Volviendo a los operadores

Operadores de Asignación

```
a = 0;
i = i + 1;
d = d / 2;
m = m * 3;
r = r % 10;
```

```
a = 0;
i += 1;
d /= 2;
m *= 3;
r %= 10;
```

Operadores de incremento

Operadores de incremento

Pre incremento

```
++i; // equivalente a i = i + 1;

--i; // equivalente a i = i - 1;
```

Operadores de incremento

Pre incremento

```
++i; // equivalente a i = i + 1;

--i; // equivalente a i = i - 1;
```

Post incremento

```
i++; // equivalente a i = i + 1;

i--; // equivalente a i = i - 1;
```

Operadores de incremento

Operadores de incremento

La diferencia entre el pre y el post incremento (o decremento) se puede ver cuando están dentro de una expresión

Operadores de incremento

La diferencia entre el pre y el post incremento (o decremento) se puede ver cuando están dentro de una expresión

Pre incremento

```
i = 0;
printf("%d", ++i);
```

```
1
```

Operadores de incremento

La diferencia entre el pre y el post incremento (o decremento) se puede ver cuando están dentro de una expresión

Pre incremento

```
i = 0;
printf("%d", ++i);
```

```
1
```

Pos incremento

```
i = 0;
printf("%d", i++);
```

```
0
```

Operadores de incremento

La diferencia entre el pre y el post incremento (o decremento) se puede ver cuando están dentro de una expresión

Pre incremento

```
i = 0;
printf("%d", ++i);
```

```
1
```

Pos incremento

```
i = 0;
printf("%d", i++);
```

```
0
```

En el caso del pos-incremento, se *usa* el valor que traia `i` para el especificador de formato del `printf` y luego se incrementa.

Operadores lógicos

Operadores lógicos

En C hay tres operadores lógicos a nivel de variables:

Operadores lógicos

En C hay tres operadores lógicos a nivel de variables:

- AND (&&)

Operadores lógicos

En C hay tres operadores lógicos a nivel de variables:

- AND (&&)
- OR (||)

Operadores lógicos

En C hay tres operadores lógicos a nivel de variables:

- AND (&&)
- OR (||)
- NOT (!)

Operadores lógicos

En C hay tres operadores lógicos a nivel de variables:

- AND (&&)
- OR (||)
- NOT (!)

Se pueden usar en cualquier expresión, aunque generalmente se usan en condicionales y sentencias repetitivas

Operadores lógicos

Operadores lógicos

Ejemplo

```
if ( expresión 1 && expresión 2 )
  sentencia
```

Operadores lógicos

Ejemplo

```
if ( expresión 1 && expresión 2 )
  sentencia
```

La sentencia se ejecutará dependiendo del valor de las expresiones evaluadas por el operador &&

Operadores lógicos

Ejemplo

```
if ( expresión 1 && expresión 2 )
  sentencia
```

La sentencia se ejecutará dependiendo del valor de las expresiones evaluadas por el operador &&

`expresión 1`	`expresión 2`	&&
0	0	0
0	distinto de 0	0
distinto de 0	0	0
distinto de 0	distinto de 0	1

Operadores lógicos

Operadores lógicos

Ejemplo

```
if ( expresión 1 || expresión 2 )
  sentencia
```

Operadores lógicos

Ejemplo

```
if ( expresión 1 || expresión 2 )
  sentencia
```

En el caso de || la tabla de verdad es la siguiente

Operadores lógicos

Ejemplo

```
if ( expresión 1 || expresión 2 )
  sentencia
```

En el caso de $||$ la tabla de verdad es la siguiente

`expresión 1`	`expresión 2`	$\|\|$
0	0	0
0	distinto de 0	1
distinto de 0	0	1
distinto de 0	distinto de 0	1

Operadores lógicos

Operadores lógicos

El tercer operador lógico es la negación NOT (!)

Operadores lógicos

El tercer operador lógico es la negación NOT (!)

