UCA
Universidad de Cádiz

Introducción a la Programación
Grado en Ingeniería Informática

Teoría - Curso 2015-2016

Contenido 3 – Programación Estructurada

Tema 3.- Programación Estructurada

3.1.-Características de la Programación Estructurada

Características de la Programación Estructurada:

- La programación estructurada es el conjunto de técnicas que incorporan:
  - Recursos abstractos
  - Diseño descendente
  - Estructuras básicas:
    - Secuenciales.
    - Selectivas.
    - Repetitivas.

3.2.-Estructura Secuencial

Diagrama de flujo:

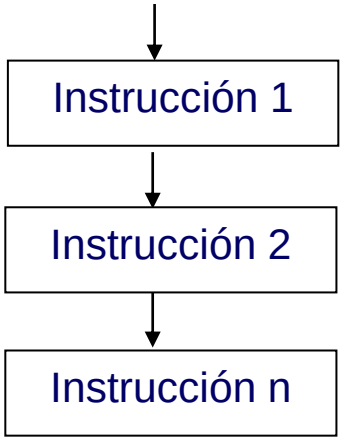
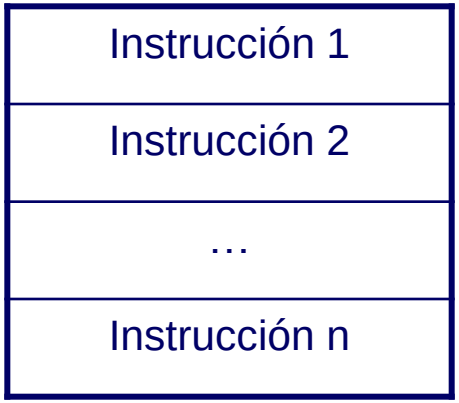


Diagrama N-S:



Pseudocódigo:

```
Instrucción 1
Instrucción 2
   …..
Instrucción n
```

3.2.-Estructura Secuencial

Ejemplo: Cálculo de la suma y producto de dos números

Diagrama de flujo:

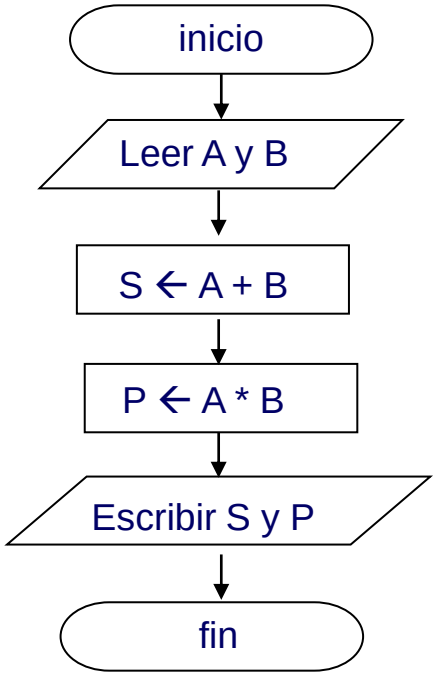


Diagrama N-S:

Algoritmo suma
Leer (A, B)
S← A+ B
P← A* B
Escribir (s, p)
Fin_algoritmo

Pseudocódigo:

```
Algoritmo suma_producto

Principal
  var
    entero : A, B, S, P
  inicio
     escribir ("introduce dos nº")
     leer (A, B)
     S← A+ B
     P← A* B
     escribir (S, P)
  fin_principal
fin_algoritmo
```

3.3.-Estructura Selectiva

La ejecución de un conjunto de instrucciones dependerá del resultado de la evaluación de una determinada condición. La condición se escribe usando expresiones que devuelven un valor lógico (V o F) al evaluarse.

Tipos:

- Simple
- Doble
- Múltiple

3.3.1-Estructura Selectiva Simple

Diagrama de flujo:

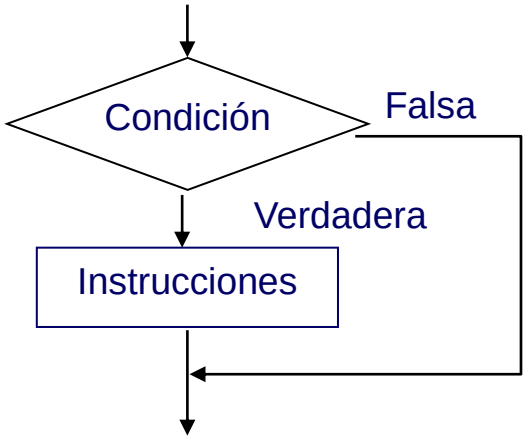
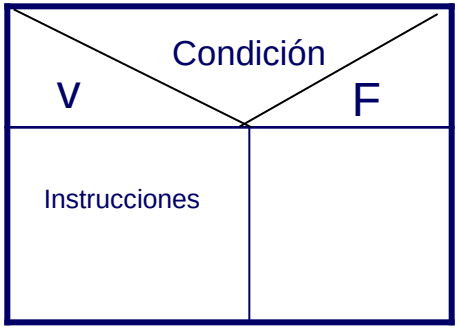


Diagrama N-S:



Pseudocódigo:

si <condición> **entonces**
    <instrucciones>
fin_si

3.3.1-Estructura Selectiva Simple

Ejemplo: Leer un número por teclado y comprobar si el número es mayor que cero

Diagrama de flujo:

