

Informática I - Electrónica

Código de Materia 950452
R1091 Viernes Turno Noche

Oscar Paniagua
Gaston Coustau

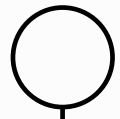


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

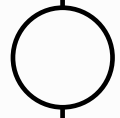


Premio Nacional a la
Calidad 2016-2019

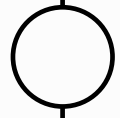
Control de Flujo



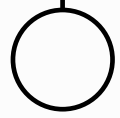
Operadores Relacionales



Operadores Lógicos



Sentencias de Control de Flujo. Estructuras de Selección



Sentencias de Control de Flujo. Estructuras de Repetición

Control de Flujo | Operadores Relacionales

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

```
#include <stdio.h>

void f1(void);
void f2(void);

int main(){

    printf("Soy main()\n");

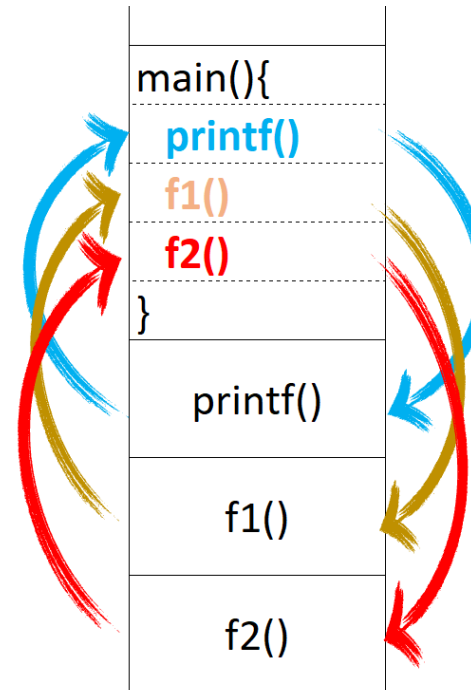
    f1();

    f2();

    return 0;
}

void f1(void){
    printf("Soy f1()\n");
}

void f2(void){
    printf("Soy f2()\n");
}
```



- Por ahora podemos llamar (call) a una función, para que se ejecute(realice lo que se programó), y vuelva el control a la función que la invocó, continuando inmediatamente en la línea siguiente de código.
- Como haríamos si quisiéramos “tomar decisiones”, por ejemplo, que alguna función se ejecute si se cumple cierta “condición”
- Vamos a ver como lo podemos hacer en “Lenguaje C”

Control de Flujo | Operadores Relacionales

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición



- El lenguaje C posee **operadores** para evaluar **expresiones lógicas**, los cuales permiten tomar decisiones durante la ejecución del programa.
- El resultado de las **expresiones lógicas** es un valor entero (int) y pueden ser:
 - 1 (Verdadero)
 - 0 (Falso)

OPERADORES RELACIONALES O DE COMPARACIÓN

Operador	Significado (condición verdadera)
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual a
!=	Distinto de ó no igual que

>_

03-01-operadores-relacionales.c

Control de Flujo | Operadores Lógicos

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

OPERADORES LÓGICOS

Operador	Significado
&&	Operador lógico AND (Y)
	Operador lógico OR (O)
!	Operador lógico NOT (NEGACIÓN)



Estos operadores se combinan con los anteriores ampliando y potenciando las posibilidad de evaluación de expresiones.

>_

03-01-operadores-logicos.c

Control de Flujo | Operadores Lógicos

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición



OPERADOR LÓGICO “AND”

- Para que la evaluación del conjunto sea considerada verdadera, todas las condiciones individuales deben ser verdadero.
- Con que una sola de las condiciones sea falsa, la resultante es falsa.
- El operador AND es análogo a la multiplicación, con que uno de los multiplicando sea cero (falso) el resultado será cero (falso).

Condicion (1)	AND	Condicion (2)	Resultante
1 (verdadero)	&&	1 (verdadero)	1 (verdadero)
1 (verdadero)	&&	0 (falso)	0 (falso)
0 (falso)	&&	1 (verdadero)	0 (falso)
0 (falso)	&&	0 (falso)	0 (falso)

Control de Flujo | Operadores Lógicos

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición



OPERADOR LÓGICO “OR”

- Para que la evaluación del conjunto sea considerada verdadera, basta con que solo una de las condiciones sea verdadera.
- Para que la evaluación resulte falsa, se requiere que todas las condiciones sean falsas.
- El operador OR es análogo a la suma (suponiendo números positivos), con que uno de los sumandos sea distinto de cero (verdadero) el resultado será distinto de cero (verdadero).

