

UT 2: Estructuras de control.


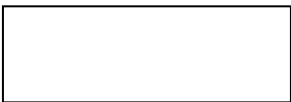
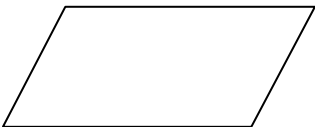
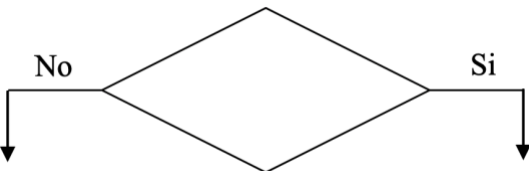

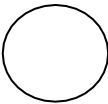
1.	<i>Notación para el diseño de algoritmos con diagramas de flujo</i>	<i>2</i>
1.1.	Instrucciones básicas que se pueden utilizar en los Diagramas de Flujo	3
1.2.	Estructuras útiles con el símbolo de “Decisión”	5
1.3.	“Variables” que podemos necesitar en las estructuras iterativas (Bucles)	6
2.	<i>Pseudocódigo</i>	<i>6</i>
3.	<i>Notación para el diseño de algoritmos con pseudocódigo</i>	<i>7</i>
3.1.	Sección de Declaración de Constantes	8
3.2.	Sección de Declaración de Variables	8
3.3.	Instrucciones de Entrada / Salida (E/S)	8
3.4.	Instrucción de Asignación:	9
3.5.	Comentarios	9
3.6.	Instrucciones Alternativas o Condicionales	9
3.7.	Instrucciones Iterativas (Bucles)	11

1. Notación para el diseño de algoritmos con diagramas de flujo

Se trata de una notación gráfica que pretende facilitar la escritura o la comprensión de algoritmos. Fue muy útil al principio de la informática y todavía se usa como apoyo para explicar ciertos algoritmos. Si los algoritmos son complejos, este tipo de esquemas no son adecuados, de ahí que actualmente, sólo se usen con fines educativos y no en la práctica.

Estos diagramas utilizan símbolos especiales que están normalizados por organismos de estandarización como ANSI e ISO.

La lista de símbolos que generalmente se usan en este tipo de diagramas son:

	Terminal: sirve para indicar el inicio o el fin del algoritmo. Dentro del símbolo hay que escribir "Inicio" o "Fin".
	Proceso: contiene una operación secuencial normalmente un cambio de valor o asignación.
	Entrada / Salida: contiene una instrucción de lectura o e escritura de datos.
	Decisión: contiene una decisión. Si se cumple el algoritmo continúa por la flecha del "Sí". De otro modo sigue por la flecha del "No".
	Dirección: conecta los diversos símbolos para indicar la dirección que sigue el flujo del algoritmo.
	Conexión: permite enlazar el diagrama de flujo entre dos partes distantes del algoritmo que están en la misma página. Para conectar se debe poner el mismo número en el conector de entrada y en el de salida.

Todo diagrama de flujo:

- Debe comenzar por el Terminal “Inicio”.
- Debe finalizar con un único Terminal “Fin”.
- Entre el Terminal “Inicio” y el Terminal “Fin” deben existir uno o más símbolos conectados mediante el símbolo Dirección.

1.1. Instrucciones básicas que se pueden utilizar en los Diagramas de Flujo

- Instrucción “Asignación”: Instrucción u operación que atribuye a una variable un valor determinado.