`expresión 1`	!
0	1
distinto de 0	0

Precedencia de operadores (actualizada)

Precedencia de operadores (actualizada)

Operador	Asociatividad
`()`	Izq. a Der.
`+ - (tipo) ++ -- !`	Der. a Izq.
`* / %`	Izq. a Der.
`+ -`	Izq. a Der.
`< <= > >=`	Izq. a Der.
`== !=`	Izq. a Der.
`|| &&`	Izq. a Der.
`=`	Der. a Izq.
`+= -= /= *= %=`	Der. a Izq.

Sentencias de repetición (o iterativas)

Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Permite repetir sentencias **mientras** se cumpla una condición

Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Permite repetir sentencias **mientras** se cumpla una condición

La sintaxis es

```
while (expresión)
  sentencias
```

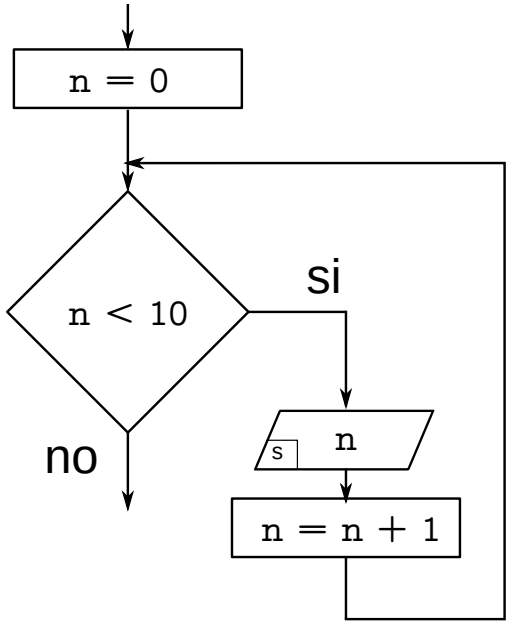
Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

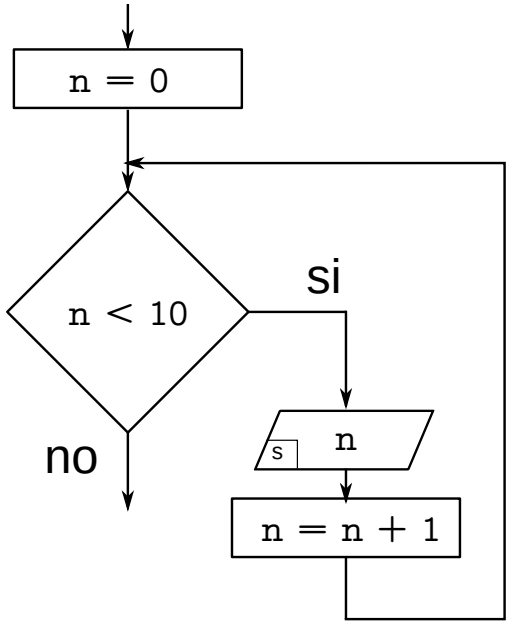
Ejemplo



Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Ejemplo




```c
n = 0;
while ( n < 10 ) {
  printf("%d", n);
  n = n + 1;
}
```

Sentencias de repetición (o iterativas)

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Igual que antes, permite repetir sentencias **mientras** se cumpla una condición

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Igual que antes, permite repetir sentencias **mientras** se cumpla una condición

La sintaxis es

```
do
  sentencias
while (expresión);
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Igual que antes, permite repetir sentencias **mientras** se cumpla una condición

La sintaxis es

