

Tema 4: Estructuras de Selección y Control

Introducción a la programación I

Ana Isabel Sierra de las Heras

Marcos Novalbos

Francisco Javier Garcia Algarra

Rodrigo Alonso Solaguren-Beascoa

Alfonso Castro Escudero

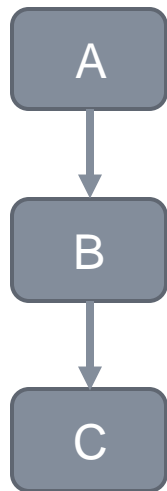
Índice

1. Estructura de control
2. Estructuras de alternativa o selección
 1. If con una alternativa
 2. If con dos alternativas: if-else
 3. If_else anidados
 4. Sentencia de control: switch
3. Estructuras de iteración o repetitivas
 1. Sentencia while
 2. Repetición: bucle for
 3. Repetición: bucle do .. while

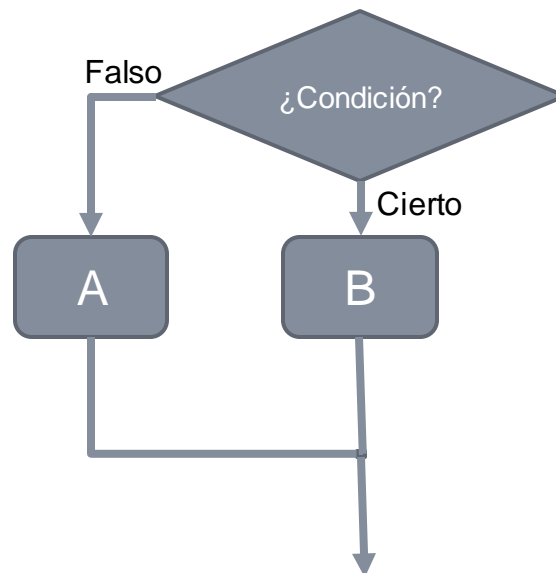
1. Estructuras de control

- Las estructuras de control gestionan el flujo de ejecución de un programa o función.
- Las instrucciones o sentencias se organizan en tres tipos:

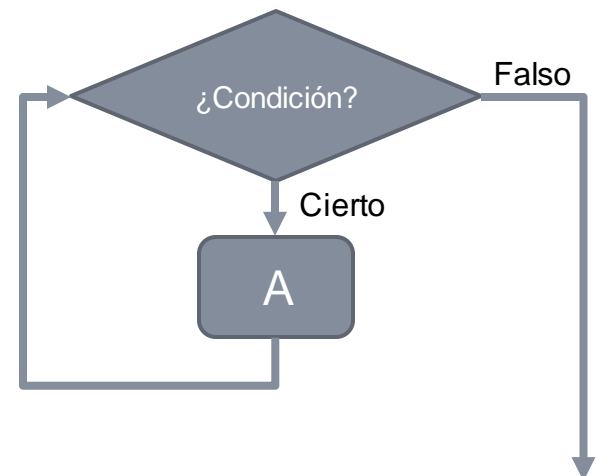
Secuencial



Alternativa o selección



Iteración o Repetitiva



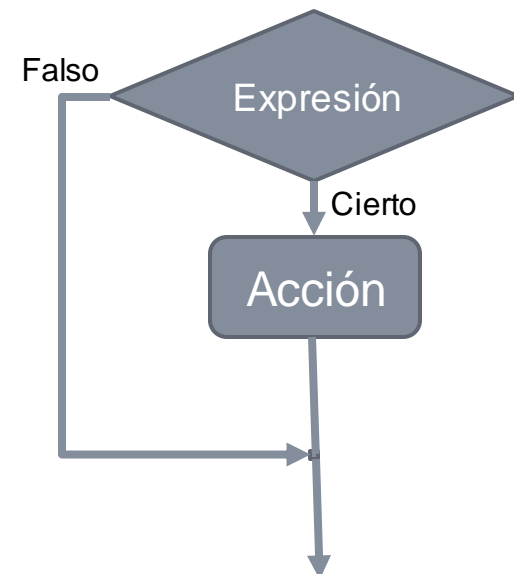
2. Estructuras de alternativa o selección. if con una alternativa

- El formato más sencillo es **if con una alternativa** y tiene la siguiente sintaxis:

```
if (expresión lógica)  
    acción;
```

- Si la expresión lógica es verdadera se ejecuta la acción, en caso contrario no se ejecuta la acción.
- Si hay más de una acción dentro del bloque de ejecución, entonces son obligatorias las llaves de apertura y cierre

```
if (expresión lógica) {  
    accion1;  
    ...  
    accionN;  
}
```



2. Estructuras de alternativa o selección.

if con una alternativa

- Ejemplo 1:
 - Pide un número entero por teclado y si es par imprímelo por pantalla diciendo que es par.

