Cap. III Estructuras Básicas de un Algoritmo

DEPARTAMENTO DE PROGRAMACIÓN DE COMPUTADORAS
DESARROLLO DE SOFTWARE I

PROF. MITZI MURILLO DE VELÁSQUEZ Msc.

Cap. III Estructuras Básicas de un Algoritmo

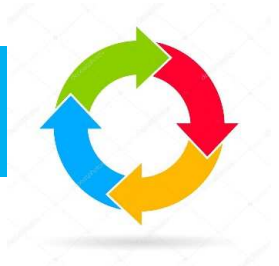
3.4 Estructuras Repetitivas

- **Contador / Acumulador**
- **Mientras**
- **Hasta que**
- **Para**

3.4.5 Ciclos definidos (contador) e indefinidos (centinela/bandera y respuesta por el usuario)

3.5 Resolver problemas y aplicar pruebas de escritorio

Estructuras de de Repetición



Son aquellas que controlan la repetición de un conjunto de instrucciones mediante la evaluación de una condición.

Las estructuras de repetición se presentan en tres formas:

- Estructura **Mientras**
- Estructura **para**
- Estructura **Hasta que**

Toda estructura de repetición está compuesta por los siguientes elementos:

1. **Condición** : expresión a evaluar (determina la entrada o no al ciclo).
2. **Bloque de instrucciones:** conjunto de instrucciones que deben ejecutarse.
3. **Ciclo o iteración** : proceso de ejecución del ciclo un número de veces.

Estructuras de de Repetición

Existen dos procedimientos básicos para controlar las iteraciones que deben realizar:

Ciclos Definidos y Ciclos Indefinidos.

- **Ciclos Definidos**, son aquellos donde se conoce con anticipación el número de veces que se ejecutará el proceso, por ejemplo **: calcule el promedio de las notas de 3 parciales.**
- **Ciclos Indefinidos**, son aquellos donde no se conoce con anticipación cuántas veces se ejecutará el proceso, por ejemplo : **calcular las ventas del supermercado en el dia de hoy?**

En ambos casos es importante controlar a través de una condición las iteraciones que se deben realizar para saber cuándo finaliza el ciclo. Esto evitará que existan ciclos infinitos.

Los ciclos definidos se controlan usando **contadores.**

Los ciclos indefinidos se controlan usando :

- **respuesta del usuario**
- **Valor centinela**

También es importante mencionar que existen 2 variables que permiten controlar y manejar valores dentro de un ciclo. Estas son : el contador y el acumulador.

Estructuras de de Repetición

1. Contador : Es una **variable** puede ser incrementada o decrementada en un valor constante de tipo entero.

Formatos:

Variable contador = variable contador + [constante numérica] ;

Variable contador = variable contador - [constante numérica];

Variable contador : variable de tipo entero

Constante numérica : **constante entera** que incrementa o decrementa a la variable contador

EJEMPLOS:

cont = cont + 1 ;

triple = triple + 3 ;

poblac = poblac - 5 ;

lista = lista – 1;

Estructuras de de Repetición

Para qué usamos un contador ?

- **contar las veces que se ejecutan los ciclos un número constante de veces,** es decir, contar repeticiones
- **contar** eventos
- Sirve de **variable de control** en las estructuras de repetición

Ejemplo 1: contar cuántos estudiantes se conectaron a Teams hoy ?

```
estud = estud + 1 ;
```

Ejemplo 2 :
Contar de 2 en 2 hasta 20

```
par = par + 2 ;
```

Ejemplo 3 :
contar el número de monedas de una alcancía

```
monedas = monedas + 1 ;
```

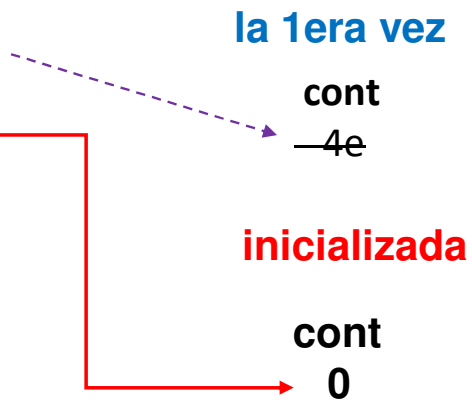
Estructuras de de Repetición

Para trabajar con un contador, debemos asegurarnos de que el mismo sea **inicializado**.

