Informática I

Claudio Paz

claudiojpaz@gmail.com

Mayo 2019

Unidad 5

Control de flujo en lenguaje C

Primero, algunas definiciones

Sentencias

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

- Simples (las terminadas en ;)

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

- Simples (las terminadas en `;`)
- Compuestas (simples encerradas entre `{}`)

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

- Simples (las terminadas en `;`)
- Compuestas (simples encerradas entre `{}`)
- de Selección (`if`, `if-else` y `switch`)

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

- Simples (las terminadas en `;`)
- Compuestas (simples encerradas entre `{}`)
- de Selección (`if`, `if-else` y `switch`)
- de Repetición (`while`, `do-while` y `for`)

Sentencias

Unidades *sintácticas* de programación que expresan una acción que debe ser llevada a cabo, de las que se espera un resultado

Pueden ser:

- Simples (las terminadas en `;`)
- Compuestas (simples encerradas entre `{}`)
- de Selección (`if`, `if-else` y `switch`)
- de Repetición (`while`, `do-while` y `for`)
- de Salto (`break`, `continue`, `return` y ~~`goto`~~)

Expresiones

Expresiones

Es una combinación de una o más constantes, variables o funciones que se interpretan siguiendo las reglas de precedencia y devuelven un resultado

Expresiones

Es una combinación de una o más constantes, variables o funciones que se interpretan siguiendo las reglas de precedencia y devuelven un resultado

Pueden ser:

Expresiones

Es una combinación de una o más constantes, variables o funciones que se interpretan siguiendo las reglas de precedencia y devuelven un resultado

Pueden ser:

- operaciones (aritméticas, lógicas, de relación, etc.)

Expresiones

Es una combinación de una o más constantes, variables o funciones que se interpretan siguiendo las reglas de precedencia y devuelven un resultado

Pueden ser:

- operaciones (aritméticas, lógicas, de relación, etc.)
- llamados a función

Expresiones

Es una combinación de una o más constantes, variables o funciones que se interpretan siguiendo las reglas de precedencia y devuelven un resultado

Pueden ser:

- operaciones (aritméticas, lógicas, de relación, etc.)
- llamados a función
- variables o constantes

Sentencias de Selección

Sentencias de Selección

Selección simple (`if`)

Sentencias de Selección

Selección simple (`if`)

En lenguaje C, el condicional simple se codifica con la palabra clave `if`

Sentencias de Selección

Selección simple (`if`)

En lenguaje C, el condicional simple se codifica con la palabra clave `if`

La condición que se evalúa se coloca entre paréntesis. Puede ser cualquier expresión

Sentencias de Selección

Selección simple (`if`)

En lenguaje C, el condicional simple se codifica con la palabra clave `if`

La condición que se evalúa se coloca entre paréntesis. Puede ser cualquier expresión

Si la condición se evalúa verdadera se *ejecuta* la sentencia que siguen a la condición.

Sentencias de Selección

Selección simple (`if`)

En lenguaje C, el condicional simple se codifica con la palabra clave `if`

La condición que se evalúa se coloca entre paréntesis. Puede ser cualquier expresión

Si la condición se evalúa verdadera se *ejecuta* la sentencia que siguen a la condición.

```
if ( expresión )
  sentencia
```

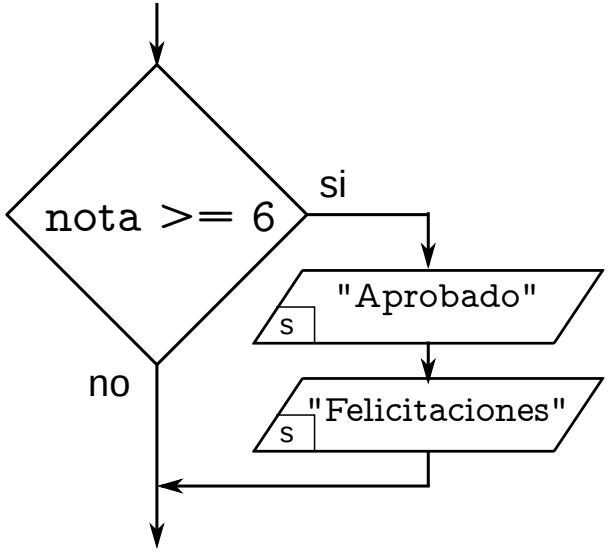
Sentencias de Selección

Selección simple (`if`)

Sentencias de Selección

Selección simple (`if`)