2. Estructuras de alternativa o selección. if con una alternativa

- Ejemplo 1:
 - Pide un número entero por teclado y si es par imprímelo por pantalla diciendo que es par.

```
#include <stdio.h>

void main(){
    int numero;

    printf ("Introduce un número: \n");
    scanf ("%d", &numero);
    if ( numero%2 == 0 )
        printf ("El número %d es par\n", numero);
}
```

2. Estructuras de alternativa o selección.

if con una alternativa

- Ejemplo 1:
 - Pide un carácter por teclado y si es 's' mayúscula o minúscula imprímelo por pantalla.

2. Estructuras de alternativa o selección. if con una alternativa

- Ejemplo 2:
 - Pide un carácter por teclado y si es 's' mayúscula o minúscula imprímelo por pantalla.

```
#include <stdio.h>

void main(){
    char Character;

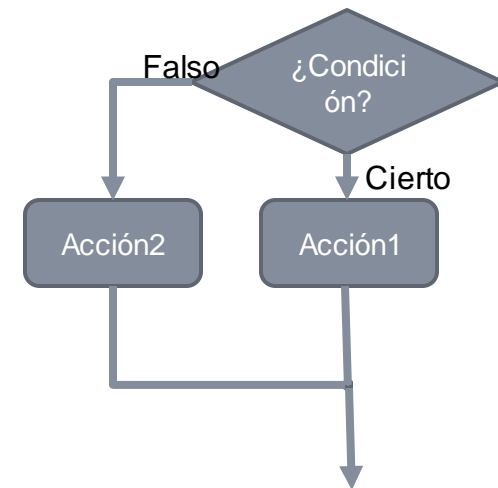
    printf ("Introduce un Character: \n");
    scanf ("%c", &Character);
    if ((Character == 'S') || (Character == 's'))
        printf ("El Character es %c\n", Character);
}
```


2. Estructuras de alternativa o selección. **if con dos alternativas**

- La sentencia **if** puede tener dos **alternativas** y tiene la siguiente sintaxis:

```
if (Condición)
    Acción1;
else
    Acción2;
```

- Si la condición es verdadera se ejecuta la acción1, en caso contrario se ejecuta la acción2.



2. Estructuras de alternativa o selección. if con dos alternativas

- **Si los bloques a ejecutar en cualquiera de las alternativas tienen más de una acción o instrucción, entonces es obligatorio el uso de las llaves:**

```
if (Condición) {  
    acción1;  
    acción2;  
    ...  
    accionN;  
}  
else {  
    acción1;  
    acción2;  
    ...  
    accionN;  
}
```

2. Estructuras de alternativa o selección.

if con dos alternativas

- Ejemplo 2:
 - Pide un número entero por teclado y si es par imprímelo por pantalla diciendo que es par. Si no, imprime por pantalla un mensaje que lo indique

