

PROBLEMA

1. Dado un vector de dimensión N, obtener la suma de sus elementos.

$$v = [\begin{array}{ccccccc} 1 & 2 & 4 & 6 & 3 & 1 & 5 \end{array}] = 21$$

ANALISIS

SOLUCION

Leer n como dimension para vector v=[]

Leer x como elemento del vector

suma <- suma*x[V[posicion]]

para leer elemento usar for y paa mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: suma

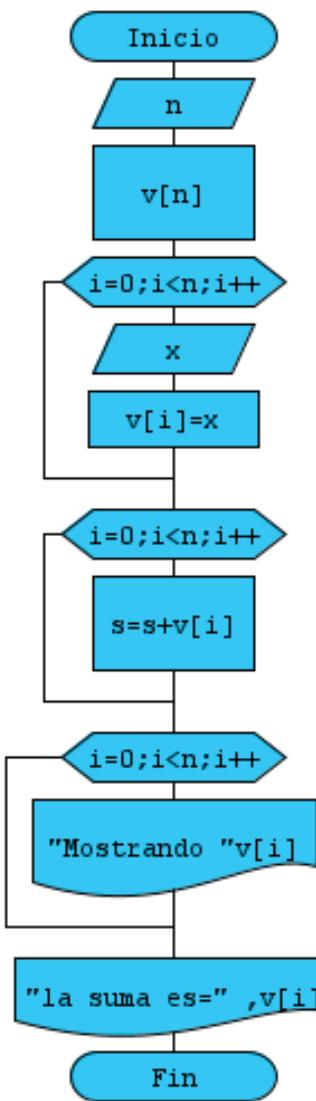
Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros

Definir proceso: suma <- suma*x[V[posicion]]

2 DISEÑAR DIAGRAMA

ALGORITHM



SOURCE IN PYTHON

```

import timeit
s=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor")))
for i in range(n):
    s=s+B[i]
# mostrando vector
print(B)
#mostrando suma
print("la suma de v[nelementos]=",s)
total_time= timeit.timeit(
    number=1000)
mult1 = mult1 * x
print("el resultado es: ",mult)
    
```

DESK CHECK

```

Ingrese dimision=7
ingrese un valor=1
ingrese un valor=2
ingrese un valor=4
ingrese un valor=6
ingrese un valor=1
ingrese un valor=5
ingrese un valor=3
[1, 2, 4, 6, 1, 5, 3]
la suma de v[nelementos]= 22
    
```

DESK CHECK

```

n = 7
x = 1
x = 2
x = 4
x = 6
x = 1
x = 5
x = 3
    
```

Prueba de escritorio (PDE's)

Inicio |

n	v	s	i	x
7	v[0]=1	0	0	1
	v[1]=2	1	1	2
	v[2]=4	3	2	4
	v[3]=6	7	3	6
	v[4]=3	13	4	3
	v[5]=1	16	5	1
	v[6]=5	21	6	5
		22	7	

PROBLEMA

2. Obtener la suma de los elementos pares y la suma de los elementos impares de un vector:

v=[] [1 2 4 6 3 1 5] =21

spar = 12, simpar= 10

ALGORITHM

ANALISIS

SOLUCION

Leer n como dimension para vector v=[]

Leer x como elemento del vector

sumaPar <- sumaPar*V[posicionPares]

sumalmpar <- sumalmpar*V[posicionImpares]

para leer elemento usar for y para mostrar usar

for

1 ANALIZAR EL PROBLEMA

Especificar resultado: suma

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros

Definir proceso:

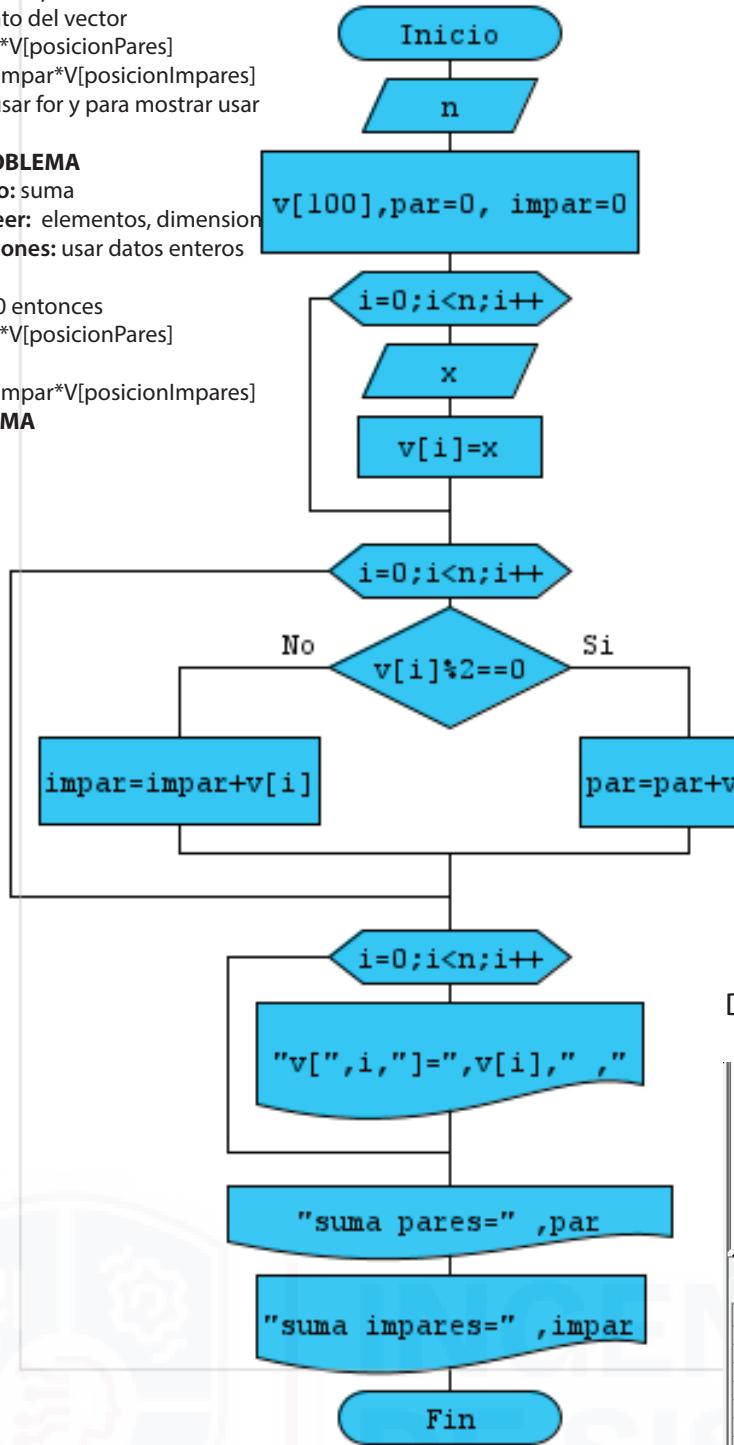
si V[posicion]%2==0 entonces

sumaPar <- sumaPar*V[posicionPares]

sino

sumalmpar <- sumalmpar*V[posicionImpares]

2 DISEÑAR DIAGRAMA



SOURCE IN PYTHON

```

import timeit
par=0
impar=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
for i in range(n):
    if B[i]%2==0:
        par = par + B[i]
    else:
        impar = impar + B[i]
# mostrando vector
print(B,"")
#mostrando suma
print("la suma de v[n pares]=",par)
print("la suma de v[n impares]=",impar)
total_time= timeit.timeit( number=1000)
  
```

DESK CHECK

```

Ingrese dimision=7
ingrese un valor=1
ingrese un valor=2
ingrese un valor=4
ingrese un valor=6
ingrese un valor=3
ingrese un valor=1
ingrese un valor=5
[1, 2, 4, 6, 3, 1, 5]
la suma de v[n pares]= 12
la suma de v[n impares]= 10
  
```

DESK CHECK

n = 7					
x = 1					
x = 2					
x = 3					
x = 6					
x = 5					
x = 4					
x = 1					
v[0]=1 ,v[1]=2 ,v[2]=3 ,v[3]=6 ,v[4]=5 ,v[5]=4 ,v[6]=1 ,suma pares=12suma impares=10					
Prueba de escritorio (PDE's)					
Inicio					
n	v	par	impar	i	x
7	v[0]=1 0	0	0	0	1
	v[1]=2 2	2	1	1	2
	v[2]=3 8	8	4	2	3
	v[3]=6 12	12	9	3	6
	v[4]=5		10	4	5
	v[5]=4			5	4
	v[6]=1			6	1
				7	

PROBLEMA

3. Dado un vector de dimensión N, mostrar solamente sus elementos pares.

v=[]
 ANALISIS
 spar = 2, 4, 6

ALGORITHM

SOLUCION

Leer n como dimension para vector v=[]

Leer x como elemento del vector

mostrar V[posicionPares]

para leer elemento usar for y para mostrar usar
for

1 ANALIZAR EL PROBLEMA

Especificar resultado: suma

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros

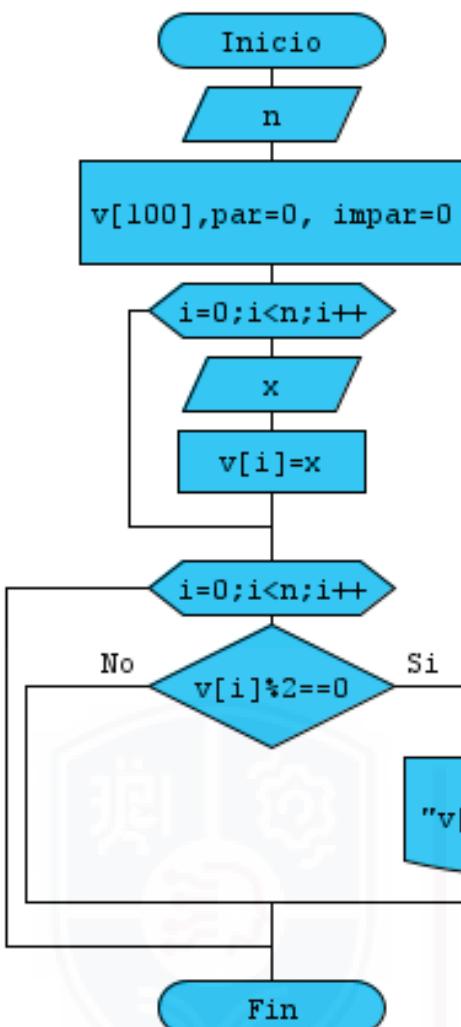
Definir proceso:

si V[posicion] % 2 == 0 entonces

mostrar V[posicionPares]

2 DISEÑAR DIAGRAMA

DESK CHECK



Prueba de escritorio (PDE's)

M	Inicio
n	
x	
v	
i	
v[0]=3	0 0 0 3
v[1]=2	1 2
v[2]=1	2 1
v[3]=6	3 6
v[4]=5	4 5
v[5]=4	5 4
v[6]=1	6 1

SOURCE IN PYTHON

```

import timeit
par=0
impar=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
    # mostrando vector
print(B, "")
for i in range(n):
    if B[i]%2==0:
        print("B[",i,"]=",B[i])
total_time= timeit.timeit(number=1000)
  
```

DESK CHECK

```

Ingrese dimision=7
ingrese un valor=3
ingrese un valor=2
ingrese un valor=1
ingrese un valor=6
ingrese un valor=5
ingrese un valor=4
ingrese un valor=1
[3, 2, 1, 6, 5, 4, 1]
B[ 1 ]= 2
B[ 3 ]= 6
B[ 5 ]= 4
  
```

PROBLEMA

4. Dado un vector, reemplazar los valores pares por ceros.

ANALISIS
 $v = [1 \ 2 \ 4 \ 6 \ 3 \ 1 \ 5]$
 $spar = 2=0, 4=0, 6=0$

SOLUCION

Leer n como dimension para vector $v = []$

Leer x como elemento del vector

mostrar $V[n]elementos]$

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: suma

Identificar datos a leer: elementos, dimension

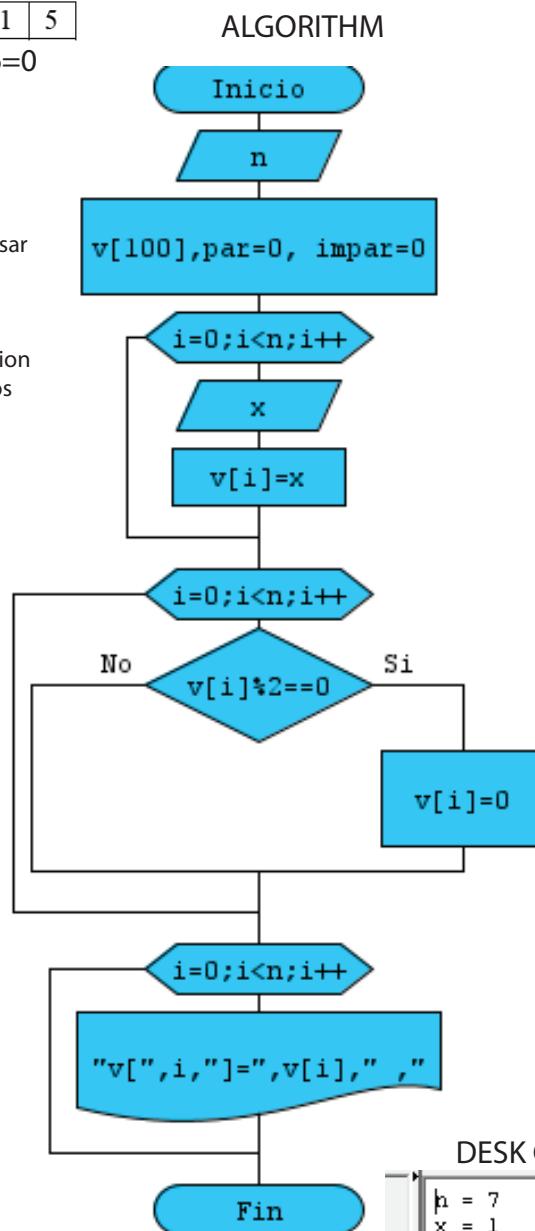
Determinar restricciones: usar datos enteros

