

ALGORITMIA Y PROGRAMACIÓN

ESTRUCTURAS ITERATIVAS



Contenido

□ Estructuras de Repetición

- For
- Ciclos anidados
- Contadores y acumuladores
- While
- Ejercicios

Estructuras de Repetición

- **Ejercicio:** Suponga que debe mostrar los números del 1 a 100 en ventanas separadas en una aplicación.

Estructuras de Repetición

Ejercicio: Suponga que debe mostrar los números del 1 a 100 en Ruby.
Se podría hacer con la siguiente función:

```
def mostrarNumeros()

    puts ("El número es: " + 1)
    puts ("El número es: " + 2)
    puts ("El número es: " + 3)
    puts ("El número es: " + 4)

    .....

    puts ("El número es: " + 100)

end
```

Y si nos piden listar del 1 al 5000?

Estructuras de Repetición

Son un grupo de instrucciones que permite la ejecución repetitiva de otro grupo de instrucciones. Hay una variable asociada al **ciclo** o **estructura de repetición** que controla el número de veces que se repetirán las instrucciones.

Existen 2 estructuras de repetición:

for

while

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

1. Análisis del problema

Entradas: ??

Salidas: ??

Proceso:??

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

1. Análisis del problema

Entradas: n: Entero

Salidas: ??

Proceso:??

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

1. Análisis del problema

Entradas: n: Entero

Salidas:

imprimir (número 1)

imprimir (número 2)

.

.

imprimir (número n)

Proceso:??

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

1. Análisis del problema

Entradas: n:Entero

Salidas:

imprimir (número 1)

imprimir (número 2)

.

.

imprimir (número n)

Proceso:

desde 1 hasta n

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

1. Análisis del problema

Entradas: n:Entero

Salidas:

imprimir (número 1)

imprimir (número 2)

.

.

imprimir (número n)

Proceso:

desde 1 hasta n

Imprimir 1,2,3,.....n

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

2. Diseñar el algoritmo y escribirlo en pseudocódigo

Inicio

n: entero

n= leer (“Digite un número”)

???

Fin

Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

2. Diseñar el algoritmo y escribirlo en pseudocódigo

Inicio

n: entero

i: entero (Variable de control)

n = leer ("Digite un número")

Para i = 1 hasta n; incrementar i en 1

Haga

Imprimir (i)

Fin_para

Fin

Estructuras de Repetición

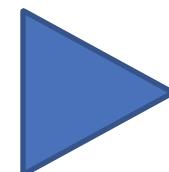
Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

3. Codificar el algoritmo usando algún lenguaje de programación

pseudocódigo

Ruby

Para i = 1 hasta n; incrementar i en 1
Haga
 Imprimir (i)
Fin_para



Estructuras de Repetición

Problema: Desarrollar un programa que solicite un número y muestre los números desde el 1 hasta el número solicitado.

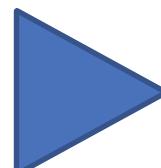
3. Codificar el algoritmo usando algún lenguaje de programación

pseudocódigo

Para i = 1 hasta n; incrementar i en 1
Haga
 Imprimir (i)
Fin_para

Ruby

**Estructura
De
Repetición
for**



Estructuras de Repetición

Se usa **para** repetir una instrucción o un conjunto de instrucciones,
Desde un inicio, **Mientras** una condición se cumpla y con un
incremento o decremento.

Un ciclo **for** tiene tres partes:

Expresión de inicio,
Expresión de condición y
Expresión de incremento/decremento.

Para i = 1 hasta n; incrementar i en 1

Haga

Imprimir (i)

Fin_para

Estructura de la instrucción for

Se debe declarar la variable de control

Iniciar la variable de control

probar el valor final de la variable de control para continuar el ciclo

```
for i in 0.step(100)  
    puts (i)  
end
```

Imprime los números desde **0 hasta 99**

Por defecto el incremento es de **1 en 1**

Estructura de la instrucción for

Se debe declarar la variable de control

Iniciar la variable de control

probar el valor final de la variable de control para continuar el ciclo

Incremento o decremento de la variable de control

```
for i in 0.step(100,2)
    puts (i)
end
```

En este caso el incremento será **de 2 en 2**.