Qué significa inicializar ? **significa asignar un valor inicial a una variable.**

Ejemplo 1:

```
entero cont ;
------
cont = 0;
```



Estructuras de de Repetición

Reglas para crear una variable contador :

1. **declarar** la variable contador
2. **inicializar el contador** con un valor que puede ser 0 ó cualquier otro valor.
   ejemplo : entero cont; cont = 0;
3. **incrementar/decrementar** con valor constante
   cont = cont + **1**;

Ejemplos :

1. **entero cont;**
2. **cont = 0 ;**
3. **cont = cont + 2;**

1. **entero con;**
2. **con = 5;**
3. **con = cont – 1;**

Estructuras de Repetición

2. Acumulador: es una variable cuya función es almacenar cantidades **variables**, resultantes de sumas sucesivas.

Formato :

variable acumulador = variable acumulador + variable;

Ejemplos: totedad = totedad + edad; **activo = activo – saldo;**

totlibros = totlibros + libros;

Al igual que la variable contador, una variable acumuladora debe ser inicializada.

Los acumuladores trabajan con datos de tipo entero o de tipo flotante

Estructuras de de Repetición

Ejemplo: calcular el peso total de varias reses.

```
flotante peso;
flotante  ptotal;
ptotal = 0.0;
------
leer(peso);
ptotal = ptotal + peso;
```

Cuánto dinero se ha obtenido de la venta de boletos de hoy ?

```
flotante dinero;
flotante   ventab;
 ventab  = 0.0;
-----
ventab = ventab+ dinero ;
```

Calcular cuántos pedidos de libros realizó la librería en este mes ?

```
entero  libros;
 entero  totlibro;
 totlibro  = 0;
-----
 totlibro = totlibro + libros ;
```

Estructuras de de Repetición

a. Ciclos Definidos: Son aquellos donde se conoce el número de veces que es necesario ejecutarlo.

a.1 **Ciclo controlado por Contador**

Requiere :

- nombre de la variable de control o contador
- valor inicial de la variable de control
- incremento o decremento
- establecer la condición que compruebe el valor final de la variable de control.

Estructuras de Repetición

b. Ciclos Indefinidos :

Son aquellos donde no se conoce con anticipación el número de veces que se ejecutará el ciclo.

b.1 **Ciclo controlado por respuesta** : se utiliza en aquellas ocasiones en que se le da al usuario la opción de terminar el ciclo.

Requiere:

- el nombre de la variable de control
- valor inicial de la variable de control
- **la condición que comprueba el valor final de la variable de control**
- Colocar dentro del bloque de instrucciones la solicitud de la respuesta del usuario, para la variable de control.

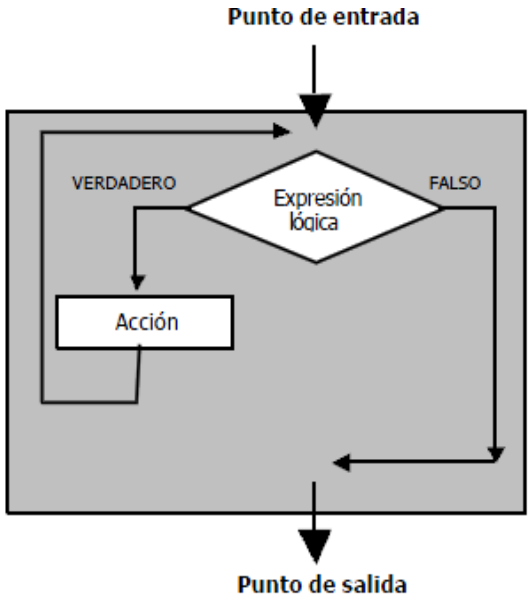
Estructuras de Repetición

1. mientras:

Permite ejecutar una sentencia, simple o compuesta, mientras la condición sea cierta.

Formato:

```
mientras(condición) {
      sentencia 1;
      sentencia 2;
      sentencia n;
 }
```



La ejecución sucede si :

- se evalúa la expresión

•Si la primera vez el resultado es cero (falso), no se ejecuta el ciclo ni una sola vez y se pasa a la sig. Instrucción.
•si el resultado es cierto, se ejecuta la sentencia dentro del mientras y se repite el proceso.

Estructuras de Repetición

Permite ejecutar una sentencia, simple o compuesta, mientras la condición sea cierta.