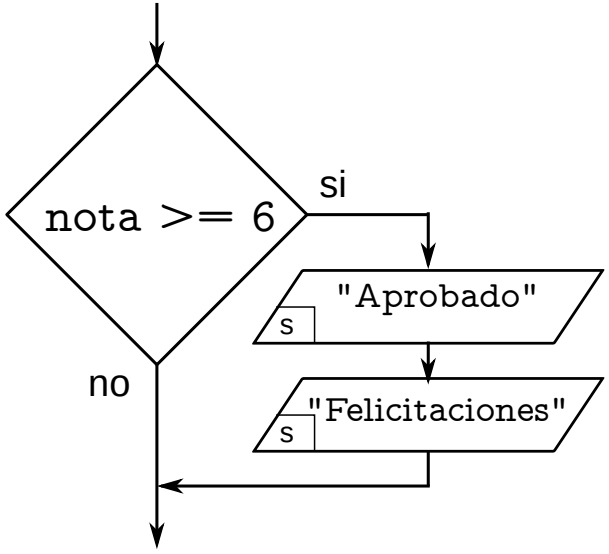
Ejemplo



Sentencias de Selección

Selección simple (`if`)

Ejemplo




```c
if ( nota >= 6 ) {
  printf("Aprobado\n");
  printf("Felicitaciones\n");
}
```

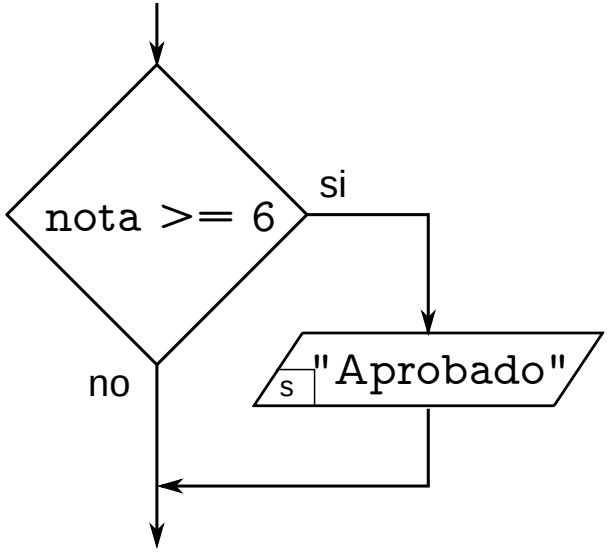
Sentencias de Selección

Selección simple (`if`)

Sentencias de Selección

Selección simple (`if`)

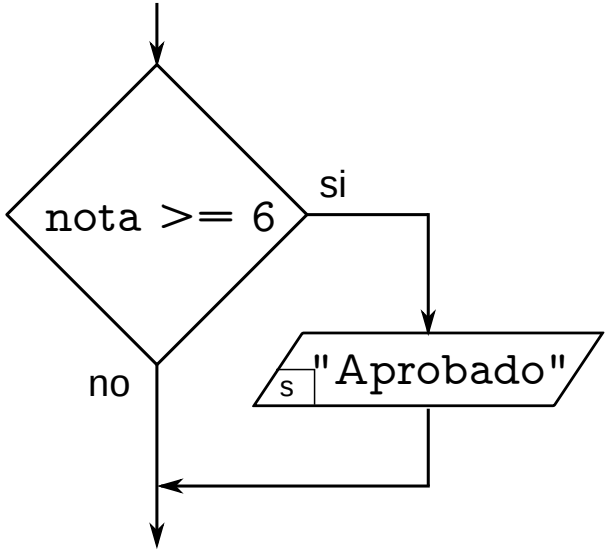
Ejemplo



Sentencias de Selección

Selección simple (`if`)

Ejemplo

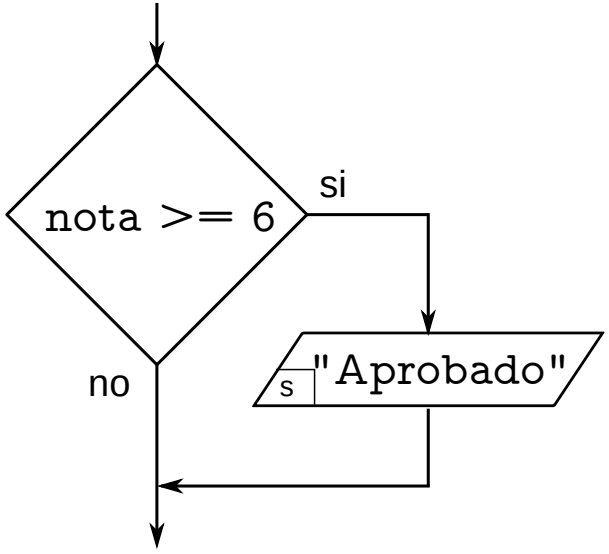



```c
if ( nota >= 6 ) {
  printf("Aprobado\n");
}
```

Sentencias de Selección

Selección simple (`if`)