2. Estructuras de alternativa o selección. if con dos alternativas

- Ejemplo 2:
 - Pide un número entero por teclado y si es par imprímelo por pantalla diciendo que es par. Si no, imprime por pantalla un mensaje que lo indique

```
#include <stdio.h>

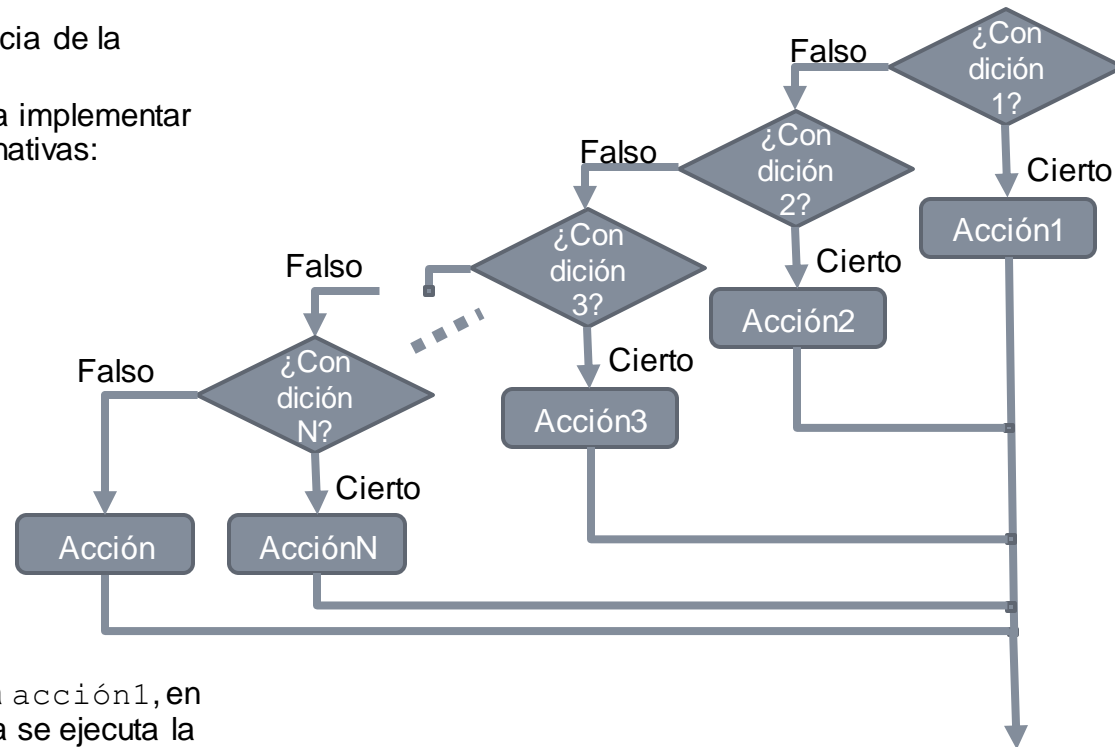
void main(){
    char Character;

    printf ("Introduce un Character: \n");
    scanf ("%c", &Character);
    if ((Character == 'S')||(Character == 's'))
        printf ("El Character es %c\n", Character);
    else
        printf ("El Character no es s o S\n");
}
```

2. Estructuras de alternativa o selección. if anidados

- Una sentencia **if es anidada** cuando la sentencia de la rama falsa es a su vez una sentencia if.
- Una sentencia if anidada se puede utilizar para implementar decisiones con varias alternativas o multi-alternativas:

```
if (Condición1)
    Acción1;
else if (Condición2)
    Acción2;
else if (Condición3)
    Acción3;
...
else if (CondiciónN)
    AcciónN;
else
    Acción;
```



- Si la condición1 es verdadera se ejecuta la acción1, en caso contrario si la condición2 es verdadera se ejecuta la acción2, en caso contrario si la condición3 es verdadera se ejecuta la acción3, ..., en caso contrario si la condiciónN es verdadera se ejecuta la acciónN, en caso contrario se ejecuta la acción

2. Estructuras de alternativa o selección.

if anidados

- Ejemplo 3:
 - Pide un número entero de tres cifras por teclado. Comprueba si la tercera cifra (centena) del número es 1, 2, 3 o 5 escribiendo por pantalla que la centena del número es 1, 2, 3 o 5. Si no es así, el programa deberá escribir que la centena no es 1, 2, 3 o 5.

2. Estructuras de alternativa o selección. if anidados

■ Ejemplo 3:

- Pide un número entero de tres cifras por teclado. Comprueba si la tercera cifra (centena) del número es 1, 2, 3 o 5 escribiendo por pantalla que la centena del número es 1, 2, 3 o 5. Si no es así, el programa deberá escribir que la centena no es 1, 2, 3 o 5.

```
int numero = 0;
int centena = 0;

printf ("Introduce un numero entero de 3 cifras: \n");
scanf ("%d",&numero);

centena = numero / 100;

if (centena == 1)
    printf ("La centena del numero es %d\n", centena);
else if (centena == 2)
    printf ("La centena del numero es %d\n", centena);
else if (centena == 3)
    printf ("La centena del numero es %d\n", centena);
else if (centena == 5)
    printf ("La centena del numero es %d\n", centena);
else
    printf ("La centena del numero no es ni 1 ni 2 ni 3 ni 5\n");
```

