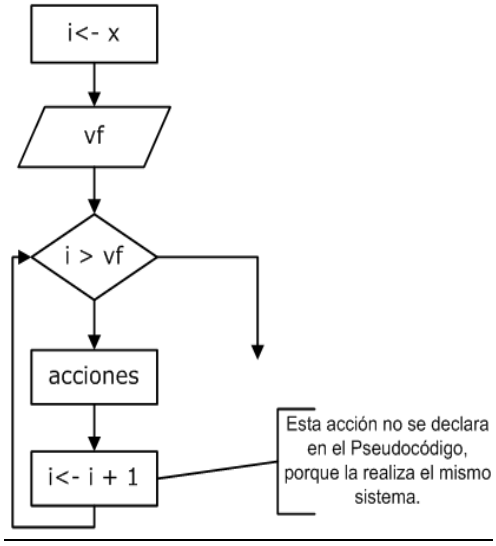
ARQUITECTURA DE
PLATAFORMAS Y SERVICIOS DE TI

III SEMESTRE



Puntos a tratar:

- Estructuras repetitivas,
- Desde/Para (for)
- Desarrollo de ejercicios de aplicación de la estructura for.

Objetivo de la sesión:

Al concluir la sesión el estudiante estará en la capacidad de implementar la estructura repetitiva for.

ESTRUCTURAS REPETITIVAS - for

INSTITUTO DE EDUCACIÓN SUPERIOR TECNOLÓGICO PÚBLICO "CHINCHA"

TEMA 05

UD FUNDAMENTOS DE PROGRAMACIÓN

ING. BARRIOS QUISPE, RICHARD JHONSON
Ingeniero de sistemas

2025

Tema 05:

ESTRUCTURAS REPETITIVAS - for

1. ESTRUCTURAS REPETITIVAS

Las computadoras están especialmente diseñadas para todas aquellas aplicaciones en las cuales una operación o conjunto de ellas deben repetirse muchas veces.

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan ***bucles***, y se llama ***iteración*** al hecho de repetir la ejecución de una secuencia de acciones.

Un ejemplo aclarará la cuestión: Supongamos que se desea sumar una lista de números escritos desde el teclado. El medio conocido hasta ahora es leer los números y añadir sus valores a una variable SUMA que contenga las sucesivas sumas parciales. La variable SUMA se hace igual a cero y a continuación se incrementa el valor del número cada vez que uno de ellos se lea. El algoritmo que resuelve este problema es:

```
Algoritmo  Suma
var
    entero:  Suma, número.
inicio
    SUMA ← 0
    leer (numero)
    SUMA  ←  SUMA  + numero
    leer (numero)
    SUMA  ←  SUMA  + numero
    leer (numero)
    SUMA  ←  SUMA  + numero
fin
```

y así sucesivamente para cada número de la lista. En otras palabras, el algoritmo repite muchas veces las acciones:

```
    leer (numero)
    SUMA  ←  SUMA  + numero
```

Tales opciones repetitivas se denominan ***bucles o lazos.*** La acción (o acciones) que se repiten en un bucle se denominan ***iteración.*** Las dos principales preguntas a realizarse en el diseño de un bucle son: ¿qué contiene el bucle? Y ¿cuántas veces se puede repetir?

Cuando se utiliza un bucle para sumar una lista de números, se necesita saber cuántos números se han de sumar. Para ello necesitaremos conocer algún medio para *detener* el bucle. En el ejemplo siguiente usaremos la técnica de solicitar al usuario el número que desea, por ejemplo, **"N"**.

```
Algoritmo  Suma_numero
var
        entero: N, TOTAL
        real: NUMERO, SUMA
inicio
        leer (N)
        TOTAL  ←  N
        SUMA  ←  0
```

```
        mientras  TOTAL  >  0 hacer
                leer (NUMERO)
                SUMA  ←  SUMA  +  NUMERO
                TOTAL  ←  TOTAL  -  1
        fin_mientras
        escribir ('la  suma de los ', N, 'números es', SUMA)
fin
```

La condición de salida se puede indicar en el interior del bucle, pero lo normal es que la condición se indique al final o al principio del bucle, y así se consideran tres tipos de instrucciones o estructuras repetitivas o iterativas generales y una particular que denominaremos **iterar**, que contiene la salida en el interior del bucle.