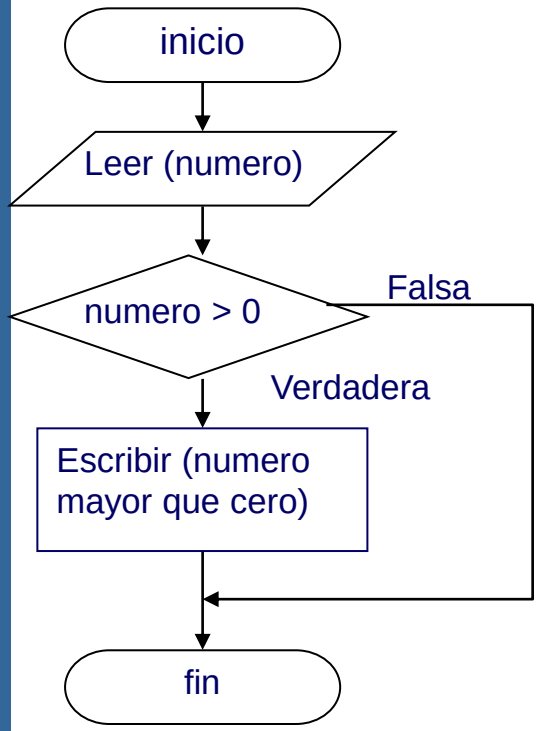
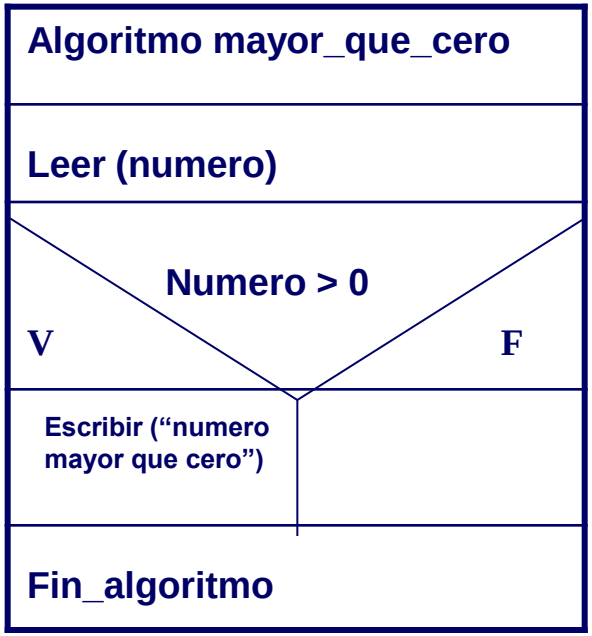


Diagrama N-S:



Pseudocódigo:

```
Algoritmo mayor_que_cero

 Principal
 var
    entero: numero
  inicio
    escribir ("introduce nº")
    leer(numero)
    si (numero >0) entonces
        escribir(" número mayor
                  que cero")
     fin_si
  fin_principal
fin_algoritmo
```

3.3.2.-Estructura Selectiva Doble

Diagrama de flujo:

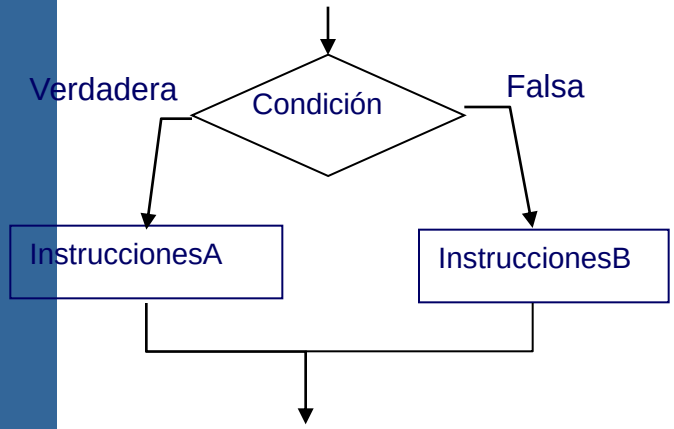
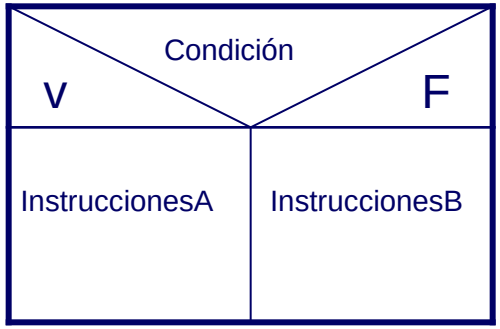


Diagrama N-S:



Pseudocódigo:

si <condición> **entonces**
    <instrucciones A>
si_no
    <instrucciones B>
fin_si

3.3.2.-Estructura Selectiva Doble

Ejemplo: Algoritmo que compruebe si un número introducido por teclado es par o impar

Diagrama de flujo:

