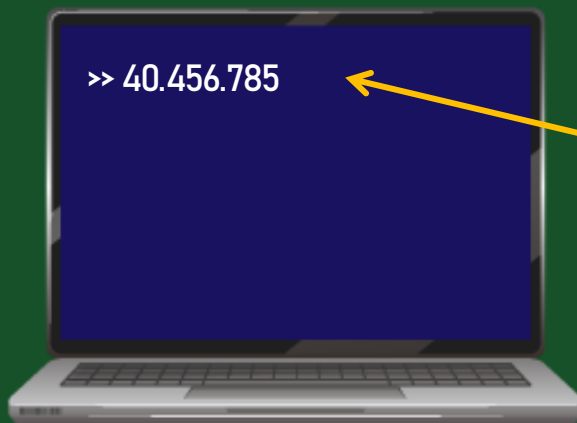
← Lee la próxima edad



edad	dni
5	53.321.125
20	

# Estructura Iterativa: MIENTRAS()

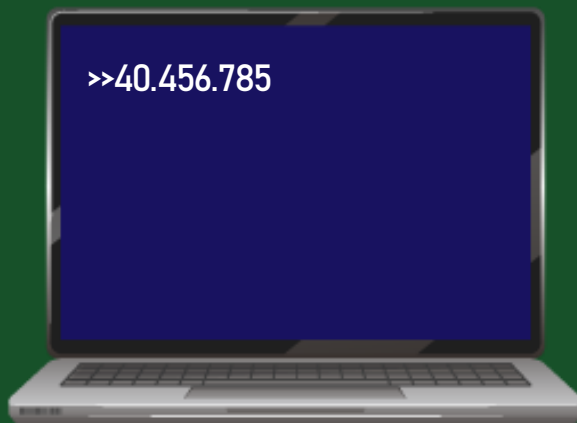
```
...  
LEER(edad)  
MIENTRAS( edad > 0)      ← Controla si edad es mayor a 0. Como edad = 20 por lo  
                           tanto ingresa al bucle.  
|   LEER(dni)             ← Lee dni  
|   SI (edad > 18) ENTONCES ← Controla si edad es mayor a 18. Como edad = 20  
|   |   ESCRIBIR(dni)      ← Escribe el dni  
|   Fin_si  
|   LEER(edad)            ← Lee la próxima edad  
Fin_mientras
```



edad	dni
5	53.321.125
20	40.456.785
18	

# Estructura Iterativa: MIENTRAS()

```
...  
LEER(edad)  
MIENTRAS( edad > 0)      ← Controla si edad es mayor a 0. Como edad = 18 por lo  
                            tanto ingresa al bucle.  
|   LEER(dni)             ← Lee dni  
|   SI (edad > 18) ENTONCES ← Controla si edad es mayor a 18. Como edad = 18, no  
|   |   ESCRIBIR(dni)      mayor, entonces no ingresa a la selección.  
|   |   Fin_si  
|   LEER(edad)            ← Lee la próxima edad  
Fin_mientras  
...
```



edad	dni
5	53.321.125
20	40.456.785
18	43.128.875
0	



# Estructura Iterativa: MIENTRAS()

...  
LEER(edad)

MIENTRAS( edad > 0)



Controla si edad es mayor a 0. Como edad = 0 y no mayor, por lo tanto no cumple con la condición.

LEER(dni)

SI (edad > 18) ENTONCES

ESCRIBIR(dni)

Fin\_si

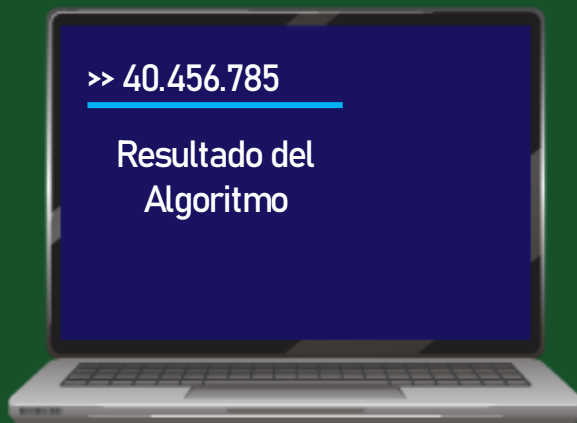
LEER(edad)