Iterar	(loop) ()
Mientras	(while)
Repetir	(repeat)
Desde (para)	(for)

Los tres casos de estructuras repetitivas dependen de la situación y modo de la condición. La condición se evalúa tan pronto se encuentra en el algoritmo y su resultado producirá los tres tipos de estructuras citadas.

1. La condición de salida del bucle se realiza al principio del bucle **(estructura mientras)**

```
Algoritmo SUMA 1
var
        entero: n
        real: S
Inicio
        // inicializar K,S a cero
        K ← 0
        S ← 0
        Leer (n)
        Mientras  K < n hacer
                K ← K + 1
                S ← S + K
        Fin_mientras
        Escribir (S)
Fin
```

2. La condición de salida se origina al final del bucle, el bucle se ejecuta *hasta que* se verifica una cierta condición **(estructura repetir)**

```
Repetir
        K ← K + 1
        S ← S + K
Hasta_que K > n
```

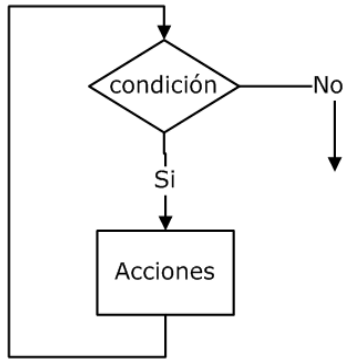
3. La condición de salida se realiza con un contador que cuenta el número de iteraciones **(estructura desde - para)**

```
Desde i = Vi hasta Vf hacer
        S ← S + i
Fin_desde
```

SINTAXIS

Sintaxis de la estructura repetitiva Mientras ("while")

Diagrama de Flujo



Pseudocódigo en castellano

mientras condición **hacer**
acción s1
.
.
acción sn
fin_mientras

Sintaxis en Java (while)

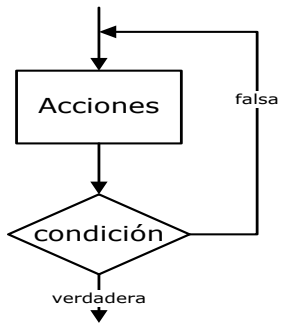
```
while (Condición) {
    <instrucciones>;
}
```

Sintaxis en Java (do/while)

```
do{
    <instrucciones>;
} while (Condición);
```

Sintaxis de la estructura repetitiva Repetir ("Repeat")

Diagrama de Flujo



Pseudocódigo en castellano

repetir
acción s1
acción s2
.
.
acción sn
hasta_que <condición>

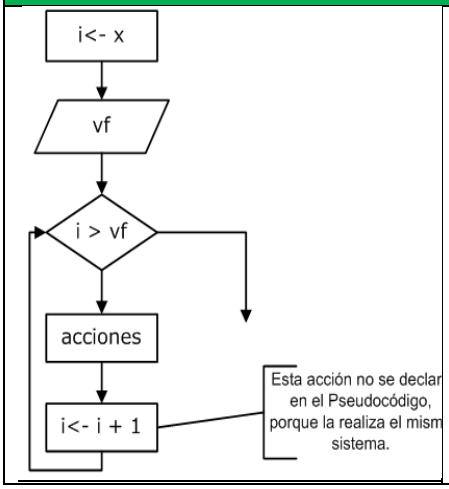
Pseudocódigo en inglés 1

repeat
<instruccion1>
<instruccionN>
until <condición>

Pseudocódigo en inglés 2

do until <condición>
<instrucciones>
end do

Sintaxis de la estructura repetitiva Desde ("for")



Pseudocódigo en castellano

desde i=x hasta y
<inc/dec> z hacer
<instrucciones> ;
fin_desde

Sintaxis en Java

```
for (i=vi; i<=vf; i++) {
    <instruccion1>;
        .
        .
    <instruccionN>;
}
```