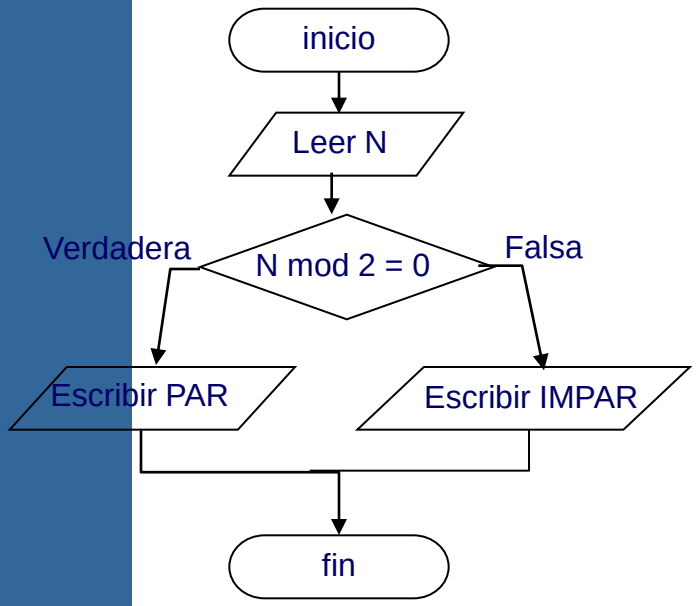
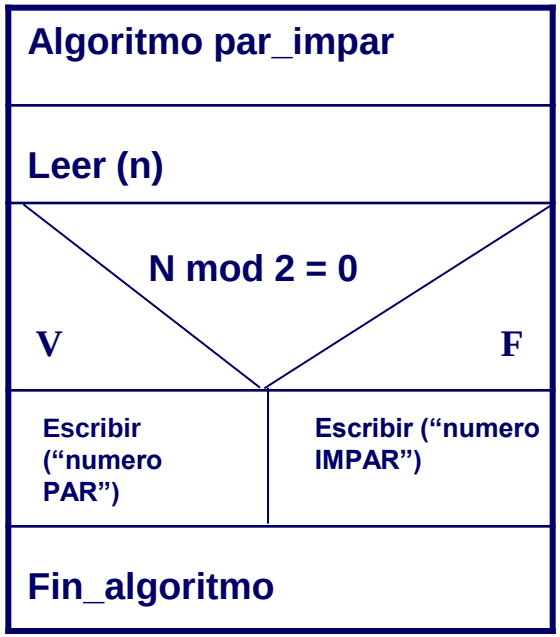


Diagrama N-S:



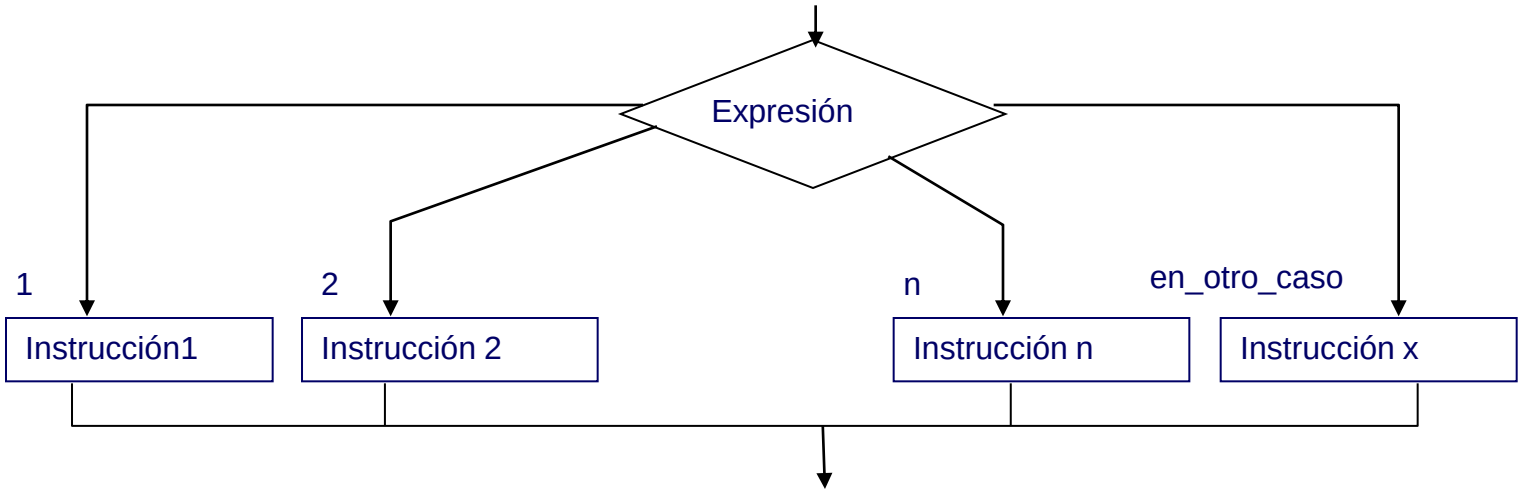
Pseudocódigo:

```
Algoritmo par_impar

Principal
  var
   entero: n
  inicio
   escribir (" introduce nº")
   leer (n)
   si (n mod 2 =0) entonces
                escribir("PAR")
    si_no
                escribir("IMPAR")
    fin_si
  fin_principal
fin_algoritmo
```

