



Sesión 4 – Semana 2

Estructuras de Repetición

W W W . M A K A I A . O R G

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



Contenido

- Introducción
- Estructura de repetición **para**
- Estructura de repetición **mientras**
- Estructuras de repetición anidadas
- Prueba de escritorio
- Variables usadas frecuentemente con las estructuras de repetición.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG



Introducción

- En los algoritmos desarrollados hasta el momento hemos utilizado variables, entrada y salida de datos, asignaciones, expresiones, estructuras secuenciales y estructuras de selección.

Sin embargo, muchos problemas requieren de características de repetición, en las que algunos cálculos o secuencias de instrucciones se repiten varias veces, utilizando diferentes conjuntos de datos.

- Las estructuras de repetición permiten que una secuencia de instrucciones se ejecuten varias veces mientras se cumpla una condición. Estudiaremos las estructuras de repetición **para** y **mientras**.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG



Introducción

- También son conocidas como ciclos o bucles.
- El ciclo **para** se utiliza cuando se conoce el numero exacto de veces que se debe ejecutar una secuencia de instrucciones.
- El uso mas común del ciclo **mientras** es cuando se desconoce el numero exacto de veces que se debe de ejecutar una secuencia de instrucciones.

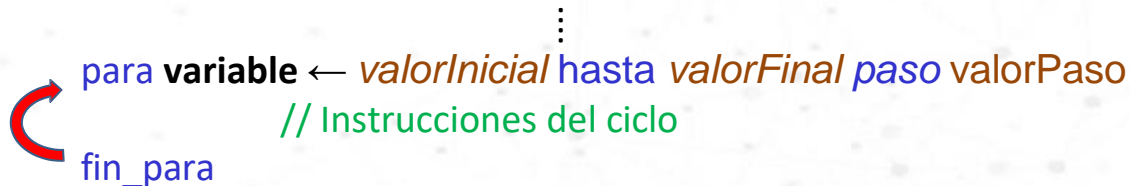
W W W . M A K A I A . O R G

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



W W W . M A K A I A . O R G

Sintaxis de la estructura de repetición **para** en pseudocódigo (1/2)




```
⋮  
para variable ← valorInicial hasta valorFinal paso valorPaso  
    // Instrucciones del ciclo  
fin_para  
⋮
```

Con el **para** se pueden presentar dos casos:

1. El **valorInicial** asignado a la **variable** controladora es menor que el **valorFinal**. Las instrucciones que están entre las palabras reservadas **para** y **fin_para** se ejecutan siempre y cuando el valor de la **variable** sea menor o igual que el **valorFinal**.

Una vez se llegue al **fin_para** la **variable** controladora se incrementa automáticamente en el valor que indique el **valorPaso** y se vuelve a verificar si la **variable** controladora es menor o igual que **valorFinal**. Si la **variable** controladora es mayor que **valorFinal** termina el **para** y se ejecutan las instrucciones que se encuentren después del **fin_para**. Cuando no se especifica el valor del paso se entiende que es de una unidad.



Sintaxis de la estructura de repetición **para** en pseudocódigo (2/2)

2. El **valorInicial** asignado a la **variable** controladora es mayor que el **valorFinal**. Las instrucciones que están entre las palabras reservadas **para** y **fin_para** se ejecutan siempre y cuando el valor de la **variable** sea mayor o igual que el **valorFinal**.

Una vez se llegue al **fin_para** la **variable** controladora se **decrementa** automáticamente en el valor que indique el **valorPaso** y se vuelve a verificar si la **variable** controladora es mayor o igual que **valorFinal**. Si la **variable** controladora es menor que **valorFinal** termina el **para** y se ejecutan las instrucciones que se encuentren después del **fin_para**. Para este caso el **valorPaso** debe ser expresado con un número negativo, de no hacer así el ciclo sería infinito.



Prueba de Escritorio

- Consiste en hacer un seguimiento de los valores que toman las variables de un algoritmo a medida que se siguen cada uno de los pasos que se establecen en el algoritmo.
- Para hacerla se disponen todas las variables del algoritmo en columnas diferentes y se listan, bajo sus nombres, los valores que toman a medida que se siguen los pasos especificados.
- Es útil para entender que hace un algoritmo o para verificar que un algoritmo sea correcto.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG



Variables usadas frecuentemente con las estructuras de repetición

- **Contador**: Variable de tipo **entero** usada para contar. Se incrementa (o disminuye) en un valor constante en cada iteración del ciclo.
- **Acumulador**: Variable que se usa para almacenar valores numéricos distintos que generalmente se suman (o multiplican) en cada iteración de un ciclo.
- **Bandera** o **centinela**: Variable de tipo **lógico** o **entero** utilizada en la condición del ciclo **mientras** para decidir si se itera o no. Es útil cuando no sabemos el numero exacto de veces que se debe iterar.

