Programación I

Pizarras y Ejemplos de clase, 1 al 10 de octubre 2025

Este documento contiene las pizarras y ejemplos de código que se utilizaron en las clases de Programación I del grupo G1M3, durante la primera mitad el mes de octubre de 2025.

1 . Clase 1 octubre 2025

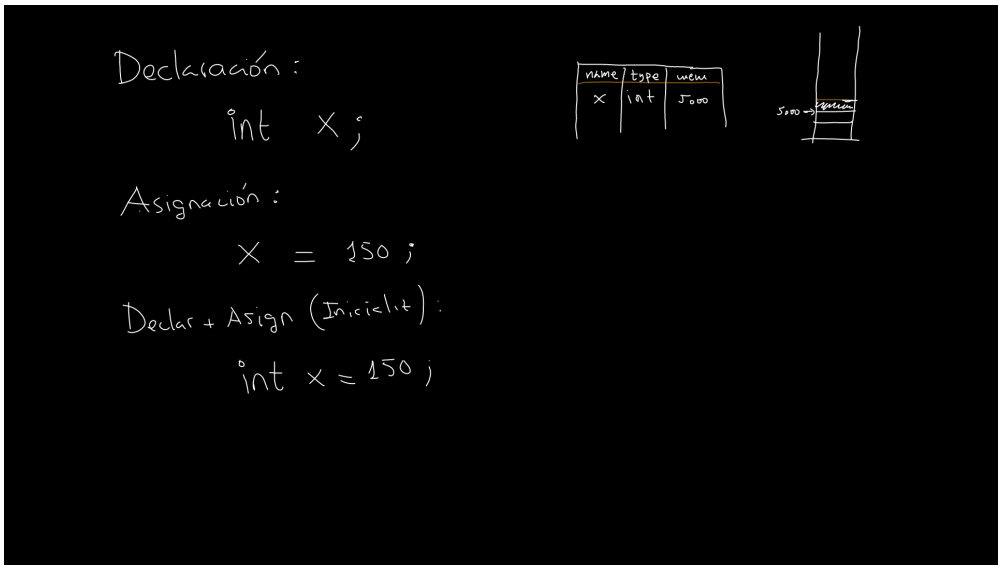


Figura 1: Declaración y asignación de variables.

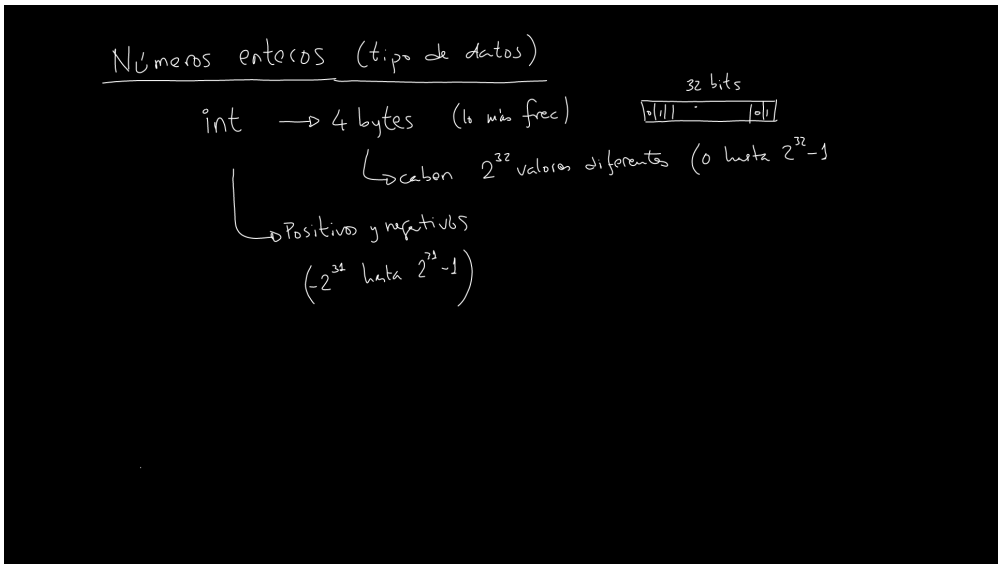


Figura 2: Tipo de datos *int*.

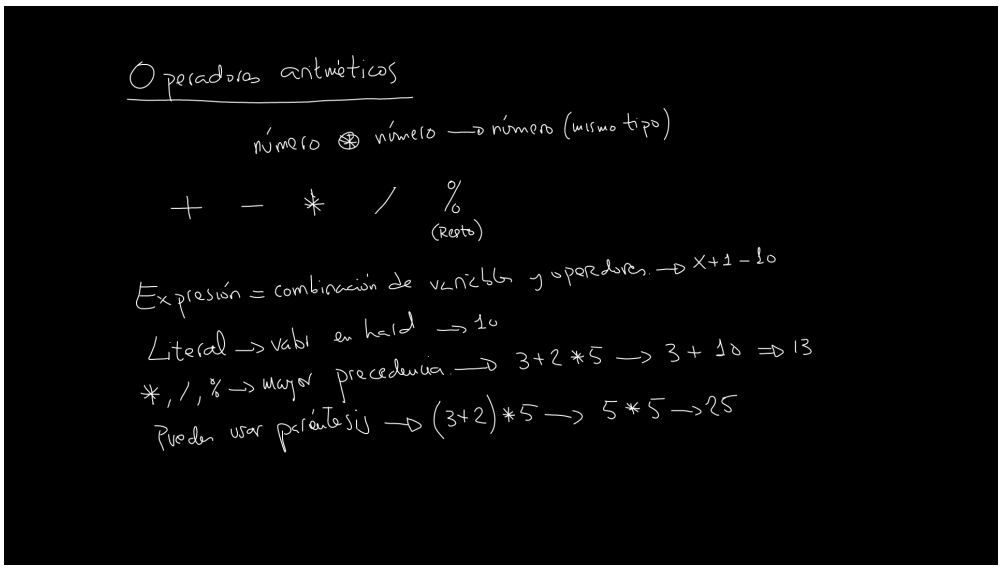


Figura 3: *int*: operadores aritméticos.

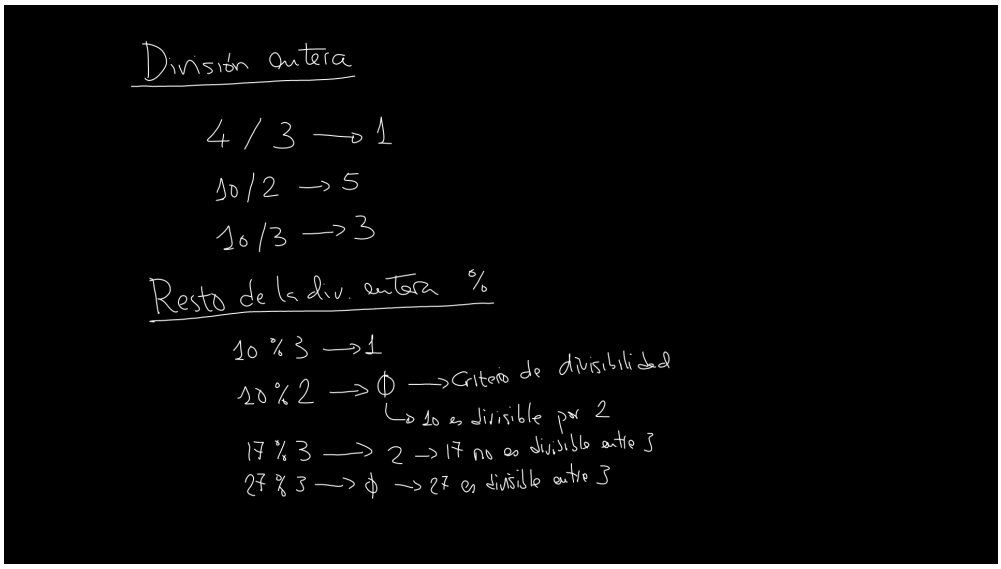


Figura 4: int: división entera y resto de la división entera.