Sintaxis: **variable ← expresión**

Ejemplo: **edad ← 10**

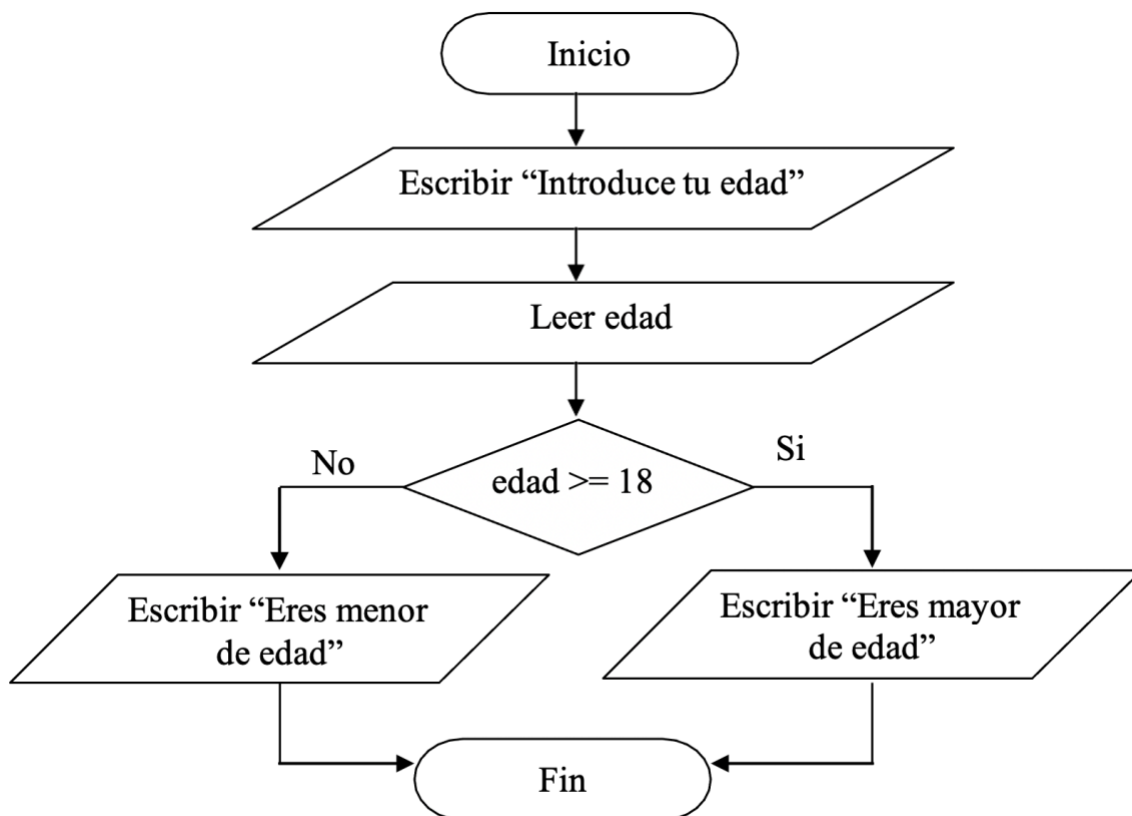
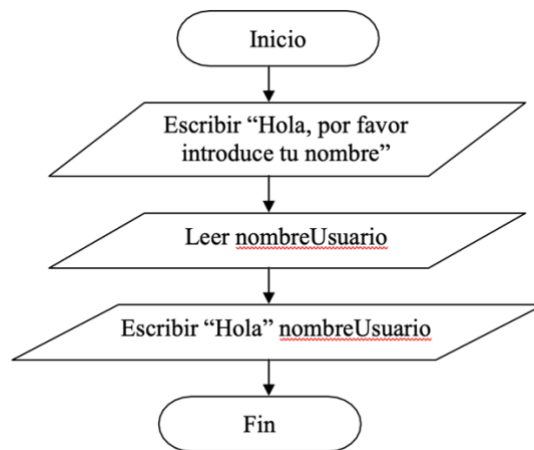
Consideraciones importantes a tener en cuenta:

- El operador de Asignación es: ←
 - A la izquierda de la asignación debe existir obligatoriamente una única variable.
 - A la derecha de la asignación debe existir una expresión.
 - Lo primero que se evalúa es la expresión de la parte derecha de la asignación. Una vez que se ha obtenido su valor, se asigna o concede su valor a la variable de la parte derecha.
 - A la variable sólo se le puede asignar un valor que sea del mismo tipo de dato que el tipo de dicha variable.
 - Si una variable es de tipo numérica, carácter, cadena o booleana (tipos básicos) sólo puede contener un único valor. Es decir en el momento en que se asigne un nuevo valor a una variable, desaparece el valor antiguo que contenía.
 - Las asignaciones utilizan el símbolo gráfico “Proceso” de los diagramas de flujo.
- **Instrucciones de Entrada y Salida de información:**
 - **Operación de “Lectura”:**
 - Permite introducir o asignar un valor a una variable desde un dispositivo externo de entrada (teclado).
 - Utiliza el símbolo gráfico “Entrada/Salida” con la palabra clave “Leer” seguida de la variable que recogerá el valor introducido por el usuario mediante el teclado.

- **Operación de “Escritura”:**

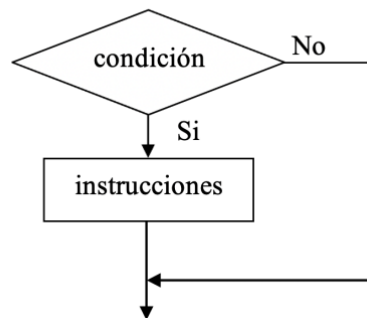
- Permite mostrar o visualizar una información por un dispositivo externo de salida (pantalla).
- Utiliza el símbolo gráfico “Entrada/Salida” con la palabra clave “Escribir” seguida de la información que se quiera mostrar por pantalla. Dicha información puede ser un mensaje de texto en cuyo caso habrá que encerrarlo entre comillas dobles o puede provenir del contenido de una variable (sin comillas dobles !!).

Ejemplos:

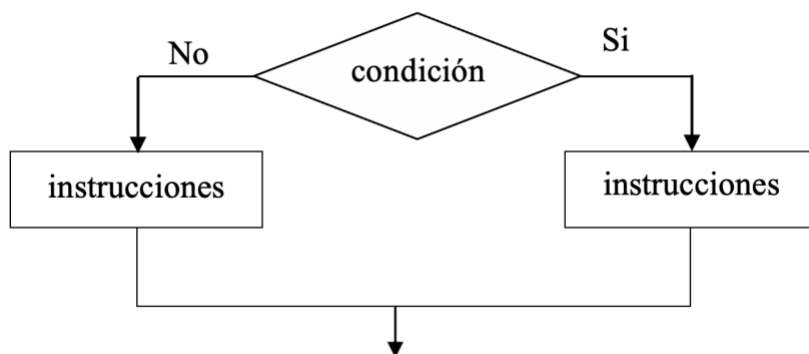


1.2. Estructuras útiles con el símbolo de “Decisión”

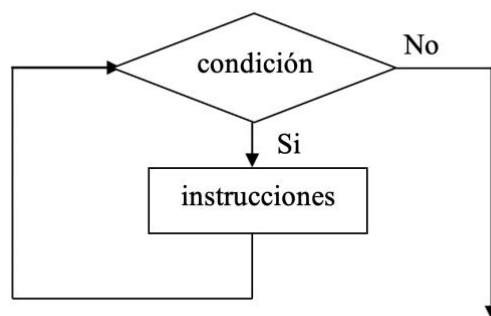
- **Estructura alternativa simple (estructura “si”)**



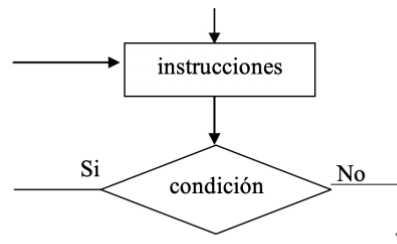
- **Estructura alternativa doble (estructura “si ... sino”)**



