

Tema 3: Sentencias de Control

Oscar Perpiñán Lamigueiro

- 1 **Introducción**
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

Introducción

- Sin sentencias de control los programas se ejecutan de manera secuencial

Sentencias de Control

Sentencias Condicionales ejecutan unas secuencias u otras según el cumplimiento de unas condiciones.

- `if, if - else`
- `switch - case`

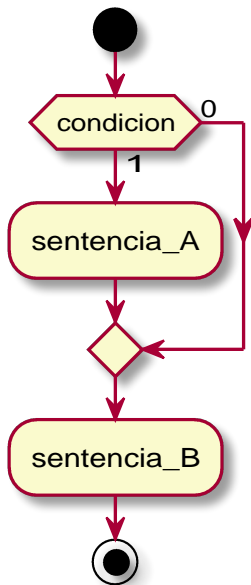
Sentencias Repetitivas repiten un conjunto de sentencias en unas determinadas condiciones.

- `for`
- `while, do - while`

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

if



if

- Si se cumple condicion (su resultado es **diferente de 0**) se ejecuta la sentencia_A.
- **Siempre** se ejecuta la sentencia_B.

```
if (condicion)
    sentencia_A;
sentencia_B;
```

- Para ejecutar un conjunto de sentencias hay que agruparlas **entre llaves**.

```
if (condicion)
{
    sentencia_A1;
    sentencia_A2;
    ...
}
sentencia_B1;
sentencia_B2;
...
```

Ejemplo if

```
# include <stdio.h>
int main ()
{
    int n;
    printf("Escribe un número entero\n");
    scanf("%d", &n);
    if (n % 2 == 0) // Condición
    { // Uso de llaves
        printf("Se cumple la condición: ");
        printf("El número %d es par.\n", n);
    } // Fin de if
    printf("Gracias por participar.\n");
    return 0;
}
```

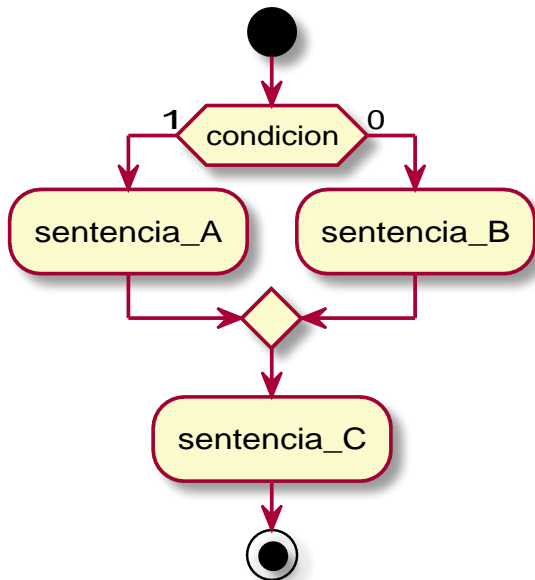

Ejemplo if

```
# include <stdio.h>
int main ()
{
    int n;
    printf("Escribe un número entero\n");
    scanf("%d", &n);
    if (n % 2 == 0) // Sin llaves para sentencias simples
        printf("El número %d es par.\n", n);
    // Fin de if
    printf("Gracias por participar.\n");
    return 0;
}
```

Ejemplo if

```
# include <stdio.h>
int main ()
{
    int n;
    printf("Escribe un número entero\n");
    scanf("%d", &n);
    if (n % 2) //Esta condición no es un booleano
        //La sentencia se ejecuta cuando la condición
        //no es igual a cero (n % 2 != 0)
        printf("El número %d es impar.\n", n);
    printf("Gracias por participar.\n");
    return 0;
}
```

if - else



if - else

if Si se cumple condicion (su resultado es **diferente de 0**) se ejecuta la sentencia_A.

else En caso contrario se ejecuta la sentencia_B.