Ejemplo 1 Ejemplo usando la división entera y el resto de la división

```c
#include <stdio.h>

int main() {
   int x = 10;
   int y = 3;
   int z_1 = 10/3;
   int z_2 = 10%3;

   printf("%d entre %d es %d\n", x, y, z_1);
   printf("El resto de dividir %d entre %d es %d\n", x, y, z_2);

   return 0;
}
```

La salida por pantalla:

```
10 entre 3 es 3
El resto de dividir 10 entre 3 es 1
```

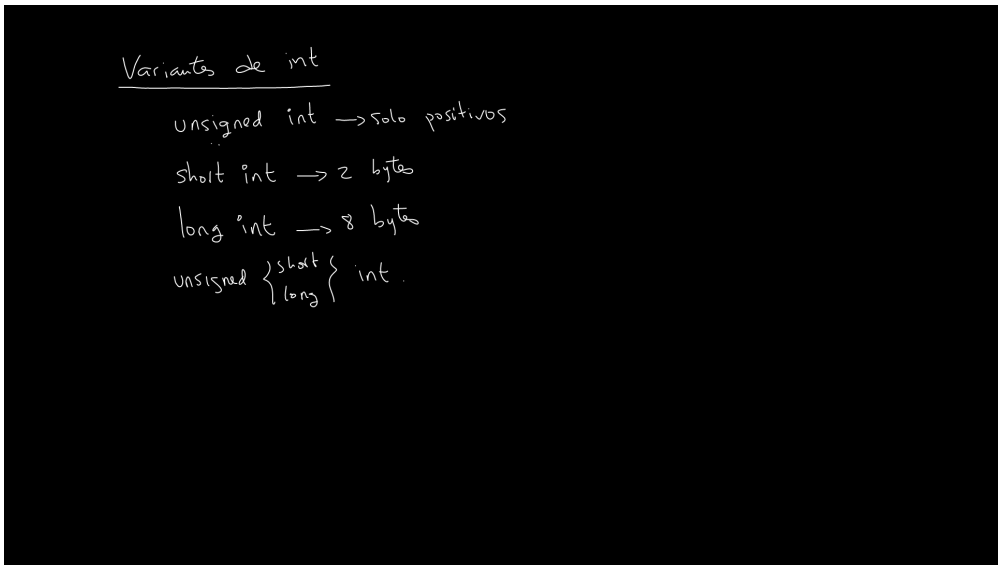


Figura 5: Variantes del tipo *int*.

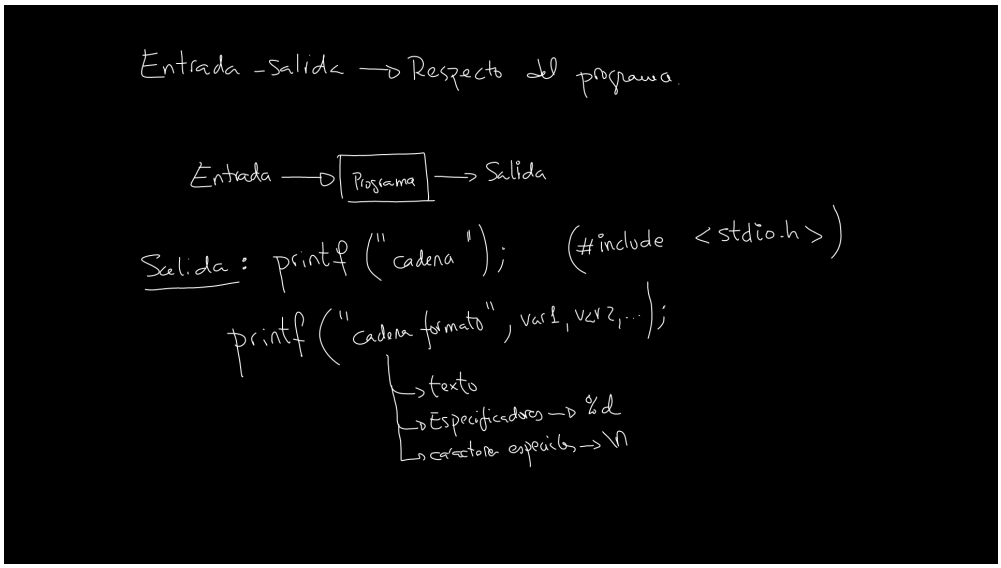


Figura 6: Concepto de entrada salida.

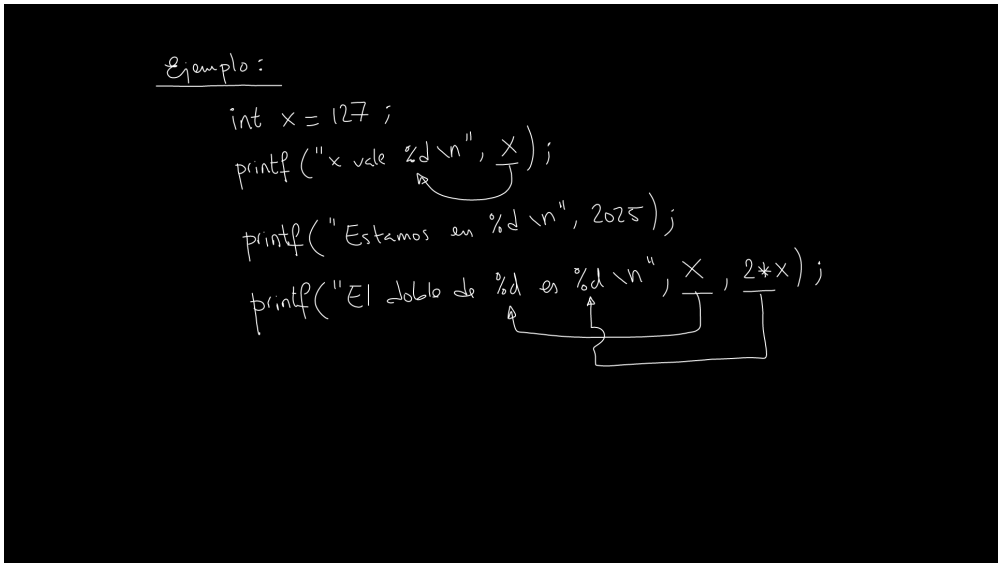


Figura 7: *int*: formato para salidas por pantalla.