2. Estructuras de alternativa o selección.

if anidados

- Ejemplo 3bis:
 - Pide un carácter b, r, n o v y devuelve un color que empieza por ese carácter o un mensaje indicando que no hay ningún color asignado

2. Estructuras de alternativa o selección. if anidados

▪ Ejemplo 3bis:

- Pide un carácter b, r, n o v y devuelve un color que empieza por ese carácter.

```
char character;

printf("Introduce uno de estos caracteres por pantalla: b, r, n, v \n");
scanf("%c", &character);

if (character == 'b' || character == 'B')
    printf("Has escogido el color blanco\n");

else if (character == 'r' || character == 'R')
    printf("Has escogido el color rojo\n");

else if (character == 'n' || character == 'N')
    printf("Has escogido el color negro\n");

else if (character == 'v' || character == 'V')
    printf("Has escogido el color verde\n");

else
    printf("No has escogido ningun color\n");
```

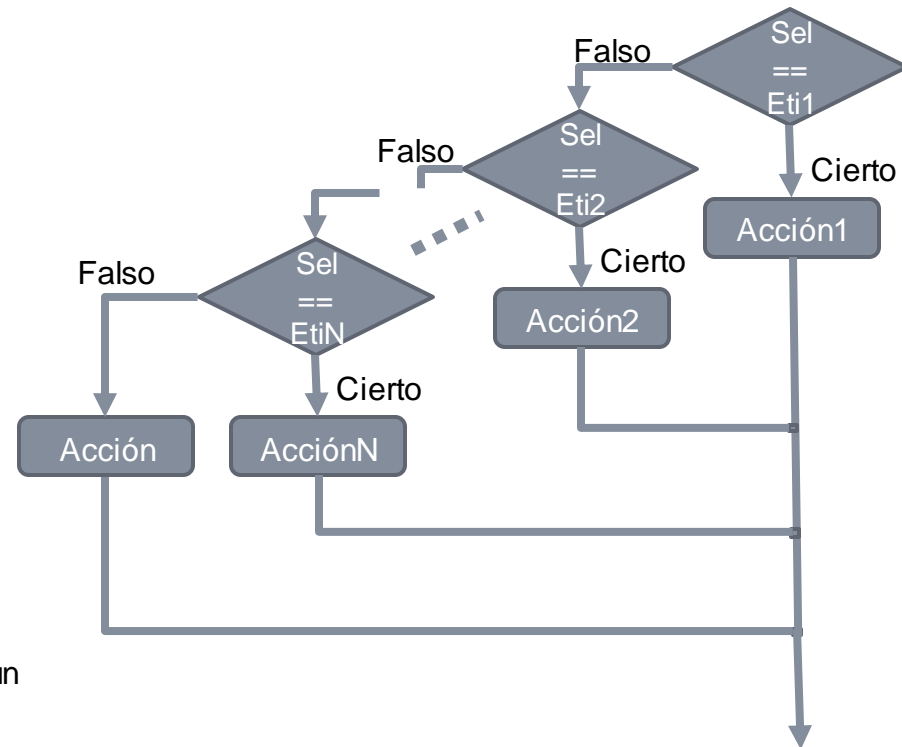
2. Estructuras de alternativa o selección.

switch

- La sentencia **switch** se utiliza para seleccionar una de entre múltiples alternativas.

```
switch (selector) {  
    case etiqueta1: Acción1;  
                    break;  
    case etiqueta2: Acción2;  
                    break;  
    ...  
    case etiquetaN: AcciónN;  
                    break;  
    default : Acción; /*opcional*/  
}
```

- La expresión `selector` es evaluada y debe producir un resultado de tipo `int` o `char`
- Cada etiqueta es un valor único constante y debe tener un valor diferente a las otras etiquetas
- Si el valor de `selector` es igual a una de las etiquetas case entonces se ejecutará la acción asociada y continuará hasta que se encuentra la sentencia `break` o el final de la sentencia `switch`.



2. Estructuras de alternativa o selección.

switch

- Ejemplo 4:

- Pide un número entero por teclado de tres cifras. Comprueba si la cifra de sus centenas es 1, 2, 3 o 5 escribiendo por pantalla que la centena del número es 1, 2, 3 o 5. Si no es así, el programa deberá escribir que la centena no es 1, 2, 3 o 5. Realizar el ejemplo mediante un switch-case.