Condición (1)	OR	Condición (2)	Resultante
1 (verdadero)		1 (verdadero)	1 (verdadero)
1 (verdadero)		0 (falso)	1 (verdadero)
0 (falso)		1 (verdadero)	1 (verdadero)
0 (falso)		0 (falso)	0 (falso)

Control de Flujo | Operadores Lógicos

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos**
- Sentencias de Control de Flujo.
Estructuras de Selección
- Sentencias de Control de Flujo.
Estructuras de Repetición







OPERADOR LÓGICO “NOT”

- Es un operador unario que invierte el resultado de la condición o evaluación.
- El mismo antecede al operando cambiando su significado

NOT	Condición	Resultante
!	1 (verdadero)	0 (falso)
!	0 (falso)	1 (verdadero)

Control de Flujo | Operadores Lógicos

Control de Flujo

-  Operadores Relacionales
-  Operadores Lógicos
-  Sentencias de Control de Flujo. Estructuras de Selección
-  Sentencias de Control de Flujo. Estructuras de Repetición

EJEMPLO

```
h=3; z=5; a=0;
```

```
printf("%d", (h == 1) || (z <= 1));
```

 0 - Falso

```
printf("%d", (h != 1) && (z == 1));
```

 0 - Falso

```
printf("%d", ( h ) || (z == 1));
```

 1 - Verdadero

```
printf("%d", !a );
```

 1 - Verdadero

```
printf("%d", h && z && a);
```

 0 - Falso

```
printf("%d", h && z && !a);
```

 1 - Verdadero

```
printf("%d", !(h && z && a));
```

 1 - Verdadero

```
b= ((h!=z) && (a)) || (z>0) ;
```

```
printf("%d", b);
```

 1 - Verdadero

```
b= !((h!=z) || (a)) && (z>0) ;
```

```
printf("%d", b);
```

 0 - Falso

Control de Flujo | Sentencias

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

Sentencias

Sentencias Simples

Una **sentencia simple** es una expresión seguida de un punto y coma (;)

Sintaxis:

<Sentencia>;

en donde el ; es obligatorio

- Las sentencias simples se ejecutan una tras otra, en el mismo orden en que han sido escritas.
- Para cambiar ese orden están las **sentencias de control de flujo**, las cuales dependiendo de una condición deciden qué sentencia, o bloque de sentencias, es la siguiente en ejecutarse.

Sentencias Compuestas

Una **sentencia compuesta** o bloque es un grupo de sentencias agrupadas entre llaves { }.

- Son sintácticamente equivalentes a una sentencia simple.

Sentencias de Control de Flujo

Estructuras de Selección

If – else
switch

Permiten ejecutar unas sentencias u otras en función de una condición

Estructuras de Repetición

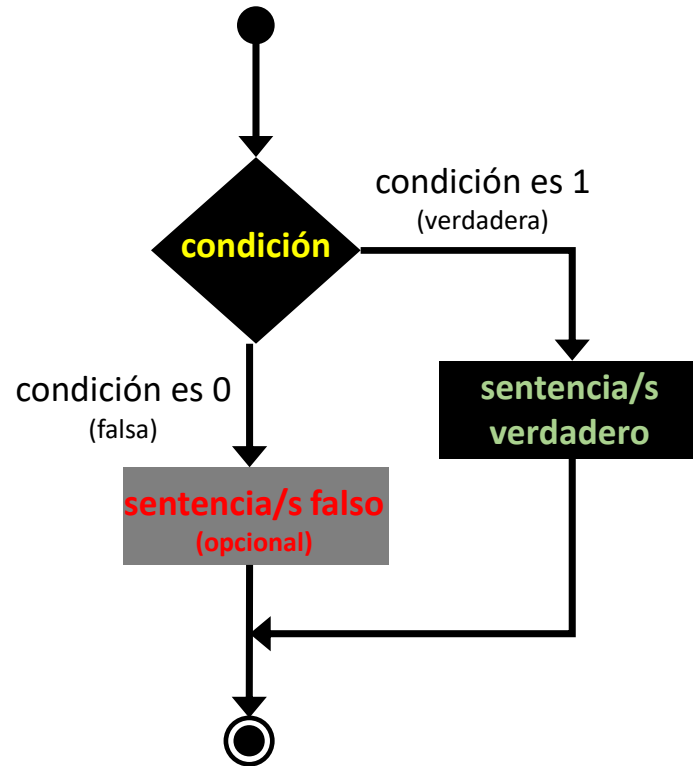
while
do while
for

Permiten iterar el flujo de ejecución de un programa dentro de un bloque de sentencias mientras se verifique una condición, es decir, mientras la condición sea cierta