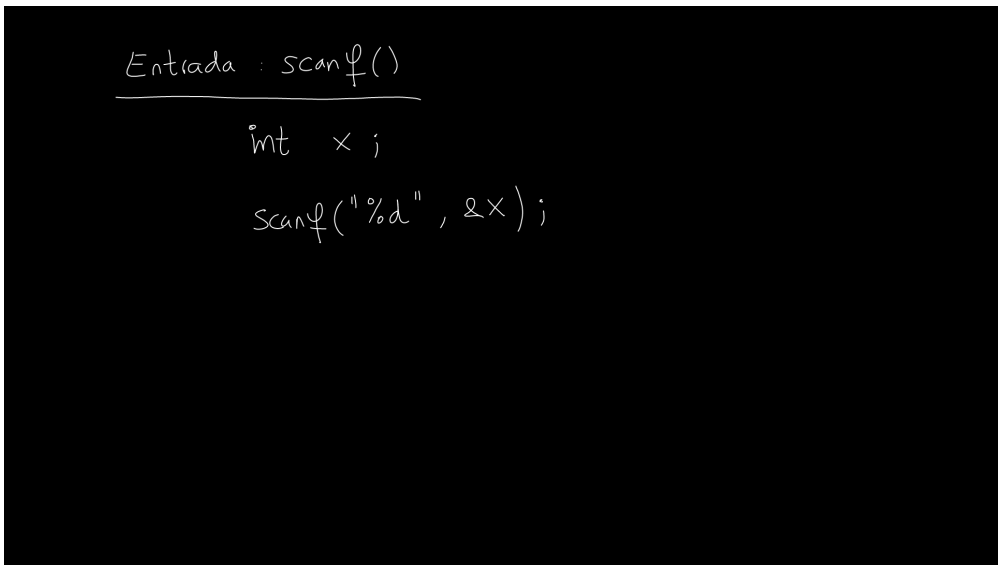


Figura 8: *int*: formato para entradas desde el terminal.

Ejemplo 2 Ejemplo de entrada y salida desde terminal.

```
#include <stdio.h>

int main() {
   int n;
   printf(”Teclea un número: ”);
   scanf(”%d”, &n);

   printf(”Has tecleado: %d\n”, n);

   return 0;
}
```

La salida por pantalla:

```
Teclea un número: 127
Has tecleado: 127
```

Ejemplo 3 Ejemplo de entrada desde terminal y división entera.

```
#include <stdio.h>

int main() {
   int x;
   printf(”n: ”);
   scanf(”%d”, &x);

   printf(”Resto división entre 2: %d\n”, x%2);
   printf(”Resto división entre 3: %d\n”, x%3);

   return 0;
}
```

La salida por pantalla:

```
n: 39
Resto división entre 2: 1
Resto división entre 3: 0
```

Ejemplo 4 Para escribir el símbolo % en pantalla hay que poner % %.

```c
#include <stdio.h>

int main() {
   int n;
   printf("n= ");
   scanf("%d", &n);

   printf("n= %d\n", n);
   printf("%d %% 2 = %d\n", n,n%2);

   return 0;
}
```

La salida por pantalla:

```
n= 17
n= 17
17 % 2 = 1
```

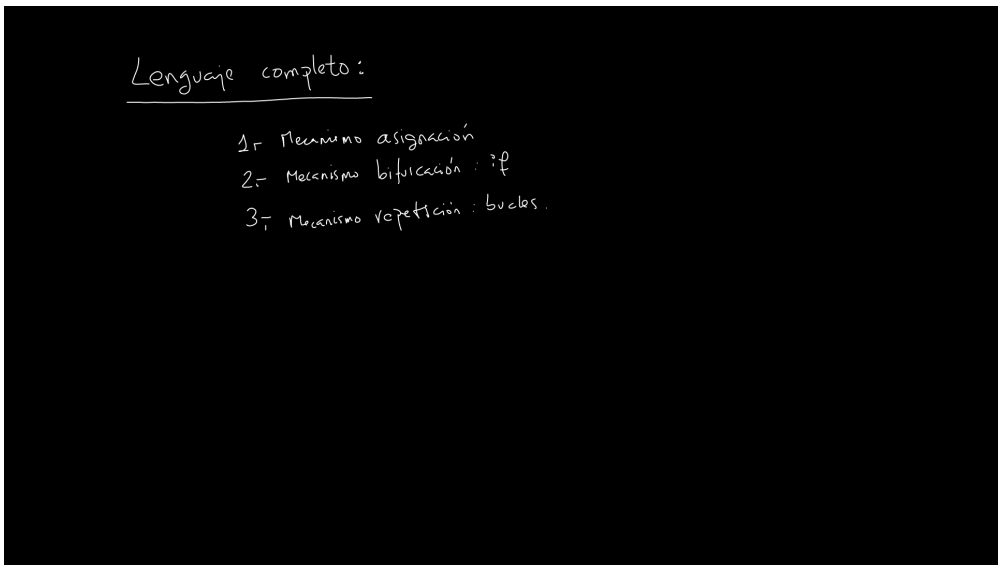
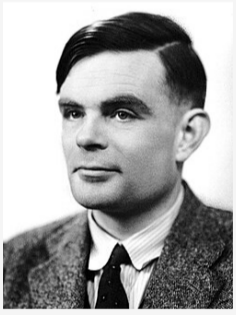


Figura 9: Condiciones para que un lenguaje sea *Turing completo*.

Alan Turing



Alan Turing fue un matemático británico de mediados del siglo XX que jugó un papel trascendental en el desarrollo de la ciencia de la computación.

Puedes echar un vistazo al artículo que hay sobre él en la Wikipedia:

https://es.wikipedia.org/wiki/Alan_Turing

También te puede gustar la película *The imitation game*, que en España se ha titulado *Descifrando el enigma*. La película narra la carrera contrarreloj de Alan Turing (Benedict Cumberbatch) y su equipo de descifrado de códigos, entre los que se encontraba la también matemática Ann Mitchell, en su intento de romper el cifrado de la máquina Enigma de la Alemania Nazi en el ultrasecreto Cuartel General de Comunicaciones del Gobierno del Reino Unido, situado en la mansión de Bletchley Park durante los días más oscuros de la Segunda Guerra Mundial.

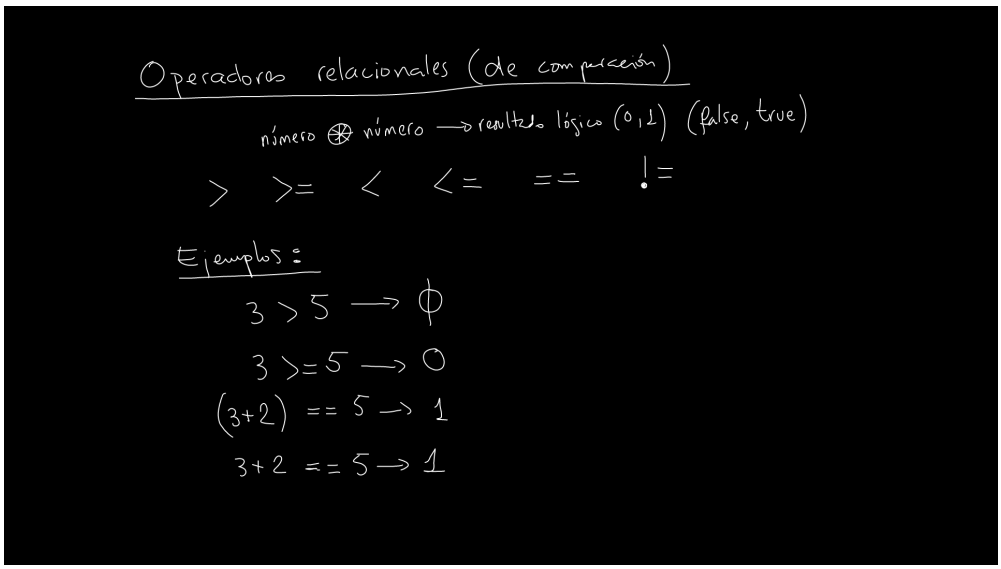


Figura 10: Operadores *relacionales* o *de comparación*.