2. Estructuras de alternativa o selección.

switch

■ Ejemplo 4:

- Pide un número entero de tres cifras por teclado. Comprueba si la tercera cifra del número es 1, 2, 3 ó 5 escribiendo por pantalla que la centena del número es 1, 2, 3 ó 5. Si no es así, el programa deberá escribir que la centena no es 1, 2, 3 ni 5.

```
int numero = 0;
int centena = 0;
printf ("Introduce un numero entero de 3 cifras: \n");
scanf ("%d",&numero);

centena = numero / 100;

switch (centena){
case 1:
    printf ("La centena del numero es %d\n", centena);
    break;
case 2:
    printf ("La centena del numero es %d\n", centena);
    break;
case 3:
    printf ("La centena del numero es %d\n", centena);
    break;
case 5:
    printf ("La centena del numero es %d\n", centena);
    break;
default:
    printf ("La centena del numero no es ni 1 ni 2 ni 3 ni 5\n");
}
```

2. Estructuras de alternativa o selección.

switch

- Ejemplo 4bis:
 - Pide un carácter b, r, n, v y devuelve un color que empieza por ese carácter o un mensaje indicando que no hay ningún color asignado. Realizar el ejemplo mediante un switch-case.

2. Estructuras de alternativa o selección.

switch

■ Ejemplo 4bis:

- Pide un carácter b, r, n o v y devuelve un color que empieza por ese carácter o un mensaje indicando que no hay ningún color asignado.

```
char character;
printf("Introduce uno de estos caracteres por pantalla: b, r, n, v \n");
scanf("%c", &character);

switch (character) {
    case 'b':
        printf("Has escogido el color blanco\n");
        break;
    case 'r':
        printf("Has escogido el color rojo\n");
        break;
    case 'n':
        printf("Has escogido el color negro\n");
        break;
    case 'v':
        printf("Has escogido el color verde\n");
        break;
    default:
        printf("No has escogido ningun color\n");
}
```

2. Estructuras de alternativa o selección.

switch

- Ejemplo 4bis:
 - Pide un carácter b, r, n o v y devuelve un color que empieza por ese carácter o un mensaje indicando que no hay ningún color asignado. Realizar el ejemplo mediante un switch-case.
 - Contemplar el uso de mayúsculas

2. Estructuras de alternativa o selección.

switch

■ Ejemplo 4bis:

- Pide un carácter b, r, n o v y devuelve un color que empieza por ese carácter o un mensaje indicando que no hay ningún color asignado. Acepta mayúsculas

```
char caracter;  
printf("Introduce uno de estos caracteres por pantalla: b, r, n, v \n");  
scanf("%c", &caracter);  
  
switch (caracter) {  
    case 'b':  
    case 'B':  
        printf("Has escogido el color blanco\n");  
        break;  
    case 'r':  
    case 'R':  
        printf("Has escogido el color rojo\n");  
        break;  
    case 'n':  
    case 'N':  
        printf("Has escogido el color negro\n");  
        break;  
    case 'v':  
    case 'V':  
        printf("Has escogido el color verde\n");  
        break;  
    default:  
        printf("No has escogido ningun color\n");  
}
```

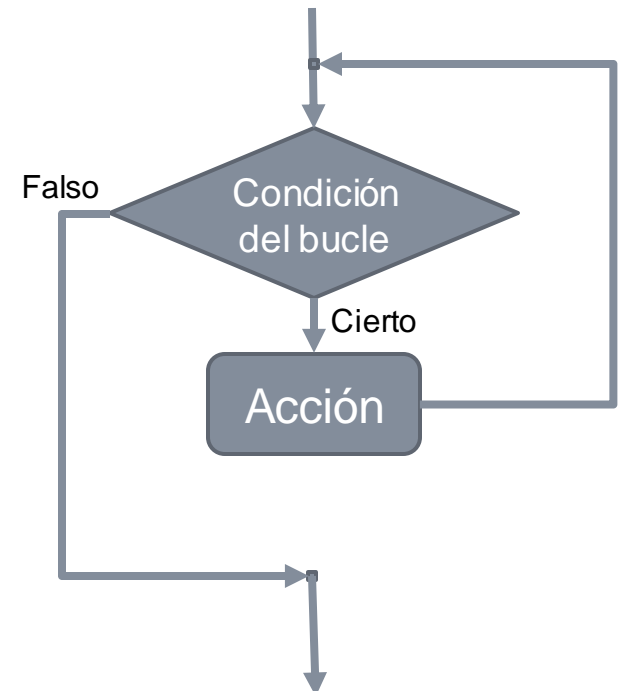

3. Estructuras de iteración o repetitivas

while

- La sentencia `while` va acompañada de una condición (expresión lógica) que controla la secuencia de repetición, es decir, controla el número de veces que se debe ejecutar el bucle.
- La condición va delante del cuerpo del bucle y fuerza a que sea evaluada antes de que comience a ejecutarse el cuerpo del bucle.

```
while (condición del bucle) {  
    Acción1  
    Acción2;  
    ...  
    AcciónN;  
}
```