Fin\_mientras



Sale de la estructura Mientras

...



edad	dni
5	53.321.125
20	40.456.785
18	43.128.875
0	



# Estructura Iterativa: MIENTRAS()

A tener en cuenta:

- Es conveniente usar esta estructura cuando NO sabemos con anterioridad la cantidad de veces que se repetirá una serie de acciones.
- Se debe tener en cuenta que para evitar un bucle infinito (esto implica que no se puede salir de la estructura iterativa) siempre se debe leer un nuevo dato o incrementar/decrementar dentro de la estructura, ya que ese dato forma parte de la condición.

Por ejemplo:

LEER(cantidad)

MIENTRAS(cantidad > 0)

acciones

cantidad ← cantidad - 1

Fin\_mientras

Si no existiera esta línea, y cantidad fuera igual a 4, no saldría de la estructura, debido a que ese dato jamás es modificado



# Sentencia While

La sentencia while ejecuta una sentencia, simple o compuesta, cero o más veces, dependiendo de una condición.

Su sintaxis es:

```
while ( condición )
```

```
{ sentencia ; }
```

Donde:

**condición** es cualquier expresión numérica, relacional o lógica.

**sentencia** es una sentencia simple o compuesta.

# Sentencia While

```
while ( condición )  
{ sentencia ; }
```

La ejecución de la sentencia while sucede así:

Se evalúa la condición . Si el resultado de la evaluación es 0 (falso), la sentencia no se ejecuta y se pasa el control a la siguiente sentencia en el programa. Si el resultado de la evaluación es distinto de 0 (verdadero), se ejecuta la sentencia y el proceso descrito se repite desde el Inicio.

# Ejemplo

Ya conocemos la sintaxis de la estructura **while()** en C

Entonces estamos listos para implementar el algoritmo visto en el ejemplo



# Ejemplo

Algoritmo MayorDeEdad

ENTRADA: edad, dni: enteros

SALIDA: dni: entero

A1. LEER(edad)

A2. MIENTRAS( edad > 0)

LEER(dni)

SI (edad >= 18) ENTONCES

ESCRIBIR(dni)

Fin\_si

LEER(edad)