Estructura for

Un ciclo for siempre debe especificar tres partes:

- Expresión de inicio
- Expresión de condición (o prueba)
- Expresión de incremento/decremento

```
for i in 0.step(n)
    puts (i)
end
```

```
for i in 0.step(n,2)
    puts (i)
end
```

Estructura for

Variable de control: Es la variable que se utiliza para contar la cantidad de iteraciones realizadas y **es usada en la condición** que determina el límite de repeticiones a realizar.

Ejemplos:

i=0 k=m j=1



```
for i in 0.step(n)
    puts (i)
end
```

i, k , j en cada caso será la variable de control del ciclo.

Estructura for

```
def ejemplo1()
    n=100
    for i in 0.step(n+1,1)
        puts (i)
    end
end
```

101

0
1
2
3
...
101

Qué imprimen estas funciones?

```
def ejemplo2()
    n=0
    for i in 100.step((n-1), -1)
        puts (i)
    end
end
```

[n-1, 0] -1

100
99
98
97
...
1

Estructura for

□ Ejemplo:

```
def listarNumeros()
    for i in 0.step(5)
        puts("Número #{i}")
    end
end
```

Valor de i	Número Iteración	Salida
0	1	"Número 0"
1	2	"Número 1"
2	3	"Número 2"
3	4	"Número 3"
4	5	"Número 4"
5	6	"Número 5"
6	Se interrumpe	

El número de iteraciones indica la cantidad de veces que se repite la(s) instrucción(es) que estén en el cuerpo del ciclo.

En el ejemplo, la instrucción: **puts** ("Número #{i}") se ejecuta 6 veces.

Estructura for

- Ejemplo:

```
def cicloPrueba()
    for i in 18.step(12, -2)
        puts(i)
    end
end
```

Inic' o Términ.o

INC/DEC

18
16
14
12

- Cuántas veces se ejecutará la instrucción: **puts (i)**
- Qué valores imprime?

Estructura for

- Ejemplo:

```
def cicloPrueba()
    for i in 18.step(12, -2)
        puts(i)
    end
end
```

Valor de i	Número Iteración	Salida
18	1	18
16	2	16
14	3	14
12	4	12
10	Se interrumpe	

La instrucción:
puts (i)
se ejecuta 4 veces

Estructura for

- Ejemplo:

```
def muestraAlgo()
    for i in 0.step(20)
        if (i%2 == 0)
            puts ("El valor de i es #{i}")
        end
    end
end
```

Inicio Término Final = 1

0
2
4
6
8
10
12
14
16
18
20

||
Veces

- Cuantas veces se ejecutará la instrucción:

```
puts ("El valor de i es #{i}")
```

Estructura for

- Ejemplo:

```
def muestraAlgo2()
    for j in 0.step(101)
        if (j % 3) == 0
            puts ("El valor de j es #{j}")
        else
            puts ("#{j} NO es ...")
        end
    end
end
```

Divisible por 3

- Que debo completar en el mensaje “NO es ...” ?

```
puts ("#{j} NO es ...")
```

Estructura for

Una vez el ciclo se interrumpe se ejecuta la instrucción ubicada después de él. Por ejemplo:

```
a=0
```

```
for k in 0.step(5;2)
```

```
    a+=1
```

```
puts ("El valor de a es: #{a}")
```

$$a = a + 1$$



Estructura for

Una vez el ciclo se interrumpe se ejecuta la instrucción ubicada después de él. Por ejemplo:

```
a=0  
  
for k in 0.step(5,2)  
    a+=1  
end  
puts ("El valor de a es: #{a}")
```

- La variable **a** inicia con el valor de **cero** (0).
- Se ingresa al ciclo y se ejecuta tres (3) veces la instrucción **a+=1**
- Luego se muestra en pantalla el valor de **a**, es decir, 3.