Definir proceso:

si $V[posicion]\%2==0$ entonces
 $v[posiciones]=0$

mostrar $V[posicionPares con ceros]$

2 DISEÑAR DIAGRAMA



SOURCE IN PYTHON

```

import timeit
par=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
# mostrando vector
print(B, "")
for i in range(n):
    if B[i]\%2==0:
        B[i]=0
    print("B[",i,"]",B[i])
print(B)
total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

Ingrese dimision=7
ingrese un valor=2
ingrese un valor=4
ingrese un valor=6
ingrese un valor=3
ingrese un valor=1
ingrese un valor=1
ingrese un valor=5
[2, 4, 6, 3, 1, 1, 5]
B[ 0 ]= 0
B[ 1 ]= 0
B[ 2 ]= 0
[0, 0, 0, 3, 1, 1, 5]
tiempo en ejecucion 1.3899996702093631e
    
```

DESK CHECK

```

n = 7
x = 1
x = 2
x = 3
x = 6
x = 5
x = 4
x = 1
v[0]=1 ,v[1]=0 ,v[2]=3 ,v[3]=0 ,v[4]=5 ,v[5]=0 ,v[6]=1 ,
    
```

Prueba de escritorio (PDE's)

Inicio

n	v	...	i	x
7	v[0]=1	0	0	1
	v[1]=2		1	2
	v[2]=3		2	3
	v[3]=6		3	6
	v[4]=5		4	5
	v[5]=4		5	4
	v[6]=1		6	1

PROBLEMA

5. Dado un vector de dimensión N, obtener la cantidad de números pares e impares..

v=[] 1 2 4 6 3 1 5
spar = 3 impar=4

ANALISIS

SOLUCION

Leer n como dimension para vector v=[]

Leer x como elemento del vector

mostrar V[nelementos]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: suma

Identificar datos a leer: elementos, dimension

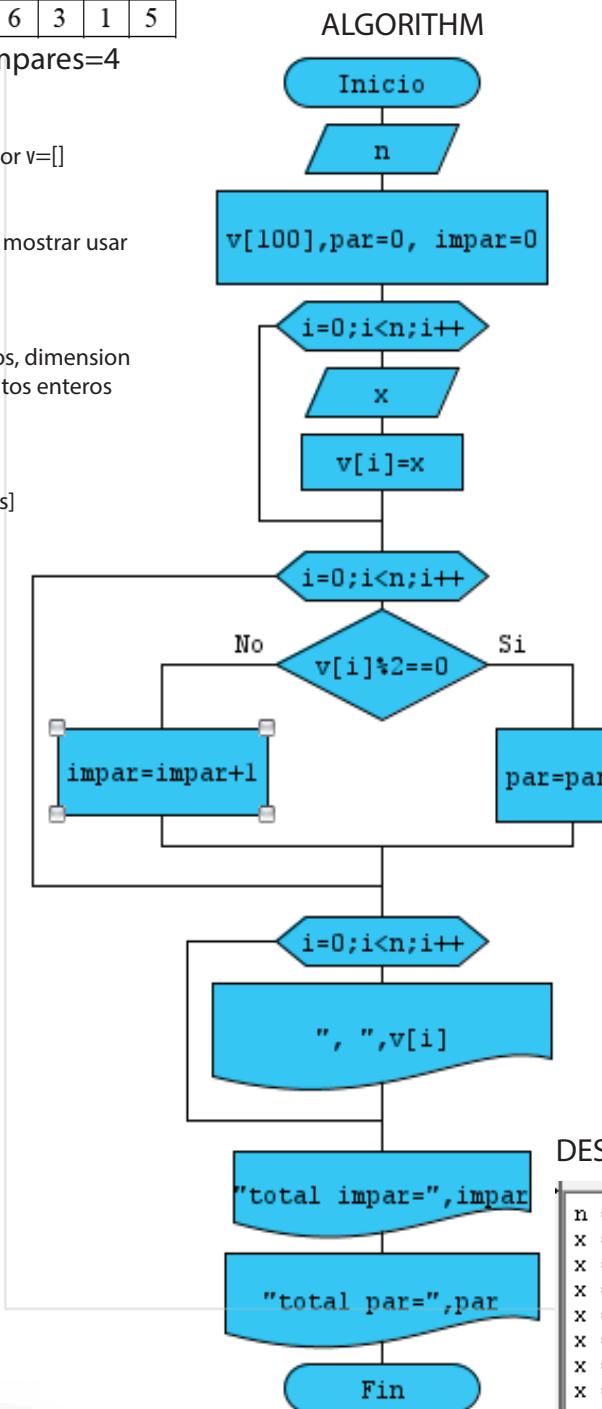
Determinar restricciones: usar datos enteros

Definir proceso:

si $V[\text{posicion}] \% 2 == 0$ entonces
 $v[\text{posiciones}] = 0$

mostrar V[posicionPares con ceros]

2 DISEÑAR DIAGRAMA



SOURCE IN PYTHON

```

import timeit
par=0
impar=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
# mostrando vector
print(B, "")
for i in range(n):
    if B[i]%2==0:
        par=par+1
    else:
        impar=impar+1
print(B)
print("totalpares=",par)
print("total Impares=",impar)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

Ingrese dimision=7
ingrese un valor=1
ingrese un valor=2
ingrese un valor=3
ingrese un valor=6
ingrese un valor=5
ingrese un valor=4
ingrese un valor=1
[1, 2, 3, 6, 5, 4, 1]
[1, 2, 3, 6, 5, 4, 1]
totalpares= 3
total Impares= 4
tiempo en ejecucion 1.2600001355167478e-05
    
```

DESK CHECK

```

n = 7
x = 3
x = 2
x = 1
x = 4
x = 5
x = 6
x = 1
, , 2, 1, 4, 5, 6, 1
total impar=4total par=3
    
```

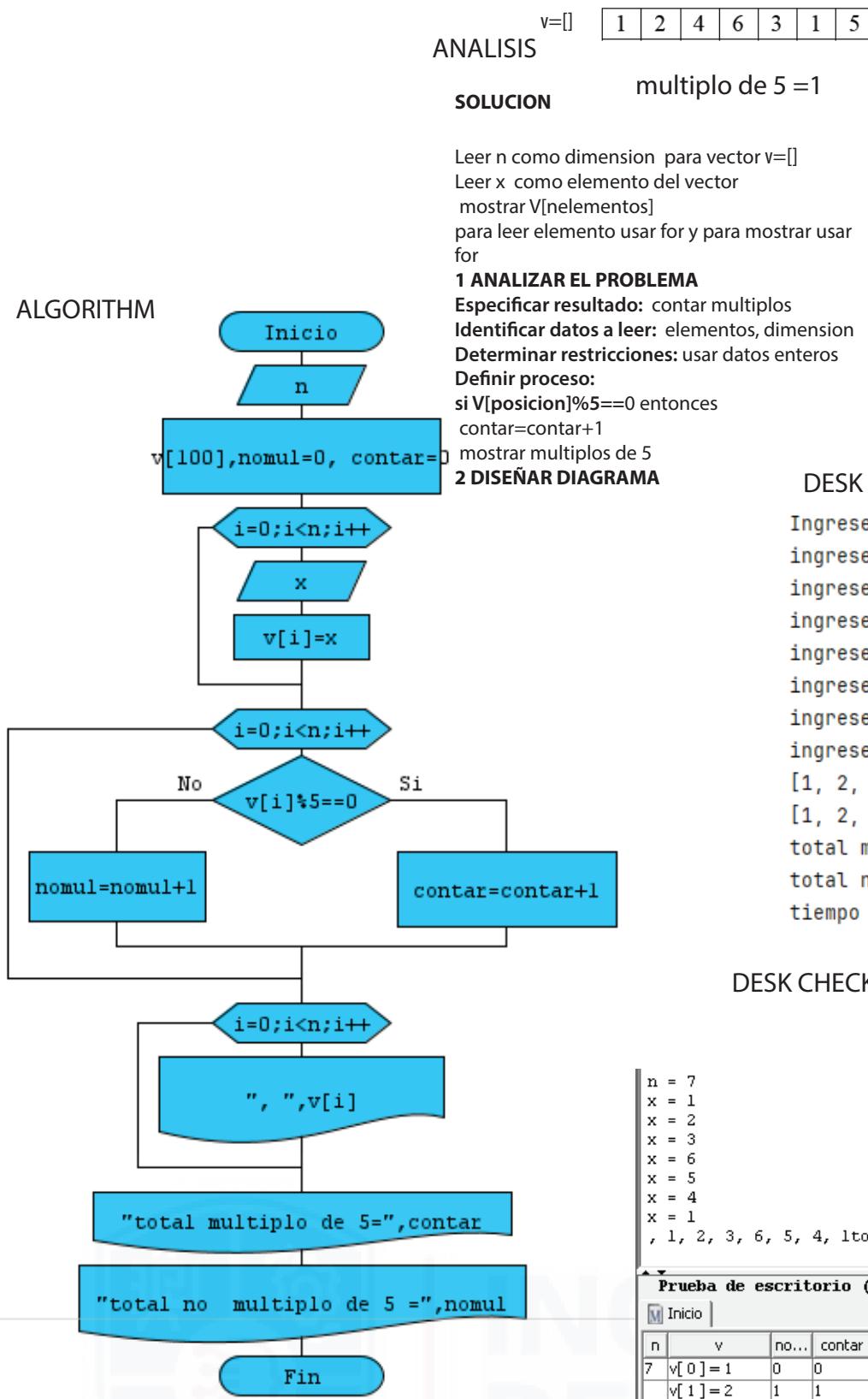
Prueba de escritorio (PDE's)

n	v	par	impar	i	x
7	v[0]=3	0	0	0	3
	v[1]=2	1	1	1	2
	v[2]=1	2	2	2	1
	v[3]=4	3	3	3	4
	v[4]=5		4	4	5
	v[5]=6			5	6
	v[6]=1			6	1
					7

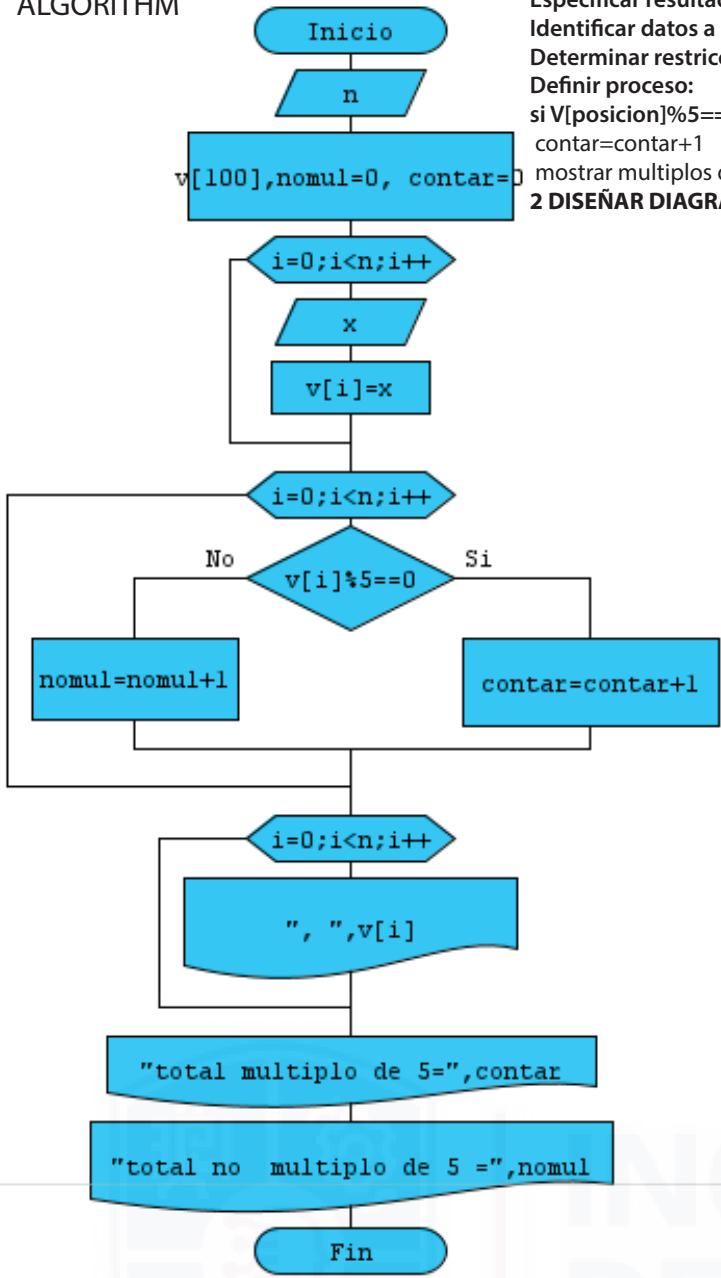


PROBLEMA

6. Dado un vector de dimensión N, obtener la cantidad de múltiplos de 5.



ALGORITHM



ANALISIS

SOLUCION

multiplo de 5 = 1

Leer n como dimension para vector v = []
Leer x como elemento del vector
mostrar V [n elementos]
para leer elemento usar for y para mostrar usar
for

1 ANALIZAR EL PROBLEMA

Especificar resultado: contar múltiplos
Identificar datos a leer: elementos, dimensiones
Determinar restricciones: usar datos enteros

Definir proceso:

si V[posicion] % 5 == 0 entonces

contar=contar+1

mostrar multiplos de 5

2 DISEÑAR DIAGRAMA

2 DISEÑAR DIAGRAMA

SOURCE IN PYTHON

```
import timeit
nomul=0
contar=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
    # mostrando vector
print(B, "")
for i in range(n):
    if B[i]%5==0:
        contar=contar+1
    else:
        nomul=nomul+1
print(B)
print("total mutiplos de 5=",contar)
print("total no multiplos=",nomul)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
```

DESK CHECK

```
Ingrese dimension=7
ingrese un valor=1
ingrese un valor=2
ingrese un valor=3
ingrese un valor=6
ingrese un valor=5
ingrese un valor=4
ingrese un valor=1
[1, 2, 3, 6, 5, 4, 1]
[1, 2, 3, 6, 5, 4, 1]
total mutiplos de 5= 1
total no multiplos= 6
tiempo en ejecucion 5.7600002037361264e-05
```

DESK CHECK

```
n = 7
x = 1
x = 2
x = 3
x = 6
x = 5
x = 4
x = 1
, 1, 2, 3, 6, 5, 4, ltotal multiplo de 5=ltotal no multiplo de 5 =6
```

Prueba de escritorio (PDE's)

 Inicio

n	v	no...	contar	i	x
7	v[0]=1	0	0	0	1
	v[1]=2	1	1	1	2
	v[2]=3	2		2	3
	v[3]=6	3		3	6
	v[4]=5	4		4	5
	v[5]=4	5		5	4
	v[6]=1	6		6	1

PROBLEMA

7. Pedir un número entero y positivo que será el tamaño de un vector, llenarlo por teclado asegurándose que solo ingresen números positivos de un solo dígito y mostrar por pantalla.

