

Guía de aprendizaje

Estructuras de control y condicionales

Exp 2 – Semana 4

Fundamentos de Programación (PRY2201)

**Facilitador disciplinar:** Josué Oteiza

**Asesor par:** Luis Videla

# Índice

[Introducción a la semana 3](#_Toc158887535)

[Resultado de aprendizaje 4](#_Toc158887536)

[Conceptos relevantes 4](#_Toc158887537)

[Preguntas activadoras 5](#_Toc158887538)

[Actividad 5](#_Toc158887539)

[Estructuras de control 6](#_Toc158887540)

[Bucles o Ciclos 6](#_Toc158887541)

[Estructuras de control 6](#_Toc158887542)

[Estructuras Condicionales 18](#_Toc158887543)

[Operaciones lógicas y condicionales 24](#_Toc158887544)

[Links de Interés 27](#_Toc158887545)

[Cierre de la semana 28](#_Toc158887546)

[Referencias 29](#_Toc158887547)

[Lecturas de la semana 29](#_Toc158887548)

[Apuntes 30](#_Toc158887549)

# Introducción a la semana

¡Ya te encuentras en la Experiencia 2 de la asignatura de Fundamentos de la Programación! Durante esta semana, profundizaremos en los bucles y ciclos en Java que, si bien recuerdas, son estructuras fundamentales que permiten la repetición controlada de bloques de código, facilitando la automatización de tareas y la manipulación de datos. Estas construcciones proporcionan una manera eficiente y elegante de ejecutar instrucciones de manera iterativa, permitiendo que un conjunto de operaciones se repita hasta que se cumpla una condición específica. En Java, los bucles más comunes incluyen "for", "while" y "do-while", cada uno con sus propias características y aplicaciones particulares. Estos elementos son esenciales para mejorar la eficiencia y la flexibilidad de los programas, brindando a los desarrolladores la capacidad de gestionar repeticiones de manera estructurada y controlada.

# Resultado de aprendizaje

### El estudiante será capaz de:

**RA2.** Aplica elementos básicos de Java para el desarrollo de programas que solucionen problemáticas planteadas.

### Indicador de logro:

**IL4.**  Utiliza Algoritmos en Java en la creación de programas funcionales que solucionen las problemáticas planteadas.

# Conceptos relevantes

|  |  |  |
| --- | --- | --- |
| Estructuras de control en Java | Tipos de datos | Operadores |
| Prácticas de codificación | Algoritmos de búsqueda | Algoritmos de ordenamiento |

# Preguntas activadoras

* ¿Cómo crees que la comprensión de los algoritmos puede mejorar la resolución de problemas en la vida diaria?
* ¿Cuál es la importancia de las estructuras de control condicionales y los bucles en la escritura de programas?
* ¿Qué desafíos podrían surgir al trabajar con diferentes tipos de datos en un programa? ¿Cómo abordarías la conversión de tipos de datos para garantizar un flujo de programa adecuado?

# Actividad

### Descripción de la actividad

En esta cuarta semana realizarás una actividad formativa individual con encargo llamada "Utilizando estructuras de control y condicionales en Java" donde, a través de una problemática planteada, deberás crear un algoritmo para crear una solución. Tal solución debe contener estructuras de control y estructuras condicionales dentro del proceso.

# Estructuras de control

## Bucles o Ciclos

En la experiencia pasada, vimos varios elementos que nos ayudaron a construir nuestro primer algoritmo; revisamos estrategias de abstracción y tipos de algoritmos, aplicamos pseudocódigo y utilizamos variables, expresiones aritméticas y lógicas en la resolución de problemas algorítmicos. Además, conocimos lo que son las estructuras de control y estructuras condicionales. A partir de esto último, comenzaremos profundizando en los bucles o ciclos y luego, en las estructuras condicionales en Java.

Hagamos un repaso de lo que conocemos como estructuras de control.

## Estructuras de control

Si bien recuerdas, en semanas previas mencionamos que las estructuras de control son instrucciones que permiten controlar y modificar el flujo de ejecución de un programa. Si bien hay varios tipos, nos centraremos en las **estructuras iterativas o repetitivas** más comunes, conocidas como bucles o ciclos.

Importante

En diferentes lenguajes de programación, puedes encontrar que los términos “bucle” y “ciclo” se utilizan de manera intercambiable. Por ejemplo, en Java se usa el término "ciclo", mientras que en otros lenguajes como Python se utiliza "bucle". En la práctica, los programadores a menudo utilizan el término que se sienta más natural en el contexto del lenguaje de programación que están utilizando, pero para no confundirnos, de aquí en adelante hablaremos de “ciclo”, siguiendo el lenguaje de programación Java.