Estructura for

□ Ejercicio 1:

Desarrollar un programa Ruby que pregunte al usuario el número de estudiantes de un curso, luego pregunte el nombre de cada uno de ellos. Finalmente, se debe mostrar un listado con todos los estudiantes.

```
>>>
Digite la cantidad de estudiantes: 3
Digite el nombre del estudiante 1: Maria
Digite el nombre del estudiante 2: Pablo
Digite el nombre del estudiante 3: Carlos
Los nombres de los 3 estudiantes son:
Maria
Pablo
Carlos

>>>
```

Estructuras de Repetición

Problema: Desarrollar un programa Ruby que pregunte al usuario el número de estudiantes de un curso, luego pregunte el nombre de cada uno de ellos. Finalmente, se debe mostrar un listado con todos los estudiantes.

1. Análisis del problema

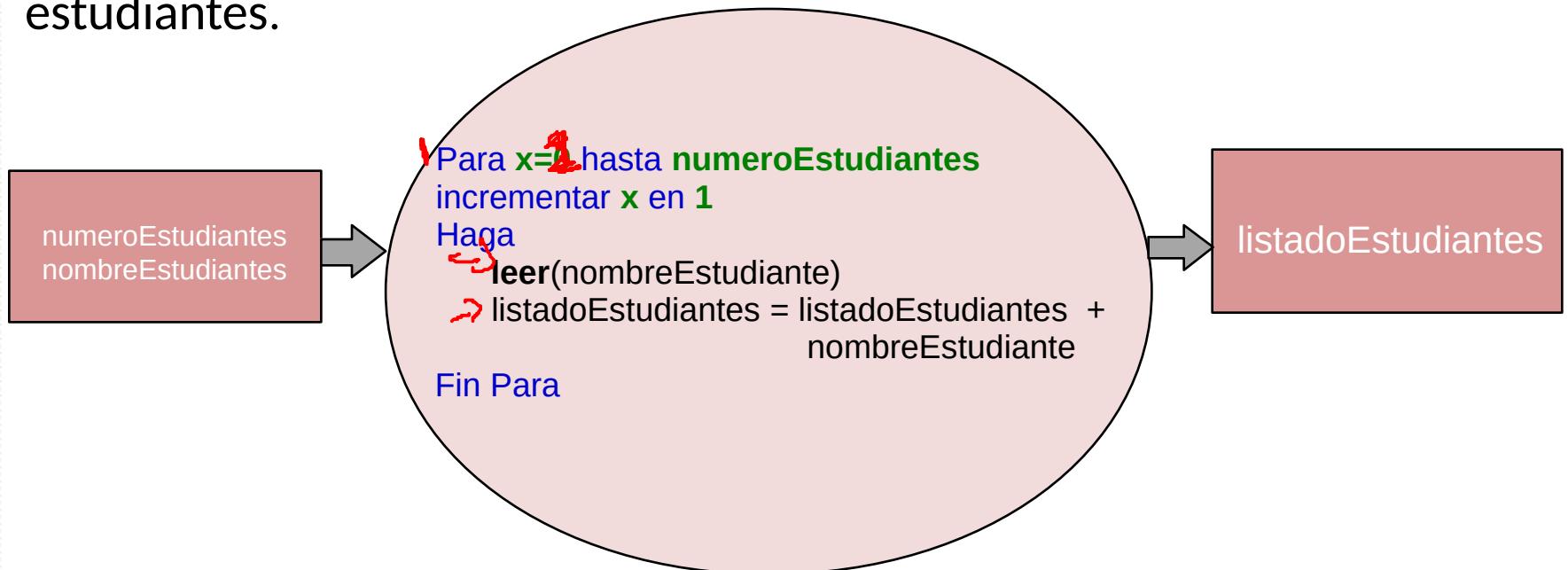
Entradas: numeroEstudiantes, nombreEstudiante,
Salidas: listadoEstudiantes

Proceso:??

Entrada Cadrado de texto

Estructuras de Repetición

Problema: Desarrollar un programa Ruby que pregunte al usuario el número de estudiantes de un curso, luego pregunte el nombre de cada uno de ellos. Finalmente, se debe mostrar un listado con todos los estudiantes.