ANALISIS $v = []$

1	2	4	6	3	1	5
---	---	---	---	---	---	---

SOLUCION

Ler n como dimension para vector $v = []$

validar que sea un entero positivo

Ler x como elemento del vector

validar que sean numeros positivos y de digito

1

mostrar $V[n \text{elementos}]$

para leer elemento usar for y para mostrar usar

for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, positivos, de un dígito

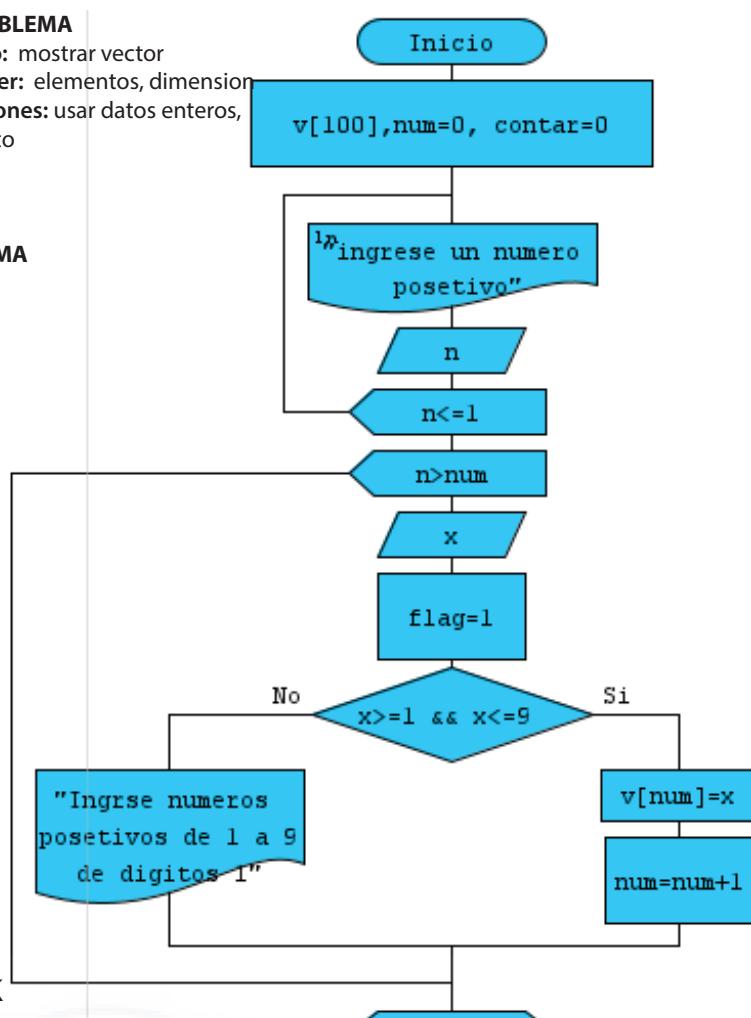
Definir proceso:

si $x > 1 \&& x \leq 9$

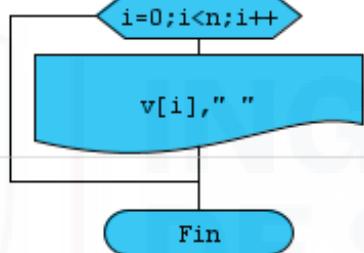
llenar el vector

2 DISEÑAR DIAGRAMA

ALGORITHM



DESK CHECK



```

ingrese un numero, positivo
n = 6
x = 3
x = 2
x = 1
x = 5
x = 0
Ingrse numeros ,positivos de 1 a 9,de digitos ix = 1
3 2 1 5 5 1
  
```

v	num	n	...	x	i
v[0]=3	0	6	1	0	0
v[1]=2	1		1	2	1
v[2]=1	2		1	1	2
v[3]=5	3		1	5	3
v[4]=5	4		1	5	4
v[5]=1	5		1	0	5
v[6]=6	6		1	1	6

SOURCE IN PYTHON

```

import timeit
nomul=0
contar=0
n=int(input("Ingrese dimision="))
B=[]
for i in range(n):
    B.append(int(input("ingrese un valor=")))
# mostrando vector
print(B, "")
for i in range(n):
    if B[i]%5==0:
        contar=contar+1
    else:
        nomul=nomul+1
print(B)
print("total mutiplos de 5=",contar)
print("total no multiplos=",nomul)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
  
```

DESK CHECK

PROBLEMA

7. Pedir un número entero y positivo que será el tamaño de un vector, llenarlo por teclado asegurándose que solo ingresen números positivos de un solo dígito y mostrar por pantalla.

ANALISIS $v = []$

1	2	4	6	3	1	5
---	---	---	---	---	---	---

SOLUCION

Ler n como dimension para vector $v = []$

validar que sea un entero positivo

Ler x como elemento del vector

validar que sean numeros positivos y de dígito 1

mostrar $V[n \text{elementos}]$

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, positivos, de un dígito

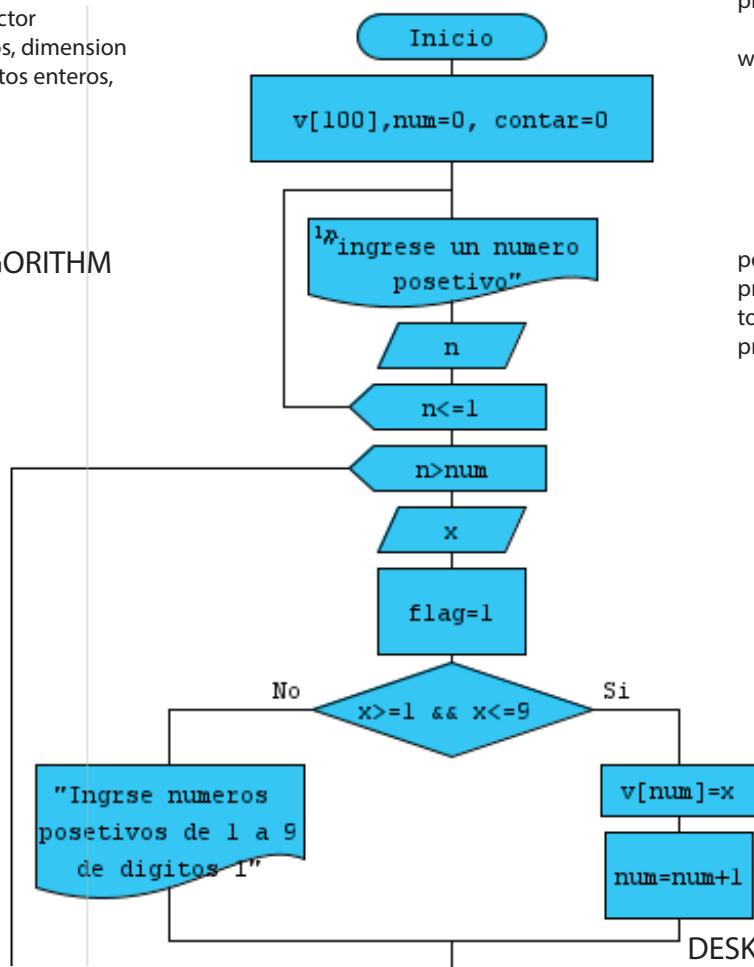
Definir proceso:

si $x > 1 \&& x \leq 9$

llenar el vector

2 DISEÑAR DIAGRAMA

ALGORITHM



DESK CHECK

```

ingrese un numero, positivo
n = 6
x = 3
x = 2
x = 1
x = 5
x = 5
x = 0
Ingrse numeros ,positivos de 1 a 9,de digitos 1x = 1
3 2 1 5 5 1
  
```

Prueba de escritorio (PDE's)

M Inicio |

v	num	n	...	x	i
v[0]=3	0	6	1	0	3
v[1]=2	1		1	2	1
v[2]=1	2		1	1	2
v[3]=5	3		1	5	3
v[4]=5	4		1	5	4
v[5]=1	5		1	0	5
	6		1	1	6

SOURCE IN PYTHON

```

numero = -1
contar=0
import timeit
B=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo: "))
    if n>0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

print(f"Has ingresado un número positivo: {n}")

while n>contar:
    x=int(input("ingrese un digito"))
    if x>=1 and x<=9:
        B.append(x)
        contar=contar+1
    else:
        print("ingrese numeros de un dígito positivos")
print(B)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
  
```

DESK CHECK

```

Introduce un número positivo: 5
Has ingresado un número positivo: 5
ingrese un digito1
ingrese un digito2
ingrese un digito3
ingrese un digito6
ingrese un digito5
[1, 2, 3, 6, 5]
tiempo en ejecucion 1.2800000149582047e-05
  
```

PROBLEMA

8. Realizar un algoritmo que te permita llenar dos vectores diferentes del mismo tamaño y mostrar la multiplicación de posiciones de la vector_1 por el vector_2.

ANALISIS A=[] [1 2 4 6 3 1 5] B=[] [1 2 4 6 3 1 5]

C=[1 4 16 36 9 1 25]

SOLUCION

Ler n como dimension para vector A[], B[]

C[]

mostrar C[nelementos]= A[n] * B[n]

para leer elemento usar for y para mostrar usar
for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector multipli
cado entre c=vectorA* vectorB

Identificar datos a leer: elementos, dimensio

Determinar restricciones: usar datos enteros,
positivos, de un digito

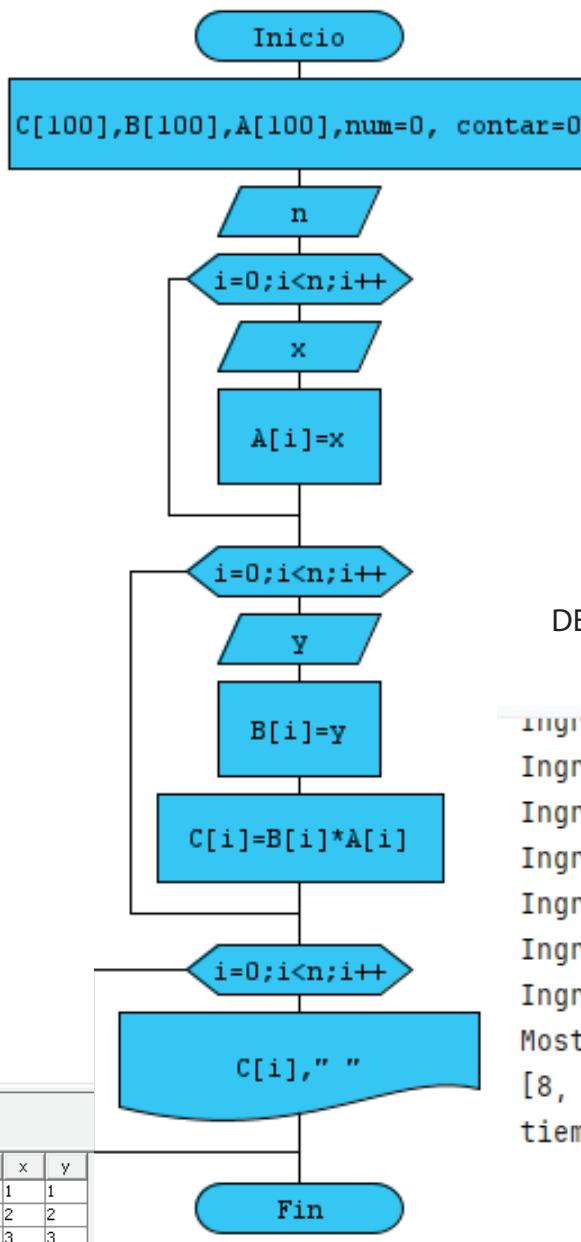
Definir proceso:

c[posecion]= A[posecion]* B[posecion]

llenar el vector

2 DISEÑAR DIAGRAMA

ALGORITHM



DESK CHECK

```

n = 5
x = 1
x = 2
x = 3
x = 6
x = 5
y = 1
y = 2
y = 3
y = 6
y = 5
2 4 6 12 10
    
```

Prueba de escritorio (PDE's)

C	B	A	n	i	x	y
C[0] = 2	B[0] = 1	A[0] = 1	0 0 5	0	1	1	
C[1] = 4	B[1] = 2	A[1] = 2			1	2	2
C[2] = 6	B[2] = 3	A[2] = 3			2	3	3
C[3] = 12	B[3] = 6	A[3] = 6			3	6	6
C[4] = 10	B[4] = 5	A[4] = 5			4	5	5
							5