- **Estructura iterativa (bucle “mientras”)**



- **Estructura iterativa (bucle “repetir ... mientras”)**



1.3. “Variables” que podemos necesitar en las estructuras iterativas (Bucles)

- **Variable “contador”**: variable cuyo valor se incrementa o se decrementa en una cantidad constante en cada iteración o repetición del bucle.
- **Variable “acumulador”**: variable cuyo valor se incrementa o decrementa en una cantidad no constante en cada iteración o repetición del bucle.
- **Variable “interruptor, bandera o flag”**: variable que contiene un valor lógico o booleano (es decir que sólo puede tomar por valores Verdadero o Falso) y que nos indica si ha acontecido o no un suceso.

2. Pseudocódigo

Para diseñar algoritmos estructurados se aconseja el uso de un lenguaje especial llamado pseudocódigo, que puede ser traducido posteriormente fácilmente a cualquier lenguaje de programación.

El pseudocódigo permite:

- Diseñar algoritmos de forma estructurada (resolver cualquier problema mediante instrucciones secuenciales, alternativas e iterativas).
- Introducir el diseño modular en los algoritmos.

Hay que tener en cuenta que existen multitud de pseudocódigos, es decir no hay un pseudocódigo 100% estándar. Pero sí hay gran cantidad de detalles aceptados por todos los que escriben pseudocódigos. A continuación se verá el pseudocódigo más aceptado en España.

El pseudocódigo está formado por instrucciones escritas en un lenguaje que posteriormente al ser convertido en lenguaje de programación será entendido por el ordenador. Por ello en pseudocódigo sólo se pueden utilizar ciertas instrucciones muy

concretas, de tal forma que la escritura de dichas instrucciones debe cumplir reglas muy estrictas.

Las únicas instrucciones permitidas son:

- **Instrucciones de Entrada / Salida:** para leer o escribir datos.
- **De proceso:** operaciones que realiza el algoritmo (normalmente asignaciones).
- **De control de flujo:** instrucciones alternativas o iterativas (bucles y condiciones).
- **De declaración:** mediante las que se crean variables, constantes y subprogramas.
- **Llamadas a subprogramas:** permiten el diseño modular.
- **Comentarios:** Notas o aclaraciones que se escriben junto al pseudocódigo para explicar su funcionamiento.

3. Notación para el diseño de algoritmos con pseudocódigo

Estructura básica de un pseudocódigo

Algoritmo nombreDelAlgoritmo

Const

<lista constantes>



*Sección de declaración
de **constantes***

Var

<lista variables>



*Sección de declaración
de **variables***

Inicio

<instrucciones>

Fin

- **Todo pseudocódigo comienza por “Algoritmo”** seguido del nombre que queramos dar al algoritmo.
- A continuación se detalla en caso de que se vaya a utilizar alguna constante en el algoritmo, **la sección de declaración de constantes**.
- En caso de que exista alguna variable en el algoritmo, hay que escribir la **sección de declaración de variables**.
- Todas las instrucciones que resuelven el algoritmo en pseudocódigo deben estar encabezadas por la palabra **inicio** y cerradas por la palabra **fin**.

Además hay que tener en cuenta:

- Aunque no importan las mayúsculas y minúsculas a la hora de escribir en pseudocódigo, se aconseja **escribir en minúsculas** debido a que existen muchos lenguajes en los que sí importa el hecho de escribir en mayúsculas o minúsculas (lenguajes sensibles a mayúsculas y minúsculas).
- Se aconseja **escribir de forma tabulada**, para que se vea más claro que se encuentra entre el inicio y el fin tanto del algoritmo principal como de las estructuras alternativas e iterativas

3.1. Sección de Declaración de Constantes

- En la sección de declaración de constantes se especifican todas las constantes que se van a utilizar en el algoritmo.
- Por cada constante hay que indicar el nombre y el valor fijo que van a recibir, utilizando el operador=
- Los nombres de las constantes se suelen escribir en **mayúsculas** para diferenciarlos a simple vista de los nombres de las variables.

Const

PI = 3.141592

NOMBRE = "Jose"

3.2. Sección de Declaración de Variables

- En la sección de declaración de variables se especifican todas las variables que se van a utilizar en el algoritmo.
- Por cada variable hay que indicar el **nombre y el tipo de dato** de la variable, utilizando el operador:
- Los nombres de las variables se suelen escribir en **minúsculas**.

var

edad: entero

sueldo: real

3.3. Instrucciones de Entrada / Salida (E/S)

Las mismas instrucciones que las utilizadas en los Diagramas de Flujo.

- **Instrucción de Entrada:** recoge la información que el usuario ha introducido mediante el teclado del ordenador para asignárselo a una variable.

Leer variable

- **Instrucción de Salida:** muestra o visualiza información por la pantalla del ordenador. En caso de que se muestre una variable, lo que realmente aparecerá por la pantalla será el valor que almacena dicha variable.

Escribir " ... texto ... "
Escribir variable

3.4. Instrucción de Asignación:

La misma operación que la utilizada en los Diagramas de Flujo.

Variable \leftarrow expresión

3.5. Comentarios

Los comentarios son notas o aclaraciones que se escriben junto al pseudocódigo para explicar su funcionamiento (práctica muy aconsejable).

Para introducir un comentario hay que escribir los símbolos // al principio de cada línea comentada.

Los comentarios no tienen ningún efecto sobre la ejecución del pseudocódigo, tan solo son aclaraciones tanto para el propio programador como para las personas que lean su algoritmo.

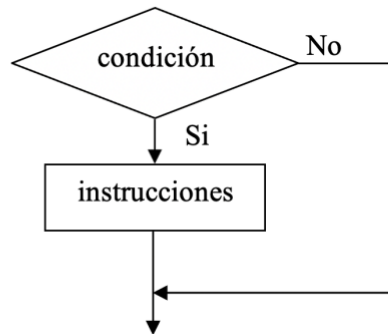
```
Inicio
    // Un comentario. Aquí podemos escribir lo que queramos.
    Instrucciones
    // Otro comentario
fin
```

3.6. Instrucciones Alternativas o Condicionales

- **Instrucción condicional simple:** instrucción que tras evaluar una expresión lógica, realiza una serie de instrucciones en caso de que la expresión lógica sea verdadera.

Su sintaxis es: Si (expresión lógica) entonces
 Instrucciones si se cumple la expresión lógica
 Fin si

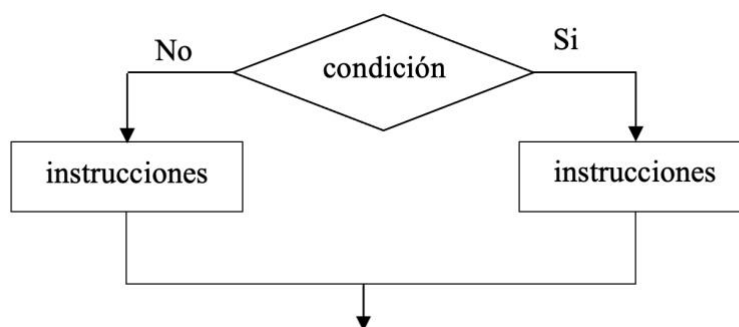
Su estructura equivalente en diagrama de flujo es:



- **Instrucción condicional compuesta o doble:** es igual que la anterior, sólo que se añade un apartado sino que contiene las instrucciones que se ejecutarán si la expresión evaluada es falsa.

Su sintaxis es:
 Si (expresión lógica) entonces
 Instrucciones si se cumple la expresión lógica
 Sino
 Instrucciones si no se cumple la expresión lógica
 Fin si

Su estructura equivalente en diagrama de flujo es:



- **Instrucción alternativa “según sea”:** instrucción que permite construir alternativas múltiples a partir del valor de una variable de tipo entero o de tipo carácter.

Dentro de la instrucción “Según sea” se indicarán los posibles valores con los que se comparará la variable. Si el valor de la variable coincide con alguno de los valores indicados se ejecutarán las instrucciones correspondientes. Si no

coincide con ningún valor, se ejecutarán las instrucciones correspondientes a la sección “en otro caso”.

Su sintaxis es:

```
Según sea <variable>
    <valor_1>: instrucciones valor_1
    <valor_2>: instrucciones valor_2
    ....
    en otro caso:
        instrucciones en otro caso
Fin según sea
```

Ejemplo:

```
Algoritmo ejemploSegunSea
Var
    opc: carácter
inicio
    escribir "Elige una opción A, B o C: "
    leer opc
    según sea opc
        'A': escribir "Has seleccionado la opción A"
        'B': escribir "Has seleccionado la opción B"
        'C': escribir "Has seleccionado la opción C"
        en otro caso:
            escribir "Opción errónea"
    fin según sea
Fin
```

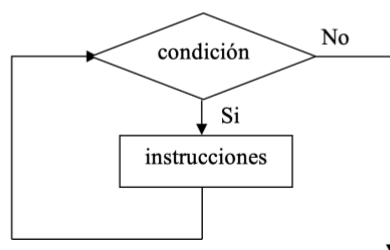
3.7. Instrucciones Iterativas (Bucles)

- **Bucle “Mientras”:** instrucciones que se ejecutan continuamente mientras una expresión lógica sea verdadera.

Su sintaxis es:

```
Mientras (expresión lógica) hacer
    Instrucciones que se van a repetir
Fin mientras
```

Su estructura equivalente en diagrama de flujo es:



Ejemplo:

```

cont ← 1
mientras (cont <= 100) hacer
    escribir cont
    cont ← cont + 1
fin mientras
  
```

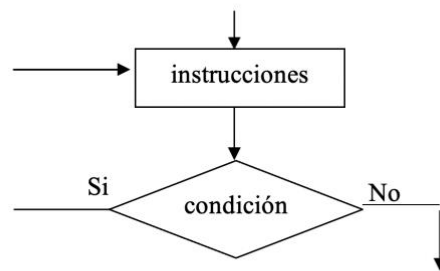
- **Bucle “Repetir Mientras”:** la única diferencia respecto a la anterior está en que la expresión lógica se evalúa después de haber ejecutado las instrucciones. Es decir, el bucle como mínimo se ejecuta una vez.

Su sintaxis es:

```

Hacer
    Instrucciones que se van a repetir
mientras (expresión lógica)
  
```

Su estructura equivalente en diagrama de flujo es:



Ejemplo:

```

cont ← 1
hacer
    escribir cont
    cont ← cont + 1
mientras (cont <= 100)
  
```

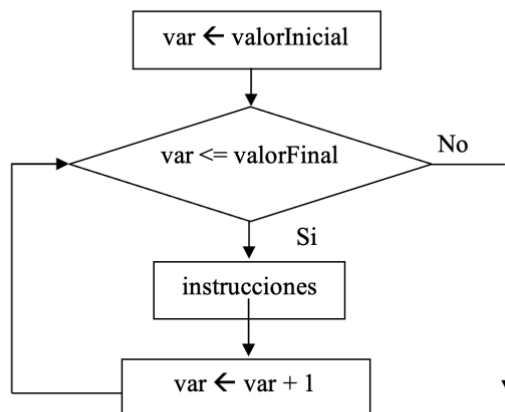
- **Bucle “Desde”:** bucle que se repite un determinado número de veces.

Su sintaxis es:

```

Desde <Variable> = <valorInicial> hasta <valorFinal> hacer
    Instrucciones que se van a repetir
Fin desde
  
```

Su estructura equivalente en diagrama de flujo es:



Ejemplo:

Desde `l = 1` hasta 100 hacer
Escribir `l`
Fin desde