Control de Flujo | Estructuras de Selección | if..else

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición



```
if (condición)
    sentencia_verdadero;
else
    sentencia_falso;
```

```
if (condición){
    sentencia_verdadero_1;
    sentencia_verdadero_2;
    sentencia_verdadero_3;
}
else {
    sentencia_falso_1;
    sentencia_falso_2;
    sentencia_falso_3;
}
sentencia_x;
```

> _

03-03.1-if-else.c

Control de Flujo | Estructuras de Selección | if..else | else if

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo.
Estructuras de Selección**
- Sentencias de Control de Flujo.
Estructuras de Repetición



- Se utiliza para expresar una toma de decisión múltiple.
- Existe mas de una “condición”
- No es más que una sucesión de **if-else**.
- El último **else**, cumple el rol de “si no se cumple ninguna de las condiciones anteriores, entonces...”
- Al igual que un simple if, éste ultimo **else** no es obligatorio

```
if (condición){  
    sentencia o bloque;  
}  
else if (condición) {  
    sentencia o bloque;  
}  
else if (condición) {  
    sentencia o bloque;  
}  
else {  
    sentencia o bloque;  
}
```

>_

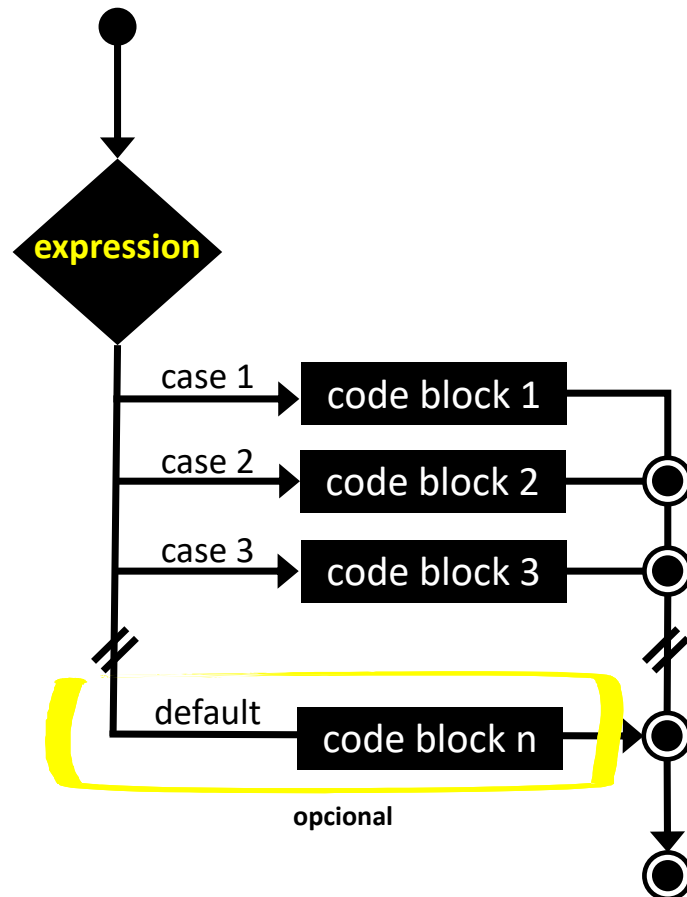
03-03.2-if-else-else if.c
03-03.3-if-else-else if.c

Control de Flujo | Estructuras de Selección | switch

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

“switch”, es una alternativa al uso del **if-else**, aunque no es exactamente lo mismo.



```
switch (variable o expresión)
{
  case valor1:
    sentencia;
    sentencia;
    break;
  case valor2:
    sentencia;
    sentencia;
    break;
  case valor n:
    sentencia;
    sentencia;
    break;
  default:
    sentencia;
    sentencia;
    break;
}
```

The code snippet shows a switch statement with four cases. The "break;" statement in the first case is highlighted with a yellow circle and labeled "opcional". The "default:" block is highlighted with a yellow rectangle and labeled "opcional".

Control de Flujo | Estructuras de Selección | switch

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección**
- Sentencias de Control de Flujo. Estructuras de Repetición

```
switch (variable o expresión)
{
case valor1:
    sentencia;
    sentencia;
    break;
case valor2:
    sentencia;
    sentencia;
    break;
case valor n:
    sentencia;
    sentencia;
    break;
default:
    sentencia;
    sentencia;
    break;
}
```



- **valor** es un valor o expresión constante entera y no puede estar repetido.
- **default** es opcional, se ejecuta si no se cumple ningún **case**.
- Un **switch** puede tener dentro de un **case** diversos códigos, inclusive otro **switch**.
- Los **case** son puntos de entrada.
- Los **break** son puntos de salida.
- El valor obtenido en los paréntesis de **switch**, (**variable o expresión**) se compara con cada **case**. Cuando son coincidentes comienzan a ejecutarse las sentencias hasta que se encuentre un **break** o bien finalice el **switch**.
- De no existir ninguna coincidencia, se ejecutarán las sentencias asociadas al **default**, si es que la misma existe. En caso contrario sale del bloque.
- En caso de **switch** anidados, el **break** te saca del bloque (**switch**) al que pertenece.