FORMATO: Mientras controlado con Contador

1. declarar variable de control
2. **Inicializar la variable de control**

mientras (condición) 3. condición

```
{   sentencia 1;
    sentencia 2;
     :
    sentencia n;
   variable_control  = incremento/decremento ;
 }
```

FORMATO: Mientras controlado con respuesta

1. declarar variable de control
2. Inicializar variable de control generalmente una variable de tipo carácter

mientras **(condición)** **3. Condición**

```
  {  sentencia 1;
     :
     sentencia n;
     imprimir("Mensaje al usuario");

     leer ( variable de control);    4.  leer respuesta
  }
```

Estructuras de Repetición

Permite ejecutar una sentencia, simple o compuesta, mientras la condición sea cierta.

FORMATO: Mientras controlado con valor centinela

1. declarar variable de control
2. Leer la variable de control

mientras **(condición)** **3. Condición**
{ sentencia 1;
:
sentencia n;

leer (variable de control**);** **4. leer respuesta**
}

Estructuras de Repetición

Mientras usando Contador

1. **declarar variable de control**
2. **Inicializar la variable de control**

mientras (condición) **3. condición**

```
{   sentencia 1;
    sentencia 2;
     :
    sentencia n;                          4. incremento
   variable_control  = incremento/decremento ;
 }
```

Considere:

1. declarar la variable que controla el ciclo
2. Inicialice la variable que controla ecl ciclo.
3. Establezca una condición que evalúa la entrada al ciclo siempre que sea cierta
   - Se definen las instrucciones a ejecutar

4. Dentro del bloque de instrucciones debe haber un incremento/decremento que permita que la condición finalice y no haya un ciclo infinito.
   - Cuando la codición sea falsa, se termina el ciclo y continuamos en la siguiente instrucción.

1. Mientras con Contador

Análisis y Diseño

Entrada	
Proceso	generar 5 números (cont) cont = cont + 1
Salida	cont

variables de memoria	U.A.L	
cont	cont < 5	**PANTALLA**
~~0~~	0 < 5 cierto	**n: 1**
~~1~~	1 < 5 cierto	**n : 2**
~~2~~	2 < 5 cierto	**n: 3**
~~3~~	3 < 5 cierto	**n: 4**
~~4~~	4 < 5 cierto	**n: 5**
5	5 < 5 **falso**	

Ejemplo 1: Imprimir los 5 primeros números

```
Algoritmo CincoNumeros {
   entero cont;  ← 1. declarar variable contador
    cont = 0;← 2. Inicializar
    mientras (cont < 5)  ← 3. condición
     {
        cont = cont + 1; ←  4. incremento
        imprimir("n :", cont);
      }
}
```

1. Mientras con Contador

Análisis y Diseño

Entrada	**x, y**
Proceso	Repetir 10 veces (cont) Calcular z = x * y Calcular tot = tot + z cont = cont + 1
Salida	z , **tot**

Definición del Problema :Elabore un algoritmo que lea 10 valores para x y y, calcule e imprima el valor de z, utilizando la función: Z = Y ^{x} X. Además sume el valor calculado.

```
Algoritmo FuncionArit{
// Area declarativa
  entero  cont , x , y,  z, tot;
// Cuerpo del algoritmo
  cont  = 1;    tot=0;← Inicializar
  mientras (cont <= 10)
        {
            leer(x, y);
            z  =   x * y ;
            tot = tot   +  z;
            cont =  cont + 1; ← incremento
            imprimir ( " Valor de Z : ", z);  }
    imprimir("La suma de los valores de z : ", tot);
}
```

1. Mientras con Contador

Elabore un algoritmo que lea 10 valores para x y y, calcule e imprima el valor de z, utilizando la función: $Z = Y^{x} X$. Además sume el valor calculado.

variables de memoria					UAL			
cont	x	y	z	tot			cont <= 3	PANTALLA
~~1~~	3	4	12	0 12	3 *4 12	0 + 12 12	1 < = 3 cierto	Lea x y y : 3 4 Valor de z = 12
~~2~~	4	5	20	32	4 * 5 20	12 + 20 32	2 < = 3 cierto	Lea x y y : 4 5 Valor de z = 20
~~3~~	2	~~9~~	18	50	2 * 9 18	32 + 18 50	3< = 3 cierto	Lea x y y : 2 9 Valor de z = 18
4							4 < =3 falso	La suma de los valores de z = 50

```
Algoritmo FuncionArit {
// Area declarativa
  entero cont , x , y, z, tot;
// Cuerpo del algoritmo
  cont = 1;   tot=0;← Inicializar
  mientras (cont <= 3)  ← condición
     {  imprimir ( " Lea x y y :")
           leer(x, y);
           z =  x * y ;
           tot = tot  +  z;
           cont = cont + 1; ← incremento
           imprimir ( " Valor de Z : ", z);
     }
    imprimir("La suma de los valores de z : ", tot);
}
```