2. ESTRUCTURA REPETITIVA for.

En Java, la estructura repetitiva for permite ejecutar un bloque de código un número determinado de veces. Su sintaxis básica consta de tres partes: la inicialización, que se ejecuta una sola vez antes de que comience el bucle; la condición, que se evalúa antes de cada iteración para determinar si el bucle continúa o se detiene; y la actualización, que modifica la variable de control después de cada repetición.
Por ejemplo, el siguiente código:

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Iteración " + i);
}
```

imprimirá los números del 1 al 5, ya que el bucle inicia con i = 1, se ejecuta mientras i <= 5, e incrementa i en cada iteración. La estructura **for** es ideal cuando se conoce de antemano la cantidad de repeticiones necesarias.

3. DESARROLLO DE EJERCICIOS DE APLICACIÓN DE LA ESTRUCTURA CONDICIONAL MÚLTIPLE.

EJER001: Obtener la suma de los primeros N números naturales positivos.

CODIFICACIÓN:

```
public class Suma_N {
    public static void main(String[] args) {
        //Variables
        int i,n,s = 0;
        //Entrada
        Scanner teclado = new Scanner(System.in);
        System.out.print("Numero: ");
        n = teclado.nextInt();
       //Proceso
          i = 1;
          while(i <= n){
                s = s + i;
             i = i + 1;
          }

        //Salida
        System.out.println("");
        System.out.println("Suma: " + s);
    }
}
```

EJER002: Dado un rango de números enteros, obtener la cantidad de números pares que contiene.

CODIFICACIÓN:

```
public class Busca_Pares {
    public static void main(String[] args) {
        //Variables
        int i,ni,nf,cp = 0;
        //Entrada
        Scanner teclado = new Scanner(System.in);
        System.out.print("Num. Incial: ");
        ni = teclado.nextInt();
        System.out.print("Num. Final: ");
        nf = teclado.nextInt();
        //Proceso
              i = ni + 1;
              while(i < nf){
                    if(i % 2 == 0){
```

```
                        cp += 1;
                    }
                i++;
            }
        //Salida
        System.out.println("");
        System.out.println("Cant. Pares: " + cp);
    }
}
```

EJER003: Obtener la suma de pares e impares de los primeros N números enteros positivos.

CODIFICACIÓN:

```
public class Suma_Pares_E_Impares {
    public static void main(String[] args) {
        //Variables
        int i,n,sp = 0, si = 0;
        //Entrada
        Scanner teclado = new Scanner(System.in);
        System.out.print("Numero: ");
        n = teclado.nextInt();
        //Proceso
            for(i = 1; i <= n; i += 2){
                        si += i;
            }
            for(i = 2; i <= n; i += 2){
                sp += i;
            }
        //Salida
        System.out.println("");
        System.out.println("Suma pares: " + sp);
        System.out.println("Suma impares: " + si);
    }
}
```

EJER004: Hallar cuantos múltiplos de M hay en un rango de números enteros.

CODIFICACIÓN:

```
public class Cuenta_Multiplos {
    public static void main(String[] args) {
        //Variables
        int ni,nf,nm,c=0,i;
        //Entrada
        Scanner teclado = new Scanner(System.in);
        System.out.print("Num. Inicial: ");
        ni = teclado.nextInt();
        System.out.print("Num. Final: ");
        nf = teclado.nextInt();
                System.out.print("Num. Multiplo: ");
        nm = teclado.nextInt();
        //Proceso
        for(i = ni; i<=nf; i++){
          if(i % nm == 0)
```

```
                    c += 1;
                }
            //Salida
            System.out.println("");
            System.out.println("Cantidad: " + c);
        }
    }
```

EJERCICIOS DE AUTOESTUDIO PARA EL SIGUIENTE TEMA:

1. Dado un rango de números enteros, obtener la cantidad de números pares que contiene.

2. Dado un número, devolver el dígito mayor.

3. Obtener la suma de pares e impares de los primeros N números enteros positivos.

4. Calcule la suma de los cuadrados y los cubos de los N primeros números naturales.

Material redactado con información proveniente de diversas fuentes.