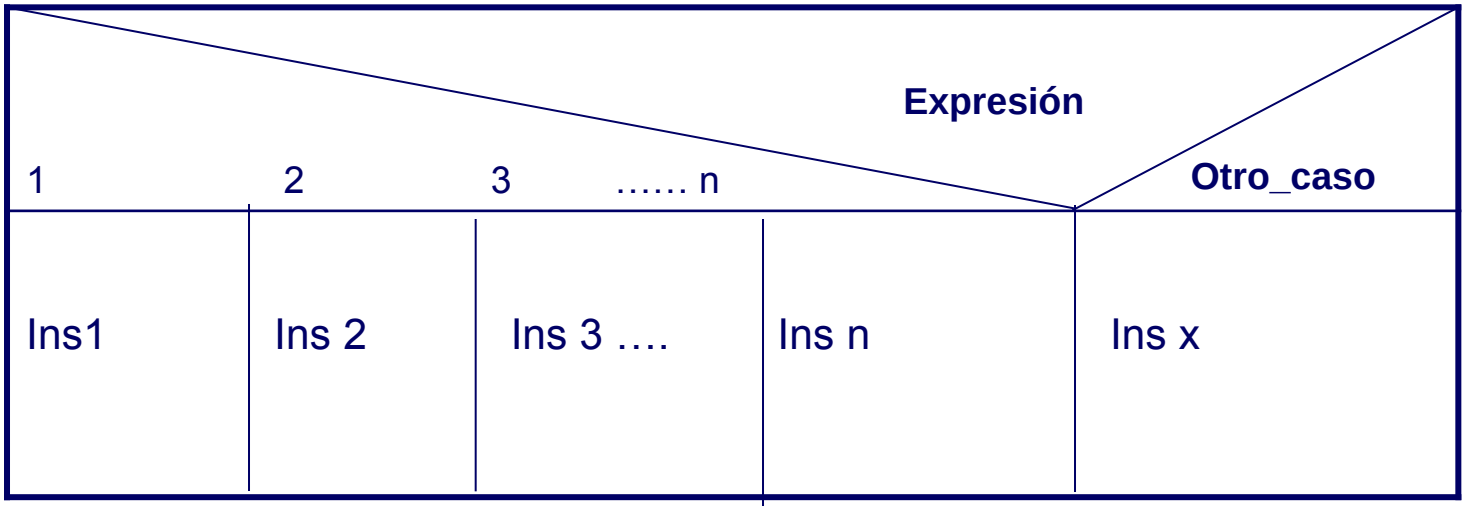
3.3.3.-Estructura Selectiva Múltiple

Diagrama de flujo:



3.3.3.-Estructura Selectiva Múltiple

Diagrama NS:



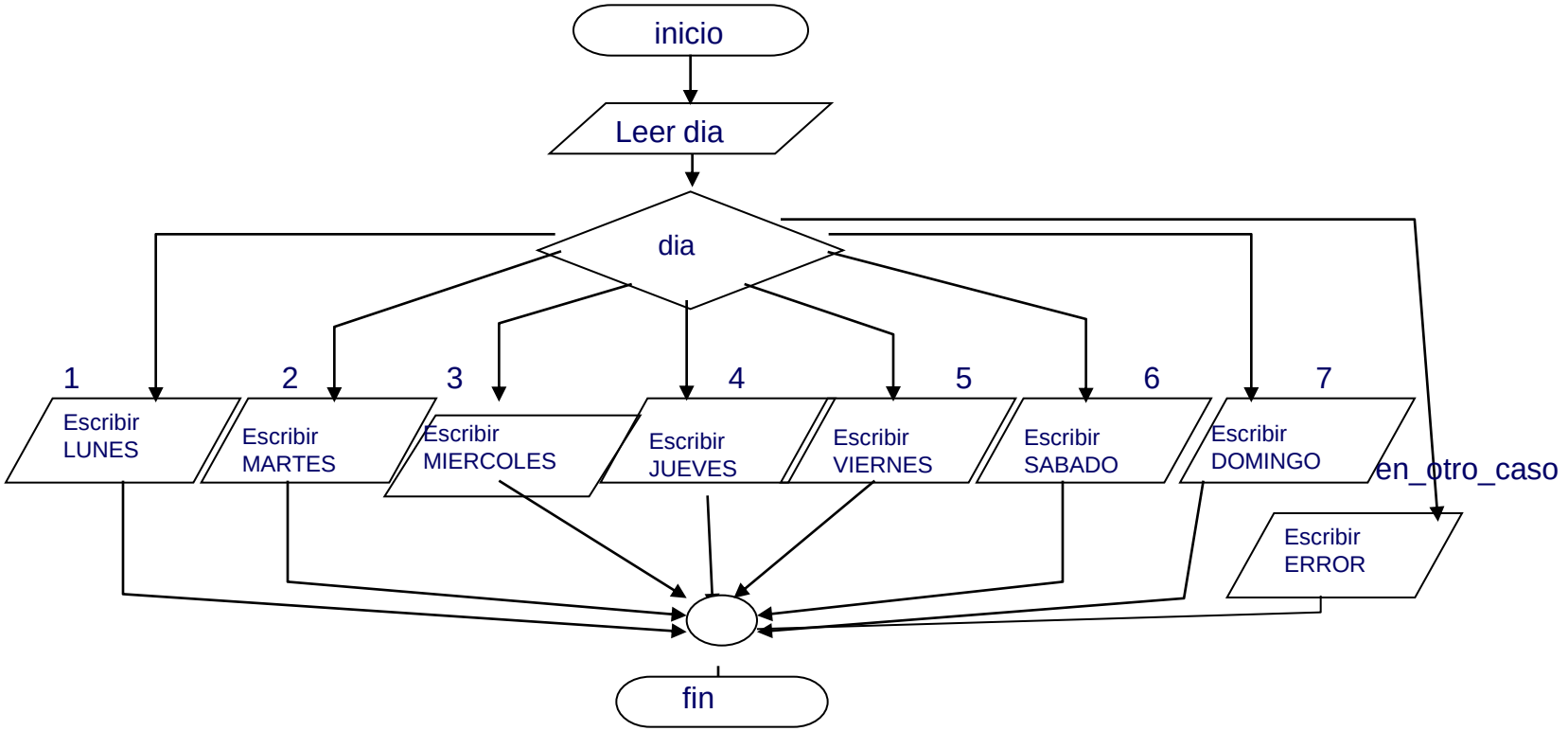
3.3.3.-Estructura Selectiva Múltiple

Pseudocódigo:

segun_sea (expresión) **hacer**
1: instrucción/es 1
2: instrucción/es 2
3: instrucción/es 3
…..
n: instrucción/es n
en_otro_caso: instrucción/es x
fin_segun