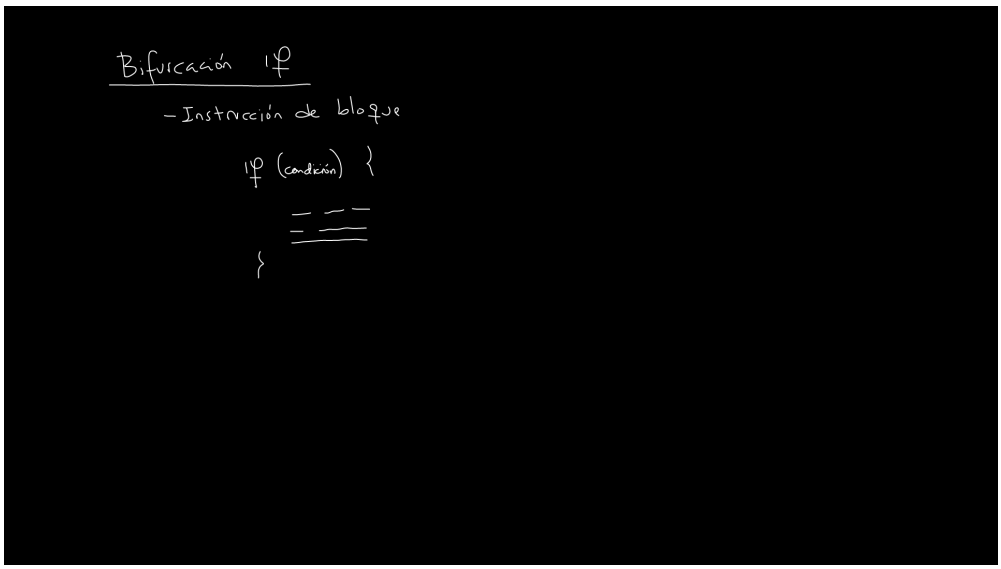


Figura 11: Bifurcaciones: esquema de la instrucción *if*.

2 . Clase 3 octubre 2025

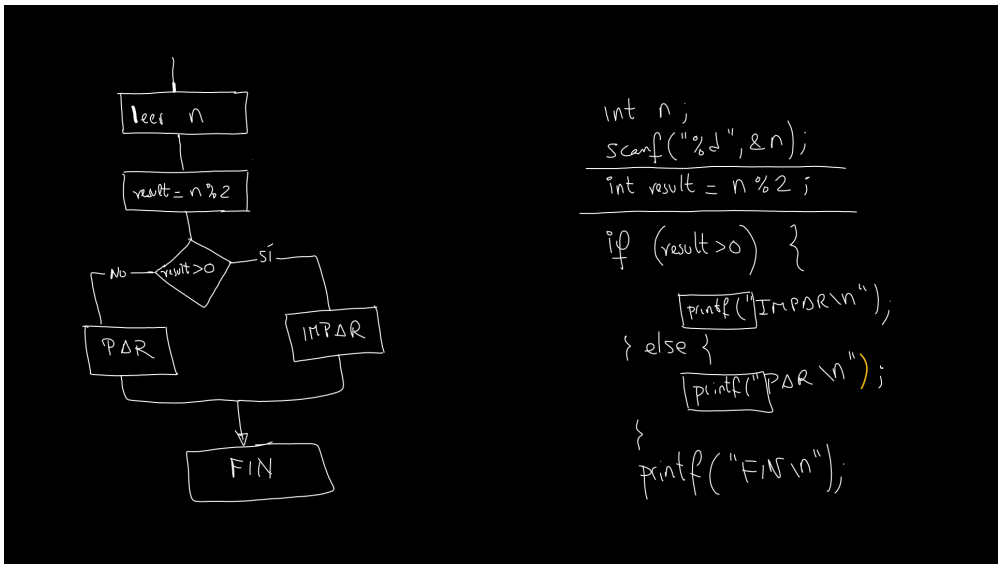


Figura 12: Bifurcación *if ... else*.

Ejemplo 5 Ejemplo *if ... else*: comprobar si número par.

```
#include <stdio.h>

int main() {

   // Entrada
   int n;
   printf("n= ");
   scanf("%d", &n);

   // Procesamiento
   int result = n%2;

   // Salida
   if(result>0) {
      printf("Impar\n");
   } else {
      printf("Par\n");
   }
   printf("FIN\n");

   return 0;
}
```

La salida por pantalla, para dos ejecuciones sucesivas del programa:

```
n= 19
Impar
FIN

n= 22
Par
FIN
```

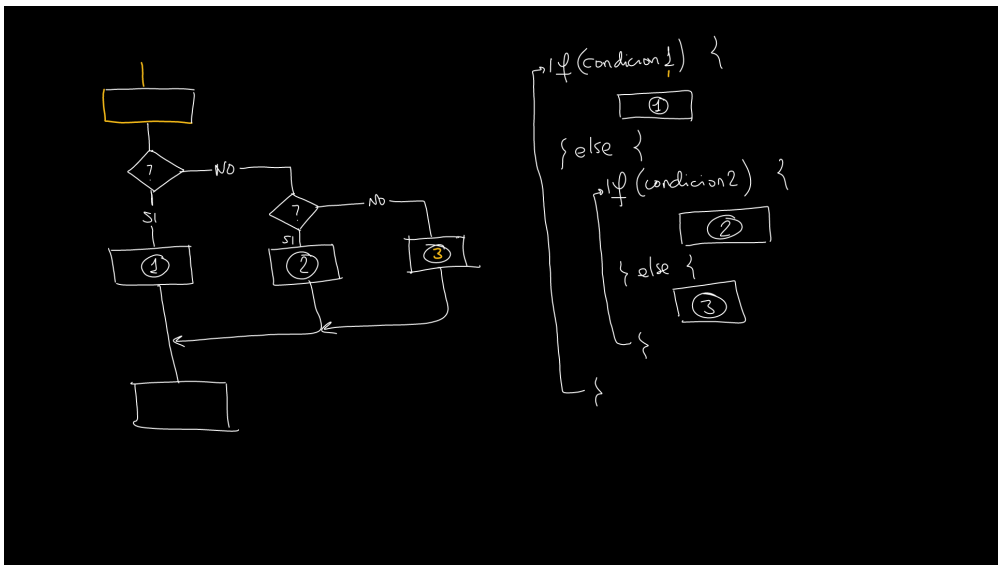


Figura 13: Bifurcaciones *if* anidadas.