Estructuras de Repetición

Problema: Desarrollar un programa Ruby que pregunte al usuario el número de estudiantes de un curso, luego pregunte el nombre de cada uno de ellos. Finalmente, se debe mostrar un listado con todos los estudiantes.

1. Análisis del problema

Entradas: numeroEstudiantes, nombreEstudiante,
Salidas: listadoEstudiantes

Proceso:

```
Para x=1 hasta numeroEstudiantes incrementar x en 1
Haga
leer(nombreEstudiante)
listadoEstudiantes = listadoEstudiantes + nombreEstudiante
Fin Para
```

Estructura for

□ Ejercicio 1:

Inicio

numeroEstudiantes: entero
listadoEstudiantes: texto
nombreEstudiante: texto

leer(numeroEstudiantes)

Para **x=0** hasta **numeroEstudiantes** incrementar **x** en **1**
Haga

leer(nombreEstudiante)

listadoEstudiantes = listadoEstudiantes + nombreEstudiante

Fin Para

imprimir(listadoEstudiantes)

Fin

Estructura for - Ciclos Anidados

□ Ejemplo:

Suponga que se desea crear un programa en Ruby que permita imprimir en pantalla el factorial de cada número existente en la serie 1 a n, siendo n un número menor a 20 el cual es digitado por un usuario.

Para i=1 hasta n+1 incrementar i en 1

Haga

 Para j=1 hasta i+1 incrementar j en 1

 Haga

 instrucción 1

 instrucción 2

 ...

 Fin Para

 instrucción 4

Fin Para

NOTA: Usamos la condición $n \leq 20$ con fines académicos y evitar resultados con números exageradamente grandes.

Estructura for - Ciclos Anidados

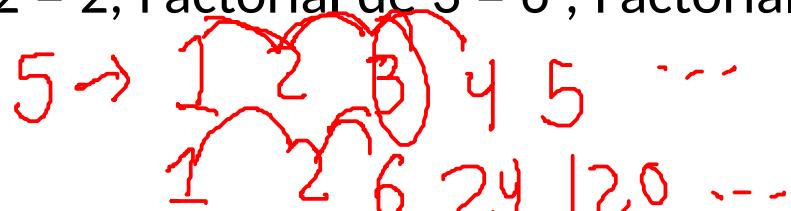
Un **ciclo anidado** es un ciclo, de cualquier tipo, que contiene otro ciclo de cualquier tipo.

En el ejercicio propuesto se observa que se debe crear un ciclo para manejar la serie de 1 a n, y un ciclo anidado para calcular el factorial de n_i .

Por ejemplo: si el usuario digita el número 5 el sistema debe recorrer la serie 1 a 5, y para cada número devolver su factorial, así:

Factorial de 1 = 1, Factorial de 2 = 2, Factorial de 3 = 6 , Factorial de 4 = 24 y el Factorial de 5 = 120.

Nota:



El factorial de un número n es la multiplicación de todos los números de la serie 1 a n, entonces, el factorial de 5 es $1 \times 2 \times 3 \times 4 \times 5 = 120$, el factorial de 3 es $1 \times 2 \times 3 = 6$.

Estructura for - Ciclos Anidados

Inicio

factorial, n: entero
respuesta: texto

leer(n)

SI n <= 20 **Entonces**

Desde i=1 hasta n incrementar i en 1

Haga

factorial=1

Desde j=1 hasta i-1 incrementar j en 1

Haga

factorial = factorial*j

Fin Para

resposta = respuesta + "El factorial de " + j + "es: " factorial;

Fin Desde

imprima(respuesta)

SINO

imprima("Ha digitado un numero superior a 20")

Fin

Estructura for - Ciclos Anidados

Inicio

factorial,n: entero
respuesta: texto

leer(n)

SI n<= 20

Para i=1 hasta n
Haga

factorial=1

Para j=1 hasta i
Haga

factorial = factorial*j

Fin Para

 respuesta = respuesta + "El factorial de " + j + "es: " factorial;

Fin Para

imprima(respuesta)

SINO

imprima("Ha digitado un numero superior a 20")

Fin

El ciclo principal, basado en la variable de control **i**, contiene un ciclo anidado basado en la variable de control **j**.