SOURCE IN PYTHON

```

n = -1
contar=0
import timeit
B=[]
C=[]
A=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo:"))
    if n > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

print(f"Has ingresado un número positivo: {n}")

for i in range(n):
    x=int(input("Ingrese un valor="))
    A.append(x)
for i in range(n):
    y=int(input("Ingrese un valor="))
    B.append(y)
for i in range(n):
    C.append(A[i]*B[i])
print("Mostrando vector multiplicado")
print(C)
total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

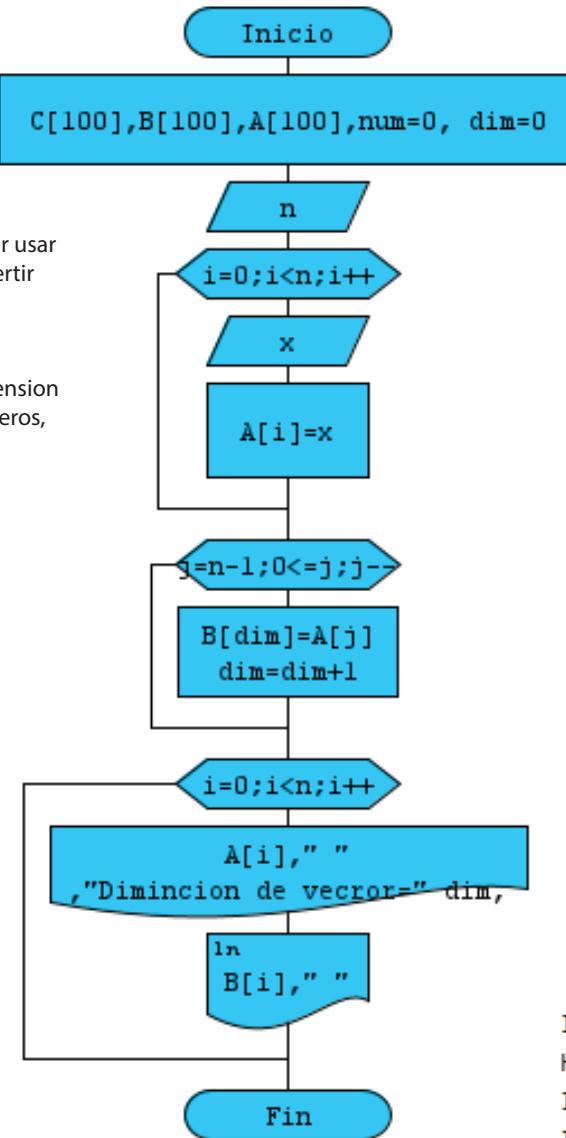
Ingrese un valor=6
Ingrese un valor=9
Ingrese un valor=3
Ingrese un valor=2
Ingrese un valor=2
Ingrese un valor=2
Mostrando vector multiplicado
[8, 18, 9, 8, 10, 12, 14]
tiempo en ejecucion 1.429999974789097
    
```

PROBLEMA

9. Dado un vector V[] de dimensión N, (añadir elementos) calcular la dimensión y mostrar por pantalla vector invertido.

ANALISIS $v = [1 \ 2 \ 4 \ 6 \ 3 \ 1 \ 5]$

ALGORITHM



DESK CHECK

```

n = 4
x = 1
x = 2
x = 3
x = 5
1 5
2 3
3 2
5 1
    
```

Prueba de escritorio (PDE's)

Inicio

C	B	A	...	n	i	x	j
	B[0]=5	A[0]=1	0	0	4	0	1
	B[1]=3	A[1]=2		1		1	2
	B[2]=2	A[2]=3		2		2	3
	B[3]=1	A[3]=5		3		3	5
				4			-1
				n			

SOURCE IN PYTHON

```

n = -1
dim=0
import timeit
B=[]
A=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo:"))
    if n> 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

print(f"Has ingresado un número positivo: {n}")
#llenado de vector de dimension n
for i in range(n):
    x=int(input("Ingrese un valor="))
    A.append(x)
#invertir vector A[] en B[]
for i in range(n-1,-1,-1):
    B.append(A[i])
    dim=dim+1
print("Mostrando vector A")
print(A)
print("Mostrando vector B invertido")
print(B)
print("Mostrando dimencion de vector=",dim)

total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

Introduce un número positivo: 5
Has ingresado un número positivo: 5
Ingrese un valor=1
Ingrese un valor=2
Ingrese un valor=3
Ingrese un valor=6
Ingrese un valor=5
Mostrando vector A
[1, 2, 3, 6, 5]
Mostrando vector B invertido
[5, 6, 3, 2, 1]
Mostrando dimencion de vector= 5
tiempo en ejecucion 1.4000004739500582e-05
    
```

PROBLEMA

11. Dado un vector V de n elementos se pide la suma de aquellos elementos que se encuentran en posiciones pares en la variable spar y la suma de los elementos que se encuentran en las posiciones impares simp.

ANALISIS

v=[]

1	4	7	16	5	100	49	1	81	100
---	---	---	----	---	-----	----	---	----	-----

SOLUCION

$$\begin{aligned} \text{spar} &= 1+7+5+49+81=143 \\ \text{simp} &= 4+16+100+1+100=221 \end{aligned}$$

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Mostrar suma de elementos pares

Mostrar suma de elementos impares

para leer elemento usar for y para mostrar usar for c

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimision

Determinar restricciones: usar datos enteros, positivos

Definir proceso:

si vector[posicion] modulo 2==0
pares += vector A[dimension]

por falso

impares += vector A[dimension]

2 DISEÑAR DIAGRAMA

DESK CHECK

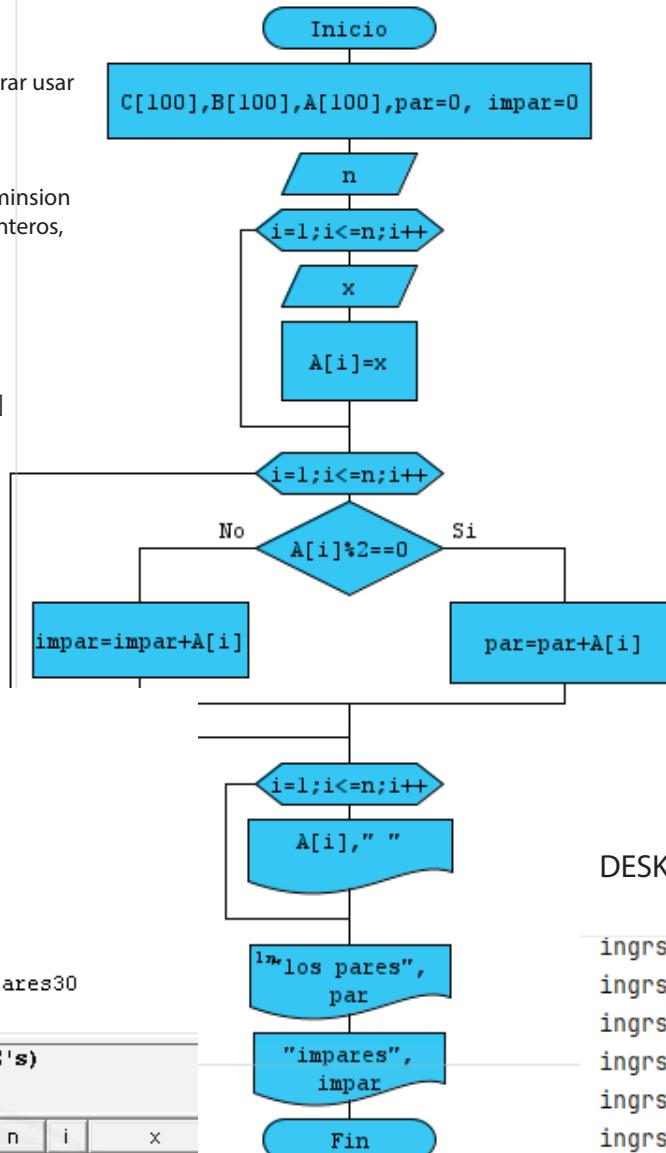
```
n = 10
x = 1
x = 2
x = 3
x = 6
x = 5
x = 4
x = 7
x = 8
x = 9
x = 10
1 2 3 6 5 4 7 8 9 10 los pares30
impares25
```

Prueba de escritorio (PDE's)

Inicio |

C	B	A	par	impar	n	i	x
		A[1] = 1	0	0	10	1	1
		A[2] = 2	2	1		2	2
		A[3] = 3	8	4		3	3
		A[4] = 6	12	9		4	6
		A[5] = 5	20	16		5	5
		A[6] = 4	30	25		6	4
		A[7] = 7				7	7
		A[8] = 8				8	8
		A[9] = 9				9	9
		A[10] = 10				10	10
							11

ALGORITHM



SOURCE IN PYTHON

```

n = -1
impar=0
par=0
import timeit
A=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo:"))
    if n> 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

print(f"Has ingresado un número positivo: {n}")

for fila in range (n):
    m=int(input(F"ingrese un valor {fila}= "))
    A.append(m)
#llenado de vector de dimisión n
for i in range(n):
    if A[i]% 2==0:
        par=par+A[i]
    else:
        impar = impar + A[i]
print("Mostrando vector A=")
print(A)
print("Total pares=",par)
print("Total impares=",impar)

total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)

```

DESK CHECK

```

ingrse un valor 4= 5
ingrse un valor 5= 100
ingrse un valor 6= 49
ingrse un valor 7= 1
ingrse un valor 8= 81
ingrse un valor 9= 100
Mostrando vector A=
[1, 4, 7, 16, 5, 100, 49, 1, 81, 100]
Total pares= 220
Total impares= 144
tiempo en ejecucion 1.3999997463542968e-05

```


PROBLEMA

13. Dado un vector N, intercambiar el segundo elemento con el penúltimo elemento de un vector

ANALISIS

V[] =	5	7	9	13	12	8
V[]	5	12	9	13	7	8

SOLUCION

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Leer x como elemento del vector A=[]

Mostrar A[nelementos]

intercambiar posiciones de elementos de vector

A=[]

para leer elemento usar for y para mostrar usar

for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros,

positivos

Definir proceso:

```
aux=B[con+1]
B[con+1]=B[n-2]
B[n-2]=aux
```

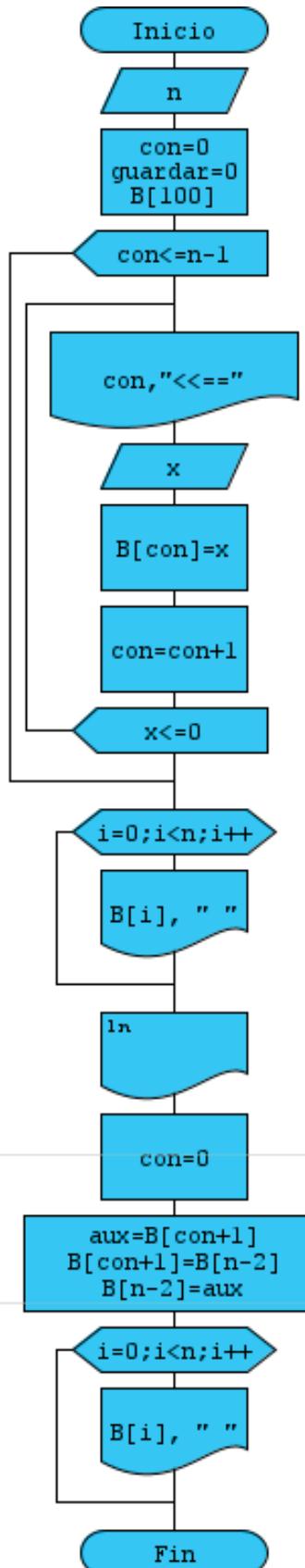
2 DISEÑAR DIAGRAMA

DESK CHECK

```
n = 5
0<==x = 1
1<==x = 2
2<==x = 3
3<==x = 5
4<==x = 4
1 2 3 5 4
1 5 3 2 4
```

Prueba de escritorio (P)						
Inicio		B x i ...				
n	...	B[0]	x	i	...	
5	0	0	B[0]=1	1	0	2
1		B[1]=2	2	1		
2		B[2]=3	3	2		
3		B[3]=5	5	3		
4		B[4]=4	4	4		
5		B[1]=5		5		
0		B[3]=2		0		

ALGORITHM



SOURCE IN PYTHON

```

n = -1
x=-1
con=0
aux=0
import timeit
B=[]
C=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo="))
    if n> 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

for i in range(n):
    x = int(input("Introduce valor en="))
    B.append(x)
print("Vector llenado")
print(B)
#Intercambiar posiciones por el segundo y penultimo
aux=B[con+1]
B[con+1]=B[n-2]
B[n-2]=aux
#mostramos vector intercambiando posiciones
print("Vector final ")
print(B)

total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)

```

SK CHECK

```

Introduce un número positivo= 5
Introduce valor en= 1
Introduce valor en= 2
Introduce valor en= 3
Introduce valor en= 4
Introduce valor en= 5
Vector llenado
[1, 2, 3, 4, 5]
Vector final
[1, 4, 3, 2, 5]
tiempo en ejecucion 1.4299992471933365e-0

```

PROBLEMA

14. Dado un vector V de N elementos enteros positivos, invertir el orden de sus elementos sin utilizar vectores auxiliares.