Los ciclos son esenciales para la automatización de tareas repetitivas y la manipulación de datos en programas Java; son un componente clave para crear algoritmos eficientes y flexibles, y sus tipos más comunes son ‘for’, ‘while’ y ‘do-while’.

Definamos a continuación, cada uno de ellos:

**Ciclo ‘*for’***

Es ideal cuando se conoce la cantidad exacta de veces que deseas repetir una acción. Se caracteriza por su estructura de inicialización, condición y actualización, lo que lo hace adecuado para la iteración controlada.

Ahora veremos un ejemplo de ciclo ‘for’:

Supongamos que queremos imprimir los números del 1 al 5 utilizando un ciclo ‘for’. El código se vería así:

**Figura 1**

*Ciclo ‘for’*

*Nota*. Ejemplo de aplicación de un ciclo ‘for’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Como puedes ver en el ejemplo, se realizan tres pasos, estos son:

**Paso 1** - Ciclo ‘for’ con inicialización, condición y expresión de actualización:

for (int contador = 1; contador <= 5; contador++) {

En el encabezado del ciclo ‘for’, se especifican tres partes:

* La inicialización ‘(int contador = 1)’,
* La condición ‘(contador <= 5)’, y
* La expresión de actualización ‘(contador++)’.

El ciclo se ejecutará mientras la condición sea verdadera.

**Paso 2** - Cuerpo del ciclo:

System.out.println("Número: " + contador);

En este bloque de código, imprimimos el valor actual de contador. Esto se repetirá para cada iteración del ciclo.

**Paso 3** - Fin del programa:

System.out.println("Fin del programa");

Después de salir del ciclo ‘for’, mostramos un mensaje indicando que el programa ha llegado a su fin.

**Ciclo ‘while’**

Se utiliza cuando no se conoce de antemano cuántas veces se debe repetir una acción, pero se ejecutará mientras una condición dada sea verdadera. Es una herramienta poderosa para iteraciones basadas en condiciones.

Veamos a continuación dos ejemplos de implementación del ciclo ‘while’:

**Ejemplo 1**

Supongamos que queremos imprimir los números del 1 al 5 utilizando un ciclo ‘while’. El código se vería así:

**Figura 2**

*Ciclo ‘while’*

*Nota*. Ejemplo de aplicación de un ciclo ‘while’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Los pasos por seguir en este ejemplo son los siguientes:

**Paso 1** - Inicialización de la variable de control:

int contador = 1;

Aquí, creamos una variable llamada ‘contador’ e inicializamos su valor en 1. Esta variable se utilizará para contar y controlar el ciclo.

**Paso 2** - Definición de la condición del ciclo:

while (contador <= 5) {

El ciclo while continuará ejecutándose mientras la condición contador <= 5 sea verdadera.

**Paso 3** - Cuerpo del ciclo:

System.out.println("Número: " + contador);

En este bloque de código, imprimimos el valor actual de contador. En este caso, se mostrarán los números del 1 al 5.

**Paso 4** - Actualización de la variable de control:

En cada iteración, incrementamos el valor de contador en 1. Esto es esencial para evitar un bucle infinito y para avanzar hacia la condición de salida.

contador++;

**Paso 5** - Fin del programa:   
Después de salir del ciclo ‘while’, mostramos un mensaje indicando que el programa ha llegado a su fin.

System.out.println("Fin del programa");

**Ejemplo 2**

El siguiente ejemplo nos muestra cómo calcular el factorial de un número en Java con un ciclo ‘while’. El factorial de un número natural es el producto de todos los números naturales menores o iguales a él. Por ejemplo, el factorial de 5 es 120, que es el producto de 1, 2, 3, 4 y 5.

Para calcular el factorial de un número en Java con un ciclo ‘while’, podemos seguir los siguientes pasos:

**Figura 3**

*Ciclo ‘while’*

1. Declara una variable para almacenar el factorial.

2. Declara una variable para almacenar el número actual.

3. Inicializa la variable ‘numeroActual’ al valor del número a calcular.

4. Inicializa el factorial a 1.

5. Imprime el factorial.

*Nota*. Ejemplo de aplicación de un ciclo ‘while’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Importante

Mientras ‘numeroActual’ sea mayor o igual a 1, realiza los siguientes pasos:

* Multiplicar el factorial por ‘numeroActual’.
* Decrementar ‘numeroActual’ en 1.