W W W . M A K A I A . O R G

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



W W W . M A K A I A . O R G


Ejemplo 1: Algoritmo para sumar los primeros n números enteros positivos

ALGORITMO

```
algoritmo SumarPrimerosNNumeros
variables
    entero: i, n, sumatoria
inicio
    muestre('Ingrese la cantidad de
    números a sumar: ')
    lea(n)
    sumatoria ← 0
    para i ← 1 hasta n paso 1
        sumatoria ← sumatoria + i
    fin_para
    muestre('La suma es: ', sumatoria)
fin
```

PROGRAMA

```
n = input('Ingrese la cantidad de
números a sumar: ');
sumatoria = 0;
for i = 1: 1: n
    sumatoria = sumatoria + i;
end
disp('La suma es: ')
disp(sumatoria)
```



Ejemplo 2: Prueba de escritorio para el algoritmo del ejemplo 1

<u>n</u>	<u>i</u>	<u>sumatoria</u>
5	1	1
	2	3
	3	6
	4	10
	5	15
6		


Ejemplo 3: Algoritmo para sumar n números ingresados por teclado.

ALGORITMO

```
algoritmo SumarNNumeros
variables
    entero: i, n
    real: numero, sumatoria
inicio
    muestre('Ingrese la cantidad de números a
    sumar: ')
    lea(n)
    sumatoria ← 0
    para i ← 1 hasta n
        muestre('Ingrese un numero: ')
        lea(numero)
        sumatoria ← sumatoria + numero
    fin_para
    muestre('La suma es: ', sumatoria)
fin
```

PROGRAMA

```
n = input('Ingrese la cantidad de
números a sumar: ');
sumatoria = 0;
for i = 1: n
    numero=input('Ingrese un numero: ');
    sumatoria = sumatoria +
    numero;
end
disp('La suma es: ')
disp(sumatoria)
```



Ejemplo 4: Algoritmo que calcula y muestra la raíz cuadrada de los números impares menores o iguales que un número n ingresado por teclado.

ALGORITMO

```
algoritmo RaizCuadradaDeImpares
variables
    entero: i, n
    real: r
inicio
    muestre('Ingrese un entero positivo: ')
    lea(n)


    para i ← 1 hasta n paso 2
        r ← raiz2(i)
        muestre('La raíz cuadrada de ', i, ' es ', r)
    fin_para
fin
```

PROGRAMA

```
n = input('Ingrese un entero positivo: ');

for i = 1 : 2 : n
    r = sqrt(i);
    fprintf('La raíz cuadrada de %d es %f \n', i, r)
end
```

WWW.MAKAIA.ORG



Ejemplo 5: Algoritmo que calcula y muestra la raíz cuadrada de los primeros n números pares en orden decreciente.

ALGORITMO

```
algoritmo RaizCuadradaDePares
variables
    entero: i, m, n
    real: r
inicio
    muestre('Ingrese un entero positivo: ')
    lea(n)
    m ← 2 * n
    para i ← m hasta 1 paso -2
        r ← raiz2(i)
        muestre('La raíz cuadrada de ', i, ' es ', r)
    fin_para
fin
```

PROGRAMA

```
n = input('Ingrese un entero positivo: ');

m = 2 * n;

for i = m : -2 : 1
    r = sqrt(i);
    fprintf('La raíz cuadrada de %d es %f \n', i, r)
end
```

WWW.MAKAIA.ORG

Ejemplo 6: Algoritmo para determinar si un numero es primo

ALGORITMO

```
algoritmo NumerosPrimos
variables
    entero: i, n, divisores
inicio
    muestre('Ingrese un entero: ')
    lea(n)
    divisores ← 0
    para i ← 1 hasta n
        si n mod i = 0 entonces
            divisores ← divisores + 1
        fin_si
    fin_para
    si divisores = 2 entonces
        muestre('El numero es primo')
    si_no
        muestre('El numero no es primo')
    fin_si
fin
```

PROGRAMA

```
n = input('Ingrese un entero: ');
divisores = 0;
for i = 1: n
    if mod(n, i) == 0
        divisores = divisores + 1;
    end
end
if divisores == 2
    disp('El numero es primo')
else
    disp('El numero no es primo')
end
```



Ejemplo 7

Escriba un algoritmo que lea las notas y los nombres de n estudiantes y muestre:

- a) La cantidad de estudiantes que ganaron.
- b) El porcentaje de estudiantes que perdieron.
- c) El nombre del estudiante con la nota mas alta.
- d) La nota más alta.
- e) La nota más baja.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG

Algoritmo para el ejemplo 7

algoritmo EstudiantesGanaron

variables

entero: i, n, ganaron

real: nota, menorNota, mayorNota,
porcentajePerdieron

cadena: nombre, nombreMayorNota

inicio

muestre('CALCULAR ESTUDIANTE S GANARON')

muestre('Ingrese la cantidad de estudiantes: ')
lea(n)

ganaron \leftarrow 0
mayorNota \leftarrow 0.0
menorNota \leftarrow 5.0
nombreMayorNota \leftarrow "

para i \leftarrow 1 hasta n

muestre('Ingrese el nombre del estudiante ', i, ': ')

lea(nombre)

muestre('Ingrese la nota: ')

lea(nota)

si nota \geq 3.0 entonces

ganaron \leftarrow ganaron + 1

fin_si

si nota > mayorNota entonces

mayorNota \leftarrow nota

nombreMayorNota \leftarrow nombre

fin_si

si nota < menorNota entonces

menorNota \leftarrow nota

fin_si

fin_para

porcentajePerdieron \leftarrow (n - ganaron) / n * 100

muestre('Ganaron: ', ganaron, ' estudiantes')

muestre('Porcentaje perdieron: ', porcentajePerdieron, '%')

muestre('Estudiante con mayor nota: ', nombreMayorNota)

muestre('Nota mayor: ', mayorNota)

muestre('Nota menor: ', menorNota)

fin


WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG

Sintaxis de la estructura de repetición **mientras** en pseudocódigo



```
⋮  
mientras expresión lógica  
    // Instrucciones del ciclo  
fin_mientras  
⋮
```

Si la **expresión lógica** evalúa a **verdadero** se ejecutan las instrucciones que están entre las palabras reservadas **mientras** y **fin_mientras**. Una vez ejecutadas las instrucciones se vuelve a evaluar la **expresión lógica**.

Dentro las instrucciones debe existir una que eventualmente haga que la **expresión lógica** evalúe a **falso**. Si la **expresión lógica** evalúa a **falso**, termina el **mientras** y se ejecutan las instrucciones que se encuentren después del **fin_mientras**.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG

Ejemplo 9: Algoritmo que suma los primeros n números enteros (Versión 2)

ALGORITMO

```
algoritmo SumarPrimerosNNumeros2
variables
    entero: i, n, s
inicio
    muestre('Ingrese la cantidad de
    números a sumar:')
    lea(n)
    i ← 1
    s ← 0
    mientras i ≤ n
        s ← s + i
        i ← i + 1
    fin_mientras
    muestre('La suma es: ', s)
fin
```

PROGRAMA

```
n = input('Ingrese la cantidad de
números a sumar: ');
i = 1;
s = 0;
while i <= n
    s = s + i;
    i = i + 1;
end
disp('La suma es:')
disp(s)
```

Ejemplo 10: Algoritmo que suma los números leídos hasta que se ingrese un numero menor o igual que cero

ALGORITMO

```
algoritmo SumarNumerosPositivos
variables
  real: numero, sumatoria
inicio
  muestre('Ingrese un numero: ')
  lea(numero)
  sumatoria ← 0
  mientras numero > 0
    sumatoria ← sumatoria + numero
    muestre('Ingrese un numero: ')
    lea(numero)
  fin_mientras
  muestre('La suma es: ', sumatoria)
fin
```

PROGRAMA

```
numero = input('Ingrese un numero: ');
sumatoria = 0;
while numero > 0
  sumatoria = sumatoria + numero;
  numero = input('Ingrese un numero: ');
end
disp('La suma es: ')
disp(sumatoria)
```

Ejemplo 11: Algoritmo que suma los números leídos hasta que se ingrese un carácter diferente a la s minúscula o mayúscula

ALGORITMO

```
algoritmo SumarNumeros
variables
    real: numero, sumatoria
    carácter: respuesta
inicio
    respuesta ← 's'
    sumatoria ← 0
    mientras respuesta = 's' o respuesta = 'S'
        muestre('Ingresa un número: ')
        lea(numero)
        sumatoria ← sumatoria + numero
        muestre('Desea sumar mas números?(s/n): ')
        lea(respuesta)
    fin_mientras
    muestre('La suma es: ', sumatoria)
fin
```

PROGRAMA

```
respuesta = 's';
sumatoria = 0;
while respuesta=='s' || respuesta=='S'
    numero = input('Ingresa un número: ');
    sumatoria = sumatoria + numero;
    respuesta = input('Desea sumar
mas números? (s/n): ', 's');
end
disp('La suma es: ')
disp(sumatoria)
```



Estructuras de repetición anidadas

Se forman escribiendo ciclos **para** o **mientras** dentro de ciclos **para** o **mientras**. Esta anidación se puede hacer tantas veces como sea necesario.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG

Ejemplo 12: Algoritmo que calcula y muestra las tablas de multiplicar hasta un numero entero n.