```
do
  sentencias
while (expresión);
```

con la diferencia que las sentencias se realizan al menos una vez.

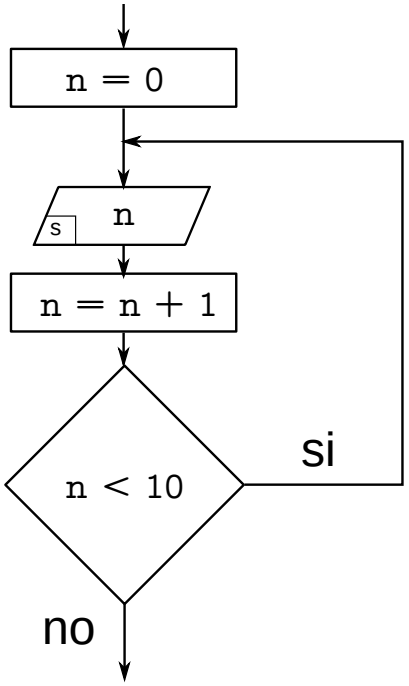
Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

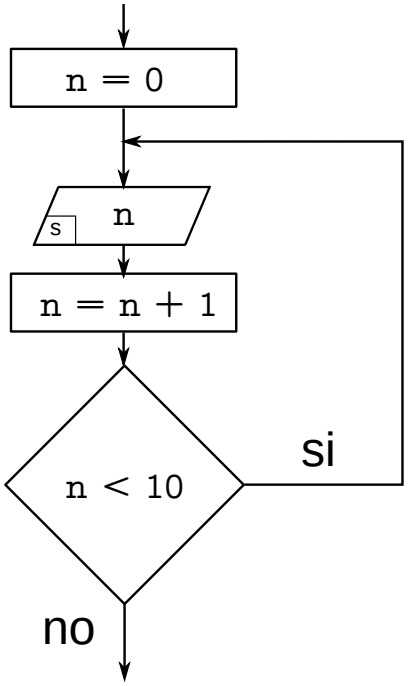
Ejemplo



Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo




```c
n = 0;
do {
  printf("%d", n);
  n = n + 1;
} while ( n < 10 );
```

Sentencias de repetición (o iterativas)

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Se usa generalmente para validar datos ingresados por el usuario

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Se usa generalmente para validar datos ingresados por el usuario

Las sentencias que se ejecutan al menos una vez son el `printf` que solicita el dato y el `scanf` que toma el valor ingresado

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Se usa generalmente para validar datos ingresados por el usuario

Las sentencias que se ejecutan al menos una vez son el `printf` que solicita el dato y el `scanf` que toma el valor ingresado

Si los datos ingresados están en el rango no permitido la condición debe evaluarse por verdadero

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

El usuario debe ingresar una calificación

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

El usuario debe ingresar una calificación

Entonces el rango permitido es desde 1 hasta 10, incluyendo a ambos

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

El usuario debe ingresar una calificación

Entonces el rango permitido es desde 1 hasta 10, incluyendo a ambos

Entonces si se ingresa un número menor que 1 se debe solicitar un nuevo ingreso

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

El usuario debe ingresar una calificación

Entonces el rango permitido es desde 1 hasta 10, incluyendo a ambos

Entonces si se ingresa un número menor que 1 se debe solicitar un nuevo ingreso

Si el número es mayor a 10 también se debe solicitar un nuevo ingreso

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Ejemplo

```
do {
  printf("Ingrese la calificación (1-10): ");
  scanf("%d", &nota);
} while ( nota < 1 || nota > 10 );
```

Sentencias de repetición (o iterativas)

Sentencias de repetición (o iterativas)

Ejemplo

Sentencias de repetición (o iterativas)

Ejemplo

Realizar un programa que calcule el promedio entre `n` notas ingresadas.

Sentencias de repetición (o iterativas)

Ejemplo

Realizar un programa que calcule el promedio entre `n` notas ingresadas.

Las notas deben ser validadas, de forma que solo se acepten aquellas entre 1 y 10.

```c
#include <stdio.h>
// u5-notas.c

int main (void) {
  int i, n;
  int nota, notas;
  float promedio;

  printf("Cuantos registros cargará: ");
  scanf("%d", &n);

  notas = 0;
  i = 0;
  while (i < n) {
    do {
      printf("Ingrese nota %d: ", i+1);
      scanf("%d", &nota);
    } while ( nota < 1 || nota > 10 );
    notas += nota;
    i++;
  }
  promedio = (float) notas / n;

  printf("El promedio de notas es %.2f", promedio);

  return 0;
}
```