Ejemplo 6 Comprobar par, impar, cero. Solución con *if* anidado.

```c
#include <stdio.h>

int main() {

   // Entrada
   int n;
   printf("n= ");
   scanf("%d", &n);

   // Procesamiento
   int result = n % 2;

   // Salida
   if (n == 0) {
      printf("Es cero\n");
   } else {
      if (result != 0) {
         printf("Impar\n");
      } else {
         printf("Par\n");
      }
   }
   printf("FIN\n");

   return 0;
}
```

La salida por pantalla, para tres ejecuciones sucesivas del programa:

```
n= 17
Impar
FIN

n= 0
Es cero
FIN

n= 34
Par
FIN
```

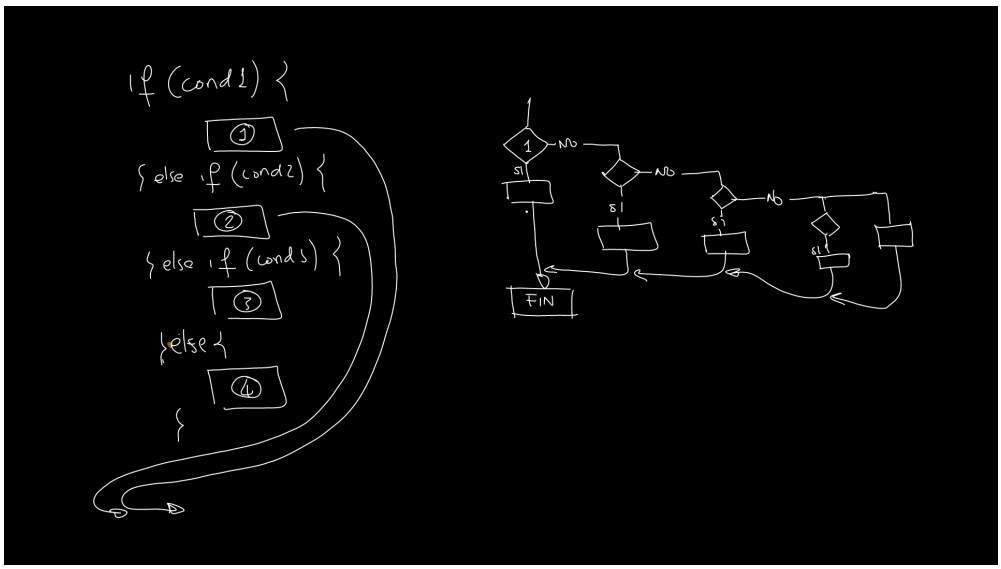


Figura 14: Bifurcación *if ... else if ... else*.

Ejemplo 7 Comprobar par, impar, cero. Solución con *if ... else if ... else*.

```c
#include <stdio.h>

int main() {

   // Entrada
   int n;
   printf("n= ");
   scanf("%d", &n);

   // Procesamiento
   int result = n % 2;

   // Salida
   if (n == 0) {
      printf("Es cero\n");
   } else if (result != 0) {
      printf("Impar\n");
   } else {
      printf("Par\n");
   }
   printf("FIN\n");

   return 0;
}
```

La salida por pantalla, para tres ejecuciones sucesivas del programa:

```
n= 17
Impar
FIN

n= 0
Es cero
FIN

n= 34
Par
FIN
```

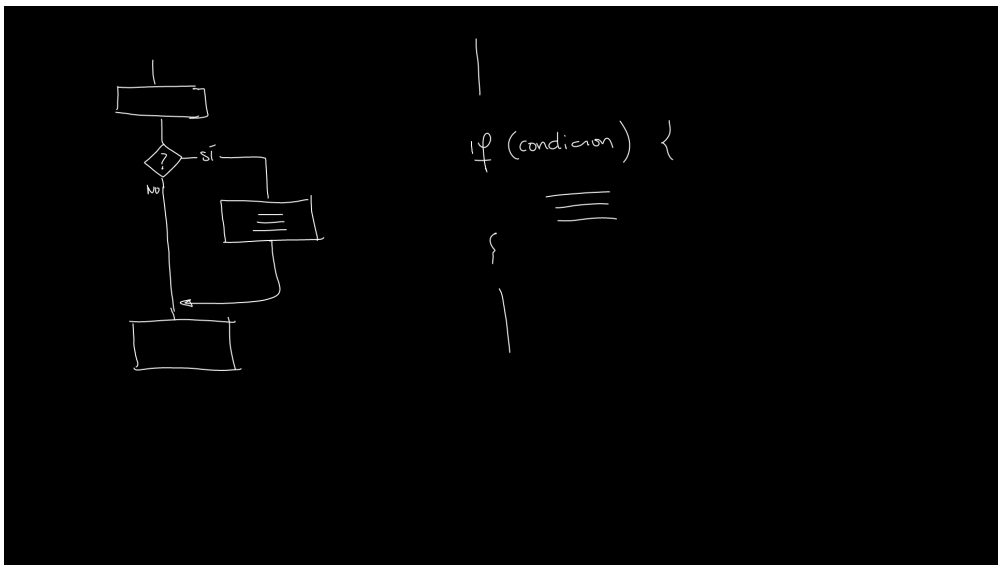


Figura 15: Bifurcación *if* sin rama *else*.

Ejemplo 8 Ejemplo de *if* sin rama *else*.

```c
#include <stdio.h>

int main() {

   // Entrada
   int n;
   printf("Introduce un número: ");
   scanf("%d", &n);

   // Procesamiento + Salida
   if (n > 0) {
      printf("El número es positivo\n");
   }
   printf("Fin del programa\n");

   return 0;
}
```

La salida por pantalla, para dos ejecuciones sucesivas del programa:

```
Introduce un número: 29
El número es positivo
Fin del programa

Introduce un número: -2
Fin del programa
```

3 . Clase 8 octubre 2025

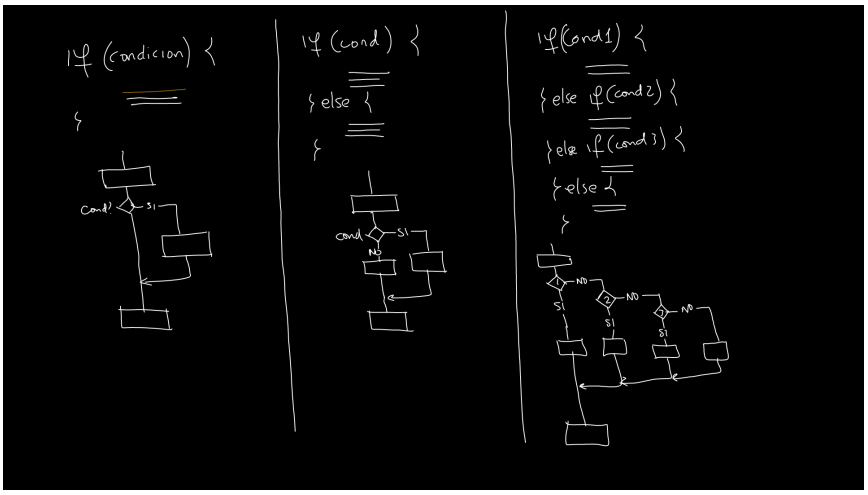


Figura 16: Resumen de bifurcacones *if*.