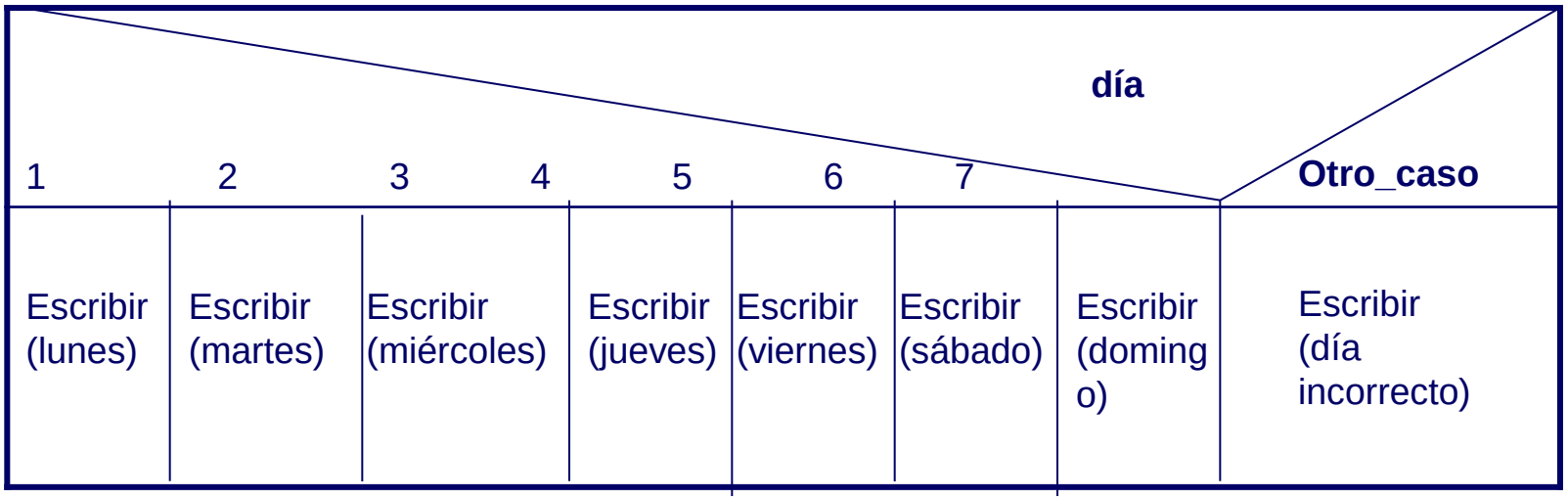
3.3.3.-Estructura Selectiva Múltiple

Ejemplo: Se desea diseñar un algoritmo que escriba los nombres de los días de la semana en función del valor de una variable *dia* introducida por teclado, que representa su posición dentro de la semana.



3.3.3.-Estructura Selectiva Múltiple

Diagrama N-S:



3.3.3.-Estructura Selectiva Múltiple

Pseudocódigo:

```
Algoritmo Nombre_dias

Principal
var
        entero: dia
inicio
        escribir (“introduzca nº de dia”)
        leer(dia)
        segun_sea (dia) hacer
            1: escribir(“LUNES”)
            2: escribir(“MARTES”)
            3: escribir(“MIERCOLES”)
            4: escribir(“JUEVES”)
            5: escribir(“VIERNES”)
            6: escribir(“SABADO”)
            7: escribir(“DOMINGO”)
            en_otro_caso
                escribir(“Día incorrecto”)
        fin_segun
 fin_principal
fin_algoritmo
```

3.4.-Estructuras Repetitivas

Permiten repetir una o varias instrucciones varias veces en función de la evaluación de una determinada condición. A estas estructuras se les denomina *bucles*, y se llama *iteración* a cada repetición de la ejecución de la secuencia de instrucciones que forman el llamado *cuerpo del bucle*.

Tipos:

- Estructura Mientras
- Estructura Repetir_hasta
- Estructura Desde

3.4.1.-Estructura Repetitiva Mientras

En la estructura repetitiva **mientras** el conjunto de instrucciones que forman el cuerpo del bucle se repite mientras la condición se evalúe como verdadera. La condición se encuentra al principio del bucle

Diagrama de flujo:

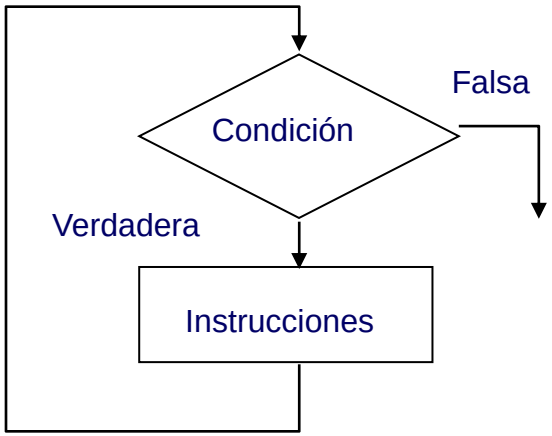
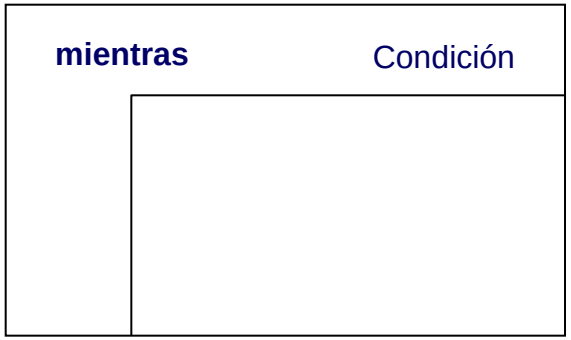