Fin\_mientras

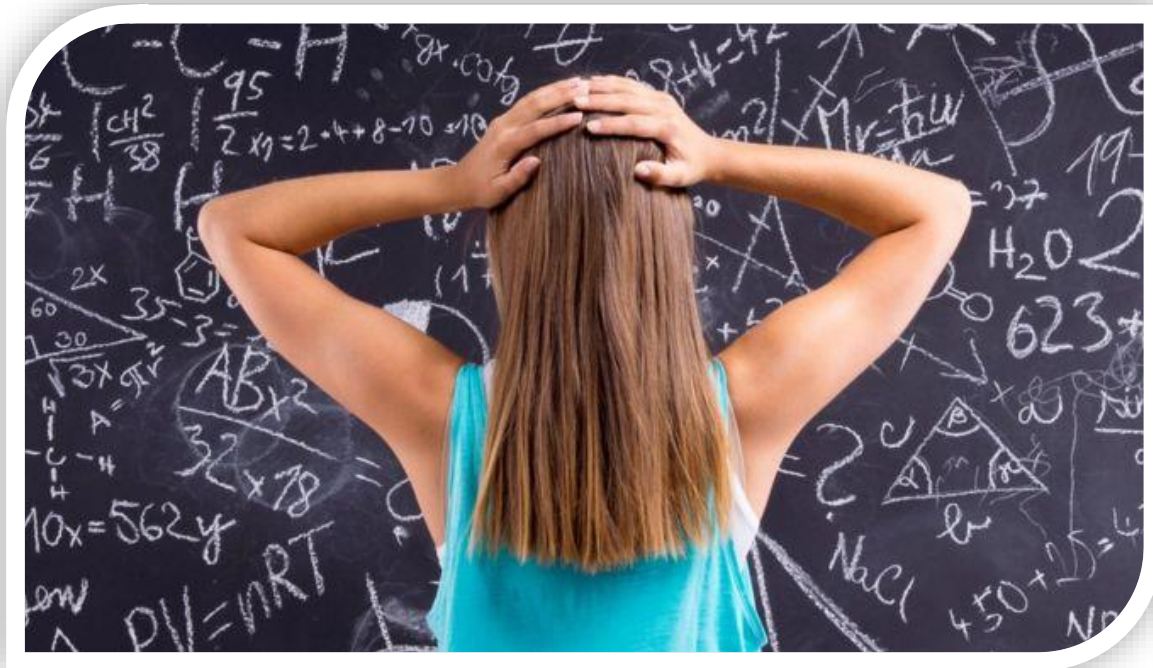
A3. PARAR

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int edad, dni;
6
7      printf("Ingrese la edad: ");
8      scanf("%d", &edad);
9
10     while(edad > 0)
11     {
12         printf("Ingrese el dni: ");
13         scanf("%d", &dni);
14         if(edad >= 18)
15         {
16             printf("La persona con DNI %d es mayor de edad", dni);
17         }
18
19         printf("\n Ingrese la proxima edad: ");
20         scanf("%d", &edad);
21     }
22
23     return 0;
24 }
```

# Ejercicio

Diseñar un algoritmo que muestre la tabla de multiplicar de un número determinado.

Además se solicita que muestre por pantalla hasta que número se desea ver la tabla. Implemente en C.



# Estructura Iterativa: Hacer-Mientras()

Al igual que la estructura Mientras, hacer-mientras es utilizado para repetir acciones bajo un condición, con la diferencia que si o si se ejecuta al menos una vez.

Su notación es:

Hacer

acciones

Mientras (condición)

--> Condición: La condición puede ser simple o compuesta. En este último caso estará "unida" por una conjunción (y) o una disyunción (o).



# Estructura Iterativa: Hacer-Mientras()

Veamos un ejemplo de su funcionamiento

Analizando un sector de pseudocódigo

# Estructura Iterativa Hacer - Mientras()

...

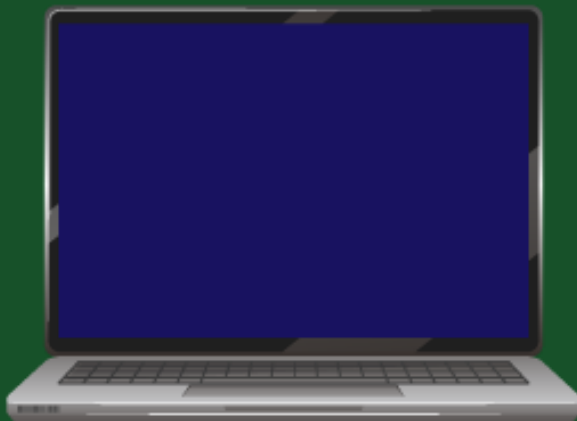
Hacer

LEER(edad, dni) ← Lee la edad y dni de la primera persona ingresada

MIENTRAS( edad < 18) ← Controla si edad es menos a 18. Como edad = 5 entonces volverá ingresar a la estructura

ESCRIBIR(dni)

...



edad	dni
5	53.321.125

# Estructura Iterativa Hacer - Mientras()

...

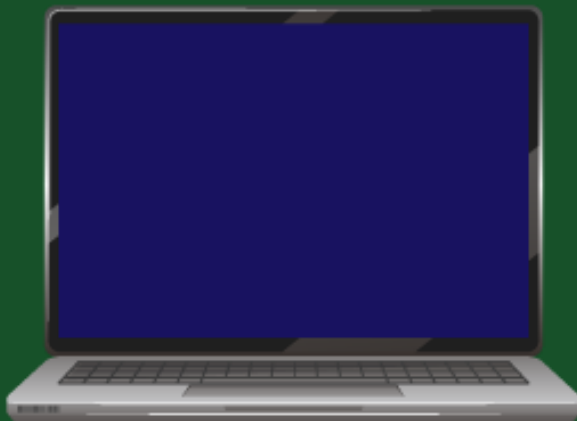
Hacer

LEER(edad, dni) ← Lee la edad y dni de la siguiente persona ingresada

MIENTRAS( edad < 18) ← Controla si edad es menor a 18. Como edad = 15 entonces volverá ingresar a la estructura.

ESCRIBIR(dni)

...



edad	dni
5	53.321.125
15	42.152.368

# Estructura Iterativa Hacer - Mientras()

...

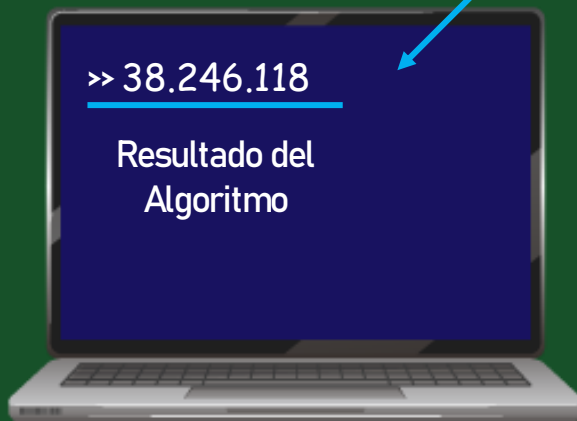
Hacer

LEER(edad, dni) ← Lee la edad y dni de la siguiente persona ingresada

MIENTRAS( edad < 18) ← Controla si edad es menor a 18. Como edad = 18 entonces sale de la estructura de iteración

ESCRIBIR(dni) ← Escribe el dni de la persona mayor a 18

...



edad	dni
5	53.321.125
15	42.152.368
18	38.246.118

# Sentencia do-while

En el lazo do-while tiene la comprobación relacional al final, en vez de tenerla al inicio como es en la estructura while. La sintaxis es:

```
do {  
  
    sentencia;  
  
    }while ( condición );
```

- Se ejecuta al menos una vez.
- La sentencia se ejecuta y después se evalúa la condición.
- Condición de salida: La condición se verifica después de la primera ejecución del bloque de. Si la condición es verdadera, el bucle continúa ejecutándose; si es falsa, el bucle termina.

# Sentencia do-while

- ❑ Retomar el ejemplo anteriormente realizado usando la estructura do-while.
- ❑ Preguntas:
  - ▣ ¿Para qué casos considera que seria de utilidad usar un do-while, en vez de un while?
  - ▣ Cual es la diferencia entre un while y un do-while?
  - ▣ ¿Se podría utilizar cualquiera de los dos de manera indistinta?

# Estructura Iterativa: HACER()

Son utilizados para repetir un número predeterminado o fijo de veces

Su notación es: **HACER 10 VECES ( $i = 1, \dots, 10$ )**

**acciones**

**Fin\_hacer**

**Variable i:**

- > Indica el número de iteraciones.
- > Se debe declarar como variable auxiliar.
- > En el diseño de algoritmo es opcional su uso.
- > En muchas ocasiones se usa dentro del bucle.



# Estructura Iterativa: HACER()

Veamos un ejemplo de su funcionamiento

Analizando un sector de pseudocódigo

# Estructura Iterativa: HACER()

...

HACER 4 VECES ( $i = 1, \dots, 4$ )

← Ingresa con  $i=1$

LEER(dni, edad)

← Lee los 2 datos solicitados

SI ( $\text{edad} > 18$ ) ENTONCES

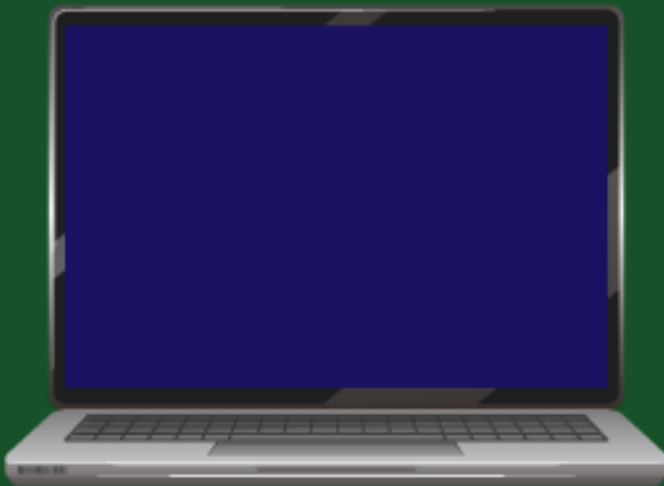
← Controla si edad es mayor a 18.  
Como edad = 5 entonces no ingresa  
a la selección

ESCRIBIR(dni)

Fin\_si

Fin\_hacer

...



i	dni	edad
1	53.321.125	5

# Estructura Iterativa: HACER()

...

HACER 4 VECES ( $i = 1, \dots, 4$ )

← Ingresar con  $i=2$

LEER(dni, edad)

← Lee los 2 datos solicitados