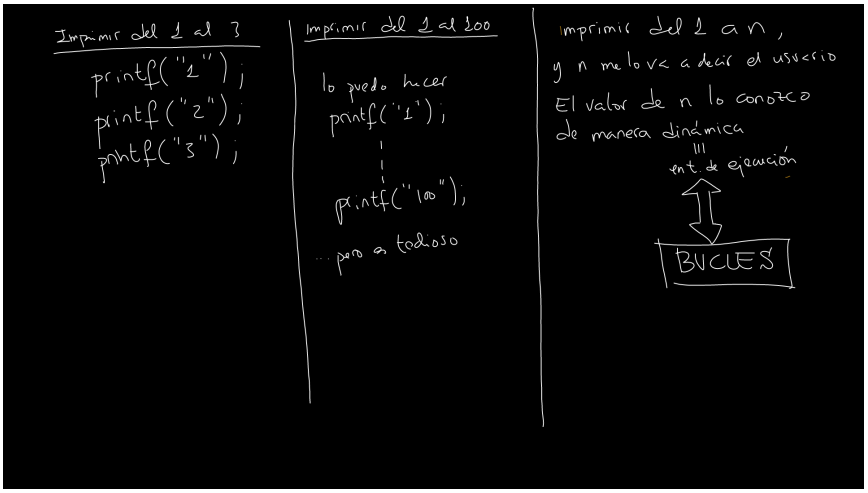


Figura 17: Necesidad de disponer de bucles.

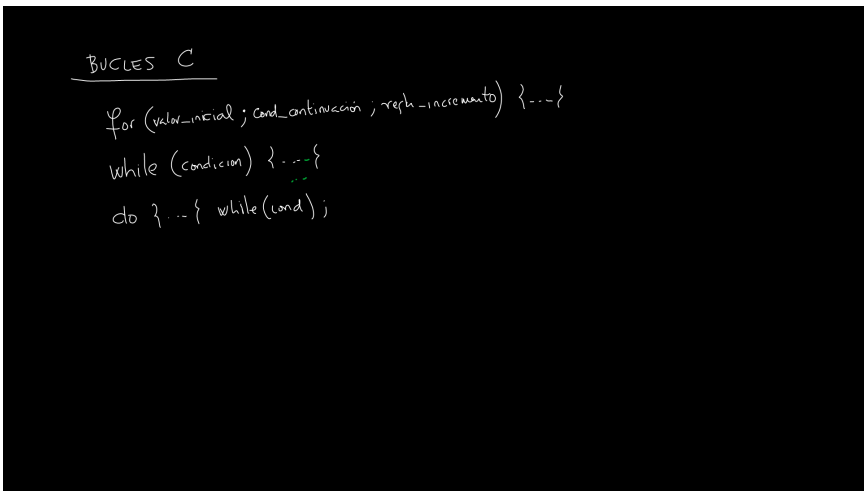


Figura 18: Los tres tipos de bucles del lenguaje C.

Bucles *while*.

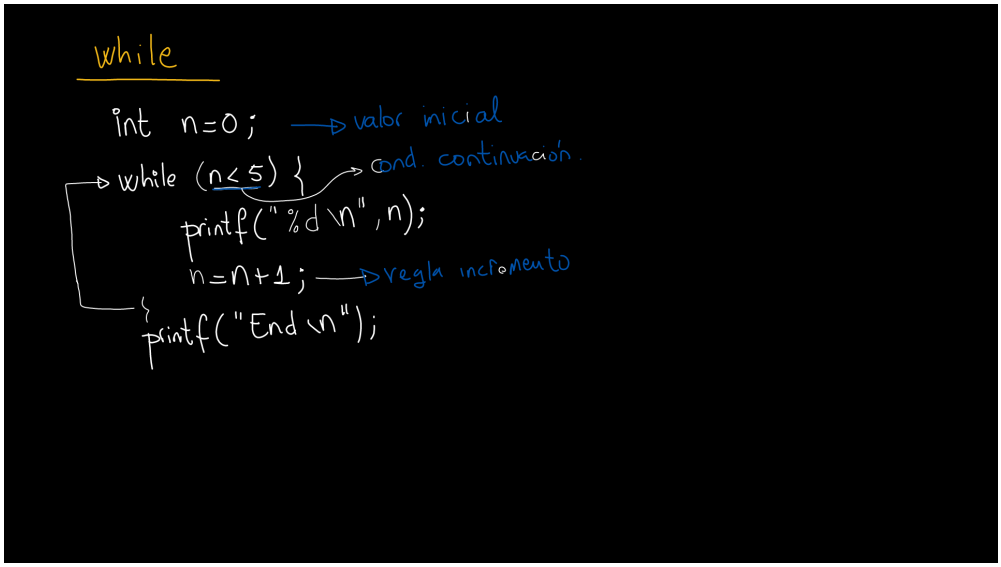


Figura 19: El bucle *while*.

Ejemplo 9 Ejemplo de bucle *while*: imprimir de 0 a 4.

```c
#include <stdio.h>

int main() {

   // Entrada
   int n = 0;

   // Procesamiento + salida
   while(n<5) {
      printf("%d\n", n);
      n = n + 1;
   }
   printf("FIN\n");
}
```

La salida por pantalla:

```
0
1
2
3
4
FIN
```

Necesidad de disponer de bucles: asignación dinámica

El programa del Ejemplo 10 imprime los valores entre 1 y el número tecleado por el usuario. El último valor a imprimir no lo conocemos en el momento de codificar el programa, solo lo conoceremos cuando se esté ejecutando el programa.

Cuando un valor no se conoce en *tiempo de compilación*, sino que se va a conocer en *tiempo de ejecución*, se dice que la asignación del valor a la variable se hace *de manera dinámica*.

Si no dispusiéramos de una instrucción para hacer bucles, no podríamos resolver este problema.

Ejemplo 10 Asignación dinámica: necesidad de disponer de bucles.

```
#include <stdio.h>

int main() {

   int ultimo;
   printf("ultimo= ");
   scanf("%d", &ultimo);
   printf("---------------\n");

   int contador = 0;
   while(contador<ultimo) {
      printf("%d\n", contador);
      contador = contador + 1;
   }
   printf("FIN\n");

   return 0;
}
```

La salida por pantalla:

```
ultimo= 4
---------------
0
1
2
3
FIN
```

Bucles *for*

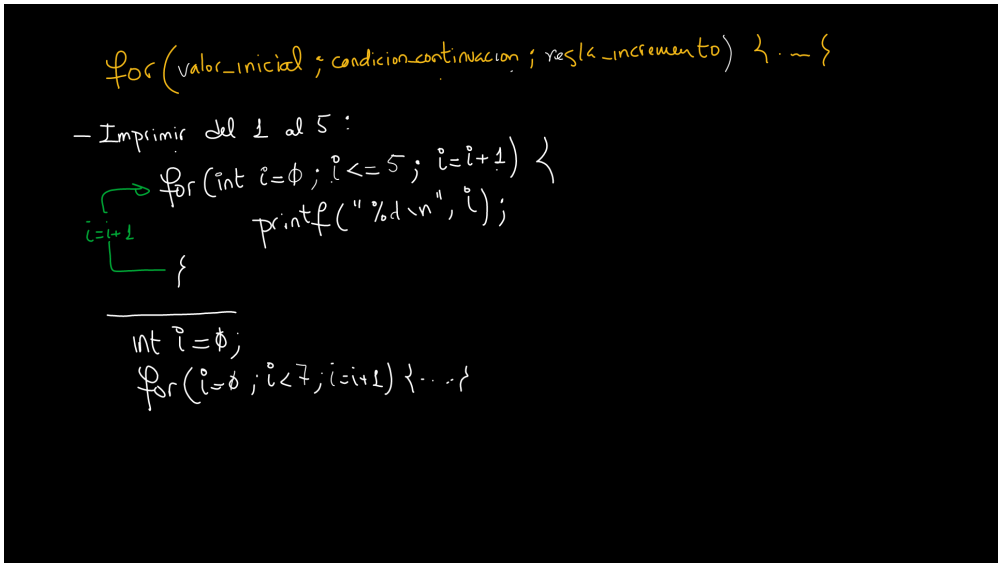


Figura 20: El bucle *for*.