- La acción (o acciones) se repiten mientras la condición del bucle permanece verdadera y termina cuando se hace falsa
- El cuerpo del bucle se ejecutará cero o más veces
- Bucle infinito: Si la variable de control no se actualiza dentro del bloque la condición no se hace falsa en ninguna iteración y el bucle se ejecutará siempre.



3. Estructuras de iteración o repetitivas

while

- Ejemplo:
- Pide una palabra por pantalla, cuenta los caracteres que tiene y devuelve el resultado

3. Estructuras de iteración o repetitivas

while

- Pide una palabra por pantalla, cuenta los caracteres que tiene y devuelve el resultado

```
int contador = 0;
char character = '\0';
int seguir = 1; //inicializamos a uno porque queremos
                //que la primera vez se ejecute siempre

printf ("Introduce una palabra: ");

while (seguir) {
    character = getchar();
    if (character == ' ' || character == '\n')
        seguir = 0;
    else
        contador++;
}
printf("La palabra contiene %d caracteres \n", contador);
```

3. Estructuras de iteración o repetitivas

while

- Ejemplo 5: . Adivinar un número de 0 al 10
 - Genera un numero aleatorio entre 0 y 10 utilizando la función rand.
 - Pide al usuario que introduzca número entero por teclado.
 - Comprueba si el número introducido coincide con el generado.
 - A la salida imprime el número de intentos para adivinarlo.

3. Estructuras de iteración o repetitivas

while

■ Ejemplo 5: Adivinar un número de 0 al 10

- Genera un numero aleatorio entre 0 y 10 utilizando la función rand. Pide al usuario que introduzca número entero por teclado. Comprueba si el número introducido coincide con el generado. A la salida imprime el número de intentos para adivinarlo.

```
#include <stdio.h>
#include <stdlib.h> /* rand */
#include <unistd.h> /* getpid */

int main (){
    int numero_aleatorio = 0, numero = 0, adivinado = 0, contador = 0;

    /* Hay que alimentar a rand, solamente una vez (seed rand) */
    srand(getpid());

    /* Generación de un número aleatorio entre 0 y 10 */
    /* La operación módulo (%) nos da el resto de dividir rand() entre 11. Este resto puede ir de 0 a 10 */
    numero_aleatorio = rand () % 11;

    while (adivinado == 0){
        printf ("\nIntroduce un numero entero entre 0 y 10: ");
        scanf ("%d",&numero);
        if (numero == numero_aleatorio)
            adivinado = 1;
        contador ++;
    }
    printf ("\nHa adivinado el numero %d en %d veces \n", numero_aleatorio,contador);

    return 0;
}
```

3. Estructuras de iteración o repetitivas

while

- Ejemplo 6: .
 - Pide al usuario que introduzca su edad e imprímela por pantalla. ¿Qué ocurre si introducimos un carácter?.

3. Estructuras de iteración o repetitivas

while

- Ejemplo 6: .
 - Pide al usuario que introduzca su edad e imprímela por pantalla. ¿Qué ocurre si introducimos un carácter?.

```
#include <stdio.h>

void main(){
    int Edad;
    int ResultadoScanf = 0;

    printf ("Introduce tu Edad: \n");
    while (ResultadoScanf==0){
        ResultadoScanf = scanf ("%d", &Edad);
        printf ("el resultado de scanf es %d\n",ResultadoScanf);
        if (ResultadoScanf==0){
            printf ("No has introducido numero. Introduce un numero: \n");
            while (getchar() !='\n');
        }
    }
    printf ("La Edad es %d\n", Edad);
}
```