Mientras controlado con Respuesta

Si no se conoce con anticipación cuántas veces que se ejecutará el ciclo., entonces necesitamos considerar darle al usuario la opción de controlar el ciclo ingresando su respuesta. Se revisará la solución usando ciclo controlado con respuesta.

Mientras controlado con Respuesta

1. Declarar variable de control

2. Inicializar variable de control

mientras **(condición)** **3. Condición**

```
{
  sentencia 1;
  :
  sentencia n;

  imprimir("Mensaje al usuario");

  leer ( variable de control);    4.  leer respuesta
}
```

Mientras controlado con respuesta

Definición del Problema :Leer un grupo de números y sumarlos.

Análisis y Diseño

Entrada	**respuesta (resp) ,números (num)**
Proceso	Repetir mientras respuesta sea cierta Calcular (suma) = suma + num
Salida	**suma**

```
Algoritmo Sumar
{
   entero num,  suma;
   caracter resp; ← 1. definición de variable de control
   suma=0;
   resp = 's' ;← 2. inicializar variable de control
  mientras( resp =='s' O  resp =='S')  ← 3. CONDICIÓN
   {  imprimir("Ingrese un número");
      leer(num);
      suma = suma + num;
      imprimir("Desea leer otro número? s/n");
      leer (resp) ;   ← 4.  leer respuesta
   }
 imprimir("La suma es :",suma);
}
```

1. Mientras controlado con Respuesta

Definición del Problema : Leer un grupo de números y sumarlos.

variables de memoria			UAL		
resp	num	suma		resp=='s' o resp == 'S	PANTALLA
~~s~~	7	~~0~~ 7	´ 0 + 7 7	s == s o s == S C o F = C	Ingrese un número: 7 Desea leer otro numero? s
~~s~~	19	26	7 + 19 26	s == s o s == S C o F = C	Ingrese un número:19 Desea leer otro numero? s
~~s~~	30	56	26 + 30 56	n == s o n == S F o F = F	Ingrese un número: 30 Desea leer otro numero? n
n					La suma es : 56

```
Algoritmo Sumar
{
   entero num,  suma;
   caracter resp; ← 1. definición de variable de control
   suma=0;
   resp = 's' ;← 2. inicializar variable de control
   mientras( resp =='s' o  resp =='S')  ← 3. condición
   {  imprimir("Ingrese un número");
      leer(num);
      suma = suma + num;
      imprimir("Desea leer otro número? s/n");
      leer (resp) ;  ← 4.  leer respuesta
   }
imprimir("La suma es :",suma);
}
```

3. Mientras controlado con Valor Centinela

Definición del Problema: Leer una serie de velocidades siempre y cuando la velocidad sea diferente de -9999. Calcular e imprimir el promedio de velocidades.

Análisis y Diseño

Entrada	**velocidades (vel)**
Proceso	Repetir mientras velocidad <> -9999 Calcular velocidad total (vtotal) = vtotal + vel Calcular total de velocid (tot) = tot + 1 Calcular promedio(prom) = vtotal/tot
Salida	**prom**

```
Algoritmo Velocidades
{
 // Bloque de declarativas
flotante vel, vtotal =0, prom;
flotante tot = 0;
//Bloque del Instrucciones
prom = 0;
imprimir("Ingrese la velocidad: ")
leer(vel); ← 1ª lectura
mientras( vel < > -9999)   ← condición
 {
   vtotal = vtotal + vel;
   tot = tot + 1;
   imprimir("Ingrese la velocidad")
   leer( vel ); ← leer valor
   }
   prom = vtotal/tot;
   imprimir ("Promedio de velocidades : ",  prom); }
}
```

2. Mientras controlado con Valor Centinela

Definición del Problema : **L**eer una serie de velocidades siempre y cuando la velocidad sea diferente de -9999. Calcular e imprimir el promedio de velocidades.