ANALISIS Si $N=5$

$$V[] = \boxed{5 \quad 7 \quad 9 \quad 13 \quad 12 \quad 0}$$

SOLUCION

$$V[] = \boxed{0 \quad 12 \quad 13 \quad 9 \quad 7 \quad 5}$$

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Leer x como elemento del vector A=[]

Mostrar A[elementos]

intercambiar posiciones de elementos de vector A=[]

para leer elemento usar for y para mostrar usar

for para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros,

positivos

Definir proceso:

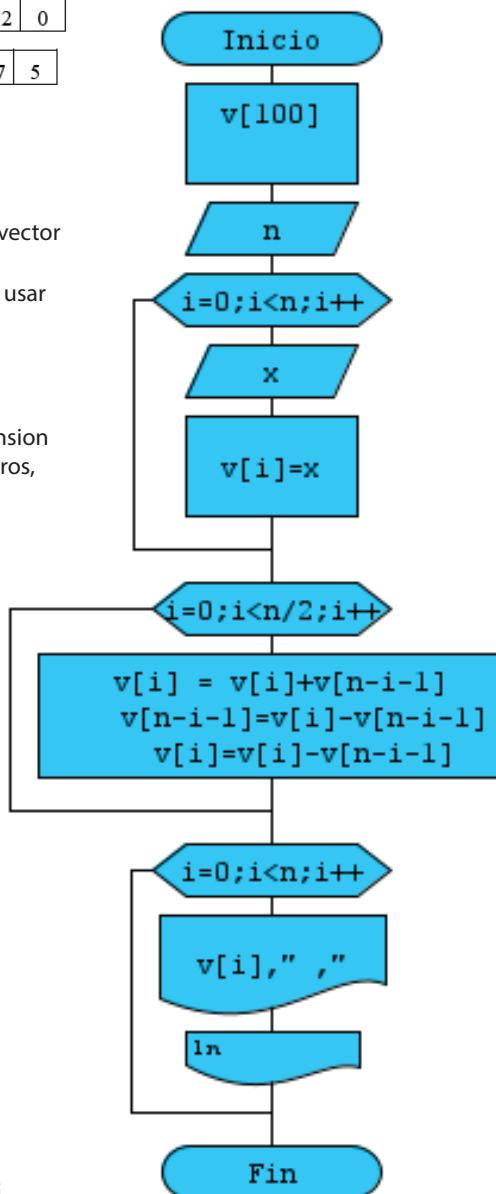
for i=0;i<n/2;i++

$$v[i] = v[i] + v[n-i-1]$$

$$v[n-i-1] = v[i] - v[n-i-1]$$

$$v[i] = v[i] - v[n-i-1]$$

ALGORITHM



DESK CHECK

```

n = 5
x = 1
x = 2
x = 3
x = 4
x = 5
5 ,
4 ,
3 ,
2 ,
1 ,

```

Prueba de escritorio (PDE):

v	n	i	x
v[0] = 1	5	0	1
v[1] = 2		1	2
v[2] = 3		2	3
v[3] = 4		3	4
v[4] = 5		4	5
v[0] = 6			
v[4] = 1			
v[0] = 5			
v[1] = 6			
v[3] = 2			
v[1] = 4			

SOURCE IN PYTHON

```
n = -1
```

```
x=-1
```

```
import timeit
```

```
v=[]
```

```
# Simulación de do...while
```

```
while True:
```

```
    n = int(input("Introduce un número positivo="))
```

```
    if n> 0:
```

```
        break
```

```
    print("El número debe ser positivo. Inténtalo de nuevo.")
```

```
for i in range(n):
```

```
    x = int(input("Introduce valor en="))
```

```
    v.append(x)
```

```
print("Vector llenado")
```

```
print(B)
```

```
n = len(v) # Longitud del vector
```

```
# Invertir el vector en su lugar sin variable temporal
```

```
for i in range(n // 2):
```

```
    v[i] = v[i] + v[n - i - 1]
```

```
    v[n - i - 1] = v[i] - v[n - i - 1]
```

```
    v[i] = v[i] - v[n - i - 1]
```

```
# Mostrar el vector invertido
print("Vector Invertido:", v)
```

DESK CHECK

```
C:\Users\soliasq\PycharmProjects\pyt
```

```
Vector Original: [1, 2, 3, 4, 5]
```

```
Vector Invertido: [5, 4, 3, 2, 1]
```

```
Vector Invertido: [5, 4, 3, 2, 1]
```

```
Process finished with exit code 0
```

PROBLEMA

15. Llenar un vector V, con la siguiente serie: $2, 2^2, 2^3, 2^4, \dots, 2^N$

ANALISIS

si $n=5$ $V[] = [2 | 4 | 8 | 16 | 32]$

ALGORITHM

SOLUCION

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Mostrar A[nelementos]

generar la serie $2, 2^2, 2^3, 2^4, \dots, 2^N$

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimensi

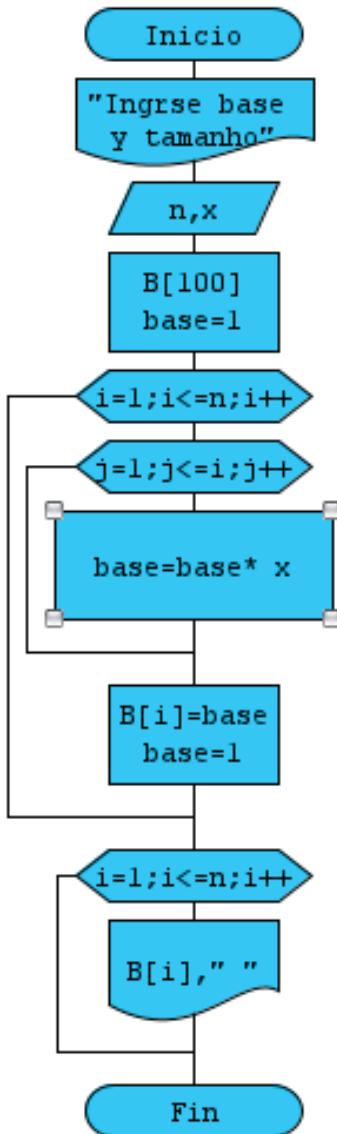
Determinar restricciones: usar datos enteros positivos

Definir proceso:

generar usando for para potencia

$\text{base} = \text{base} * x$

2 DISEÑAR DIAGRAMA



DESK CHECK

Ingrse base ,y tamanhon = 5
 $x = 2$
 $2 | 4 | 8 | 16 | 32$

Prueba de escritorio (PDE's)						
Inicio						
n	x	B	base	i	j	
5	2	B[1]=2	1	1	1	
		B[2]=4	2	2	2	
		B[3]=8	1	3	1	
		B[4]=16	2	4	2	
		B[5]=32	4	5	3	

SOURCE IN PYTHON

```

n = -1
con=0
base=1
import timeit
B=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo="))
    if n > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

for i in range(1,n+1):
    #print("vale i",i)
    for j in range(i):
        base=base*x
    B.append(base)
    base=1
print("Vector llenado")
print(B)
    
```

DESK CHECK

Introduce un número positivo= 5
Ingrse base de potencia= 2
vale i 1
vale i 2
vale i 3
vale i 4
vale i 5
Vector llenado
[2, 4, 8, 16, 32]
tiempo en ejecucion 1.4599994756281376e-05



PROBLEMA

16. Dado un vector N, almacenar la serie de Fibonacci en un vector y mostrar.

ANALISIS

SOLUCION

0 1 1 2 3 5 8...N

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Mostrar A[nelementos]

serie generada en posiciones de elementos de

vector A=[]

para leer elemento usar for y para mostrar usar

for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros,

positivos

Definir proceso:

```
a=-1
b=1
c=0
c=a+b
a=b
b=c
```

2 DISEÑAR DIAGRAMA

DESK CHECK

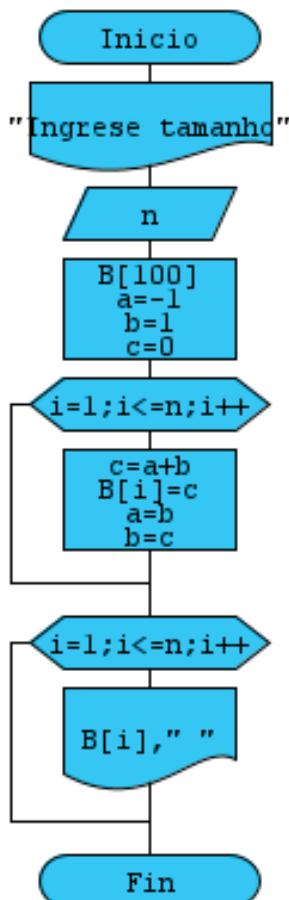
Ingrese tamanhon = 5
0 1 1 2 3

Prueba de escritorio (PDE's)

M Inicio

n	B	a	b	c
5	B[1]=0	-1	1	0
	B[2]=1	1	0	0
	B[3]=1	0	1	1
	B[4]=2	1	1	1
	B[5]=3	1	2	2
		2	3	3
				6

ALGORITHM



SOURCE IN PYTHON

```

n = -1
a=-1
b=1
c=0
import timeit
B=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo="))
    if n > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

for i in range(n):
    c=a+b
    B.append(c)
    a=b
    b=c
print("serie fifo")
print(B)

```

total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)

DESK CHECK

```

Introduce un número positivo= 5
Vector llenado
[0, 1, 1, 2, 3]
tiempo en ejecucion 3.090000245720148e-05

```

PROBLEMA

17. Llenar un vector D, con los primeros N números impares, en orden descendente.

ANALISIS

9	7	5	3	1
---	---	---	---	---

SOLUCION

Ler n como dimension para vector A=[]

validar que sea un entero positivo

generar como elemento del vector A=[]

Mostrar A[nelementos]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

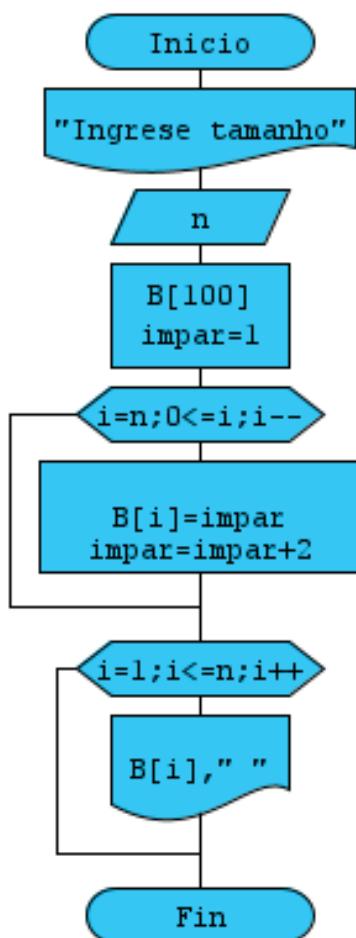
Determinar restricciones: usar datos enteros, positivos

Definir proceso:

```
B = [0] * n
for i in range(n-1,-1,-1):
    B[i]=impar
    impar=impar+2
```

2 DISEÑAR DIAGRAMA

ALGORITHM



DESK CHECK

```
Ingrese tamanhon = 5
9 7 5 3 1
```

Prueba de escritorio (PD)			
Inicio			
n	B	impar	i
5	B[5] = 1	1	5
	B[4] = 3	3	4
	B[3] = 5	5	3
	B[2] = 7	7	2
	B[1] = 9	9	1
	B[0] = 11	11	0
	13	-1	
		1	
		2	
		3	
		4	
		5	
		6	

SOURCE IN PYTHON

```

n = -1
import timeit
impar=1
B=[]

# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo="))
    if n> 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
B = [0] * n
for i in range(n-1,-1,-1):
    B[i]=impar
    impar=impar+2

print("serie de impares desc")
print(B)
total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)

```

DESK CHECK

```
Introduce un número positivo= 5
serie de impares desc
[9, 7, 5, 3, 1]
tiempo en ejecucion 1.4100005500949
```

PROBLEMA

18. Generar y desplegar un vector A de N elementos que tenga la forma siguiente:

Sea n = 10

1	10	3	20	5	30	7	40	9	50
---	----	---	----	---	----	---	----	---	----

ANALISIS

SOLUCION

Leer n como dimension para vector A=[]

validar que sea un entero positivo

Leer x como elemento del vector A=[]

Mostrar A[nelementos]

intercambiar posiciones de elementos de vector

A=[]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

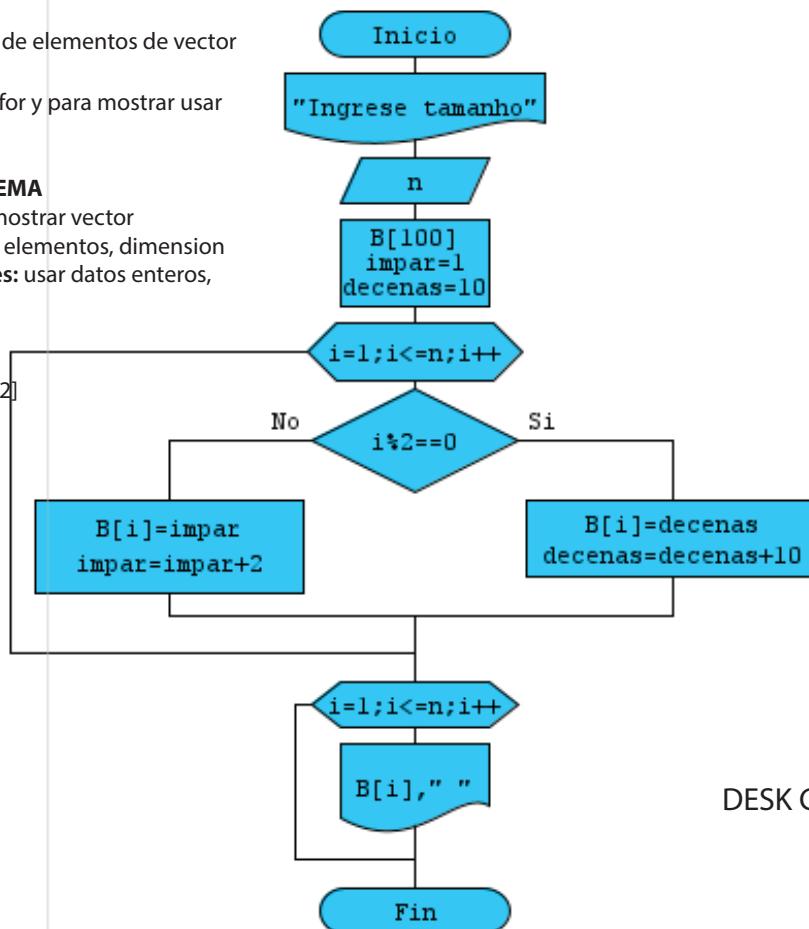
Determinar restricciones: usar datos enteros,

positivos

Definir proceso:

```
aux=B[con+1]
B[con+1]=B[n-2]
B[n-2]=aux
```

2 DISEÑAR DIAGRAMA



DESK CHECK

Ingresar tamaño = 10
10 1 20 3 30 5 40 7 50 9

Prueba de escritorio (PDE's)				
Inicio				
n	B	impar	de...	i
10	B[1] = 10	1	10	1
	B[2] = 1	3	20	2
	B[3] = 20	5	30	3
	B[4] = 3	7	40	4
	B[5] = 30	9	50	5
	B[6] = 5	11	60	6
	B[7] = 40		7	
	B[8] = 7		8	
	B[9] = 50		9	
	B[10] = 9		10	

SOURCE IN PYTHON

```

n = -1
x=-1
con=0
aux=0
import timeit
B=[]
C=[]
# Simulación de do...while
while True:
    n = int(input("Introduce un número positivo="))
    if n> 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")

for i in range(n):
    x = int(input("Introduce valor en="))
    B.append(x)
print("Vector llenado")
print(B)
#intercambiar posiciones por el segundo y penultimo
aux=B[con+1]
B[con+1]=B[n-2]
B[n-2]=aux
#mostramos vector intercambiando posiciones
print("Vector final ")
print(B)

total_time= timeit.timeit(number=1000)
print("tiempo en ejecución",total_time)

```

DESK CHECK



PROBLEMA

19. Crear una matriz con dimensiones iguales o desiguales, que reciba elementos ingresados por teclado y mostrar la matriz.