Estructura for - Ciclos Anidados

```
def listaFactorial(n)
  if n<=20
    for i in 1.step(n)
      fac = 1
      for j in 1.step(i)
        fac *= j
      end
      puts "El factorial de #{i} es: #{fac}"
    end
  else
    puts "Ha digitado un número superior a 20"
  end
end
```

$F_i = F_{i-1} \times j$

listaFactorial(10)

i = 1.step(10)

j = 1

F_{0C} = 1

j = 1.step(1)

j = 1

F_{0C} = 1 * 1

```
def listaFactorial(n)
    if n<=20
        for i in 1.step(n)
            fac = 1
            for j in 1.step(i)
                fac *= j
            end
            puts "El factorial de #{i} es: #{fac}"
        end
    else
        puts "Ha digitado un número superior a 20"
    end
```

i = 2

F_{0C} = 2

j = 1 hasta {

j = 2

F_{0C} = 1 * 2 }

j = 2

F_{0C} = 1 * 2 }

} 2

i = 3

F_{0C} = 1

j = 1 hasta 3

j = 2

F_{0C} = 1 * 1

j = 2

F_{0C} = 1 * 2

j = 3 F_{0C} = 2 * 3 = 6

```
def listaFactorial(n)
    if n<=20
        for i in 1.step(n)
            fac = 1
            for j in 1.step(i)
                fac *= j
            end
            puts "El factorial de #{i} es: #{fac}"
        end
    else
        puts "Ha digitado un número superior a 20"
    end
```

Estructura For - Acumuladores

```
def listaFactorial(n)
  if n<=20
    for i in 1.step(n)
      fac = 1
      for j in 1.step(i)
        fac *= j
      end
      puts "El factorial de #{i} es: #{fac}"
    end
  else
    puts "Ha digitado un número superior a 20"
  end
end
```

La variable **factorial** va acumulando el valor de la multiplicación y la variable **respuesta** va acumulando las cadenas usadas como respuesta en cada iteración

de cada e

Estructura for - Ciclos Anidados

- Ejercicio 2:

Desarrollar un programa Ruby que permita generar y visualizar la siguiente figura:

```
>>> imprimirLineas(7)
*
**
***
****
*****
******
*****
```

- Tenga en cuenta que el usuario ingresa al programa el número de líneas que debe tener la figura (En el ejemplo hay 7 filas).

Estructura for - Ciclos Anidados

□ Ejercicio 2:

```
imprimirLineas(numeroFilas : entero)
```

Inicio

lineas = “” : texto

Para i=1 hasta numeroFilas ~~do~~ incrementar i en 1

Haga

 Para j=1 hasta i ~~do~~ incrementar j en 1

 Haga

 lineas = lineas + “*”

 Fin Para

 lineas = lineas + “\n”

 Fin Para

imprimir(lineas)

Fin

Estructura for - Ciclos Anidados

- Ejercicio 2:

```
def imprimirLineas(n)
    lineas = ""
    for i in 1.step(n)
        for j in 1.step(i)
            #Esto es lo mismo que escribir lineas = lineas + "*"
            lineas += "*"
        end
        lineas += "\n"
    end
    puts lineas
end
```

Estructura for - Contadores

Los contadores son variables utilizadas para realizar, como su nombre lo indica, conteos de la cantidad de veces que se cumple una situación específica.

Como su objetivo principal es contar, deben ser de tipo entero y normalmente se inicializan en cero.

Los contadores y los acumuladores
pueden ser usados en cualquier
tipo de ciclos.

Estructura for - Contadores

□ Ejemplo:

Suponga que se desea crear un programa en Ruby que permita imprimir en pantalla la cantidad de números múltiplos de 3 que se encuentran en la serie 1 a n, siendo n un número digitado por un usuario.

Estructura for - Contadores

□ Ejemplo:

Suponga que se desea crear un programa en Ruby que permita imprimir en pantalla la cantidad de números múltiplos de 3 que se encuentran en la serie 1 a n, siendo n un número digitado por un usuario.