Siempre se ejecuta la sentencia_C.

```
if (condicion)
    sentencia_A;
else
    sentencia_B;
sentencia_C;
```

if - else

- Para ejecutar un conjunto de sentencias hay que agruparlas **entre llaves**.

```
if (condicion)
{
    sentencia_A1;
    sentencia_A2;
    ...
}
else
{
    sentencia_B1;
    sentencia_B2;
    ...
}
sentencia_C1;
sentencia_C2;
...
```

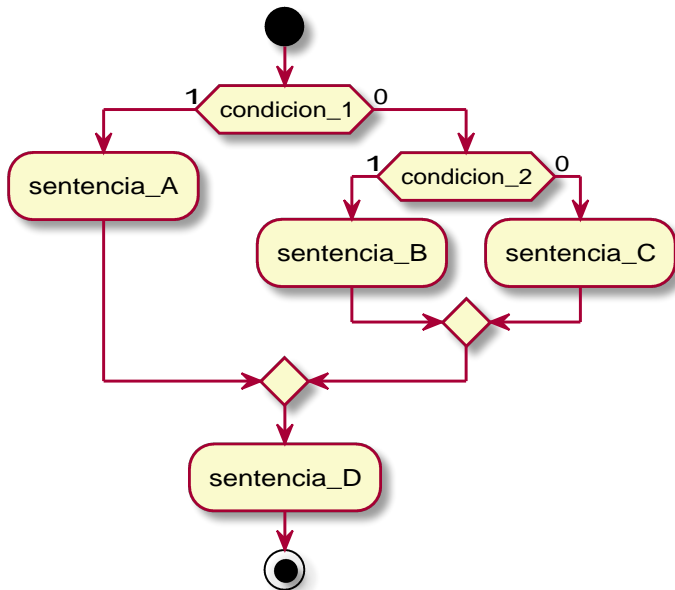
Ejemplo if-else

```
# include <stdio.h>
void main ()
{
    int n;
    printf("Escribe un número entero\n");
    scanf("%d", &n);
    if (n % 2 == 0) // condicion
    { // Inicio de if
        printf("Se cumple la condición: ");
        printf("El número %d es par.\n", n);
    }
    else
    { // Inicio de else
        printf("No se cumple la condición: ");
        printf("El número %d es impar.\n", n);
    } // Fin de if - else
    printf("Gracias por participar.\n");
}
```

Ejemplo if-else

```
# include <stdio.h>
int main ()
{
    int n;
    printf("Escribe un número entero\n");
    scanf("%d", &n);
    if (n % 2 == 0)
        printf("El número %d es par.\n", n);
    else
        printf("El número %d es impar.\n", n);
    printf("Gracias por participar.\n");
    return 0;
}
```

if - else - if



if - else - if

if Si se cumple condicion_1 se ejecuta la sentencia_A.
else En caso contrario ...
 if si se cumple condicion_2 se ejecuta la
 sentencia_B.
 else En caso contrario se ejecuta sentencia_C.
Siempre se ejecuta la sentencia_D.

```
if (condicion_1)
    sentencia_A;
else
    if (condicion_2)
        sentencia_B;
    else
        sentencia_C;
sentencia_D;
```

if - else - if

if Si se cumple condicion_1 se ejecuta la sentencia_A.

else if En caso contrario, si se cumple condicion_2 se ejecuta la sentencia_B.

else En caso contrario se ejecuta sentencia_C.

Siempre se ejecuta la sentencia_D.

```
if (condicion_1)
    sentencia_A;
else if (condicion_2)
    sentencia_B;
else
    sentencia_C;
sentencia_D;
```

Ejemplo if - else - if

```
#include <stdio.h>

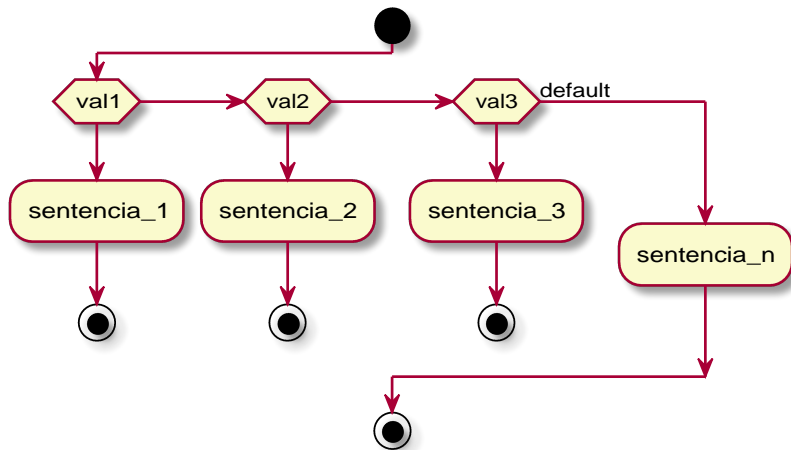
void main()
{
    int x, signo;
    printf("Escribe un número: ");
    scanf("%i", &x);

    if (x < 0)
        // se cumple la primera condición
        printf("El número es negativo.\n");
    else if (x == 0)
        // se cumple la segunda
        printf("El número es 0.\n");
    else
        // no se cumple ninguna
        printf("El número es positivo.\n");
}
```