Ejemplo



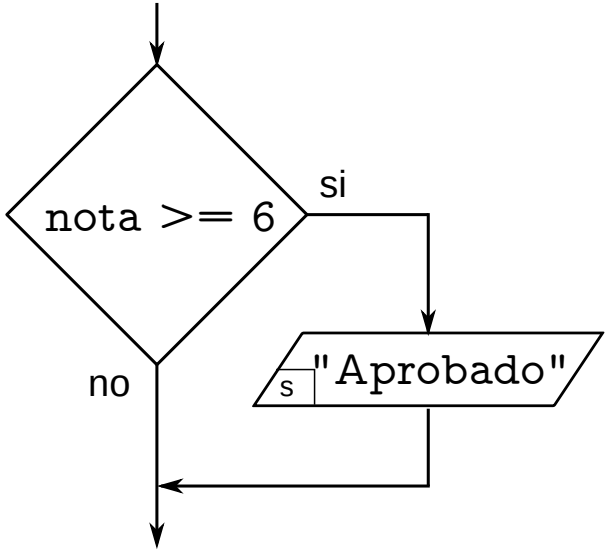

```c
if ( nota >= 6 ) {
  printf("Aprobado\n");
}
```

Si la sentencia compuesta tiene una sola sentencia simple podrían omitirse las llaves

Sentencias de Selección

Selección simple (`if`)

Ejemplo




```c
if ( nota >= 6 ) {
  printf("Aprobado\n");
}
```

Si la sentencia compuesta tiene una sola sentencia simple podrían omitirse las llaves

```c
if ( nota >= 6 )
  printf("Aprobado\n");
```

Sentencias de Selección

Selección simple (`if`)

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

En ocasiones se hacen agregados de código olvidando usar las llaves

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

En ocasiones se hacen agregados de código olvidando usar las llaves

```
if ( nota >= 6 )
  printf("Aprobado\n");
```

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

En ocasiones se hacen agregados de código olvidando usar las llaves

```
if ( nota >= 6 )
  printf("Aprobado\n");
  printf("Felicitaciones\n");
```

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

En ocasiones se hacen agregados de código olvidando usar las llaves

```
if ( nota >= 6 )
  printf("Aprobado\n");
  printf("Felicitaciones\n");
```

lo que causa errores lógicos

Sentencias de Selección

Selección simple (`if`)

Errores lógicos

En ocasiones se hacen agregados de código olvidando usar las llaves

```
if ( nota >= 6 )
  printf("Aprobado\n");
  printf("Felicitaciones\n");
```

lo que causa errores lógicos

En el ejemplo, el programa felicita siempre, sin importar el valor de nota

Sentencias de Selección

Sentencias de Selección

Selección doble (`if-else`)

Sentencias de Selección

Selección doble (`if-else`)

La sintaxis es la siguiente:

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

Sentencias de Selección

Selección doble (`if-else`)

La sintaxis es la siguiente:

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

Si la `expresión` devuelve algo distinto de cero, se ejecuta la `sentencia 1`.

Sentencias de Selección

Selección doble (`if-else`)

La sintaxis es la siguiente:

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

Si la `expresión` devuelve algo distinto de cero, se ejecuta la `sentencia 1`.

Si devuelve cero, se ejecuta la `sentencia 2`.

Sentencias de Selección

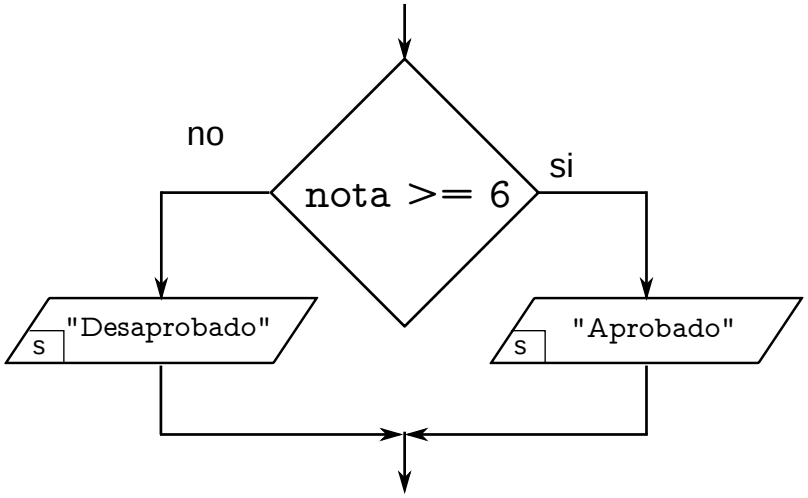
Sentencias de Selección

Selección doble (`if-else`)

Sentencias de Selección

Selección doble (`if-else`)