ANALISIS

SOLUCION

Leer n como dimension para matriz A= []

validar que sea un entero positivo

Leer filas como elemento del M = [] []

Leer columnas como elemento del M = [] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar matriz

Identificar datos a leer: elementos, dimension

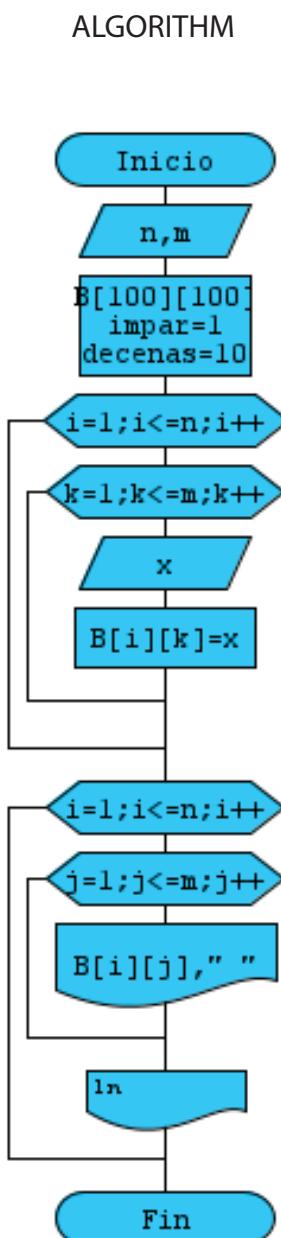
Determinar restricciones: usar datos enteros,

positivos

Definir proceso:

M[filas][columnas]=x

2 DISEÑAR DIAGRAMA



DESK CHECK

```

n = 3
m = 2
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
1 2
3 4
5 6
    
```

Prueba de escritorio (PDE's)								
Inicio								
n	m	B	...	i	k	x	j	
3	2	B[1][1]=1	1	10	1	1	1	
		B[2][1]=2		2	2	2	2	
		B[1][2]=3		3	3	3	3	
		B[2][2]=4		4	1	4	1	
		B[1][3]=5		1	2	5	2	
		B[2][3]=6		2	3	6	3	

SOURCE IN PYTHON

```

f = -1
import timeit
impar=1
M=[]

# Simulación de do...while
while True:
    f = int(input("Introduce número filas= "))
    c = int(input("Introduce número columnas= "))
    if f > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m
for i in range(f):
    Fila=[]
    for i in range(c):
        x = int(input("Introduce un valor= "))
        Fila.append(x)
    M.append(Fila)

print("la matriz es=")
print(M)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

Introduce un valor= 4
Introduce un valor= 5
Introduce un valor= 6
Introduce un valor= 3
Introduce un valor= 2
Introduce un valor= 1
Introduce un valor= 6
Introduce un valor= 25
Introduce un valor= 25
Introduce un valor= 12
Introduce un valor= 25
Introduce un valor= 25
la matriz es=
[[1, 2, 3], [4, 5, 6], [3, 2, 1], [6, 25, 25], [12, 25, 25]]
tiempo en ejecucion 1.4500023098662496e-05
    
```

PROBLEMA

20. Creamos una matriz identidad con ceros y unos. SI N=3 Y M=3

ANALISIS

matriz	
SOLUCION	

Ler n como dimension para matriz A=[]

validar que sea un entero positivo

Ler filas como elemento del M =[] []

Ler columnas como elemento del M=[] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar matriz

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, posetivos

Definir proceso:

```
si fila== columnas entonces
    M[filas][columnas]=1
caso contrario
    M[filas][columnas]=0
```

2 DISEÑAR DIAGRAMA

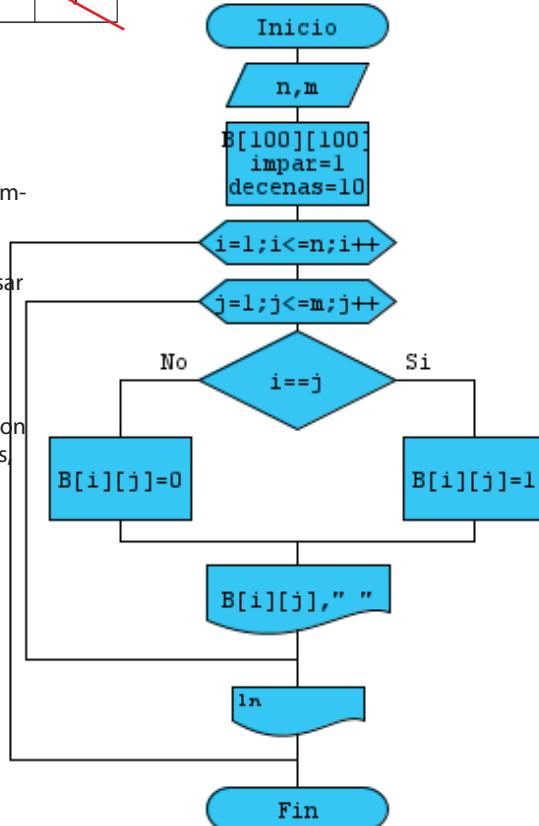
DESK CHECK

```
n = 5
m = 5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

Prueba de escritorio (PDE)

n	m	B	i	j
		B[5][1]=0	5	5
		B[1][2]=0	6	6
		B[2][2]=1	1	
		B[3][2]=0	2	
		B[4][2]=0	3	
		B[5][2]=0	4	
		B[1][3]=0	5	
		B[2][3]=0	6	
		B[3][3]=1	1	
		B[4][3]=0	2	
		B[5][3]=0	3	
		B[1][4]=0	4	
		B[2][4]=0	5	
		B[3][4]=0	6	
		B[4][4]=1	1	
		B[5][4]=0	2	
		B[1][5]=0	3	
		B[2][5]=0	4	
		B[3][5]=0	5	
		B[4][5]=0	6	
		B[5][5]=1	1	

ALGORITHM



SOURCE IN PYTHON

```
f=-1
import timeit
impar=1
M=[]

# Simulación de do...while
while True:
    f = int(input("Introduce número filas="))
    c = int(input("Introduce número columnas="))
    if f > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m
for i in range(f):
    Fila=[]
    for j in range(c):
        if i==j:
            Fila.append(1)
        else:
            Fila.append(0)
    M.append(Fila)

print("la matriz es=")
#print(M,end="")
for filas in M:
    print(filas)

total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
```

DESK CHECK

Introduce numero filas= 5

Introduce número columnas= 5

la matriz es=

```
[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
```

tiempo en ejecucion 1.3999990187585354e-05

PROBLEMA

21. Generar la matriz $n \times n$ triangular inferior izquierdo

ANALISIS

0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0

SOLUCION

Ler n como dimension para matriz A=[]
validar que sea un entero positivo

Ler filas como elemento del M = [] []

Ler columnas como elemento del M = [] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar
for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros,
positivos

Definir proceso:

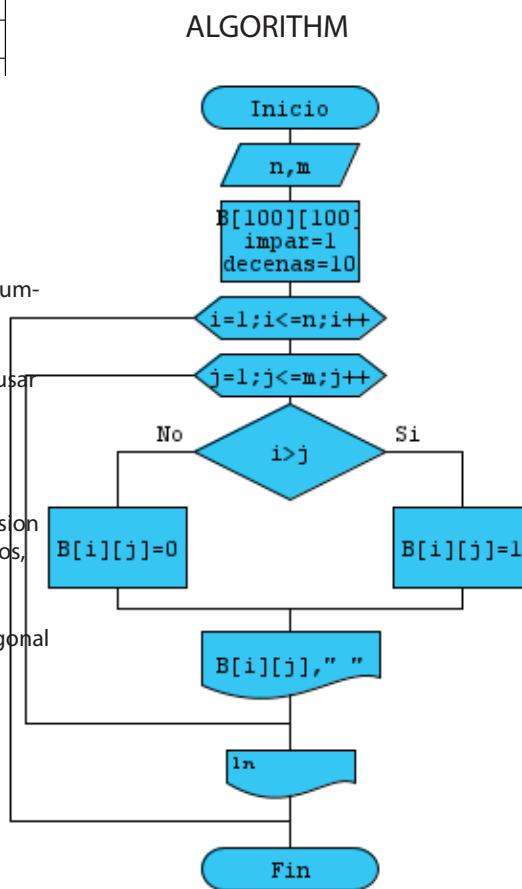
si $i > j$ nota $i \geq j$ toma mas la diagonal
superiorrr

$B[i][j]=1$

caso contrario

$B[i][j]=0$

2 DISEÑAR DIAGRAMA



DESK CHECK

n = 5
m = 5
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0

Prueba de escritorio (PDE's)						
Inicio	n	m	B	i	j
			B[3][1]=0		3	3
			B[4][1]=0		4	4
			B[5][1]=0		5	5
			B[1][2]=1		6	6
			B[2][2]=0			1
			B[3][2]=0			2
			B[4][2]=0			3
			B[5][2]=0			4
			B[1][3]=1			5

SOURCE IN PYTHON

```

f = -1
import timeit
impar=1
M=[]

# Simulación de do...while
while True:
    f = int(input("Introduce número filas= "))
    c = int(input("Introduce número columnas= "))
    if f > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
#B = [0][0] * n*m
for i in range(f):
    Fila=[]
    for j in range(c):
        if i>j:
            Fila.append(1)
        else:
            Fila.append(0)
    M.append(Fila)

print("la matriz es=")
#print(M,end="")
for filas in M:
    print(filas)

total_time=timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

Introduce número filas= 5
Introduce número columnas= 5
la matriz es=
[0, 0, 0, 0, 0]
[1, 0, 0, 0, 0]
[1, 1, 0, 0, 0]
[1, 1, 1, 0, 0]
[1, 1, 1, 1, 0]
tiempo en ejecucion 1.4100020052865148e-05
    
```

PROBLEMA

22. Generar y mostrar la siguiente matriz gusanito de tamaño $n \times m$

ANALISIS

1	2	3
6	5	4
7	8	9

12	11	10
----	----	----

SOLUCION

Ler n como dimension para matriz A= []

validar que sea un entero positivo

Ler filas como elemento del M = [] []

Ler columnas como elemento del M= [] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, posetivos

Definir proceso:

for i in range(f):

if(i%2==0):

for j in range(f):

M[i][j]=cc

cc=cc+1

else:

for j in range(c-1,-1,-1):

print(j,"=",i)

M[i][j]=cc

cc = cc + 1

2 DISEÑAR DIAGRAMA

DESK CHECK

```

n = 5
m = 5
1 ,2 ,3 ,4 ,5 ,
10 ,9 ,8 ,7 ,6 ,
11 ,12 ,13 ,14 ,15 ,
20 ,19 ,18 ,17 ,16 ,
21 ,22 ,23 ,24 ,25 .

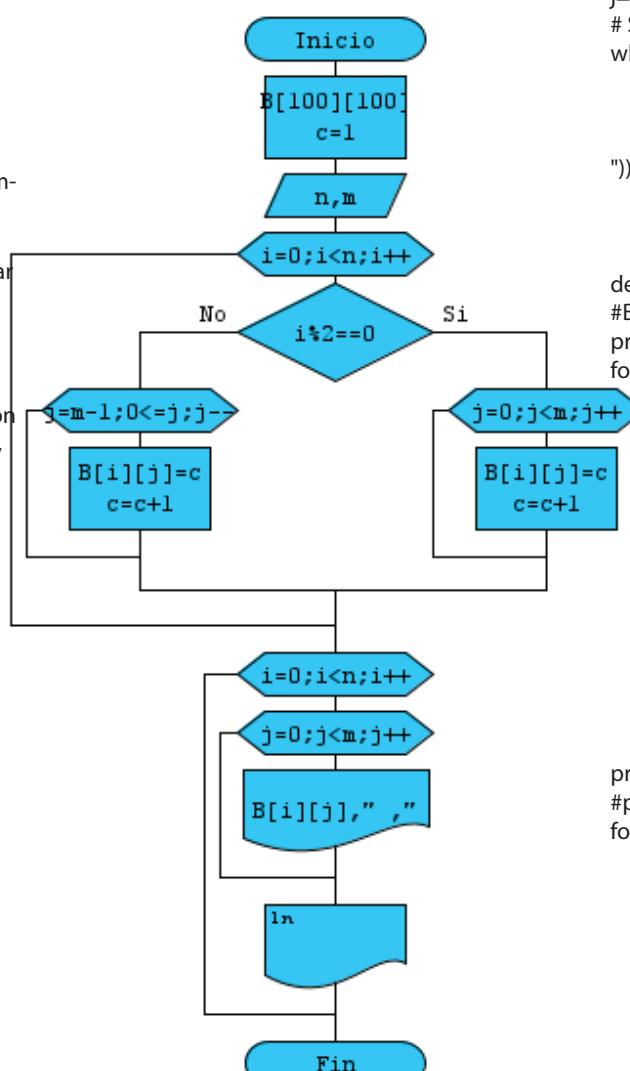
```

Prueba de escritorio (PDE's)

Inicio |

B	c	n	m	i	j
B[0][0]=1	1	5	5	0	0
B[1][0]=2	2			1	1
B[2][0]=3	3			2	2
B[3][0]=4	4			3	3
B[4][0]=5	5			4	4
B[4][1]=6	6			5	5
B[3][1]=7	7			0	4
B[2][1]=8	8			1	3
B[1][1]=9	9			2	2
B[0][1]=...10				3	1

ALGORITHM



SOURCE IN PYTHON