Como puedes ver, este código declara las variables ‘factorial’ y ‘numeroActual’. La variable ‘factorial’ almacenará el factorial del número a calcular, y la variable ‘numeroActual’ almacenará el número actual. El código luego inicializa ‘numeroActual’ al valor del número a calcular.

El código luego entra en un ciclo ‘while’ que se ejecutará mientras ‘numeroActual’ sea mayor o igual a 1. Dentro del ciclo, el código multiplica el factorial por ‘numeroActual’ y luego decrementa ‘numeroActual’ en 1.

Cuando el ciclo termina, el factorial almacenará el factorial del número calculado. El código luego imprime el factorial donde, al ejecutar este código, se imprimirá el siguiente resultado:

El factorial de 5 es: 120

**Ciclo ‘do-while’**

Similar al ciclo ‘while’, el ciclo ‘do-while’ garantiza que el bloque de código se ejecute al menos una vez antes de verificar la condición. Esto puede ser útil cuando deseas que el código se ejecute antes de realizar una comprobación.

En el siguiente ejemplo, supondremos que queremos imprimir los números del 1 al 5 utilizando un ciclo ‘do-while’. Esto quedaría de la siguiente manera:

**Figura 4**

*Ciclo ‘do-while’*

*Nota*. Ejemplo de aplicación de un ciclo ‘do-while’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Los pasos a seguir son los siguientes:

**Paso 1** - Inicialización de la variable de control:

int contador = 1;

Creamos una variable llamada ‘contador’ e inicializamos su valor en 1. Esta variable se utilizará para contar y controlar el ciclo.

**Paso 2** -Ciclo ‘do-while’:

do {

El bloque de código dentro del ‘do’ se ejecutará al menos una vez, sin importar la condición. La condición se evaluará al final del bloque.

**Paso 3** - Cuerpo del ciclo:

System.out.println("Número: " + contador);

En este bloque de código, imprimimos el valor actual de ‘contador’. Esto se repetirá mientras la condición del ‘while’ sea verdadera.

**Paso 4** - Actualización de la variable de control:

contador++;

En cada iteración, incrementamos el valor de contador en 1.

**Paso 5** - Condición del ‘while’:

} while (contador <= 5);

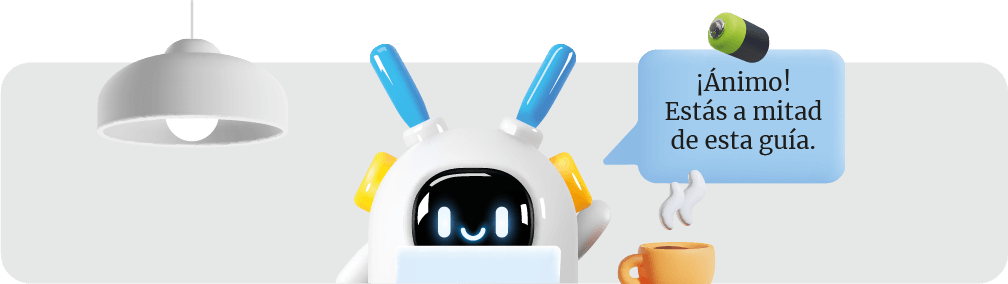
Después de ejecutar el bloque de código, la condición del ‘while’ se evalúa. Si es verdadera, el ciclo se repetirá; de lo contrario, el ciclo se detendrá.

**Paso 6** - Fin del programa:

System.out.println("Fin del programa");

Después de salir del ciclo ‘do-while’, mostramos un mensaje indicando que el programa ha llegado a su fin.

Ahora que hemos repasado los tres tipos principales de bucles en Java, es importante comprender cómo se relacionan con las estructuras condicionales, como los *condicionales* ‘if, else’, y la estructura ‘switch-case’, ¿las recuerdas?



## Estructuras Condicionales

Las estructuras condicionales se utilizan para ejecutar un bloque de código solo si se cumple una condición específica. Veamos cómo se relacionan y afectan los ciclos con estas estructuras.

Para comprender las estructuras condicionales, es esencial examinar sus componentes fundamentales. Te los mostramos a continuación:

1. **Expresión de condición:** la expresión de condición es la parte central de una estructura condicional. Es una expresión que evalúa si una condición es verdadera o falsa. Por ejemplo, en Java, una expresión de condición podría ser `x > 5`, lo que evalúa si la variable `x` es mayor que 5.
2. **Bloque de código verdadero:** este bloque contiene las instrucciones que se ejecutarán si la expresión de condición se evalúa como verdadera. Es donde especificamos qué hacer cuando la condición es cierta.
3. **Bloque de código falso (opcional):** en algunas estructuras condicionales, se incluye un bloque de código que se ejecuta si la expresión de condición es falsa (no todas las condicionales requieren este bloque). En lenguajes como Java, este bloque se define con la instrucción `else`.
4. **Evaluación de la condición:** el motor de ejecución del programa evalúa la expresión de condición. Si la expresión es verdadera, se ejecuta el bloque de código verdadero; si es falsa, se ejecuta el bloque de código falso (si está presente).