Inicio

```
contador: entero;  
contador=0;  
leer(n)  
Para i=1 hasta n+1 incrementar i en 1  
Haga  
    if ((i%3)==0){  
        contador=contador+1;  
    }  
Fin Para  
imprimir(contador);
```

Fin

Estructura for - Contadores

□ Ejemplo:

Suponga que se desea crear un programa en Ruby que permita imprimir en pantalla la cantidad de números múltiplos de 3 que se encuentran en la serie 1 a n, siendo n un número digitado por un usuario.

Inicio

```
contador: entero;  
contador=0;  
leer(n)  
Para i=1 hasta n+1 incrementar i en 1  
Haga  
    if (esMultiploDeTres(i)){  
        contador=contador+1;  
    }  
Fin Para  
imprimir(contador)  
Fin
```

Estructura for - Contadores

□ Ejemplo:

```
def multiploDeTres(n)

    contador = 0

    #Y si, podemos definir funciones dentro de funciones
    def esMultiploDeTres(a)
        if a%3 == 0
            return true
        else
            return false
        end
    end

    for i in 1.step(n)
        if(esMultiploDeTres(i)) ←
            contador+=1
        end
    end

    return contador
end
```

Estructura for - Contadores

□ Ejemplo:

```
def multiploDeTres(n)

    contador = 0

    #Y sí, podemos definir funciones dentro de funciones
    def esMultiploDeTres(a)
        if a%3 == 0
            return true
        else
            return false
        end
    end

    for i in 1.step(n)
        if(esMultiploDeTres(i))
            contador+=1
        end
    end

    return contador
end
```

Contador

Si *esMultiploDeTres* es true

Estructura for - Acumuladores

Los acumuladores son variables utilizadas para ir almacenando (acumulando) el resultado de una operación.

Pueden ser de tipo numérico (entero o real) en las cuales acumula el resultado de operaciones matemáticas, o de tipo cadena en las cuales se concatenan frases o palabras.

En el ejemplo del factorial se pueden observar las variables **factorial** y **respuesta** las cuales actúan como acumuladores numéricos y de cadena respectivamente.

Ejercicios Estructura for

1. Diseñe un algoritmo que permita detectar los números pares existentes en una serie de 1 a n, siendo n un número digitado por un usuario.
2. Diseñe un algoritmo que permita obtener la suma de todos los números enteros existentes en una serie de 1 a n y la cantidad de números pares encontrados, siendo n un número digitado por un usuario. Use un ciclo **for** en su diseño.
3. Suponga que se desea saber la nota promedio del curso de algoritmia, diseñe un algoritmo que solicite la cantidad de estudiantes del curso y el promedio de cada estudiante.

NOTA: Para cada ejercicio realice su respectiva implementación en **Ruby**

Ejercicios Estructura for

4. Suponga que el calculo de la pensión de una persona se realiza de la siguiente manera: por cada año de servicio se paga \$80 si el empleado ingresó en o después de 1995 y \$100 si ingresó antes, dicho valor (80 o 100) se multiplica por el número de cada año más la edad que tenía en el año (ej. $(100 * 1994 + 32) + (100 * 1995 + 33) + \dots$), el descuento de seguridad social en salud es del 12%. El programa debe recibir el año de ingreso y la edad del empleado en el año de ingreso, devolver el sueldo o mesada bruta, la mesada neta y el valor del descuento por salud.

Ejemplo: Para una persona que ingresó en el 2009 y que tenía 44 años en dicho año, su mesada o sueldo bruto para el 2011 es \$482.535, el descuento por salud es \$57.904 y por lo tanto su sueldo o mesada neta es \$424.630.

NOTA: Realice la respectiva implementación en Ruby.

Ejercicios Estructura for

5. Crear una aplicación que permita:

- Generar los números enteros pares entre **p** y **q**
- Generar los números enteros impares entre **a** y **b**
- Generar los primeros **z** múltiplos de **3**
- Generar la suma de **m** primeros múltiplos de **7** más los **n** primeros múltiplos de **9**

NOTA: Realice la respectiva implementación en Ruby.