ALGORITMO

```
algoritmo TablasDeMultiplicar
variables
    entero: i, j, m, n
inicio
    muestre('TABLAS DE MULTIPLICAR')
    muestre('Ingrese un entero: ')
    lea(n)
    para i ← 1 hasta n
        muestre('Tabla del ', i)
        para j ← 1 hasta 10
            m ← i * j
            muestre(i, ' * ', j, ' = ', m)
        fin_para
    fin_para
fin
```

PROGRAMA

```
disp('TABLAS DE MULTIPLICAR');
n = input('Ingrese un entero: ');

for i = 1 : n
    fprintf('\nTabla del %d \n', i)
    for j = 1 : 10
        m = i * j;
        fprintf('%d * %d = %d\n', i, j, m)
    end
end
```

Ejemplo 13: Algoritmo que determina si un numero es primo (Mejorado)

ALGORITMO

```
algoritmo NumerosPrimosMejorado
variables
    entero: j, n
    logico: primo
inicio
    muestre('Ingrese un entero:')
    lea(n)
    j ← 2
    primo ← verdadero
    mientras primo y (j < n)
        si n mod j = 0 entonces
            primo ← falso
        si_no
            j ← j + 1
        fin_si
    fin_mientras
    si primo y (n > 1) entonces
        muestre('El numero es primo')
    si_no
        muestre('El numero no es primo')
    fin_si
fin
```

PROGRAMA

```
n = input('Ingrese un entero: ');
j = 2;
primo = true;
while primo && (j < n)
    if mod(n, j) == 0
        primo = false;
    else
        j = j + 1;
    end
end
if primo && (n > 1)
    disp('El numero es primo')
else
    disp('El numero no es primo')
end
```

Ejemplo 15: Calcula la sumatoria

$$\frac{1}{3} + \frac{4}{5} + \frac{9}{7} + \dots + \frac{n^2}{2n+1}$$

ALGORITMO

```
algoritmo CalcularSumatoria
variables
    entero: i, n
    real: termino, sumatoria
inicio
    muestre('Ingrese el numero de términos: ')
    lea(n)
    sumatoria ← 0

    para i ← 1 hasta n
        termino ← i * i / (2 * i + 1)
        sumatoria ← sumatoria + termino
    fin_para

    muestre('La sumatoria es: ', sumatoria)
fin
```

PROGRAMA

```
n = input('Ingrese el numero de términos: ');
sumatoria = 0;

for i = 1 : n
    termino = i * i / (2 * i + 1);
    sumatoria = sumatoria + termino;
end

disp('La sumatoria es: ')
disp(sumatoria)
```


Ejemplo 16: Algoritmo que calcula el factorial de un numero

ALGORITMO

```
algoritmo CalcularFactorial
variables
    entero: i, n, factorial
inicio
    muestre('Ingrese un entero no negativo: ')
    lea(n)

    si  $n \geq 0$  entonces
        factorial  $\leftarrow$  1
        para  $i \leftarrow 1$  hasta n
            factorial  $\leftarrow$  i * factorial
        fin_para

        muestre('El factorial de ', n, ' es ', factorial)
    si_no
        muestre('Factorial no definido para negativos')
    fin_si
fin
```

PROGRAMA

```
n = input('Ingrese un entero no negativo: ');
if n >= 0
    factorial = 1;
    for i = 1 : n
        factorial = i * factorial;
    end
    fprintf('El factorial de %d es %d \n ', n, factorial)
else
    disp('Factorial no definido para negativos')
end
```



Ejercicios

- Solucionar nuevamente los ejemplos propuestos.
- Hacer prueba de escritorio a cada uno de los ejemplos propuestos.
- Diseñar un algoritmo para mostrar los primeros n números impares que además sean múltiplos de tres.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG



Referencias

- Joyanes Aguilar, Luis. Fundamentos de Programación: Algoritmos, estructuras de datos y objetos, 4ª edición, Madrid: McGraw-Hill, 2008.

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia



WWW.MAKAIA.ORG



■ WWW.MAKAIA.ORG

Info: comunicaciones@makaiia.org

Corporación MAKAlA

Medellín, Colombia

Carrera 43A – 34-155.

Almacentro

Torre Norte, Oficina 701

Teléfono: (+574) 448 03 74

Móvil: (+57) 320 761 01 76



@makaiiaorg