SI ( $\text{edad} > 18$ ) ENTONCES

← Controla si edad es mayor a 18. Como edad = 20 entonces ingresa a la selección

ESCRIBIR(dni)

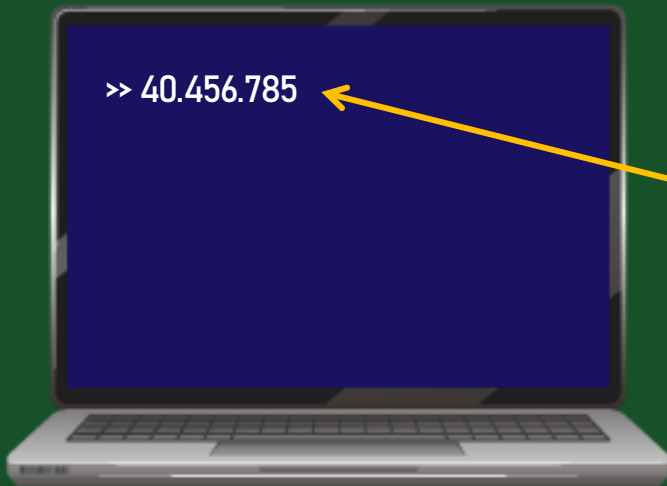
← Escribe dni

Fin\_si

← Sale de la selección

Fin\_hacer

...



i	dni	edad
1	53.321.125	5
2	40.456.785	20

# Estructura Iterativa: HACER()

...

HACER 4 VECES ( $i = 1, \dots, 4$ )

LEER(dni, edad)

SI ( $\text{edad} > 18$ ) ENTONCES

ESCRIBIR(dni)

Fin\_si

Fin\_hacer

...



Ingresa con  $i=3$

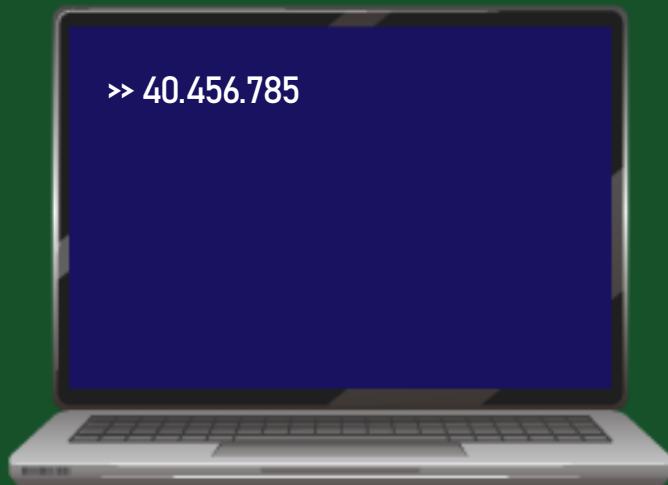


Lee los 2 datos solicitados



Controla si edad es mayor a 18.

Como edad = 18, no mayor, entonces no ingresa a la selección



i	dni	edad
1	53.321.125	5
2	40.456.785	20
3	43.128.875	18

# Estructura Iterativa: HACER()

...

HACER 4 VECES ( $i = 1, \dots, 4$ )

← Ingresa con  $i=4$

LEER(dni, edad)

← Lee los 2 datos solicitados

SI ( $\text{edad} > 18$ ) ENTONCES

← Controla si edad es mayor a 18.  
Como edad = 37 ingresa a la selección

ESCRIBIR(dni)

← Escribe dni

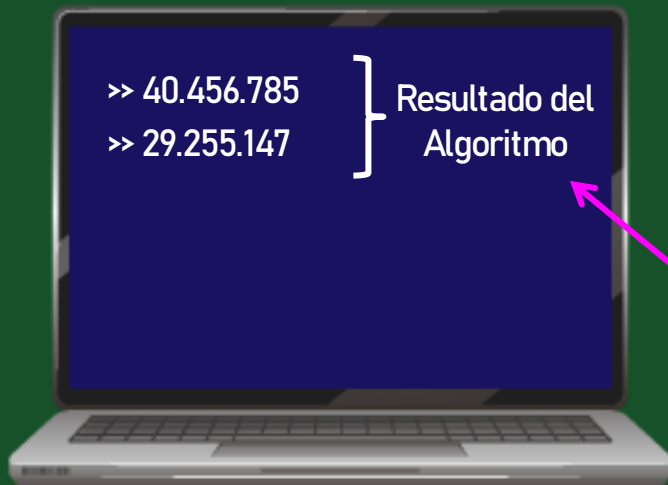
Fin\_si

← Sale de la selección

Fin\_hacer

← Sale del Hacer

...



i	dni	edad
1	53.321.125	5
2	40.456.785	20
3	43.128.875	18
4	29.255.147	37

# Estructura Iterativa: HACER()

A tener en cuenta:

- Es conveniente usar esta estructura cuando sabemos con anterioridad la cantidad de veces que se repetirá una serie de acciones.
- Podemos solicitar al usuario final que ingrese la cantidad de veces que desea que se ejecute la estructura.

Por ejemplo:

LEER(cantidad)

HACER cantidad VECES (i=1,..., cantidad)

acciones\_a\_repetir

Fin\_Hacer



# Sentencia For

A ver Marta, pase y escriba 500 veces no debo tirar papeles en clases...

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