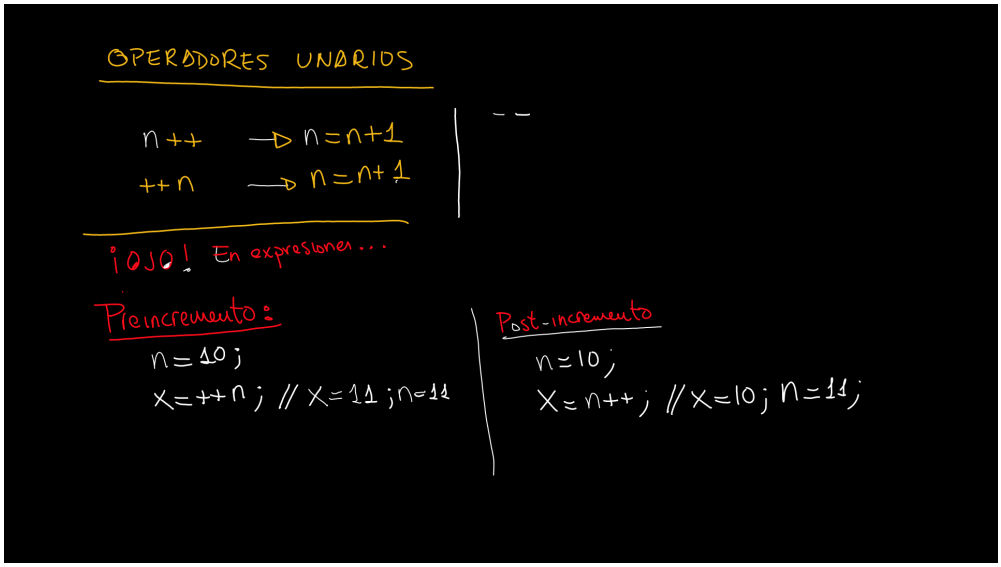


Figura 21: Operadores unarios.

Ejemplo 11 Ejemplo de bucle *for*: imprimir del 5 al 0 (orden descendente).

```
#include <stdio.h>

int main() {

   for(int i=5; i>=0; i--) {
      printf("%d\n", i);
   }
   printf("FIN\n");

}
```

La salida por pantalla:

```
5
4
3
2
1
0
FIN
```

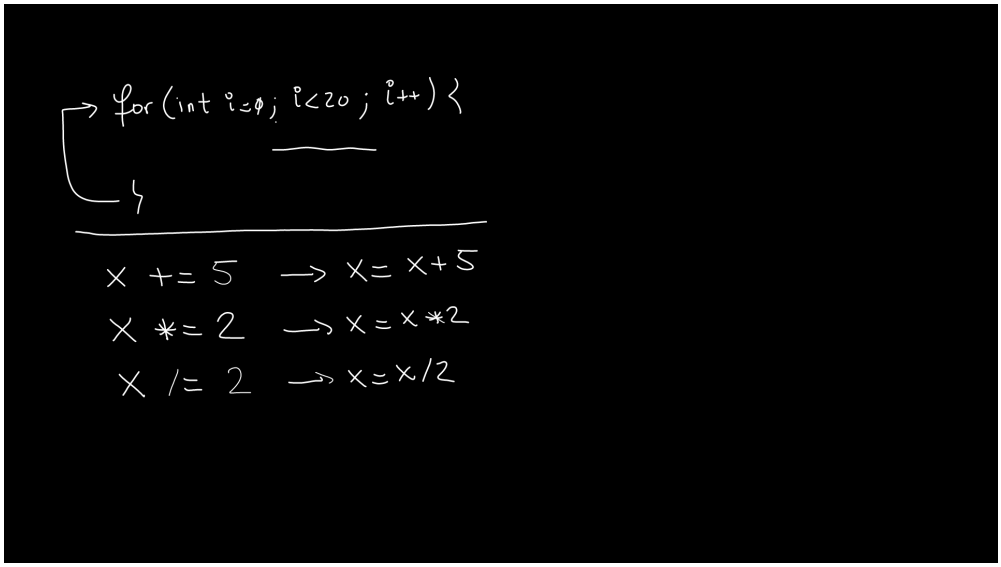


Figura 22: Operadores compuestos de asignación.

4 . Clase 10 octubre 2025

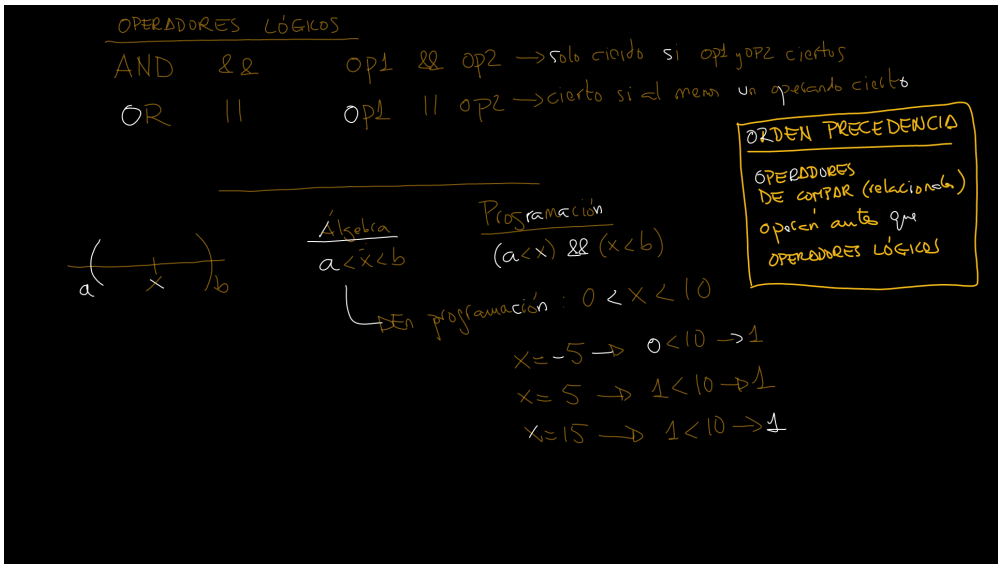


Figura 23: Operadores lógicos.