Control de Flujo | Estructuras de Selección | switch

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección**
- Sentencias de Control de Flujo. Estructuras de Repetición



- El **switch** es una alternativa al uso de **else if**, pero no siempre
- La sentencia **switch** sólo puede evaluar la igualdad, mientras que el **if** puede evaluar cualquier expresión relacional o lógica
- Una sentencia **switch** sólo trabaja con tipo de datos **int** y puede comprobar la igualdad (**==**).
- Además se pueden dar las siguientes condiciones
 - Pueden existir **case** vacíos
 - Un conjunto de sentencias dentro de un **case** no necesitan **}**.
- Sin la sentencia **break** al final de un **case**, el conjunto de sentencias contenidas en el siguiente **case** también se ejecutarán hasta encontrar un **break** o finalizar el **switch**.
- Un uso frecuente de la sentencia **switch** es procesar selecciones de menú

>_
03-04-switch.c

Control de Flujo | Estructuras de Repetición | while

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

```
while (loop_repetition_condition )  
    statement;
```

```
while (loop_repetition_condition) {  
    statement;  
    statement;  
    ...;  
}  
statement a;  
statement b;
```



- Lo primero que se hace al llegar a “while”, es evaluar la condición (**loop repetition condition**)
- Si ésta es verdadera, se ejecutan las sentencias asociadas (1 ... N) , luego se vuelve a evaluar la **condición**.
- Si ésta nueva evaluación sigue siendo verdadero, se continúa con la ejecución de las sentencias asociadas y así sucesivamente.
- El ciclo **while** finaliza, cuando la condición es falsa, en cuyo caso se continua con la ejecución de las sentencias que están a continuación (a,b).
- Si la primer evaluación de la condición da falso, las sentencias del bucle **while** no se ejecutarán.

Control de Flujo | Estructuras de Repetición | do while

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

```
do  
    statement;  
while (loop repetition condition);
```

condition);

```
do {  
    statement;  
    statement;  
    ...;  
}  
while (loop repetition condition);
```



- La comparación se encuentra al pie del lazo o ciclo y se cierra con un punto y coma (;)
- Las sentencias asociadas se ejecutan al menos una vez (independientemente de la condición)

Control de Flujo | Estructuras de Repetición | for

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

```
for (initialization expression; loop repetition condition; update expression)  
    statement;
```

```
for (initialization expression; loop repetition condition; update expression) {  
    statement;  
    ...;  
    statement;  
}
```



- **initialization** : se ejecuta cuando ingresa al ciclo, antes de evaluar la condición.
- **loop repetition condition** : igual al **while**, define si el ciclo continua o no.
- **update expression** : se ejecuta luego de haber terminado las sentencias asociadas y antes de evaluar nuevamente la condición.

- Cualquiera de las tres partes del ciclo "for" se pueden omitir, sin embargo los (;) deben permanecer
- Al igual que el **while**, si la primer evaluación da falso las sentencias asociadas no se habrán ejecutado, en este caso tampoco la operación.

Control de Flujo | Estructuras de Repetición | for

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo.
Estructuras de Selección
- Sentencias de Control de Flujo.
Estructuras de Repetición

i++?

```
#include <stdio.h>

int main(){

    int i;

    printf("\nEjemplo de for:\n");
    for ( i = 0; i < 10; i++){
        printf("i = %d\n", i);
    }

    printf("\nEjemplo de simular for con while:\n");
    i = 0;
    while ( i < 10 ){
        printf("i = %d\n", i);
        i++;
    }

    return 0;
}
```

Control de Flujo | Estructuras de Repetición

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición



Si bien ante la necesidad de repetir un conjunto de sentencias, se puede utilizar cualquier instrucción de bucle, se **recomienda** para su selección el siguiente análisis:

¿Conocemos cuántas veces hay que repetir un conjunto de instrucciones?	SI			for
	NO	¿Esas instrucciones deben ejecutarse al menos una vez?	SI	do while
			NO	while

Control de Flujo | break

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

break corta la iteración y sigue la ejecución con las sentencias a continuación del ciclo.

```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}  
  
do {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
} while (testExpression);  
  
for (init; testExpression; update) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```

Control de Flujo | continue

Control de Flujo

- Operadores Relacionales
- Operadores Lógicos
- Sentencias de Control de Flujo. Estructuras de Selección
- Sentencias de Control de Flujo. Estructuras de Repetición

continue obliga al ciclo terminar la iteración actual y continuar con la siguiente (iteración).