Si reflejamos todos estos componentes en Java, esto se realizaría de la siguiente manera:

**Figura 5**

*Estructura condicional en JavaNota*. Ejemplo de aplicación de estructura condicional en Java. *The Apache Software Foundation. (2023). Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, la **expresión de condición** es una variable booleana llamada ‘condicion’ en el primer caso y una comparación numérica (‘numero > 5’) en el segundo caso. Dependiendo de si la condición es verdadera o falsa, se ejecutará el **bloque de código** dentro del ‘if’ o el ‘else’.

Podrás ver que, en este caso, la variable ‘condicion’ se ha establecido inicialmente en ‘false’. Si cambias su valor a ‘true’, verás que el programa imprimirá el mensaje del bloque de código verdadero. Esto demuestra cómo la **evaluación de la condición** afecta el flujo de ejecución en una estructura condicional.

Además, en el ejemplo se utilizaron dos estructuras condicionales: ‘if’ y ‘else’. Junto a ellas también encontramos las estructuras ‘else-if’ y ‘switch-case’. Recordemos un poco de qué tratan cada una y cómo se pueden aplicar en diversas situaciones:

### Tipos comunes de estructuras condicionales

1. `if`: es el más básico y se utiliza para ejecutar un bloque de código si una expresión de condición es verdadera. Por ejemplo, aplica perfectamente si queremos verificar si una persona es mayor de edad y, en caso afirmativo, se imprime un mensaje relacionado con el derecho a votar:

**Figura 6**

*Condicional ‘if’*

*Nota*. Ejemplo de aplicación de estructura condicional ‘if’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. `else`: se utiliza junto con un `if` para especificar qué hacer si la expresión de condición es falsa. Continuemos con el ejemplo de la Figura 6, donde quisimos verificar si una persona es mayor de edad:

**Figura 7**

*Condicional ‘else’*

Si la condición es verdadera, se ejecuta el bloque de código dentro del ‘if’; de lo contrario, se ejecuta el bloque de código dentro del ‘else’.

*Nota*. Ejemplo de aplicación de estructura condicional ‘else’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. `else if`: se usa para evaluar múltiples condiciones en secuencia y ejecutar el bloque de código asociado a la primera condición verdadera:

**Figura 8**

*Condicional ‘else-if’*

*Nota*. Ejemplo de aplicación de estructura condicional ‘else’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, se utiliza ‘if’ para verificar si el número es positivo. Si esa condición es falsa, se evalúa la siguiente condición utilizando ‘else if’ para verificar si el número es negativo. Si ambas condiciones anteriores son falsas, se ejecuta el bloque dentro de ‘else’, indicando que el número es cero.

Importante

‘else-if’ se utiliza cuando tienes múltiples condiciones que deseas evaluar en orden, y solo se ejecutará el bloque de código correspondiente a la primera condición verdadera. ‘else’, en cambio, se ejecuta cuando todas las condiciones anteriores resultan falsas.

1. `switch-case`: se utiliza para tomar decisiones basadas en el valor de una expresión. Aunque comparte algunas similitudes con ‘if-else’, ‘switch-case’ permite comparar el valor de una variable con diferentes casos y ejecutar el bloque de código correspondiente:

**Figura 9**

*Condicional ‘switch-case’*

*Nota.* Ejemplo de aplicación de estructura condicional ‘else’ en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

La variable ‘opcion’ se evalúa en la expresión ‘switch’, y se ejecuta el bloque de código correspondiente al caso que coincide con el valor de ‘opcion’. Si no hay coincidencia con ningún caso, se ejecuta el bloque de código dentro de ‘default’. Cada caso debe finalizar con ‘break’ para salir del ‘switch’ y evitar la ejecución de los casos siguientes.

Como puedes ver, las estructuras condicionales son esenciales para escribir programas que se adapten dinámicamente a diferentes situaciones como, por ejemplo, validar entradas de usuario, controlar el flujo de un programa, manejar excepciones, personalizar la experiencia de usuario, automatizar tareas, entre otras.