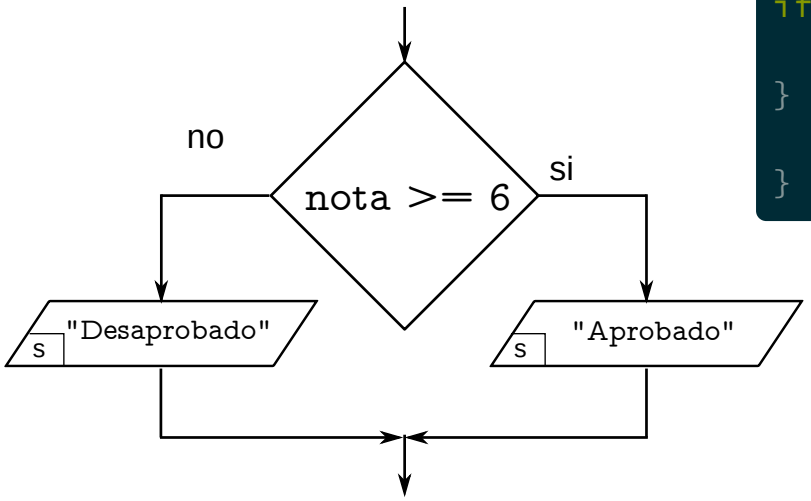
Ejemplo



Sentencias de Selección

Selección doble (`if-else`)

Ejemplo




```c
if ( nota >= 6 ) {
  printf("Aprobado\n");
} else {
  printf("Desaprobado\n");
}
```

Sentencias de Selección

Sentencias de Selección

Las sentencias *dentro* de cada rama del `if` puede ser de cualquier tipo.

Sentencias de Selección

Las sentencias *dentro* de cada rama del `if` puede ser de cualquier tipo.

Si se necesita más de una sentencia por rama, deben formar una sentencia compuesta con las `{}`.

Sentencias de Selección

Las sentencias *dentro* de cada rama del `if` puede ser de cualquier tipo.

Si se necesita más de una sentencia por rama, deben formar una sentencia compuesta con las `{}`.

Se recomienda siempre usar llaves.

Sentencias de Selección

Las sentencias *dentro* de cada rama del `if` puede ser de cualquier tipo.

Si se necesita más de una sentencia por rama, deben formar una sentencia compuesta con las `{}`.

Se recomienda siempre usar llaves.

```
if ( nota >= 6 ) {
  printf("Aprobado\n");
  if ( nota >= 9 ) {
    printf("Excelente!\n");
  }
} else {
  printf("Desaprobado\n");
}
```

Volviendo a los operadores

Volviendo a los operadores

Operadores de Asignación

Volviendo a los operadores

Operadores de Asignación

```
a = 0;

i = i + 1;

d = d / 2;

m = m * 3;

r = r % 10;
```

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.
$+ \quad - \quad (\text{tipo}) \quad ++ \quad -- \quad !$	Der. a Izq.
$* \quad / \quad \%$	Izq. a Der.
$+ \quad -$	Izq. a Der.
$< \quad <= \quad > \quad >=$	Izq. a Der.
$== \quad !=$	Izq. a Der.
$\| \| \quad \&\&$	Izq. a Der.
$=$	Der. a Izq.
$+= \quad -= \quad /= \quad *= \quad \%=$	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
```

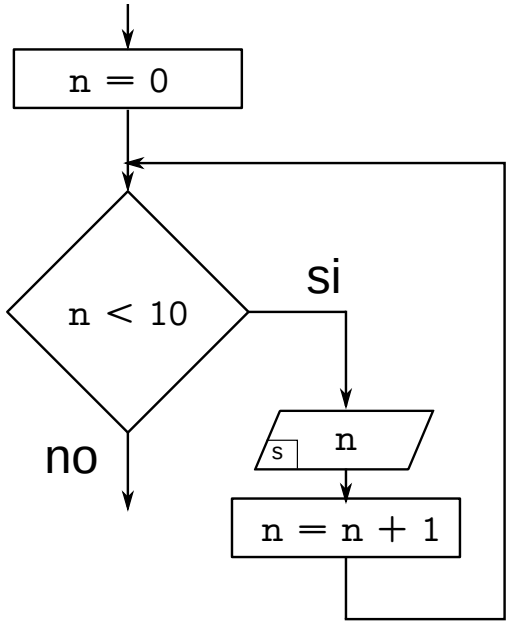
Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `while`

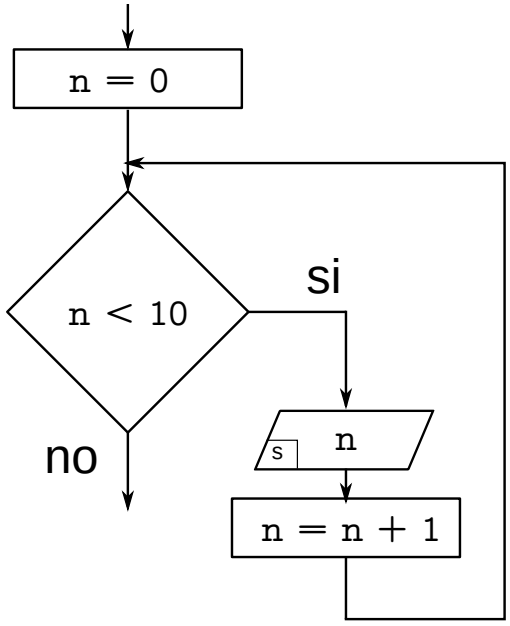
Ejemplo



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.

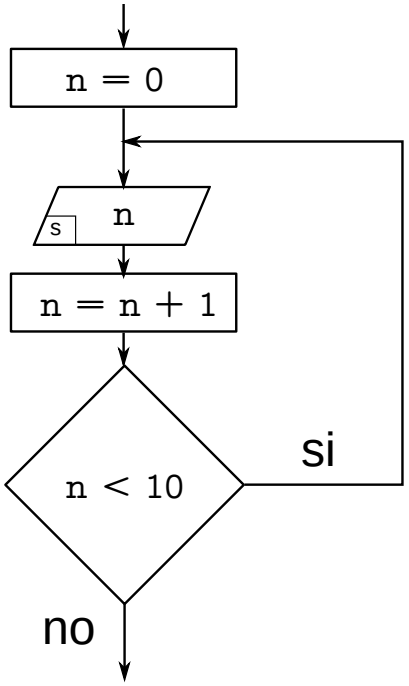
Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `do-while`

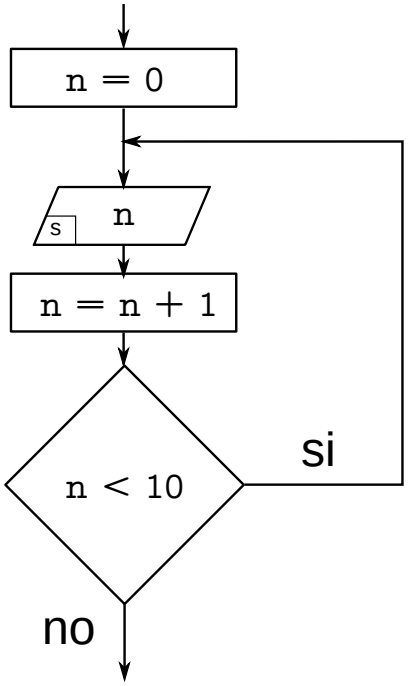
Ejemplo



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;
}
```