```
while (testExpression) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```

```
do {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
} while (testExpression);
```

```
for (init; testExpression; update) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```



C Language

- Operadores Relaciones
 - $>$ $<$ $==$ y sus combinaciones
- Operadores Lógicos
 - AND OR NOT



C Language

- Sentencias de Selección
 - If
 - If – else
 - If anidados
 - switch



C Language

- Sentencias de Repetición
 - while
 - do while
 - for

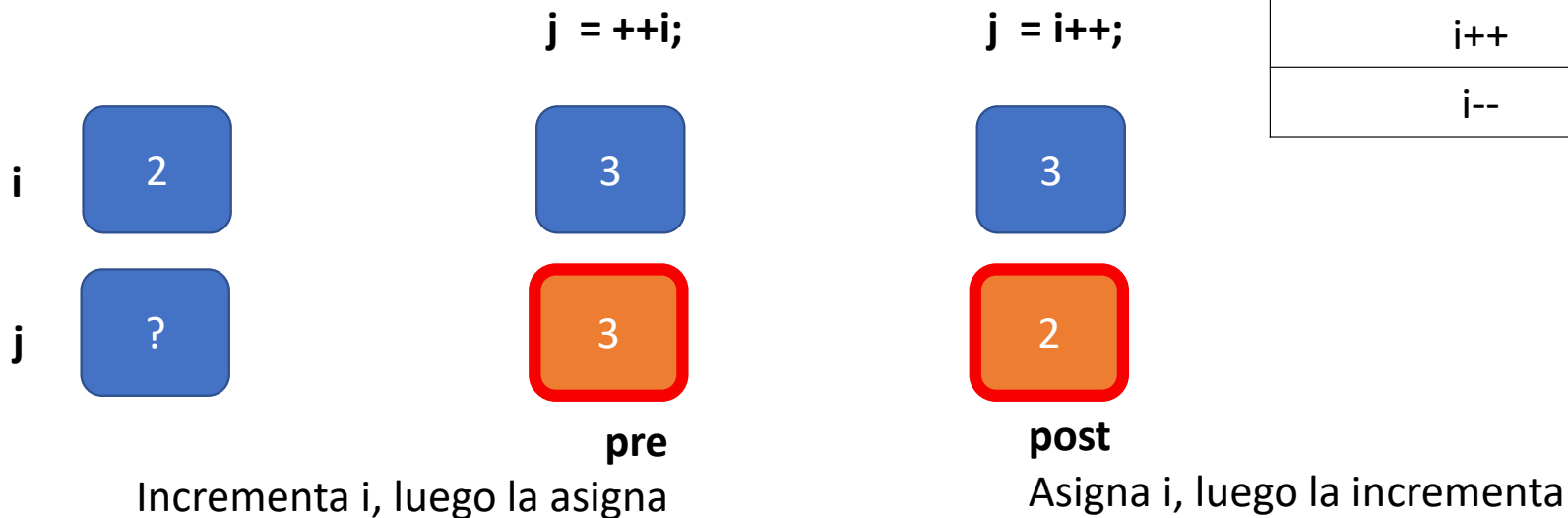
EXTRAS

Extras | Operadores de incremento y decremento



- Son utilizados generalmente en estructuras de bucles (loop)
- Los operadores de incremento (++) o decremento (--) solo tienen un operando
- Podemos utilizarlos como pre(prefix) o post(postfix)

BACK



Operador	Sintaxis
++i	pre-incremento
--i	pre-decremento
i++	post-incremento
i--	post-decremento

Extras | Precedencia de Operadores



- La precedencia de operadores determina el “orden” de “evaluación”.
- Los **operadores unarios** tienen un solo operando:
 - !, +(signo más), -(signo menos), y &(dirección)
 - Siempre son evaluados primero, excepto a una llamada a una función.

Operador	Precedencia
Llamada a función	La mas ALTA
! + ++ - -- &	
* / %	
+ -	
< <= >= >	
== !=	
& &	
=	La mas BAJA



Los operadores relacionales tienen menos **precedencia** que los operadores numéricos.

Por lo tanto, si queremos expresar que **j** es numéricamente menor que la expresión **k-1**, debemos expresarla como:

$$j < (k-1)$$