```
int main()
```

```
{
```

```
    int cont;
```

```
    for(cont=1; cont<=500; cont++)
```

```
    {
```

```
        printf("no debo tirar papeles en clases");
```

```
    }
```

```
    return 0;
```

```
}
```

Buen Intento

desmotivaciones.es

## Cosas de programadores.



# Sentencia for

La sentencia for permite ejecutar una sentencia simple o compuesta, repetidamente un número de veces conocido.

Su sintaxis es la siguiente:

**for (expresión-comienzo; condición; progresión-condición)**

**Sentencia;**

**Expresión-comienzo:** representan variables de control que serán iniciadas con los valores de las expresiones.

**Condición:** es una expresión booleana que si se omite, se supone verdadera.

**Progresión-condición:** es una o más expresiones separadas por comas cuyos valores evolucionan en el sentido de que se cumpla la condición para finalizar la ejecución de la sentencia for.

**Sentencia:** es una sentencia simple o compuesta.

# Ejemplo

Ya conocemos la sintaxis de la estructura **for()** en C

Entonces estamos listos para implementar el algoritmo visto en el ejemplo



# Ejemplo

Algoritmo MayorDeEdad

ENTRADA: edad, dni: enteros

SALIDA: dni: entero

A1. HACER 4 VECES (i= 1,...,4)

LEER(dni, edad)

SI (edad  $\geq$  18) ENTONCES

ESCRIBIR(dni)

Fin\_si

Fin\_hacer

A2. PARAR

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int edad, dni;
6
7      for (int i = 0; i < 4; i++)
8      {
9          printf("\n Ingrese la edad: ");
10         scanf("%d", &edad);
11
12         printf("\n Ingrese el dni: ");
13         scanf("%d", &dni);
14         if(edad >=18)
15         {
16             printf("La persona con DNI %d es mayor de edad", dni);
17         }
18     }
19
20     return 0;
21 }
```

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

```
int main()  
{
```

```
    printf("Hasta la próxima clase!!\n");
```

```
    return 0;
```

```
}
```