Sentencias de Selección

Sentencias de Selección

Selección múltiple (`switch`)

Sentencias de Selección

Selección múltiple (`switch`)

Se usa en casos donde hay muchos posibles valores para una variable con distintas acciones para cada valor.

Sentencias de Selección

Selección múltiple (`switch`)

Se usa en casos donde hay muchos posibles valores para una variable con distintas acciones para cada valor.

Se usa en lugar de anidar multiples `if-else`

Sentencias de Selección

Selección múltiple (`switch`)

Se usa en casos donde hay muchos posibles valores para una variable con distintas acciones para cada valor.

Se usa en lugar de anidar multiples `if-else`

Solo se pueden comparar enteros (pueden ser caracteres pero no flotantes) y solo por igualdad (no relación)

Sentencias de Selección

Sentencias de Selección

Sintaxis

```
switch (expresión) {
    case valor1:
      sentencia 1; // pueden ser muchas, no hace falta {}
      break; // opcional
    case valor2:
      sentencia 2;
      break; // opcional
    // tantos case como se quiera, mientras sean diferentes
    default:
      sentencia n;
      break; // opcional
}
```

Sentencias de Selección

Sentencias de Selección

Se compara la expresión (puede ser una variable, operación, etc.) con las *etiquetas* en los **case**

Sentencias de Selección

Se compara la expresión (puede ser una variable, operación, etc.) con las *etiquetas* en los **case**

Las etiquetas deben ser diferentes. Cuando haya una coincidencia, el flujo del programa salta hasta esa hubicación

Sentencias de Selección

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

```
i = 3;
switch (i) {
  case 1:
    printf("Primera opción\n");
    break;
  case 2:
    printf("Segunda opción\n");
    break;
  case 3:
    printf("Tercera opción\n");
    break;
  default:
    printf("Ninguna opción\n");
}
```

Sentencias de Selección

Selección múltiple (`switch`)

Sentencias de Selección

Selección múltiple (`switch`)

Ejemplo con `if-else`

```c
if (nota == 10) {
  printf("A\n");
} else {
  if (nota == 9) {
    printf("B\n");
  } else {
    if (nota == 8) {
      printf("C\n");
    } else {
      if (nota == 7) {
        printf("D\n");
      } else {
        printf("F\n");
      }
    }
  }
}
```

Sentencias de Selección

Selección múltiple (`switch`)

Ejemplo con `if-else`

```c
if (nota == 10) {
  printf("A\n");
} else {
  if (nota == 9) {
    printf("B\n");
  } else {
    if (nota == 8) {
      printf("C\n");
    } else {
      if (nota == 7) {
        printf("D\n");
      } else {
        printf("F\n");
      }
    }
  }
}
```