Diagrama N-S:



Pseudocódigo:

mientras (condición) **hacer**
instrucción 1
instrucción 2
…….
instrucción n
fin_mientras

3.4.1-Estructura Repetitiva Mientras

Ejemplo: Realizar el producto de dos números mediante el operador suma

Diagrama de flujo:

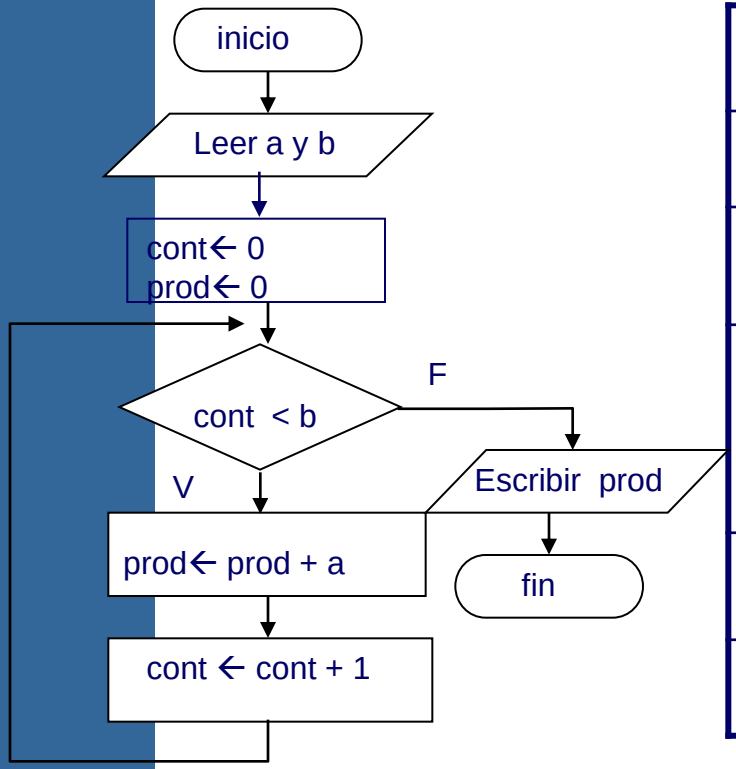


Diagrama N-S:

Algoritmo producto_sumas
leer a y b
cont ← 0 **prod ← 0**
Mientras (cont<b) **prod← prod + a** **cont ← cont + 1**
escribe prod
Fin_algoritmo

Pseudocódigo:

```
Algoritmo producto _sumas

Principal
 var
   entero : a,b, cont, prod
 inicio
   escribir(" introduzca dos nº")
   leer(a, b)
   cont← 0
   prod← 0
   mientras (cont < B) hacer
      prod← prod + A
      cont ← cont + 1
   fin_mientras
   escribir (prod)
 fin_principal
fin_algoritmo
```

3.4.2.-Estructura Repetitiva Repetir

En la estructura **repetir**, la condición se encuentra al final del bucle, y el conjunto de instrucciones que forman el cuerpo del mismo se ejecuta **hasta que** se cumpla una determinada condición.

Diagrama de flujo:

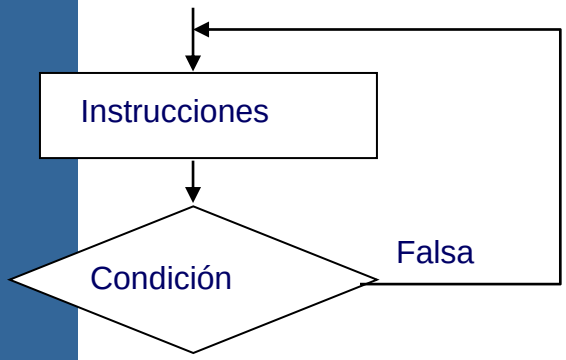
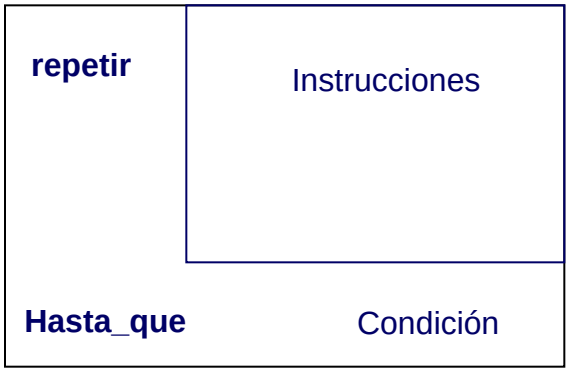


Diagrama N-S:



Pseudocódigo:

```
repetir
        instrucción 1
        instrucción 2
        …….
        instrucción n
hasta_que  (condición)
```

3.4.2-Estructura Repetitiva Repetir

Ejemplo: Realiza el algoritmo necesario para calcular el factorial de un número

Diagrama de flujo:

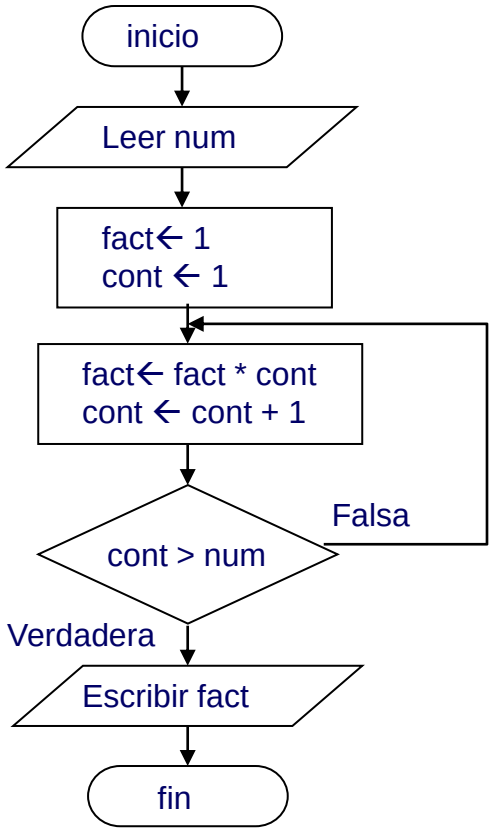


Diagrama N-S:

Algoritmo factorial
leer num
fact ← 1 **cont ← 1**
Repetir | **fact← fact*cont** **cont ← cont + 1**
Hasta_que (cont>num)
escribe fact
Fin_algoritmo

Pseudocódigo:

```
Algoritmo factorial

Principal
  var
   entero: cont, num
   real: fact
  inicio
   escribir ("introduzca un nº")
   leer (num)
   fact← 1
   cont ← 1
   repetir
      fact ← fact * cont
      cont ← cont + 1
   hasta_que (cont > num)
   escribir ("factorial", fact)
 fin_principal
fin_algoritmo
```

3.4.3.-Estructura Repetitiva Desde

La estructura **desde** ejecuta las acciones del cuerpo del bucle un número determinado de veces, controlando de modo automático las iteraciones.

Diagrama de flujo:

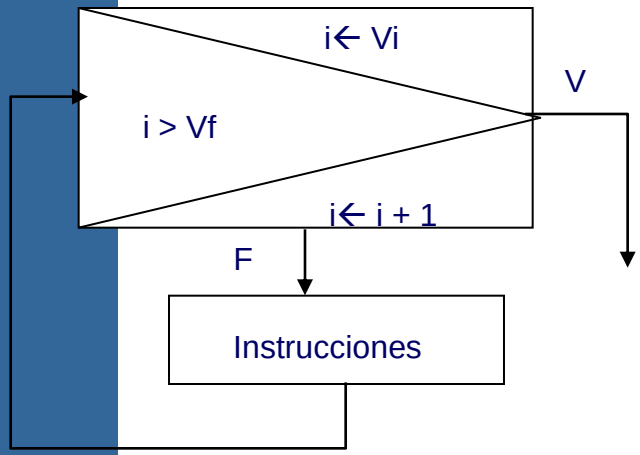
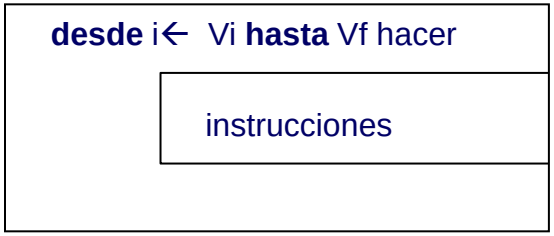


Diagrama N-S:



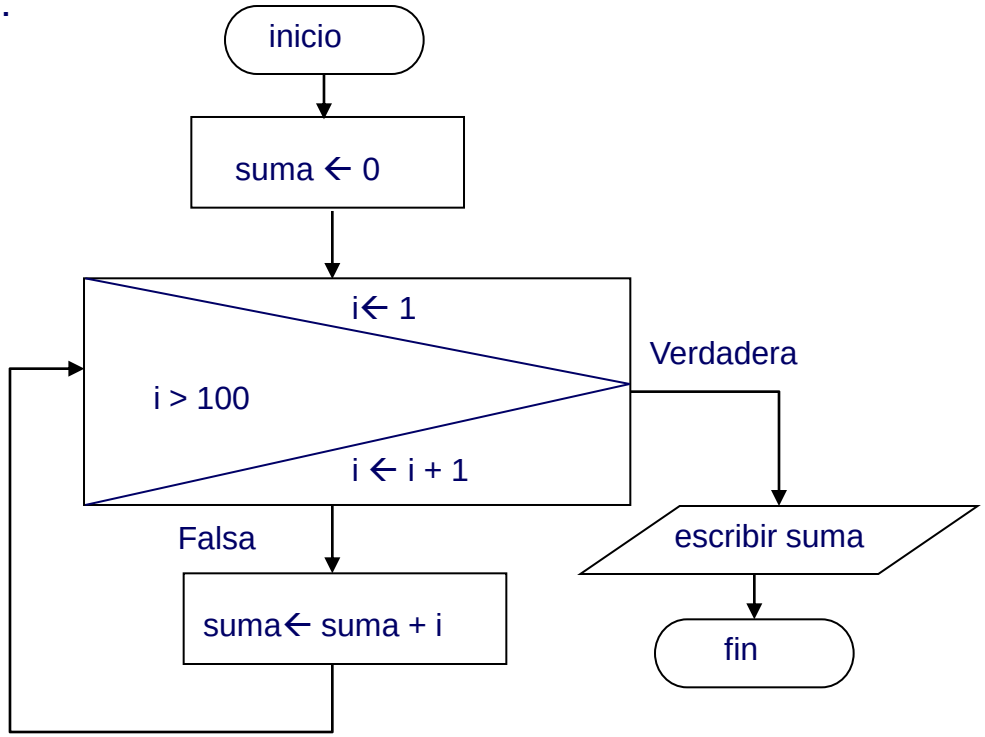
Pseudocódigo:

desde i ← Vi hasta Vf hacer
instrucciones
fin_desde

3.4.3-Estructura Repetitiva Desde

Ejemplo: Cálculo de la suma de los números enteros comprendidos entre 1 y 100.