Prueba de Escritorio

variables de memoria				UAL	
vel	vtotal	tot	prom	vel != -9999	PANTALLA
~~**120**~~	**120** **0 + 120**	**1** **0 + 1**	**0**	120 != -9999 C	Ingrese la velocidad: 120
~~**150**~~	**270** **120 + 150**	**2** **1 + 1**	**0**	**150 != -9999** **C**	Ingrese la velocidad: 150
-9999			**140.0** **270/2 =**	**-9999 != -9999** **F**	**Ingrese la velocidad: -9999**
					Promedio de velocidades : 140.0

```
Algoritmo Velocidades
{
 // Bloque de declarativas
flotante vel, vtotal =0, prom;
flotante tot = 0;
//Bloque del Instrucciones
prom = 0;
imprimir("Ingrese la velocidad: ")
leer(vel); ← 1ª lectura
mientras( vel !=  -9999)   ← condición
 {
   vtotal = vtotal + vel;
   tot = tot + 1;
   imprimir("Ingrese la velocidad")
   leer( vel ); ← leer valor
   }
   prom = vtotal/tot;
   imprimir ("Promedio de velocidades : ",   prom); }
}
```

PRÁCTICA - Sentencias de Repetición -

- **Problema1** : Defina todos los elementos necesarios para leer e Imprimir 10 números flotantes.
- **Problema 2** : Calcular la suma de los 7 primeros números pares.
- **Problema 3** : Defina todos los elementos necesarios para calcular e imprimir la sumatoria de los salarios de los 50 empleados de la Compañía El Sol. Se lee el nombre y el salario de los empleados.
- **Metodología:**

Para cada problema :

- Identifique el tipo de ciclo
- Identifique la técnica que va a aplicar en cada problema.
- Identifique la variable contador(es), de control, acumulador(es)
- Trabajo individual.

PRÁCTICA 2

Problema 1. Elabore un algoritmo que lea tres notas de un grupo de estudiantes, donde la calificación 1 vale 20%, la segunda vale el 30% y la 3 el 50%. Determinar la calificación final.
De acuerdo a la calificación determinar si el estudiante fue aprobado ó reprobado. Se aprueba con 71. Se cuenta con el Nombre, calificación1,cal2, cal3.
Imprimir nombre, calificación final y el mensaje. También se debe imprimir el total de aprobados y total de reprobados.

Problema 2. Determine en un conjunto de 100 números naturales ¿cuántos son menores de 15, mayores de 50 y cuántos están comprendidos entre 45 y 55 ?

Problema 3. LA Cía. Electrónicos Especiales desea determinar cuántos de sus empleados son mayores de 65 años, teniendo el nombre, dirección, edad y sexo. Mostrar el nombre y edad de aquellos que cumplen, además del total solicitado.

ESTRUCTURAS DE REPETICIÓN - PARA -

2. Sentencia para:

Permite ejecutar una sentencia simple o compuesta repetidamente un número conocido de veces.
Formato: para (variable = valor inicial ;condición; incremento/decremento)
{ sentencia(s); }

var. de control

valor inicial = valor inicial de la variable de control, puede ser una constante o el valor de una variable.

condic = es una expresión que comprueba la variable de control (debe ser cierta)

incre/decre = define la manera en que cambia la variable de control.

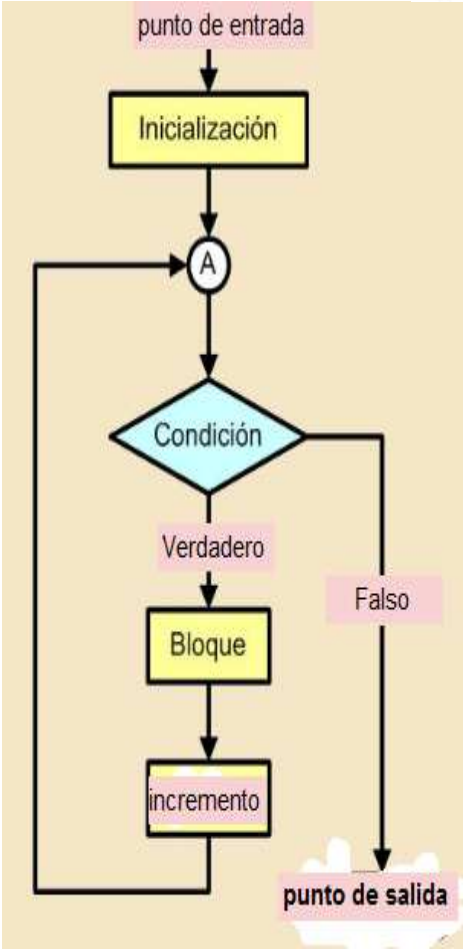
Cuando se ejecuta la sentencia for, la condición se evalúa y se comprueba antes de entrar al ciclo, el incremento es evaluado al final del ciclo.