Ejemplo con `switch`

```c
switch (nota) {
  case 10:
    printf("A\n");
    break;
  case 9:
    printf("B\n");
    break;
  case 8:
    printf("C\n");
    break;
  case 7:
    printf("D\n");
    break;
  default:
    printf("F\n");
    break;
}
```

Sentencias de repetición (o iterativas)

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Sentencias de repetición (o iterativas)

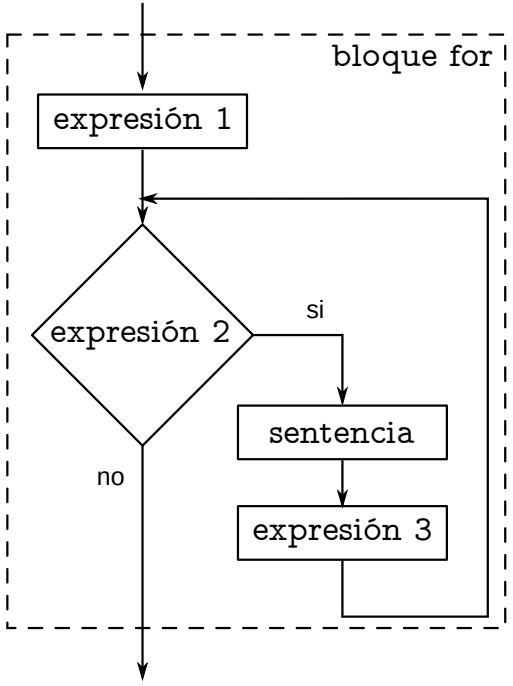
Sentencia repetitiva `for`

Tiene un funcionamiento semejante
al `while` controlado por contador

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

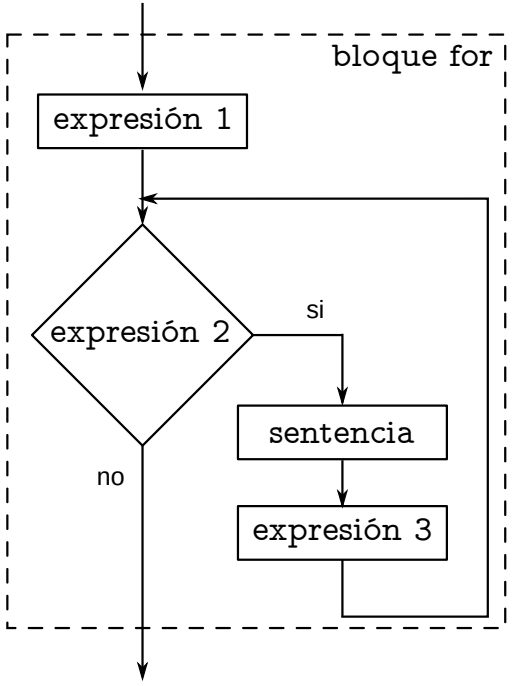


Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

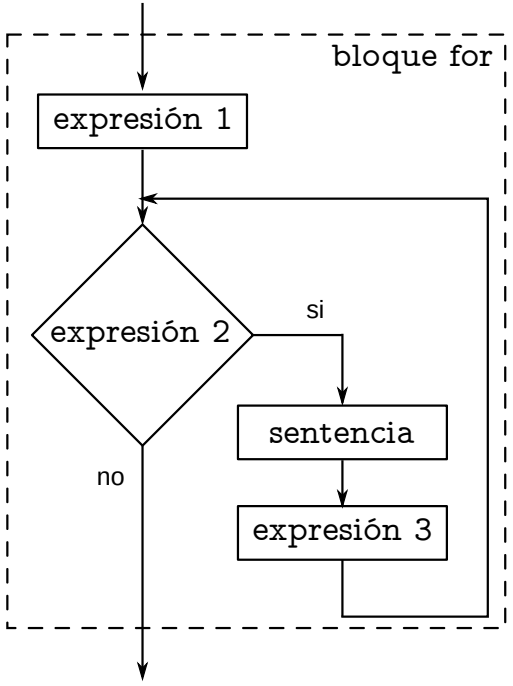
La `expresión 1` se ejecuta una vez, al principio



Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

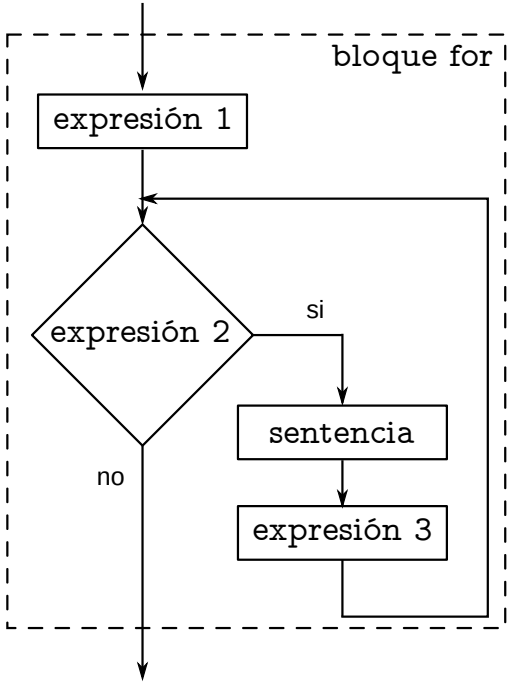


Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

La `expresión 2` sirve de condición para repetir el ciclo



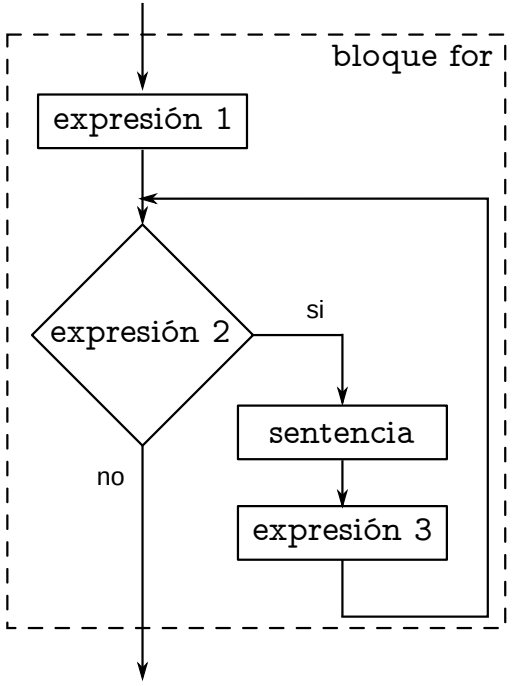
Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

La `expresión 2` sirve de condición para repetir el ciclo

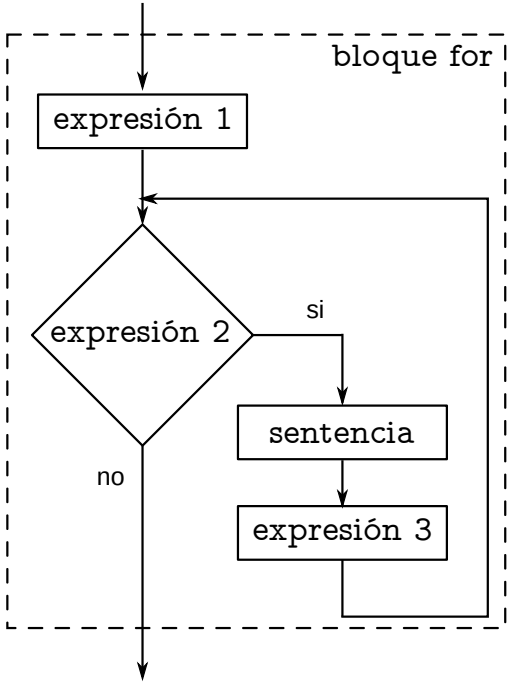
Se evalúa todos los ciclos



Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

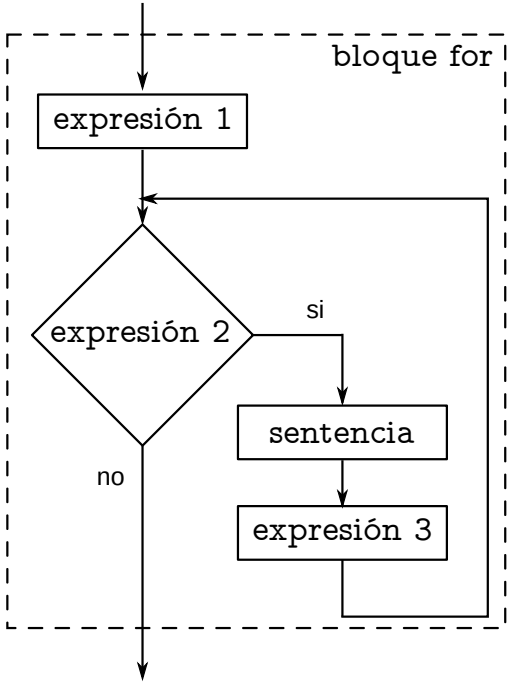


Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

La `sentencia` puede ser simple, compuesta, condicional o iterativa.