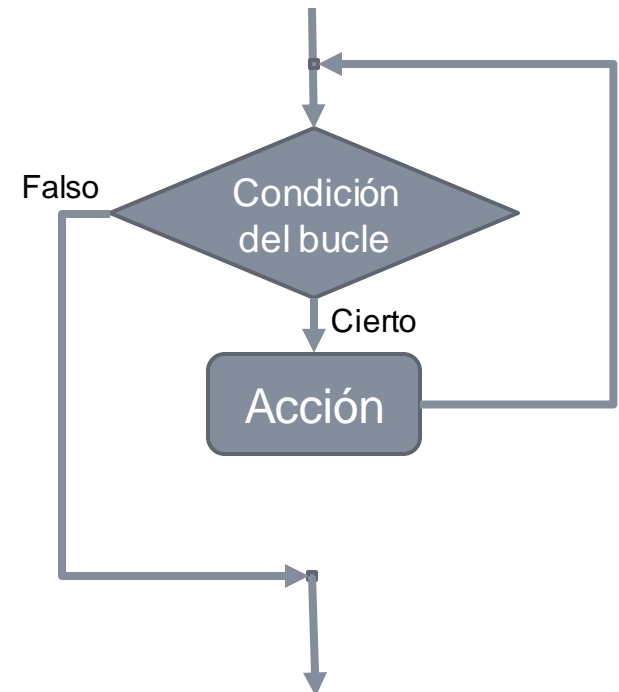
3. Estructuras de iteración o repetitivas

for

- Un bucle **for** es el más adecuado para implementar bucles controlados por contador.
- Son bucles en los que un conjunto de sentencias se ejecuta una vez por cada valor de una variable un número específico de veces
- El bucle for se diferencia del bucle `while` en que las opciones de control se sitúan solo en la cabecera de la sentencia

```
for (inicialización; CondiciónIteración; Incremento)
{
    Instrucciones;
}
```

- Inicialización: Se inicializa las variables de control del bucle
- Condición. Contiene la expresión lógica que hace que el bucle realice las iteraciones de las instrucciones, mientras que la expresión sea verdadera.
- Incremento: Incrementa o decrementa la variable de control del bucle.
- Instrucciones: acciones que se ejecutarán por cada iteración del bucle



3. Estructuras de iteración o repetitivas for

- **Ejemplo 7: Calcular el factorial de un número**
 - Pide al usuario que introduzca número entero por teclado.
 - Calcula el factorial del número introducido e imprímelo por pantalla.

3. Estructuras de iteración o repetitivas

for

- **Ejemplo 7: Calcular el factorial de un número**
 - Pide al usuario que introduzca un número entero por teclado.
 - Calcula el factorial del número introducido e imprímelo por pantalla.

```
#include <stdio.h>

int main () {
    int numero = 0;
    int factorial = 1 ;

    printf ("Introduce un numero entero mayor que 0 \n");
    scanf ("%d",&numero);

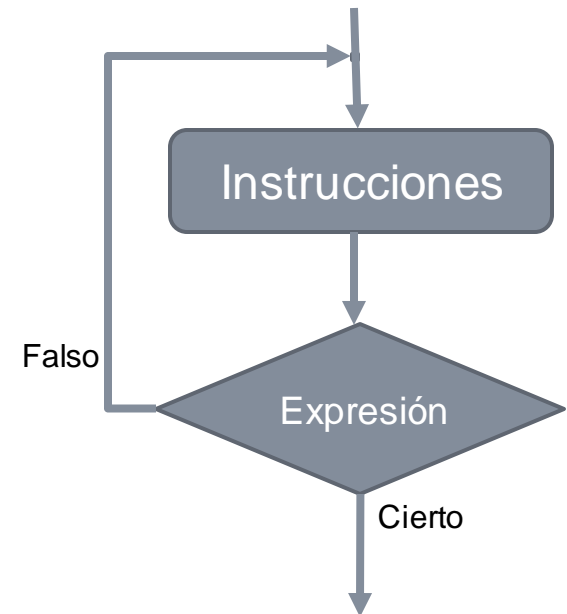
    for (int i = 1; i <= numero; i++ ){
        factorial = factorial * i;
    }
    printf ("El factorial de %d es: %d \n", numero, factorial);

    return 0;
}
```