```

f = -1
import timeit
impar=1
#M=[]
cc=1
j=0
# Simulación de do...while
while True:
    f = int(input("Introduce número filas="))
    M = [[0] * f for _ in range(f)]
    c = int(input("Introduce número columnas="))
    if f > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m
    print(M)
    for i in range(f):
        if(i%2==0):
            for j in range(f):
                M[i][j]=cc
                cc=cc+1
        else:
            for j in range(c-1,-1,-1):
                print(j,"=",i)
                M[i][j]=cc
                cc = cc + 1
    print("la matriz es=")
    #print(M,end="")
    for filass in M:
        print(filass)

```

DESK CHECK

```

la matriz es=
[1, 2, 3, 4, 5]
[10, 9, 8, 7, 6]
[11, 12, 13, 14, 15]
[20, 19, 18, 17, 16]
[21, 22, 23, 24, 25]
tiempo en ejecucion 1.2900000001536682e-05

```


PROBLEMA

24. Dada una matriz A de $n \times m$, obtener la media aritmética de cada fila de A.

ANALISIS

5	26	4	1	4
1	2	5	6	1
6	7	6	89	2
4	71	48	0	99
0	3	9	11	3

SOLUCION

Ler n como dimension para matriz M= []

validar que sea un entero positivo

Leer filas como elemento del M = [] []

Leer columnas como elemento del M = [] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, posetivos

Definir proceso:

suma cada fila de la matriz

suma=suma +B[i][j]

2 DISEÑAR DIAGRAMA

DESK CHECK

```

n = 3
x = 2
x = 1
x = 2
Suma es fila=5
x = 3
x = 4
x = 5
Suma es fila=12
x = 1
x = 2
x = 3
Suma es fila=6
2 1 2
3 4 5
1 2 3

```

Prueba de escritorio (PDE's)

n	B	...	i	j	x
3	B[0][0]=2	0	0	0	2
	B[1][0]=1	0	1	1	1
	B[2][0]=2	2	2	2	2
	B[0][1]=3	3	3	3	3
	B[1][1]=4	5	0	0	4
	B[2][1]=5	0	1	1	5
	B[0][2]=1	3	2	2	1
	B[1][2]=2	7	3	3	2
	B[2][2]=3	12	0	3	
		0	1		

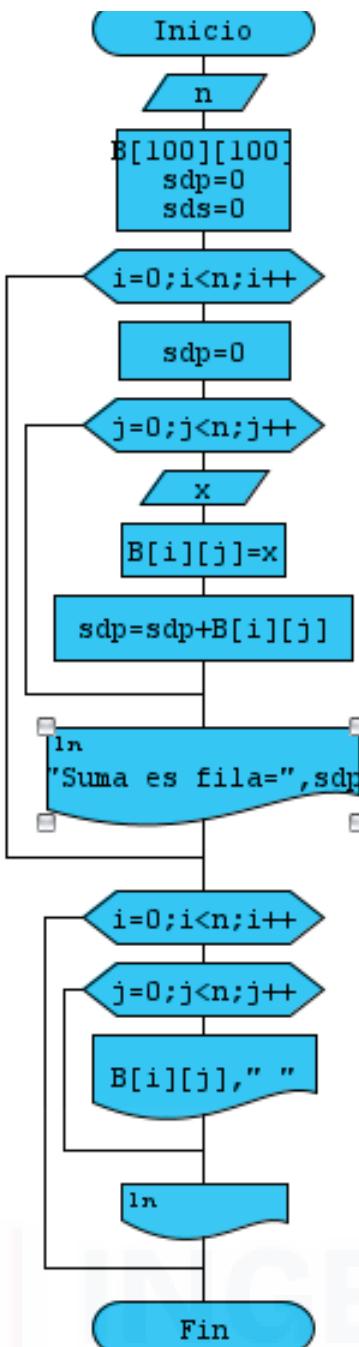
ALGORITHM

$$5 + 26 + 4 + 1 + 4 = 40/5$$

$$1 + 2 + 5 + 6 + 1 = 15/5$$

$$6 + 7 + 6 + 89 + 2 = 110/5$$

$$4 + 71 + 48 + 0 + 99 = 222/5$$



SOURCE IN PYTHON

```

n=-1
import timeit
suma=0
M=[]
# Simulación de do...while
while True:
    n=int(input("Introduce matriz cuadrada= "))

    if n>0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m

for i in range(n):
    Fila=[]
    for j in range(n):
        x = int(input("Introduce filas="))
        Fila.append(x)
    M.append(Fila)

for i in range(n):
    suma=0
    for j in range(n):
        suma = suma+ M[i][j]
    print("suma fila es=",suma)

for filass in M:
    print( filass)

# para mostrar matriz
print("la matriz es=")
#print(M,end="")
for filass in M:
    print( filass)

total_time= timeit.timeit(number=1000)
print("tiempo en ejecucion",total_time)

```

DESK CHECK

```

suma fila es= 6
suma fila es= 14
suma fila es= 6
[3, 2, 1]
[4, 5, 5]
[1, 2, 3]
la matriz es=
[3, 2, 1]
[4, 5, 5]
[1, 2, 3]
tiempo en ejecucion 1.4399993233382702e-05

```

PROBLEMA

25. Ingresar números en una matriz S de dimensión M*N y obtenga la suma total de sus elementos.

ANALISIS

SOLUCION

Leer n como dimension para matriz M=[]

validar que sea un entero positivo

Leer filas como elemento del M =[] []

Leer columnas como elemento del M =[] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

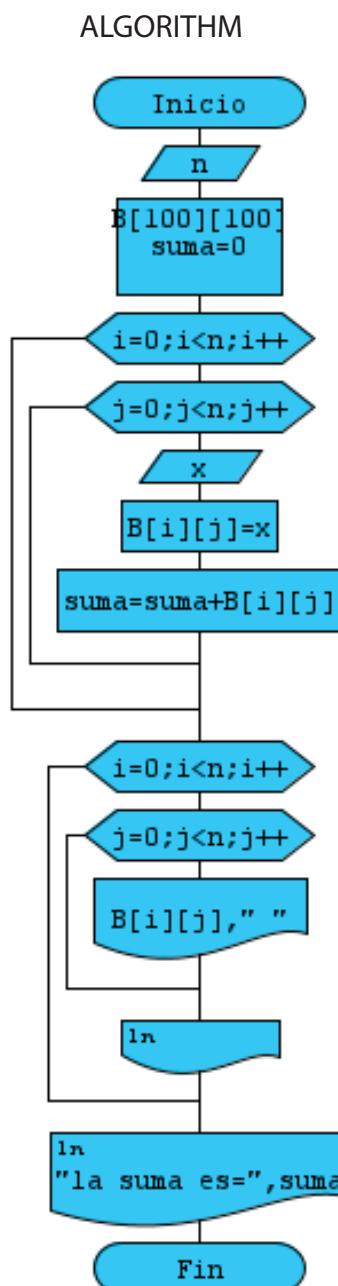
Determinar restricciones: usar datos enteros, posetivos

Definir proceso:

suma cada elemento de matriz

suma=suma +B[i][j]

2 DISEÑAR DIAGRAMA



DESK CHECK

```

n = 3
x = 3
x = 2
x = 1
x = 1
x = 2
x = 3
x = 5
x = 4
x = 6
3 2 1
1 2 3
5 4 6
la suma es=27
  
```

Prueba de escritorio (PDE's)

n	B	suma	i	j	x
3	B[0][0] = 3 0	0	0	3	
	B[1][0] = 2 3	1	1	2	
	B[2][0] = 1 5	2	2	1	
	B[0][1] = 1 6	3	3	1	
	B[1][1] = 2 7	0	0	2	
	B[2][1] = 3 9	1	1	3	
	B[0][2] = 5 12	2	2	5	
	B[1][2] = 4 17	3	3	4	
	B[2][2] = 6 21	0	6		
		27	1		
			2		

SOURCE IN PYTHON

```

c=-1
import timeit
suma=0
M=[]
# Simulación de do...while
while True:
    c= int(input("Introduce filas= "))
    f= int(input("Introduce columnas= "))

    if c>0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
#B = [0][0] * n*m

for i in range(c):
    Fila=[]
    for j in range(f):
        x = int(input("Introduce fila="))
        Fila.append(x)
    M.append(Fila)

for i in range(f):
    for j in range(c):
        suma = suma+ M[i][j]

# para mostrar matriz
print("la matriz es=")
#print(M,end="")
for filass in M:
    print(filass)
print("suma fila es=",suma)
total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
  
```

SK CHECK

```

la matriz es=
[1, 2, 3]
[6, 5, 4]
[7, 8, 9]
suma fila es= 45
tiempo en ejecucion 1.4099990949034691e-05
  
```

PROBLEMA

26. Almacene en una matriz de $M \times N$, con números enteros y obtenga la cantidad de pares e impares.

ANALISIS

SOLUCION

Ler n como dimension para matriz $M = []$
 validar que sea un entero positivo
 Leer filas como elemento del $M = [] []$
 Leer columnas como elemento del $M = [] []$
 Mostrar elementos de matriz $M[nfilas][ncolumnas]$

para leer elemento usar for y para mostrar usar
 for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros,
 positivos

Definir proceso:

suma cada fila de la matriz

SI $B[i][j] \% 2 == 0$ entonces

 spar=spar+B[i][j]

caso contrario

 impar=impar+B[i][j]

2 DISEÑAR DIAGRAMA

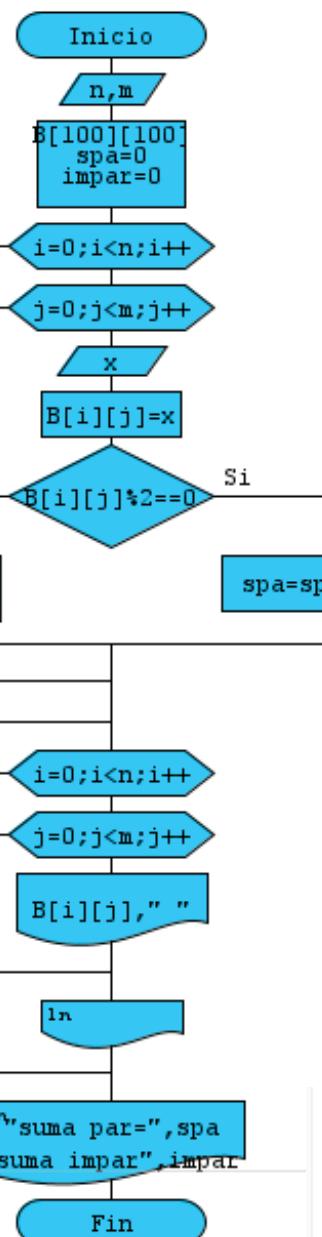
DESK CHECK

suma par=20 suma impar=25

Prueba de escritorio (PDE's)

n	m	B	spa	i	impar	j	x
3	3	$B[0][0]=1$	0	0	0	0	1
		$B[1][0]=2$	2	1	1	1	2
		$B[2][0]=3$	6	2	4	2	3
		$B[0][1]=4$	12	3	9	3	4
		$B[1][1]=5$	20	0	18	0	5
		$B[2][1]=6$	1	25	1	6	
		$B[0][2]=9$	2		2	9	
		$B[1][2]=8$	3		3	8	
		$B[2][2]=7$			0	7	

ALGORITHM



SOURCE IN PYTHON

```

c=-1
import timeit
spar=0
impar=0
M=[]
# Simulación de do...while
while True:
    c=int(input("Introduce filas="))
    f=int(input("Introduce columnas="))

    if c < 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m

    for i in range(c):
        Fila=[]
        for j in range(f):
            x = int(input("Introduce filas="))
            Fila.append(x)
        M.append(Fila)

    for i in range(f):
        for j in range(c):
            if M[i][j] % 2 == 0:
                spar = spar + M[i][j]
            else:
                impar = impar + M[i][j]

    # para mostrar matriz
    print("la matriz es=")
    #print(M,end="")
    for filass in M:
        print(filass)
    print("Suma pares=",spar)
    print("Suma impares=",impar)
    total_time= timeit.timeit(number=1000)
    print("tiempo en ejecucion",total_time)

```

DESK CHECK

la matriz es=

[1, 2, 2]

[1, 2, 5]

[4, 8, 9]

Suma pares= 18

Suma impares= 16

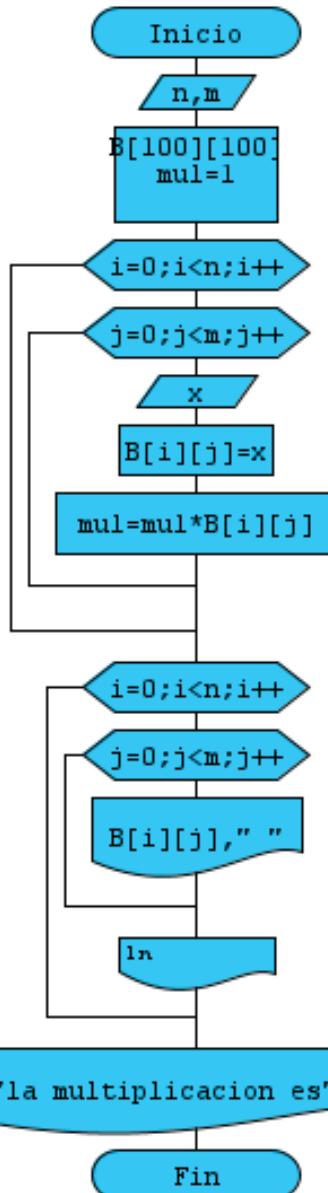
tiempo en ejecucion 1.4399993233382702e-05

PROBLEMA

27. Leer una matriz A de orden M x N y un número K. Multiplicar todos los elementos de la matriz por el número K. Mostrar la matriz resultante

ANALISIS

ALGORITHM



SOLUCION

Leer n como dimension para matriz M=[]
validar que sea un entero positivo

Leer filas como elemento del M=[] []

Leer columnas como elemento del M=[] []

Mostrar elementos de matriz M[nfilas][ncolumnas]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, positivos

Definir proceso:

suma cada fila de la matriz
smul=mul*B[i][j]

2 DISEÑAR DIAGRAMA

DESK CHECK

Consola

3 3 3
3 3 3
la multiplicacion es 19683

Prueba de escritorio (PDE's)

Inicio					
n m	B	mul	i	j	x
3 3	B[0][0]=3 1	0	0	3	
	B[1][0]=3 3	1	1	3	
	B[2][0]=3 9	2	2	3	
	B[0][1]=3 27	3	3	3	
	B[1][1]=3 81	0	0	3	
	B[2][1]=3 243	1	1	3	
	B[0][2]=3 729	2	2	3	
	B[1][2]=3 2187	3	3	3	
	B[2][2]=3 6561		0	3	
	19683		1		

SOURCE IN PYTHON