La ejecución del ciclo continuará mientras el valor de expresión 2 no sea cero, o sea mientras la condición sea cierta.

Estructura de Repetición – para

El funcionamiento de la estructura es el siguiente:

- En primer lugar, se asigna a la variable de control el valor de **inicio.**
- El ciclo se ejecuta mientras **variable de control no alcance el valor en la condición.**
- En cada iteración el valor de **variable de control es cambiado según el incremento** indicado y se ejecuta la acción o grupo de acciones encerrados en el bucle.



ESTRUCTURAS DE REPETICIÓN - para

DEFINICIÓN: Se imprimen los números del 1 al 10

Análisis y Diseño

Entrada	
Proceso	Generar números del 1 al 10 (num)
Salida	num

```
Algoritmo Números {
// Declarativas
   entero num; ←  declaración
// Bloque de instrucciones
  para (num = 1; num<10 ; num = num + 1) ← inicialización
   /Valor Final/incremento
       {
          imprimir( num);
       }
}
```

ESTRUCTURAS DE REPETICIÓN - para

DEFINICIÓN: Se imprimen los números del 1 al 10

Análisis y Diseño

variables de memoria	U.A.L	
num	num < 10	**PANTALLA**
~~1~~	1 < 10 cierto	**1**
~~2~~	2 < 10 cierto	**2**
~~3~~	3 < 10 cierto	**3**
~~4~~	4 < 10 cierto	**4**
~~5~~	5 < 10 cierto	**5**
~~6~~	6 < 10 **cierto**	6
7	**7 < 10 falso**	

Ejemplo 1: Sentencia simple.

```
Algoritmo Números
{
// Declarativas
   entero num; ← declaración
// Bloque de instrucciones
  para (num = 1; num<10 ; num = num + 1) ← inicialización
   /Valor Final/incremento
    {
        imprimir( num);
      }
}
```

ESTRUCTURAS DE REPETICIÓN -para

Ejemplo 2: Se imprimen los 50 números en orden decreciente

```
// Declarativas
   entero x; ←   declaración
// Bloque de instrucciones (Imprimir los 10 primeros numeros)
     para (x = 50;  x>1; x = x – 1) ← inicialización /Valor Final/Decremento
        {   imprimir( x);  }
```

ESTRUCTURAS DE REPETICIÓN -para

Ejemplo 3.

```
int k, x;
para ( k = 7; k < 122 ; k= k + 7)
{    imprimir("Se imprime el valor de k:",k);
     imprimir("Cuadrado de  k: ",k *k);
      x=x+1;
}
imprimir("número de veces ejecutado:", x);
```

ESTRUCTURAS DE REPETICIÓN -para

a. Ciclos Anidados

Tiene una sentencia Para colocada dentro del ciclo de otra sentencia Para.

Ejemplo: Imprimir una matriz 5*6 y rellenarla de asteriscos.

```
entero fila,  col;
caracter    c ;
c = '*' ;
imprimir(" Cuadro con Asteriscos  ");
  para (fila = 1;fila<=5; fila= fila + 1)
     { para (col = 1; col<=6; col= col + 1)
           imprimir(c);
     }
   imprimir( );
```

ESTRUCTURAS DE REPETICIÓN

Problema 1 :Calcular la suma de los 10 primeros números pares.

Problema2. Construya un algoritmo que lea un entero . Debe imprimir el entero leído, y posteriormente debe calcular e imprimir la suma de todos los enteros existentes entre 1 y el entero leído.

Problema 3. Generar la tabla de multiplicar del 5 con 10 elementos. Imprimir el título de cada tabla generada con sus elementos de la forma
5 x 1 = 5, 5 x2 =10,…
Problema 4.Construya un algoritmo que lea entero entre 1 y 9. El programa debe imprimir todos los enteros múltiplos del número que existan entre 1 y 50. Determine si es múltiplo.
Ejemplo: Para n = 8, el programa debe desplegar: 8,16, 24,32,40,48.

Problema 5. Imprima cualquier tabla de multiplicar, donde su multiplicando debe ir de 0 a 12, e imprima bajo este formato **99 * 99 = 999**