## Operaciones lógicas y condicionales

En este contexto, las expresiones booleanas determinan el curso de acción que el programa debe seguir. Al emplear operadores lógicos como AND (`&&`), OR (`||`) y NOT (`!`), podemos combinar condiciones para crear evaluaciones más complejas. Este enfoque lógico subyacente permite a los programadores modelar decisiones complejas y crear programas que se adapten dinámicamente a diferentes situaciones, mejorando así la flexibilidad y la eficiencia del código.

Ahora veamos, según la teoría de la lógica booleana, un paso a paso de cómo se utilizan las operaciones lógicas para combinar y negar condiciones:

**Figura 10**

*Operaciones lógicas*

*Nota.* Ejemplo de aplicación de una aplicación lógica en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Los pasos del ejemplo que determina si una persona puede conducir basándose en la edad y si tiene o no permiso, son los siguientes:

**Paso 1**: importamos la clase ’Scanner’ para poder leer la entrada del usuario:

import java.util.Scanner;

**Paso 2**: definimos la clase principal llamada ‘Conducir’:

public class Conducir {

**Paso 3:** creamos un objeto ‘Scanner’ llamado ‘scanner’ para leer la entrada del usuario desde la consola:

Scanner scanner = new Scanner(System.in);

**Paso 4:** solicitamos al usuario que ingrese su edad y almacenamos el valor en la variable ‘edad’:

System.out.println("Ingrese su edad:");

int edad = scanner.nextInt();

**Paso 5:** solicitamos al usuario que ingrese si tiene o no permiso de conducir y almacenamos el valor en la variable ‘tienePermiso’:

if (edad >= 18 && tienePermiso) {

System.out.println("Usted puede conducir.");

} else if (edad >= 16 && !tienePermiso) {

System.out.println("Usted puede obtener un permiso para conducir.");

} else {

System.out.println("Usted no puede conducir.");

}

**Paso 6:** utilizamos una estructura condicional ‘if-else’ con operadores lógicos. Si la edad es mayor o igual a 18 y tiene permiso, se imprime que puede conducir. Si la edad es mayor o igual a 16 y no tiene permiso, se sugiere obtener un permiso para conducir. En caso contrario, se indica que no puede conducir:

if (edad >= 18 && tienePermiso) {

System.out.println("Usted puede conducir.");

} else if (edad >= 16 && !tienePermiso) {

System.out.println("Usted puede obtener un permiso para conducir.");

} else {

System.out.println("Usted no puede conducir.");

}

**Paso 7:** finalmente, cerramos el ‘Scanner’ para evitar fugas de recursos:

scanner.close();

# Links de Interés

Te invitamos a seguir profundizando en las estructuras condicionales en Java a través del siguiente enlace:

<https://programandoointentandolo.com/2017/07/estructuras-condicionales-java.html>

También podrás revisar una amplia gama de problemas de codificación con soluciones en Java, a través del siguiente enlace:

<https://leetcode.com/problemset/?topicSlugs=java>

# Cierre de la semana

Las estructuras de control en programación ya sean iterativas o condicionales, desempeñan un papel esencial en el diseño de algoritmos. Las iterativas, como ‘for’, ‘while’ y ‘do-while’, permiten la repetición controlada de un bloque de código, facilitando la ejecución de tareas repetitivas. Por otro lado, las condicionales, como ‘if’, ‘else’, ‘else-if’ y ‘switch-case’, brindan la capacidad de tomar decisiones basadas en condiciones booleanas, adaptando el flujo del programa a situaciones específicas.

La lógica booleana, inspirada en el álgebra de Boole, subyace en estas estructuras, ya que las evaluaciones de verdad o falsedad de las expresiones condicionales guían la ejecución del código. Este enfoque lógico permite crear programas que no solo siguen una secuencia lineal, sino que también responden dinámicamente a diferentes escenarios, mejorando así la flexibilidad y eficacia del software.



# Referencias

Sanchez, R. (s.f.) *Estructuras condicionales.* Gitbooks. <https://rsanchezs.gitbooks.io/ciencia-de-datos-con-r/content/estructuras_control/condicionales/estructuras_condicionales.html>

# Lecturas de la semana

* Chapter 6: Controlling Program Flow with Loops

Burd, B. (2022). Java for Dummies. New Jersey: John Wiley & Sons. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50163-Java_fordummies/144/> Páginas 135 a 153

* Capítulo 1: Introducción a Java

Vegas Gertrudix, J. M. (2022). "Java 17: Fundamentos prácticos de programación”. Bogotá: Ediciones de la U. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50229-Java_17/46/> Páginas 50 a 59

# Apuntes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.