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

switch-case

- Permite tomar una decisión múltiple dependiendo del valor **entero** de una expresión.



switch-case

- Permite tomar una decisión múltiple dependiendo del valor **entero** de una expresión.

```
switch (expr)
{
    case val1:
        sentencia_1;
        break;
    case val2:
        sentencia_2;
        break;
    case val3:
        sentencia_3;
        break;
    ...
    default:
        sentencia_n;
        break;
}
```

Ejemplo de switch - case

```
#include <stdio.h>
void main ()
{
    float v1, v2;
    char op;
    scanf("%f %c %f", &v1, &op, &v2);
    switch(op)
    {
        case '+':
            printf("%.2f\n", v1 + v2);
            break;
        case '-':
            printf("%.2f\n", v1 - v2);
            break;
        default:
            printf("No se hacer esa operación.\n");
            break;
    }
}
```

Atención al uso de break

```
#include <stdio.h>
void main ()
{
    float v1, v2;
    char op;
    scanf("%f %c %f", &v1, &op, &v2);
    switch(op)
    {
        case '+':
            printf("%.2f\n", v1 + v2);
        case '-':
            printf("%.2f\n", v1 - v2);
        default:
            printf("No se hacer esa operación.\n");
            break;
    }
}
```


Uso de llaves con switch

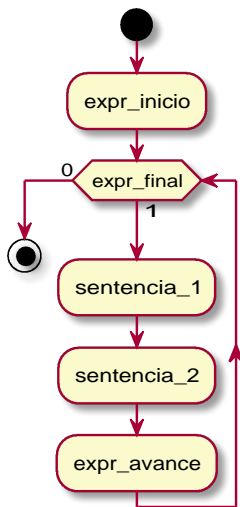
```
#include <stdio.h>
void main (){
    float v1, v2;
    char op;
    scanf("%f %c %f", &v1, &op, &v2);
    switch(op)
    {
        case '+':
            printf("Operación Suma:\n");
            printf("%.2f\n", v1 + v2);
            break;
        case '-':
            printf("Operación Resta:\n");
            printf("%.2f\n", v1 - v2);
            break;
        default:
            printf("No se hacer esa operación.\n");
            break;
    }
}
```

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - **`for`**
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

for: Flujo

Ejecuta una sentencia (simple o compuesta) un número determinado de veces hasta que el resultado de una expresión sea falso.



for : Código

Ejecuta una sentencia (simple o compuesta) un número determinado de veces hasta que el resultado de una expresión sea falso.

```
for (expr_inicio; expr_final; expr_avance)
{
    sentencia_1;
    ...
}
```

expr_inicio Expresión de inicialización (se ejecuta una sola vez).
Sirve para iniciar las variables de control del bucle.

expr_final Expresión numérica, relacional o lógica. Si es falsa se acaba el bucle (si se omite, se considera siempre verdadera y, por tanto, bucle infinito).

expr_avance Expresión (o expresiones separadas por comas) de progresión del bucle.

Ejemplo: suma de enteros

```
#include <stdio.h>

int main()
{
    // Todas las variables deben estar definidas
    // Asigno valor inicial 0 a suma
    int i, suma = 0, n = 10;

    for (i = 1; i <= n; i++)
    {
        suma += i;
    }
    printf("La suma de los %d primeros enteros es %d",
        n, suma);
    return 0;
}
```

Ejemplo: factorial

```
#include <stdio.h>

int main()
{
    int i, n = 20;
    // El factorial alcanza valores grandes
    // y siempre es positivo.
    unsigned long int fact;
    // La expresión de inicio puede ser múltiple,
    // separando por comas.
    // Aquí asigno valor inicial a fact
    for (i = 1, fact = 1; i <= n; i++)
    {
        fact *= i;
    }
    printf("El factorial de %d es %lu",
        n, fact);
    return 0;
}
```

Ejemplo: alfabeto

```
#include <stdio.h>

int main()
{
    char i;
    // Se pueden usar char en las expresiones
    for (i = 'a'; i <= 'z'; i++)
    {
        printf("%c", i);
    }
    return 0;
}
```