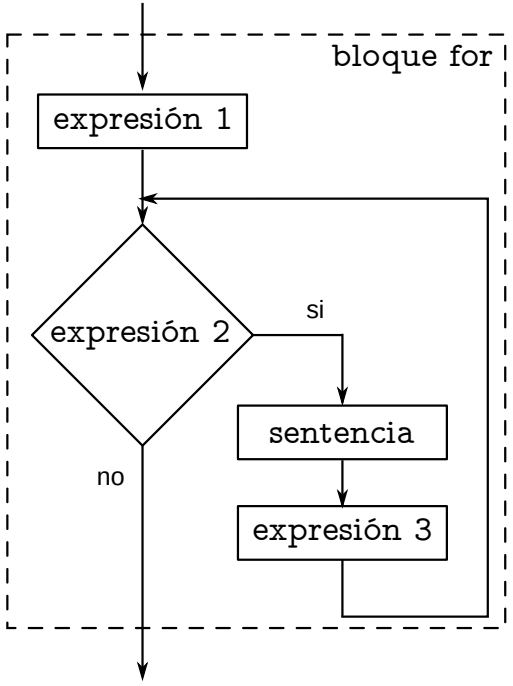
Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

La `sentencia` puede ser simple, compuesta, condicional o iterativa.

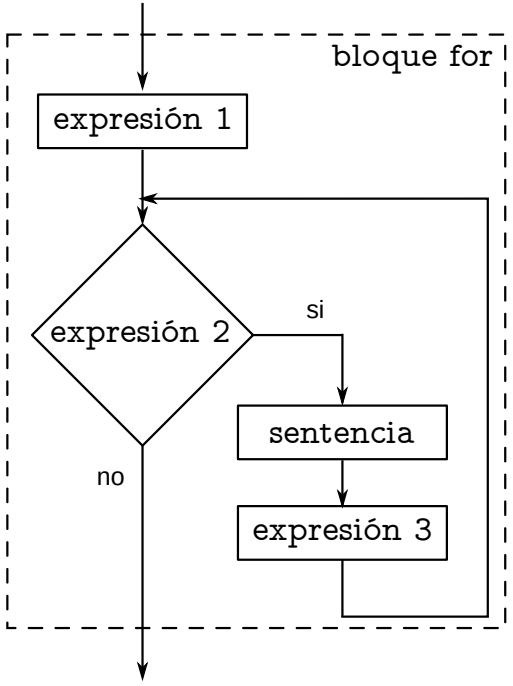
Se ejecuta todos los ciclos



Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

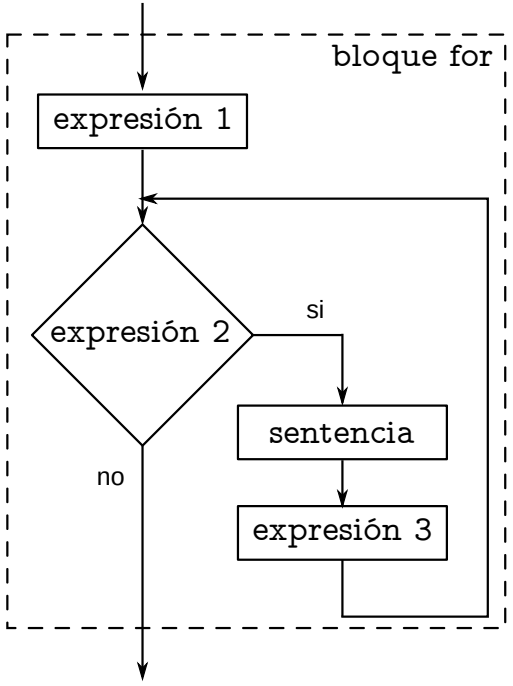


Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Tiene un funcionamiento semejante al `while` controlado por contador

La `expresión 3` se ejecuta todos los ciclos al finalizar la `sentencia`



Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

```
for ( expresión 1; expresión 2; expresión 3 )
  sentencia
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

```
for ( expresión 1; expresión 2; expresión 3 )
  sentencia
```

Las expresiones son opcionales

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

```
for ( expresión 1; expresión 2; expresión 3 )
  sentencia
```

Las expresiones son opcionales

Si falta la `expresión 2` se considera que la condición es verdad y el bucle continúa indefinidamente

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

```
for ( i = 0; i < 5; i++ )
  printf("vuelta %d\n", i);
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

```
for ( i = 0; i < 5; i++ )
  printf("vuelta %d\n", i);
```

```
vuelta 0
vuelta 1
vuelta 2
vuelta 3
vuelta 4
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

A partir del estándar C99 se puede declarar la variable de control en la `expresión 1`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

A partir del estándar C99 se puede declarar la variable de control en la `expresión 1`

```c
#include <stdio.h>
// u5-for-init-1.c

int main (void) {

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

  return 0;
}
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Pero la variable solo existe en el bloque del `for`

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Pero la variable solo existe en el bloque del `for`

```c
#include <stdio.h>
// u5-for-init-2.c

int main (void) {

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

  return 0;
}
```

Sentencias de repetición (o iterativas)

Sentencia repetitiva `for`

Pero la variable solo existe en el bloque del `for`

```c
#include <stdio.h>
// u5-for-init-2.c

int main (void) {

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

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u5-for-init-2.c
u5-for-init-2.c: In function ‘main’:
u5-for-init-2.c:9:18: error: ‘i’ undeclared (first use in this function)
   printf("%d\n", i);
                  ^
```