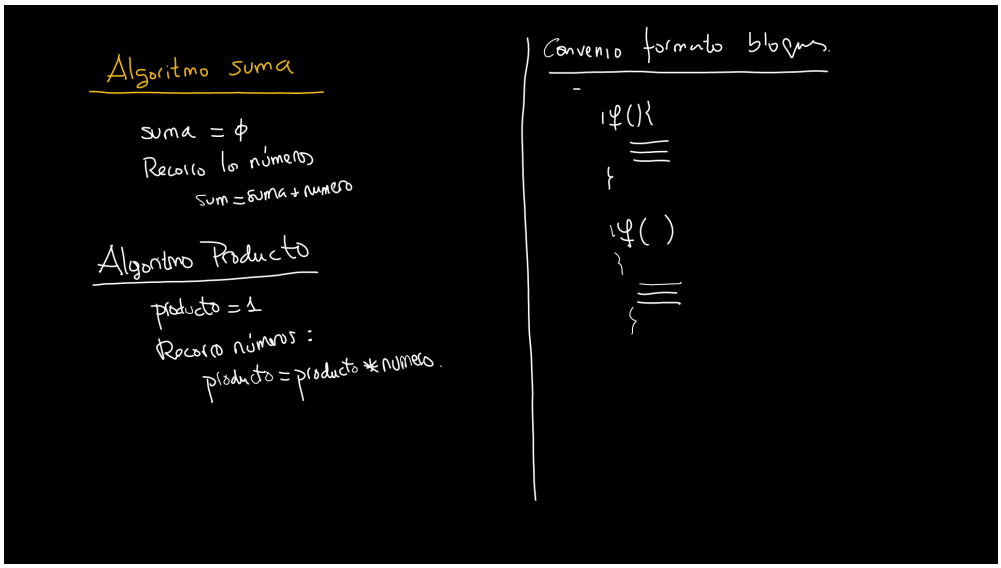


Figura 24: Algoritmos suma y producto. Convenio de formato en bloques.

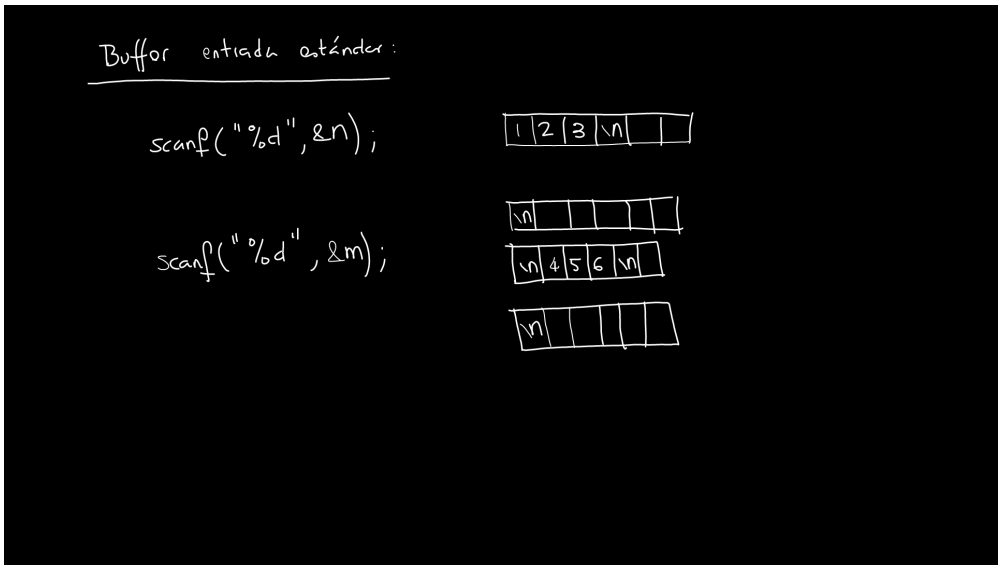


Figura 25: Cambios de línea (\n) residuales en lecturas del teclado.

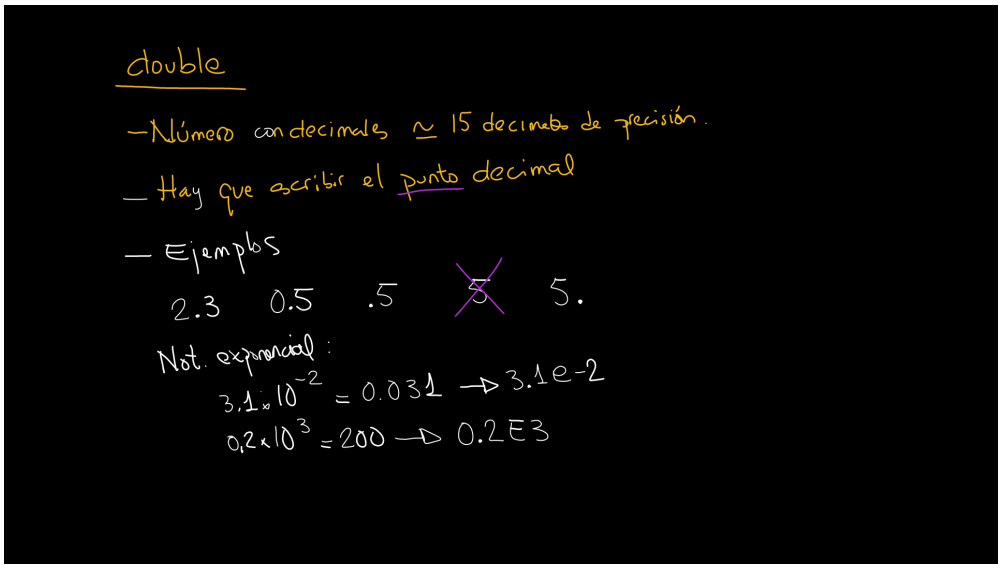


Figura 26: Tipo de datos *double*: números en *coma flotante*.

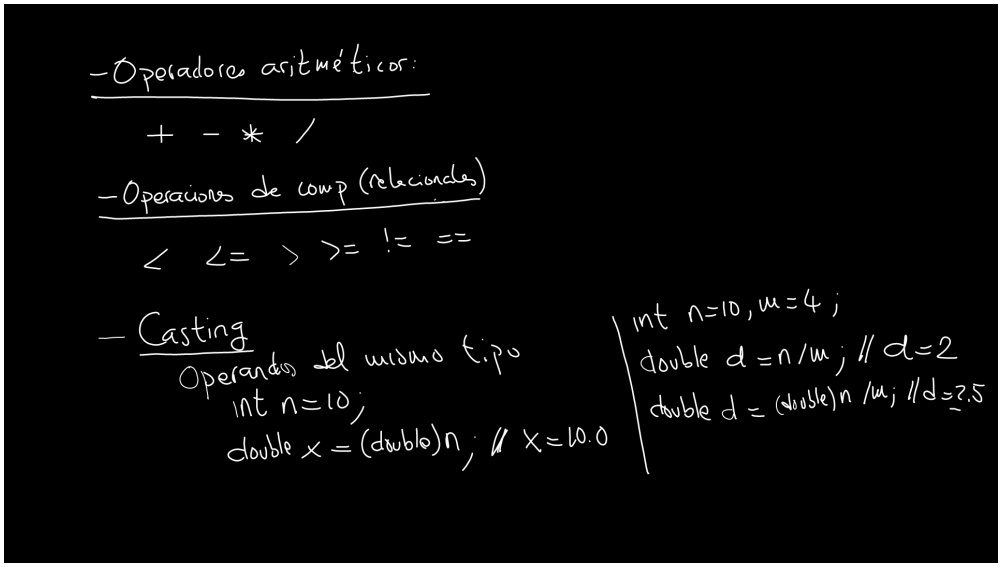


Figura 27: Operadores aritméticos y relacionales asociados al tipo *double*.