Ejemplo: bucles anidados

```
#include <stdio.h>

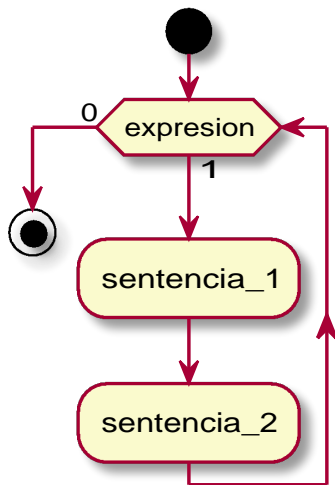
int main()
{
    int i, j;
    for (i = 1; i <= 10; i++)
    {
        printf("Tabla del %d\n", i);

        for (j = 1; j <= 10; j++)
            printf("%d x %d = %d\n",
                i, j, i * j);
    }
    return 0;
}
```

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - **`while`**
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

while: Flujo

Ejecuta una sentencia (simple o compuesta) **cero o más veces** dependiendo del resultado booleano de una expresión.



while: Código

Ejecuta una sentencia (simple o compuesta) **cero o más veces** dependiendo del resultado booleano de una expresión.

```
while (expresion)
{
    sentencia_1;
    sentencia_2;
    ...
}
```

while: Atención

- Si la primera vez que se evalúa la condición es falsa, el bloque de sentencias **no se ejecuta nunca**.
- Si la expresión es siempre verdadera **el bucle es infinito** (dentro de la sentencia debe haber una instrucción que modifique su estado).

Ejemplo

```
#include <stdio.h>

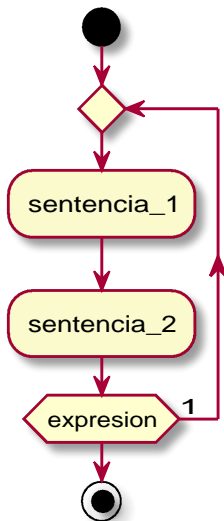
int main()
{
    int i;

    i = 5;
    while (i > 0)
    {
        printf("%d...", i);
        --i;
    }
    printf("Despegue!");
    return 0;
}
```

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

do-while: Flujo

Ejecuta una sentencia (simple o compuesta) **una o más veces** dependiendo del resultado de una expresión.



do-while: Código

Ejecuta una sentencia (simple o compuesta) **una o más veces** dependiendo del resultado de una expresión.

```
do
{
    sentencia_1;
    sentencia_2;
    ...
}
while (expresion);
```

do-while: Atención

- Si la primera vez que se evalúa la expresión es falsa, la sentencia se habrá ejecutado al menos una vez.
- Si la expresión es siempre verdadera **el bucle es infinito** (dentro de la sentencia debe haber una instrucción que modifique su estado)

Ejemplo: número entero al revés

```
#include <stdio.h>

int main()
{
    int num = 123456, cifra;

    do
    {
        cifra = num % 10;
        printf("%d", cifra);
        num = num / 10;
    }
    while (num > 0);
    printf("\n");
    return 0;
}
```

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

Equivalencia

```
#include <stdio.h>
void main() {
    int i,n = 10;
    // Bucle for
    for (i = 0; i <= n; i++)
        printf("%d ",i);
    // Bucle while
    i=0;
    while( i<= n) {
        printf("%d ",i);
        i++;
    }
    // Bucle do-while
    i=0;
    do {
        printf("%d ",i);
        i++;
    }
    while(i <= n);
}
```

¿Qué bucle elegir?

- Si se conoce el número de veces que debe ejecutarse la tarea es recomendable usar `for`.
- Si el número de veces es desconocido a priori:
 - ▶ Si debe realizarse al menos una vez se debe usar `do-while`.
 - ▶ Si no es imprescindible que se ejecute alguna vez, se puede usar `while`.

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...

break y continue

break

- Finaliza la ejecución de un bucle (si el bucle está anidado sólo finaliza él, pero no los bucles más externos).

continue

- Ejecuta la siguiente iteración del bucle.
- En un bucle while o do-while vuelve a expresión.
- En un bucle for ejecuta `expr_avance` y a continuación comprueba `expr_final`

- 1 Introducción
- 2 Sentencias Condicionales
 - `if - else`
 - `switch - case`
- 3 Sentencias Repetitivas (Bucles)
 - `for`
 - `while`
 - `do-while`
 - Equivalencia
- 4 Rupturas
- 5 ...