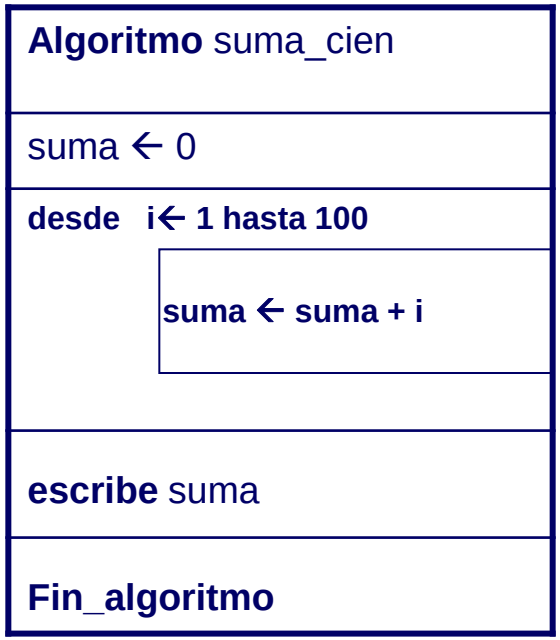
Diagrama de flujo:



3.4.3-Estructura Repetitiva Desde

Ejemplo: Cálculo de la suma de los números enteros comprendidos entre 1 y 100.

Diagrama N-S:



3.4.3-Estructura Repetitiva Desde

Ejemplo: Cálculo de la suma de los números enteros comprendidos entre 1 y 100.

Pseudocódigo:

```
Algoritmo suma_cien

Principal
 var
   entero: suma, i
 inicio
  suma ← 0
  desde i ← 1 hasta  100 hacer
            suma ← suma + i
  fin_desde
 fin_principal
fin_algoritmo
```

3.4.-Estructuras Repetitivas

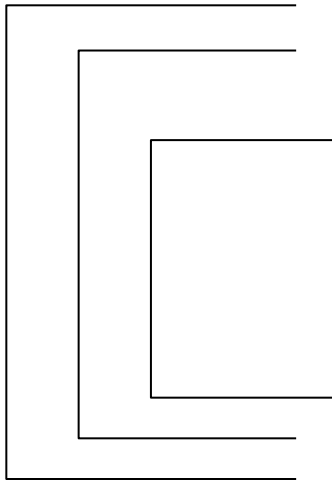
Sustitución de unas estructuras por otras:

- Podemos sustituir **siempre** una estructura **desde** por una estructura **repeti**r o por una estructura **mientras**.
- Una estructura **repetir** o **mientras**, sin embargo, se puede sustituir por una estructura **desde** sólo cuándo se conoce de antemano el número de veces que van a ejecutarse las acciones del bucle.
- Una estructura **mientras** puede sustituirse por una estructura **repetir** cuando no altere el resultado el hecho de que las acciones del bucle se ejecuten al menos una vez.
- Una estructura **repetir** siempre puede sustituirse por una estructura **mientras.** En el peor de los casos, también habría que escribir el conjunto de instrucciones del bucle antes del mismo, para que se ejecuten siempre al menos una vez.
- Estas afirmaciones pueden variar en función de cómo estén definidas las diferentes estructuras en cada lenguaje de programación.

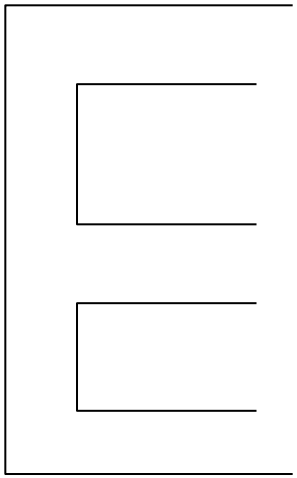
3.5.-Estructuras Anidadas

Tanto las estructuras selectivas como las estructuras repetitivas se pueden anidar, es decir, estar contenidas unas dentro de otras.

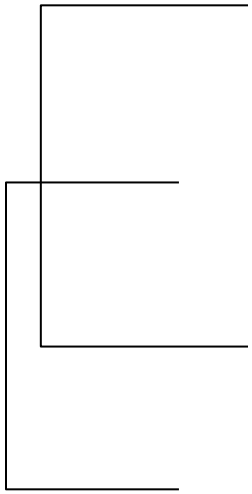
a) CORRECTO



b) CORRECTO



c) INCORRECTO



3.5.-Estructuras Anidadas

Ejemplo: escribe todos los números primos entre 2 y 1000.

```
Algoritmo primos
 Principal
 var
   lógico: encontrado
   entero : i, divisor
 inicio
   desde i← 2 hasta i ←100 hacer
       encontrado ← falso
       divisor ← 2
       mientras (divisor ≤ sqrt(i) and encontrado = falso) hacer
                si (i mod divisor = 0) entonces
              encontrado ← verdadero
                fin_si
                divisor ← divisor + 1
       fin_mientras
      si (encontrado = falso) entonces
                escribir (i)
   fin_desde
 fin_principal
fin_algoritmo
```