

Control de flujo: if, while

Control de flujo e iteración



Yosafat Coronel
GDSC ESCOM IPN
GitHub: YosafatM



Developer Student Clubs
ESCOM - IPN

Itinerario



- Flujo del programa
- Declaración y expresión
- Expresiones booleanas
- Paquetes en Dart: dart:io
- Bucles y condicionales
 - if
 - while
 - do-while
- Ejercicio

Repo del curso

Por alguna razón, olvidé decírles que hay un repo del curso en GitHub:

**[github.com/DSC-
ESCOM-IPN/2021-Curso-
Dart](https://github.com/DSC-ESCOM-IPN/2021-Curso-Dart)**

Flujo del programa (1)

Hasta ahora nuestros programas solo recorren las líneas de código de arriba hacia abajo, una forma de controlarlo es con la declaración if (el contenido en corchete indica que es opcional).

```
if (expresión booleana) {  
    // Código si al evaluar la expresión da true  
} [else {  
    //Código si al evaluar la expresión da false  
}]
```

Declaración y expresión

Declaración y expresión

En la diapositiva usamos dos términos importantes, declaración ([statement](#)) y expresión ([expression](#)). Una expresión es una [serie de literales o nombres \(identificadores\)](#) o [expresiones que generan un valor](#), es decir, a un determinado tiempo puede ser evaluado:

```
password == '1234';
```

En este caso, es una expresión condicional o booleana, y puede tener el valor de true o false.

Declaración y expresión

Por el otro lado, una declaración es una orden (que puede componerse de más órdenes). Una declaración puede estar compuesta de más declaraciones y de expresiones:

```
if (p == 4) doSomething();
```

El if, es una declaración, pero necesita de una expresión. El bloque que contiene, también es una declaración.

Expresiones Booleanas

Expresiones Booleanas

Operadores booleanos

Si ambos lados son iguales, entonces retorna true. En los String se hace carácter por carácter.

**!=, ==, >=,
>, <=, <, is**

Expresiones Booleanas

De los operadores, todos pueden tener literales o identificadores (nombres de variables) como comparación, o bien, más expresiones booleanas:

<lit> <op> <lit>
<id> <op> <id>
<ex> <op> <ex>

<tipo> nos dice que pongamos ese tipo. ex: expresión, op: operador, lit: literal. (todo bool)

Expresiones Booleanas

Los operadores OR y AND nos sirven para los casos cuando tenemos varias expresiones y queremos un valor específico para cada expresión.

*expr || expr (con
un true da true)*

OR ||
AND &&

*expr && expr (con
un false, da false)*

Flujo del programa (2)

Si tenemos más condiciones, podemos **anidar** los if, la expresiones se evaluan de arriba a abajo, si la primera da true, el resto no se lee:

```
if (expresión booleana) {  
    // Código si al evaluar la expresión da true  
} else if (otra expresión booleana) {  
    // Código si al evaluar la otra expresión da true  
} else {  
    // Código si al evaluar la otra expresión da false  
}
```

Paquetes en Dart

Paquetes en Dart

Muchas funciones en Dart (y Flutter) están ya programadas en archivos. A estas funcionalidades se les puede acceder con **import**, se le puede poner un **alias** como en Python.

'paquete:archivo/nombre' *opcional*

```
import 'dart:io' [as alias];
```

Lectura de la CL

Del paquete dart:io, stdin viene dentro del paquete. Con la siguiente función se puede leer una línea que ingrese el usuario por la línea de comandos:

```
String a;
```

```
a = stdin.readLineSync();
```

Vamos a probarlo

En el siguiente repl, se muestra cómo se usa la lectura con dart:io, vamos a modificarlo durante esta clase

replit.com/@YosafatCoronel/2-Lectura

Bucles y condicionales

While

Si queremos que una declaración se repitan de forma cíclica, podemos usar la estructura while, al terminar el código, vuelve a evaluar la expresión mientras sea true, ejecutará el código:

```
while (expresión booleana) {  
    // Código a repetir si la expresión da true  
}
```

Do - While

En caso de que queramos ejecutar primero el código y después preguntar por la evaluación de la expresión, entonces usamos el do-while.

```
do {  
    // Código a repetir si la expresión da true  
} while (expresión booleana);
```

Notar que lleva ; al final

Bucles y condicionales

Aunque ambos controlan el flujo de un programa, el if es de tipo **condicional** (su intención principal es decidir), mientras que el while es de **bucle** (aunque también decide, su uso principal es hacer entrar el flujo en bucle).

bucle: while, do-while, for

condicional: switch, if

Keyword: break

Cuando queremos **salir** antes de una **estructura de control de bucle**, con usar **break**

```
while (expresión booleana) {  
    // Algo de código  
    break;  
    // Código que no se ejecuta  
}
```

Ejercicio

Momento de codificar

Con base al repl, haz que, usando un do-while, siempre se muestre el menú, reciba la opción del usuario, y en caso de que la opción sea 'z', deje de mostrar el menú y termine el programa.



(Possible) Respuesta



replit.com/@YosafatCoronel/2-Ejercicio-Control-de-Flujo

replit.com/@YosafatCoronel/2-Ejercicio-Control-de-Flujo

Dudas y preguntas

¡Muchas gracias!