```

c=-1
import timeit
mul=1

M=[]
# Simulación de do...while
while True:
    c = int(input("Introduce filas="))
    f = int(input("Introduce columnas="))

    if c> 0:
        break
    print("El número debe ser positivo. Intentalo de nuevo.")
    #B = [0][0] * n*m

for i in range(c):
    Fila=[]
    for j in range(f):
        x = int(input("Introduce filas="))
        Fila.append(x)
    M.append(Fila)

for i in range(f):
    for j in range(c):
        mul = mul+ M[i][j]

# para mostrar matriz
print("la matriz es=")
#print(M,end="")
for filass in M:
    print( filass)
print("Multiplicacion es=",mul)

total_time= timeit.timeit( number=1000)
print("tiempo en ejecucion",total_time)
    
```

DESK CHECK

```

la matriz es=
[1, 1, 2]
[5, 3, 6]
[5, 1, 2]
Multiplicacion es= 27
tiempo en ejecucion 1.39999901875853
    
```


PROBLEMA

29. Llenar una matriz NxM y obtener la multiplicación de la diagonal principal y la diagonal secundaria:

$M[0][0] = \text{multiplicación diagonal primaria} = 1 * 3 * 6 = 18$

Multiplicación diagonal secundaria = $3 * 3 * 4 = 36$

ANALISIS

SOLUCION

Leer n como dimension para matriz B=[]

validar que sea un entero positivo

Mostrar B[n elementos][n elementos]

para leer elemento usar for y para mostrar usar for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, positivos

Definir proceso:

para diagonal principal

 si $i == j$ entonces

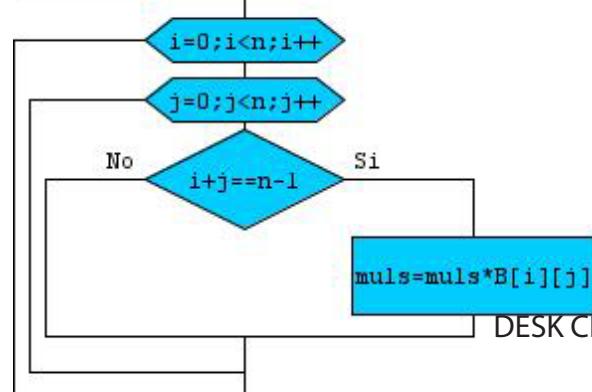
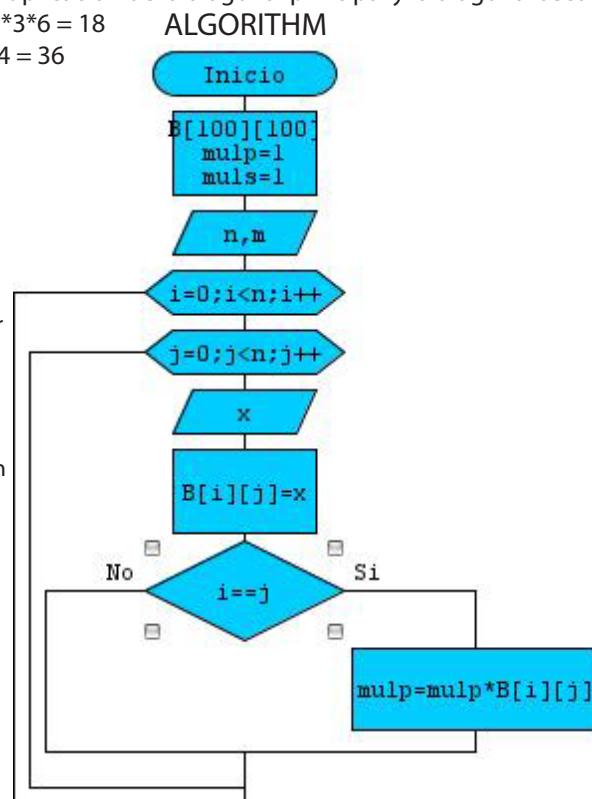
 suma=suma+B[i][j]

diagonal secundario

 si $i+j==n-1$ entonces

 suma=suma+B[i][j]

2 DISEÑAR DIAGRAMA

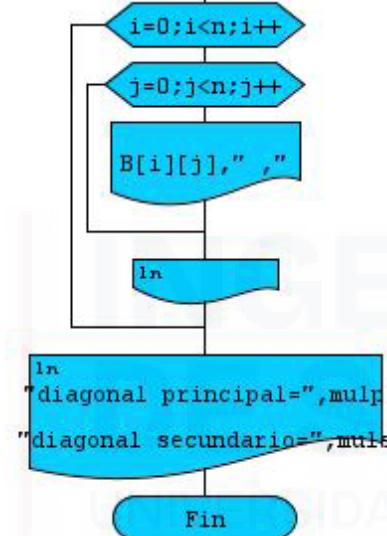


DESK CHECK

```

m = 3
x = 1
x = 2
x = 2
x = 3
x = 3
x = 2
x = 2
x = 1
x = 2
x = 1
1 ,2 ,2 ,
3 ,3 ,2 ,
1 ,2 ,1 ,
diagonal principal=3diagonal secundario=6
    
```

Prueba de escritorio (PDE's)					
M	Inicio	mulp	muls	n	m
B[0][0]=1	1	1	3	3	3
B[1][0]=2	1	2			
B[2][0]=2	3	6			
B[0][1]=3	3	6			
B[1][1]=3					
B[2][1]=2					



SOURCE IN PYTHON

```

n=-1
import timeit
sdp=0
sds=0
M=[]
# Simulación de do...while
while True:
    n= int(input("Introduce matriz cuadrada="))

    if n > 0:
        break
    print("El número debe ser positivo. Inténtalo de nuevo.")
    #B = [0][0] * n*m

    for i in range(n):
        Fila=[]
        for j in range(n):
            x = int(input("Introduce filas="))
            Fila.append(x)
        M.append(Fila)

    for i in range(n):
        for j in range(n):
            print(i,j)
            if i==j:
                sdp = sdp * M[i][j]
            for i in range(n):
                if i+j == (n - 1):
                    sds = sds * M[i][j]
                    print(sdp)

    for filass in M:
        print( filass)

    # para mostrar matriz
    print("la matriz es=")
    #print(M,end="")
    for filass in M:
        print( filass)
    print("Suma Diagonal principal=",sdp)
    print("Suma Diagonal secundario=",sds)
    total_time= timeit.timeit( number=1000)
    print("tiempo en ejecucion",total_time)
    
```

la matriz es=

[3, 3, 3]

[3, 1, 2]

[5, 2, 3]

Suma Diagonal principal= 7

Suma Diagonal secundario= 9

tiempo en ejecucion 1.4599994756281376e-05



PROBLEMA

30. Generar la siguiente matriz espejo donde $n=m$, y $n = \text{tamaño impar}$

Si $n=5$, $m = 5$

ANALISIS

SOLUCION

Leer n como dimension para vector $A=[]$

validar que sea un entero positivo

Mostrar $A[n\text{elementos}]$

generar la serie

para leer elemento usar for y para mostrar usar

for

1 ANALIZAR EL PROBLEMA

Especificar resultado: mostrar vector

Identificar datos a leer: elementos, dimension

Determinar restricciones: usar datos enteros, positivos

Definir proceso:

generar usando for para potencia

Función generar_matriz:

Argumento: n es la dimensión de la matriz (en este caso,

Genera: Una matriz de tamaño

$n \times n$ con valores incrementales comenzando en

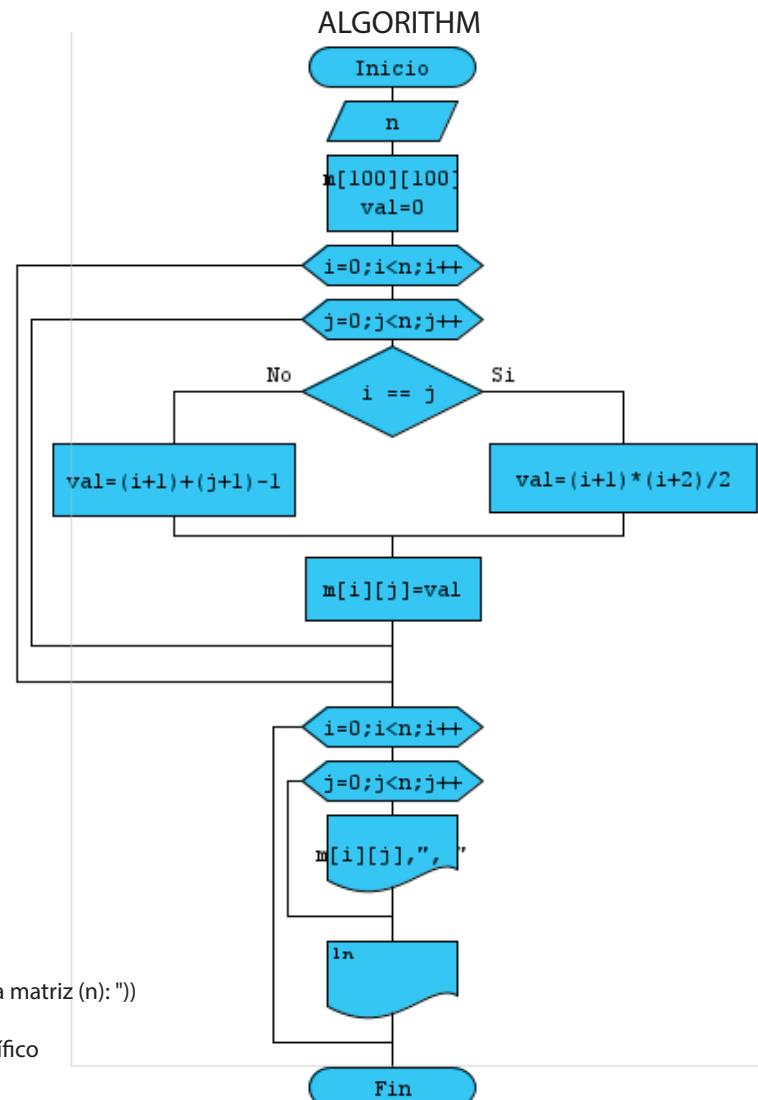
Función matriz_espejo:

Argumento: Recibe la matriz original.

Crea: Una nueva matriz con los elementos

reflejados sobre la diagonal secundaria. SOURCE IN PYTHON

2 DISEÑAR DIAGRAMA



DESK CHECK

```

n = 5
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 6, 6, 7,
4, 5, 6, 10, 8,
5, 6, 7, 8, 15,

```

Prueba de escritorio (PDE)

Inicio

n	m	...	i	j
5	m[0][0]=1	0	0	0
	m[1][0]=2	1	1	1
	m[2][0]=3	2	2	2
	m[3][0]=4	3	3	3
	m[4][0]=5	4	4	4
	m[0][1]=2	5	5	5
	m[1][1]=3	2	0	0
	m[2][1]=4	3	1	1
	m[3][1]=5	4	2	2
	m[4][1]=6	5	3	3
	m[0][2]=3	6	4	4
	m[1][2]=4	7	5	5
	m[2][2]=5	8	6	6
	m[3][2]=6	9	7	7
	m[4][2]=7	10	8	8

```

# Leer el tamaño de la matriz
n = int(input("Introduce el tamaño de la matriz (n):"))

# Generar la matriz con el patrón específico
matriz = []
for i in range(n):
    fila = []
    for j in range(n):
        if i == j:
            valor = (i + 1) * (i + 2) // 2
        else:
            valor = (i + 1) + (j + 1) - 1
        fila.append(valor)
    matriz.append(fila)

# Imprimir la matriz original
print("Matriz Original:")
for fila in matriz:
    print(fila)

# Generar la matriz espejo
espejo = []
for i in range(n):
    fila_espejo = []
    for j in range(n):
        fila_espejo.append(matriz[n - j - 1][n - i - 1])
    espejo.append(fila_espejo)

# Imprimir la matriz espejo
print("\nMatriz Espejo:")
for fila in espejo:
    print(fila)

```

DESK CHECK

Matriz Original:

```

[1, 2, 3, 4, 5]
[2, 3, 4, 5, 6]
[3, 4, 6, 6, 7]
[4, 5, 6, 10, 8]
[5, 6, 7, 8, 15]

```

Matriz Espejo:

```

[15, 8, 7, 6, 5]
[8, 10, 6, 5, 4]
[7, 6, 6, 4, 3]
[6, 5, 4, 3, 2]
[5, 4, 3, 2, 1]

```



PRÁCTICA GENERAL DE SIS - III # 3

NOMBRE: PORFIEIO ELIAS QUISPE QUISPE

PARALELO: "1A"

TURNO : MAÑANA

DOCENTE: ING. YOLANDA ESCOBAR

GESTION I-2024



INGENIERÍA
DE SISTEMAS
UNIVERSIDAD PÚBLICA DE EL ALTO