3. Estructuras de iteración o repetitivas

do-while

- Hemos visto que el caso del bucle **while** se comprueba su condición antes de ejecutar el cuerpo del bucle. Esto significa que si la condición no se cumple inicialmente, entonces el bucle no se ejecuta ninguna vez. Para los casos en que queremos garantizar que el bucle se ejecute al menos una vez existe el bucle do-while. Este tipo de estructura repetitiva hace una primera pasada por el cuerpo del bucle y después comprueba la condición, terminando el bucle si ésta es falsa, o volviendo a ejecutarlo si es cierta.
- La sintaxis es:
 - `do{`
 - `instrucción1;`
 - `instrucción2;`
 - `...`
 - `instrucciónN;`
 - `}while(condición);`
- En este caso se ejecutarán las instrucciones `instrucción1 . . . instrucciónN` y después se evaluará la condición. Si es falsa se continúa con la instrucción que sigue al bucle y si es cierta se vuelve a repetir el cuerpo del bucle (`instrucción1 . . . instrucciónN`), después se evalúa la condición de nuevo y así sucesivamente.



3. Estructuras de iteración o repetitivas

do-while

- **Ejemplo 8: Calcular el factorial de un número**
 - Pide al usuario que introduzca número entero positivo por teclado.
 - Utiliza un bucle de tipo do-while para asegurar que el número no es negativo
 - Calcula el factorial del número introducido e imprímelo por pantalla.

3. Estructuras de iteración o repetitivas

do-while

■ Ejemplo 8: Calcular el factorial de un número

- Pide al usuario que introduzca número entero positivo por teclado.
- Utiliza un bucle de tipo do-while para asegurar que el número no es negativo
- Calcula el factorial del número introducido e imprímelo por pantalla.

```
#include <stdio.h>

int main (){
    int numero = 0;
    int factorial = 1 ;

    do{
        printf ("Introduce un numero entero mayor que 0 \n");
        scanf ("%d",&numero);
    }while (numero <= 0);

    for (int i = 1; i < numero+1; i++ ){
        factorial = factorial * i;
    }
    printf ("El factorial de %d es: %d \n", numero, factorial);

    return 0;
}
```

3. Estructuras de iteración o repetitivas

while, for, do-while

while	El uso más frecuente es cuando la repetición no está controlada por contador, el test de condición precede a cada repetición del bucle, el cuerpo del bucle puede no ser ejecutado. Se debe utilizar cuando se desea saltar el bucle si la condición es falsa
for	Bucle de conteo cuando el número de repeticiones se conoce por anticipado y puede ser controlado por un contador. También es adecuado para bucles que implican control no contable del bucle con simples etapas de inicialización y de actualización. El test de la condición precede a la ejecución del cuerpo del bucle.
do-while	Es adecuada cuando se debe asegurar que al menos se ejecuta el bucle una vez.

4. Ejercicios del tema (enunciados)

Ejercicio 1

- Calcular las raíces reales de un polinomio de grado 2, expresado por el producto de tres términos cada uno integrado por un el producto de un coeficiente y las potencias de x elevadas a 2, 1 y 0.
$$a x^2 + b x + c = 0$$
- Usar la función `sqrt` de la librería `math.h` cuyo prototipo es `double sqrt(double x);` Returns the square root of x

4. Ejercicio 2

- Encontrar el máximo común divisor (MCD) de dos números enteros y positivos. Utilizar el algoritmo de Euclides
 1. Se divide el número mayor entre el menor.
 2. Si la división es exacta, el divisor es el m.c.d.
 3. Si la división no es exacta, dividimos el divisor entre el resto obtenido y continuamos de esta forma hasta obtener una división exacta.
 4. El m.c.d. es el último divisor.

4. Ejercicio 3

- Escribir un programa que calcule y visualice el más grande, el más pequeño y la media de n números ($n > 0$). El valor de n se solicitará al principio del programa y los números serán introducidos por el usuario.

4. Ejercicio 4

- Comprobar si un número entero es primo, teniendo en cuenta que un primo solo tiene 2 divisores.

4. Ejercicio 5

- Un número perfecto es un entero, que es igual a la suma de todos los enteros positivos (excluido el mismo) que son divisores del número. El primer número perfecto es el 6 ($1+2+3=6$)
- Escribir un programa que pida un numero entero positivo y determine si es perfecto

4. Ejercicio 6

- Escribir un programa que presente en modo tabla los valores de la función

$\text{seno } (2x) - x$ para $x = 0, 0.5, 1, 1.5 \dots 9.5, 10$

Utilizar la función `sin` de `<math.h>`



CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL