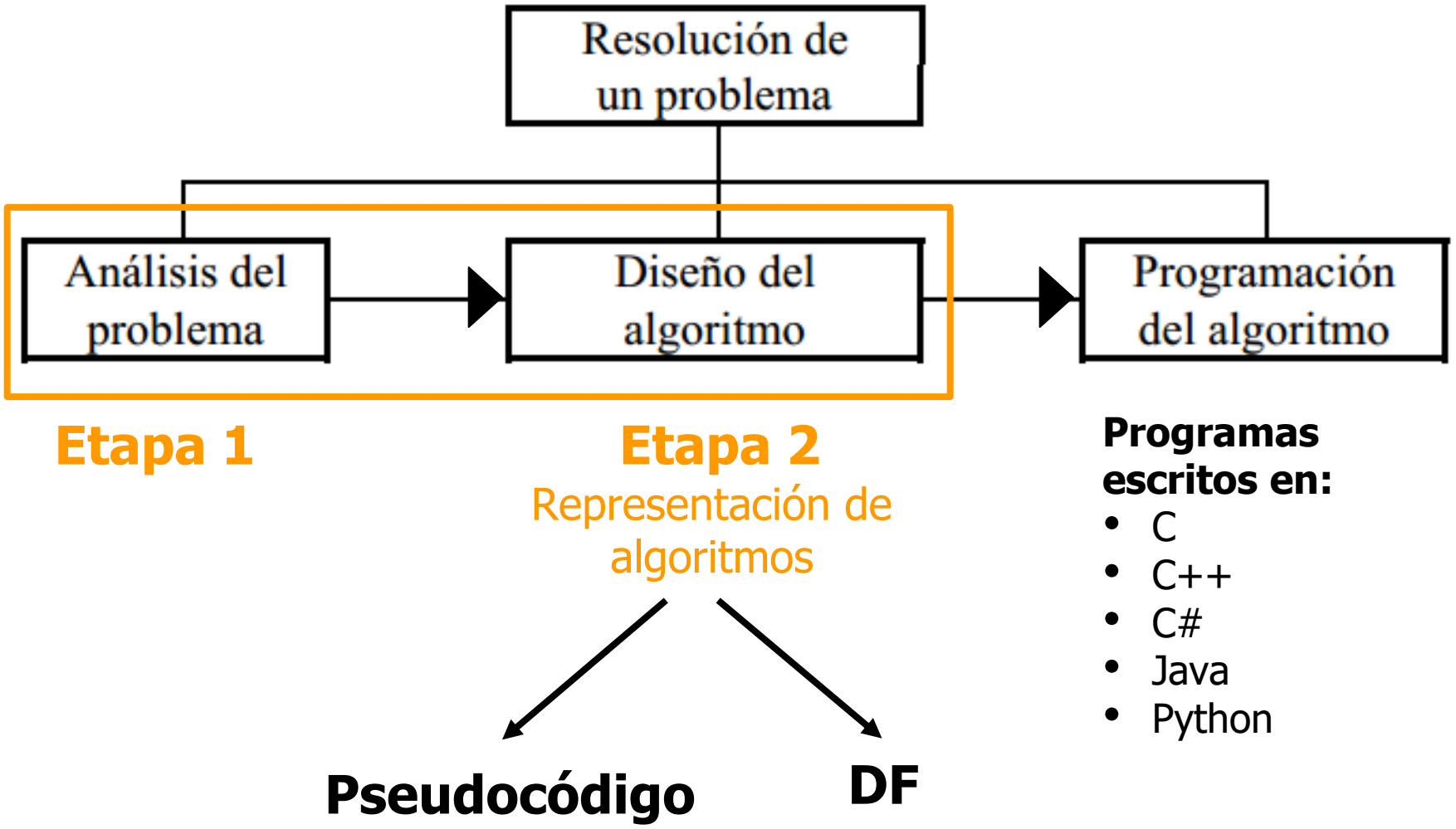
Teoría

Pseudocódigo y Diagrama de Flujo

Resolución de Problemas y Algoritmos
Fundamentos de la Programación

Ingeniería en Computación (TU - TFA)
Ingeniería en Minas (TU)
Profesorado en Ciencias de la Computación (TU - TFA)

Etapas en la Resolución de un Problema



Programas escritos en:

- C
- C++
- C#
- Java
- Python

Etapa 1: Análisis del Problema.

1. Entender el enunciado y comprender el problema.
2. Representación e interpretación de datos (entrada/proceso/salida).

Etapa 2: Diseño y desarrollo del algoritmo para resolver el problema.

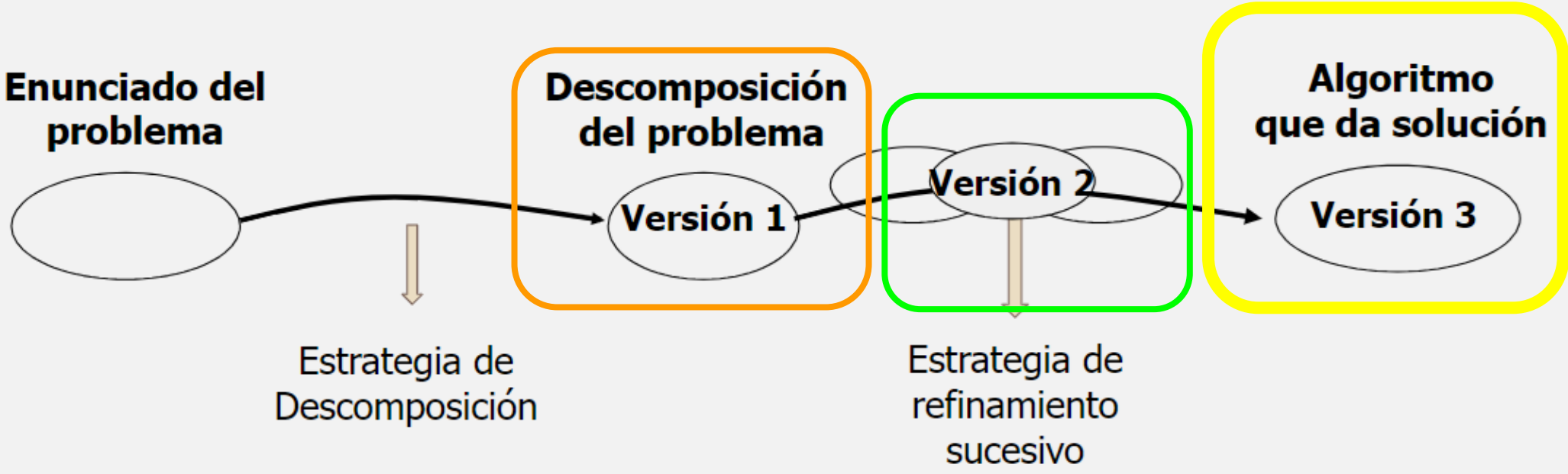
1. Descomposición de un problema en tareas.
2. Refinamiento sucesivo. Algoritmo que da solución a un problema.
3. **Representación de algoritmos: pseudocódigo y diagrama de flujo.**

Etapa 1: Análisis del Problema.

1. Entender el enunciado y comprender el problema.
2. Representación e interpretación de datos (entrada/proceso/salida).

Etapa 2: Diseño y desarrollo del algoritmo para resolver el problema.

1. Descomposición de un problema en tareas.
2. Refinamiento sucesivo. Algoritmo que da solución a un problema.
3. **Representación de algoritmos: pseudocódigo y diagrama de flujo.**



Etapa 2: Pseudocódigo y diagrama de flujo

Representación TEXTUAL O NO GRÁFICA de un algoritmo

```
INICIO
    Entero: num, absnum ;
    Leer num;
    Si num<0
        absnum= num * (-1);
    Sino
        absnum=num;
    Fin_si
    Imprimir "El valor absoluto es: " absnum;
FIN
```

Pseudocódigo es usado para la **representación no gráfica** de algoritmos de la forma más detallada posible y a su vez lo más parecida posible al lenguaje que posteriormente su utilizará para la codificación del mismo.

Representación GRÁFICA de un algoritmo

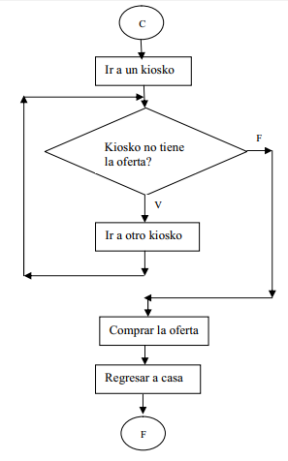


Diagrama de flujo es utilizado para **representar gráficamente** algoritmos, facilita la visualización del flujo de ejecución del mismo.

Teoría
Pseudocódigo y Diagrama de Flujo

- ✓ **Representación de algoritmos: Pseudocódigo.**
  - Representación de datos (objetos).
  - Expresiones (aritméticas/lógicas/relacionales).
  - Asignación.
  - Entrada y salida de datos.
  - Estructuras de control: secuencial, condicional y de repetición.
- ✓ **Representación de algoritmos: Diagrama de Flujo.**
  - Simbología.
  - Ejemplo completo.
- ✓ **Pseudocódigo con PSeInt.**

Problema

es una situación que se nos presenta y que mediante la aplicación de un **algoritmo** pretendemos resolver.

Problema computacional

son aquellos que para resolverlos involucran, generalmente, operaciones aritméticas, lógicas y relacionales.

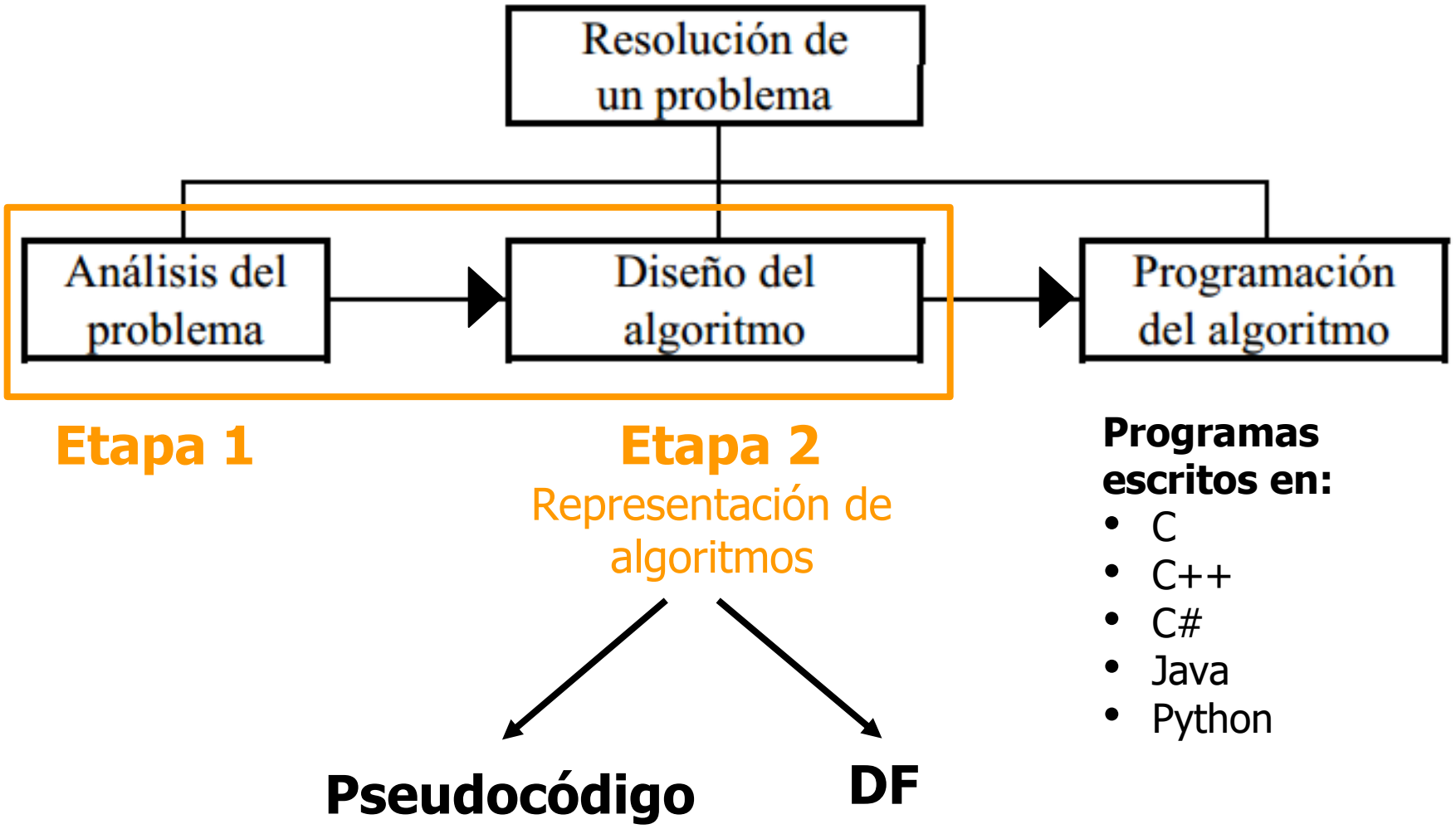
Algoritmo

es una sucesión **finita y ordenada** de acciones o pasos **precisos** que permiten resolver un problema.

Programa

es la especificación de un **algoritmo** en un lenguaje de programación para que pueda ser ejecutado por una computadora.

Etapas en la Resolución de un Problema

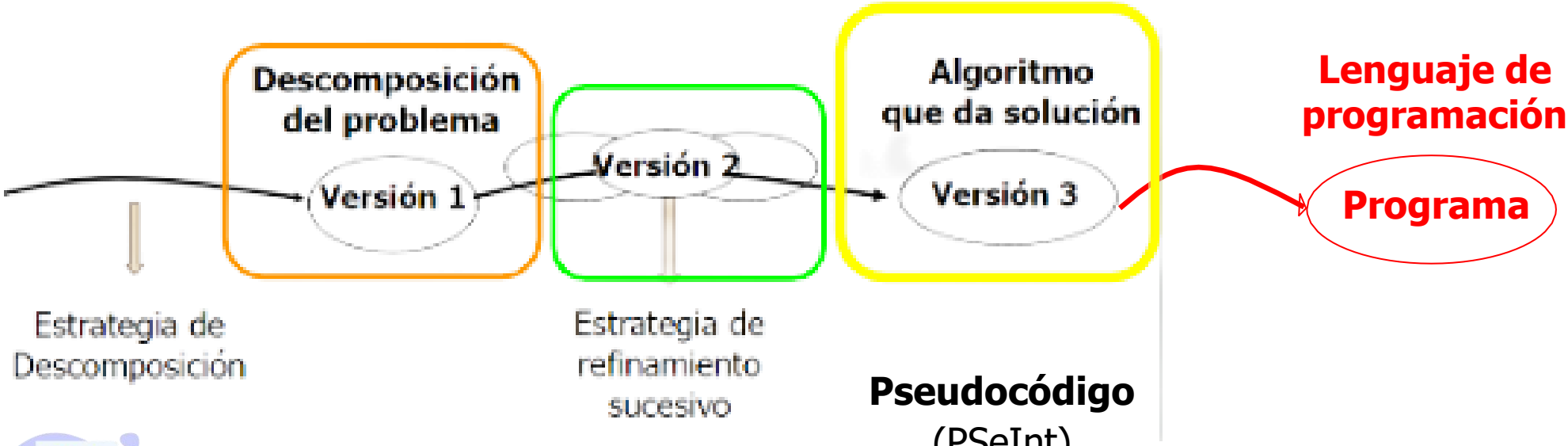


Programas escritos en:

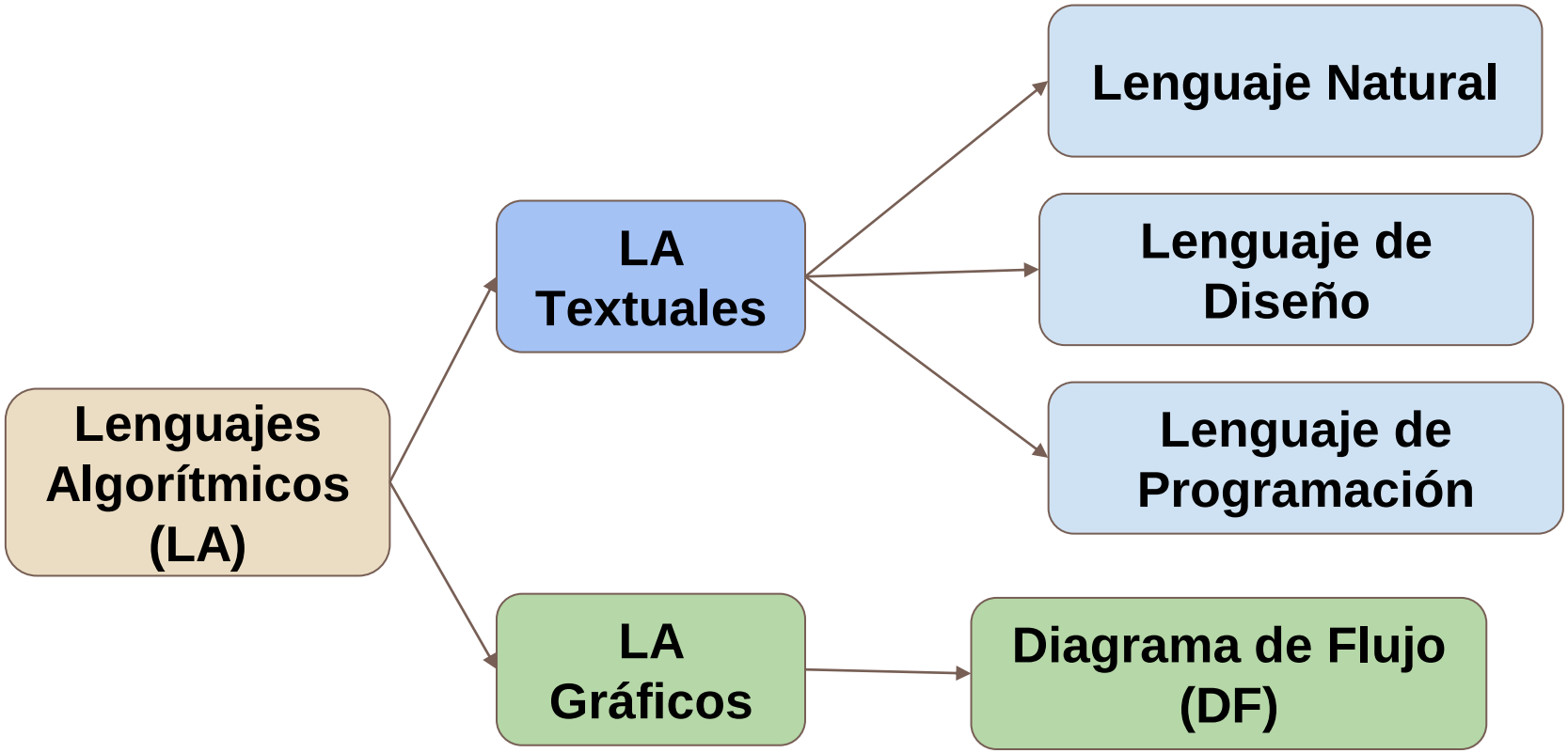
- C
- C++
- C#
- Java
- Python

Diseño del algoritmo

1. Algoritmo escrito en pseudocódigo (PSeInt).
2. Algoritmo representado en Diagrama de Flujo (DF).



Lenguajes algorítmicos



Diferencia entre
Lenguaje de diseño y Lenguaje de programación

Lenguaje de diseño, su objetivo es ser comprensible para las **personas** que van a interpretar los algoritmos escritos con él.

Los algoritmos escritos en **pseudocódigo** utilizan una combinación de lenguaje natural y elementos similares al lenguaje de programación.

Mientras que el propósito de un **lenguaje de programación** es ser comprensible por la **computadora** que va a ejecutar el **programa**.

Enunciado: descripción detallada de un problema a resolver.

Procesador: Toda entidad capaz de comprender las acciones primitivas de un algoritmo y ejecutar el trabajo indicado por el mismo.

Ambiente: El conjunto de todos los recursos necesarios para la ejecución de un algoritmo.

Algoritmo: conjunto de **acciones (pasos)** que debe realizar el procesador para resolver el problema planteado.

Problemas Computacionales

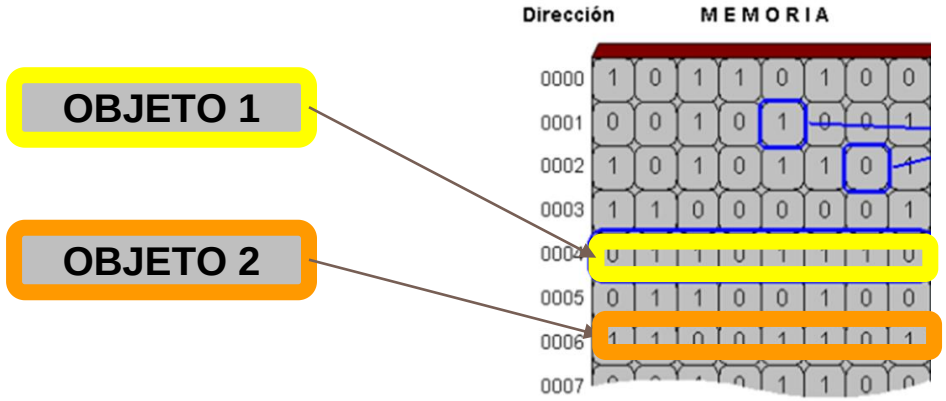
Enunciado: descripción del trabajo a realizar por la computadora.

Procesador: **computadora**

Ambiente: los **objetos** del universo del problema a definir en la **memoria de la computadora**, estos objetos deben ser capaces de almacenar **datos**.

¿Dónde se guardan los datos?
¿Quién ingresa los datos?
¿Quién recibe los datos de salida?

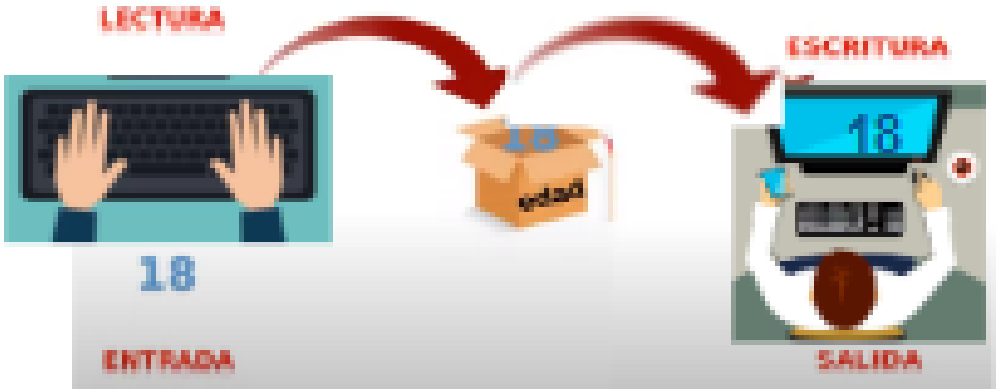
Representación de datos (objetos).



Los **objetos** en los algoritmos escritos en pseudocódigo son: las **variables** y las **constantes**.

Los principales **tipos de datos primitivos o simples** en los que pueden definirse los **objetos** son:

- Numéricos: Enteros y Reales.
- Caracteres
- Booleanos



TIPOS DE DATOS PRIMITIVOS

ENTERO: consiste de un conjunto finito de los números enteros positivos y negativos. Ejemplos: 3, -3, 345, -56.

REAL: consiste de un conjunto finito de valores de los números reales. Números con **punto decimal**, positivos y negativos. Ejemplos: 2.3, -4.6.

LÓGICO: también llamado tipo **booleano**, es el conjunto de los valores de verdad: VERDADERO y FALSO.

CARACTER: es el conjunto finito y "ordenado" de caracteres que el procesador puede reconocer. Ejemplos: 'A', '3', '+'.

las letras mayúsculas del abecedario.
las letras minúsculas del abecedario.
los caracteres numéricos del 0...9.
el caracter de espacio blanco, caracteres especiales tales como: *, +, -, _, /, (,), $, ^, %, $, < ,>, ", .

Objetos.

Variables: tienen la capacidad de almacenar UN dato y cuyo contenido **puede cambiar** durante el ciclo de vida del algoritmo.

Constantes: tienen la capacidad de almacenar UN dato y cuyo contenido **NO puede cambiar** durante el ciclo de vida del algoritmo.

Una **variable** es un objeto del ambiente cuyo valor puede cambiar y que posee además los siguientes atributos:

- un **identificador de variable, NOMBRE** que la identifica,
- un **TIPO DE DATO** que describe los valores que puede tomar la variable y las operaciones que con dicha variable pueden realizarse.

Toda **variable** debe definirse indicando el NOMBRE y el TIPO de valores que la misma puede tomar.

Definir <nombre de variable> [,<nombre de variable>]* **como** <tipo>

Ejemplos:
Definir NUMERO **como** Entero
Definir Peso, Suma **como** Real

```
//Identificador de variable

No puede tener espacios;
No puede empezar por un numero;
No puede ser una palabra
reservada;
Distingue mayusculas
```

Actividad 1: Se desea registrar la edad de un estudiante, la inicial del apellido, si está en el TFA y la nota que obtuvo en el último parcial.

A.¿Cuántas variables se deberían considerar para almacenar estos datos?

B. ¿De qué tipo es cada una?

C. Definir al menos 2 (como se muestra los ejemplos).

Expresiones (aritméticas/lógicas/relacionales).

Un procesador debe ser capaz de manipular los objetos del ambiente de un algoritmo.

Una expresión describe un cálculo a efectuar cuyo resultado es un valor único.

Una expresión consta de **operadores** y **operandos**.

Expresión	Resultado	Ejemplos
Expresión **aritmética**	Tipo numérico.	3 *-6 el resultado es -18 2.5 - 1 el resultado es 1.5
Expresión **relacional**	Tipo lógico	5 < 9 el resultado es VERDADERO -8 = 8 el resultado es FALSO
Expresión **lógica**	Tipo lógico	VERDADERO Y FALSO el resultado es FALSO NO VERDADERO el resultado es FALSO

Aritméticas

Un operando de una expresión aritmética puede ser, por ahora:

- una constante de tipo numérico,
- una variable de tipo numérico
- otra expresión aritmética, encerrada entre paréntesis.

Operador	Significado
+	suma
-	resta
*	producto
/	división
^	potencia
% ó MOD	resto de la división

Orden de precedencia	Operadores
1º	^
2º	*,/, % ó MOD
3º	+, -

Relacionales

Un operando de una expresión relacional puede ser:

- una constante de tipo numérico o caracter,
- una variable de tipo numérico o caracter.

Operador Relacional	Significado
=	igual
<	menor
<=	menor o igual
>	mayor
>=	mayor o igual
<>	distinto

Lógicas

Un operando de una expresión lógica puede ser:

- una variable de tipo lógica
- una expresión lógica.

Conectores u Operadores Lógicos	Significado
& o Y	conjunción
| o O	disjunción
~ o NO	negación

1º ~ (no lógico)
2º & (conjunción)
3º | (disjunción)

Aritméticas

Un operando de una expresión aritmética puede ser, por ahora:

- una constante de tipo numérico,
- una variable de tipo numérico
- otra expresión aritmética, encerrada entre paréntesis.

Operador	Significado
+	suma
-	resta
*	producto
/	división
^	potencia
% ó MOD	resto de la división

Definir NUME como Entero
2.5 - 1
NUME * (5 + 3.6)

Relacionales

Un operando de una expresión relacional puede ser:

- una constante de tipo numérico o caracter,
- una variable de tipo numérico o caracter.

Operador Relacional	Significado
=	igual
<	menor
<=	menor o igual
>	mayor
>=	mayor o igual
<>	distinto

Definir NUM como Real

'A' < 'M'
VERDADERO = FALSO
NUM >= 3.6

Lógicas

Un operando de una expresión lógica puede ser:

- una variable de tipo lógica o
- una expresión lógica.

Conectores u Operadores Lógicos	Significado
& o Y	conjunción
| o O	disjunción
~ o NO	negación

Definir VAL como Logico

VAL = FALSO **Y** 'A' < 'M'
NO (5 > 3.6)

¿Qué valores tienen NUM y VAL?
¿Cómo almaceno valores en NUM y VAL?

Asignación de valores.

Para asignar (dar) valor a una variable, se utiliza el **operador de asignación**.

En lenguaje de diseño (PSeInt) el símbolo es ←:

V ← E

1. **V** es el nombre de la variable a la que el procesador va a asignar (dar) el valor de E.
2. ←**,** identifica al operador de asignación.
3. **E** representa el valor a asignar y puede ser una constante, otra variable, o el resultado de la evaluación de una expresión.

A ← **6**

B ← **A**

B ← 6 +1

B ← **A** +1

Ejemplos:

Acciones	A	B
A ← **6**	6	
B ← 6 +1		7

Acciones	M	SOL
M ← **6**	6	
SOL ← M +1		7

Actividad 2: Completar la tabla de ejecución con los valores de M, NUM y VER, al ejecutar en orden:

	M	NUM	VER
M ← ‘S’			
NUM ← 10			
NUM ← (NUM * 3) - 4			
VER ← 2 < NUM			

Actividad 2: Completar la tabla de ejecución con los valores de M, NUM y VER, al ejecutar en orden:

	M	**NUM**	**VER**
M ← 'S'			
NUM ← 10			
NUM ← (NUM * 3) - 4			
VER ← 2 < NUM			

¿De qué tipo de dato deberían estar definidas M, NUM y VER?

Definir M **como ?**

Definir NUM **como ?**

Definir VER **como ?**

Entrada y Salida de datos.

LEER: es la acción primitiva que permite la **entrada** de uno o más valores a objetos del ambiente a través de un dispositivo.

Una lectura es una asignación, toma valores del medio externo (por ejemplo, del teclado) y lo asigna a las variables del ambiente.

Leer <nombre de la variable>

ESCRIBIR: es la acción primitiva que permite la **salida** de valores del ambiente a través de un dispositivo (por ejemplo, monitor). Esta acción toma uno o más valores del ambiente y lo comunica al medio externo.

Escribir <nombre de la variable>

Entrada y Salida de datos.

Acciones	edad	Pantalla
Definir edad como Entero	indefinido	
Escribir “Ingrese la edad”		Ingrese la edad
Leer edad	18	
Escribir edad		18

Entrada y Salida de datos.

Acciones	edad	Pantalla
Definir edad como Entero	indefinido	
Escribir “Ingrese la edad”		Ingrese la edad
Leer edad	18	
Escribir edad		18

¿Qué vimos hasta ahora?.

- Definir variables (nombre y tipo de dato).
- Dar valores a las variables:
  1. ←
  2. **Leer**

Leer NUM, VAR

- Operar con variables.
- Mostrar en la pantalla

Escribir “Ingrese edad”
Escribir NUM

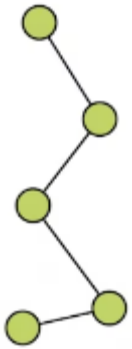
Estructuras de control.

En la programación estructurada, se disponen de estructuras de control de flujo.

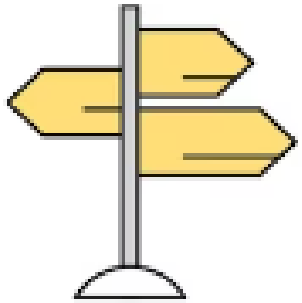
Esto hace referencia al orden en el que se ejecutarán las instrucciones de un programa, desde su comienzo hasta que finaliza.

Vamos a utilizar **3 estructuras de control**:

Secuencial



Selección



Iteración o repetición

```
1  Proceso ParImpar
2      Definir nro Como Entero
3      Escribir "Ingrese un número cualquiera"
4      Leer nro
5  FinProceso
```

Secuencial: las tareas (acciones) se deben realizar en el orden en que se escriben, es decir, primero una, luego otra, desde la primera hasta la última (o de arriba hacia abajo).

```
5      si nro MOD 2 = 0 Entonces
6          Escribir "El numero es par"
7      sino
8          Escribir "El numero es impar"
9      FinSi
```

Selección (Condicional): las tareas (acciones) se realizarán dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.

```
5      Mientras nro <= 0 hacer
6          Escribir "Ingrese un número cualquiera"
7          Leer nro
8      FinMientras
```

```
11     Mientras cont <= 4 Hacer
12         Escribir "Ingrese un número cualquiera"
13         Leer nro
14         si nro MOD 2 = 0 Entonces
15             Escribir "El numero es par"
16         sino
17             Escribir "El numero es impar"
18         FinSi
19         con<-cont+1
20     FinMientras
```

Iteración (Repetición): una tarea o conjunto de tareas (acciones) se deben realizar en forma repetitiva una **cantidad de veces específica** o dependiendo de una **condición** que se debe cumplir.

Condicional (selección)

Si <condición> **Entonces**
    <alternativa verdadera1>
Finsi

```
si nro MOD 2 = 0 Entonces
    Escribir "El numero es par"
FinSi
```

Si <condición> **Entonces**
    <alternativa verdadera>
Sino
    <alternativa falsa>
Finsi

```
si nro MOD 2 = 0 Entonces
    Escribir "El numero es par"
sino
    Escribir "El numero es impar"
FinSi
```

Repetición (Iteración)

Iteración simple

Mientras <condicion> **Hacer**

<secuencia de acciones>

FinMientras

```
cont<-1
Mientras cont <= 4 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
    cont<-cont+1
FinMientras
```

Iteración condicional

Mientras <condicion> **Hacer**

<secuencia de acciones>

FinMientras

```
Mientras nro <= 0 hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
FinMientras
```

Etapa 2: Pseudocódigo y diagrama de flujo

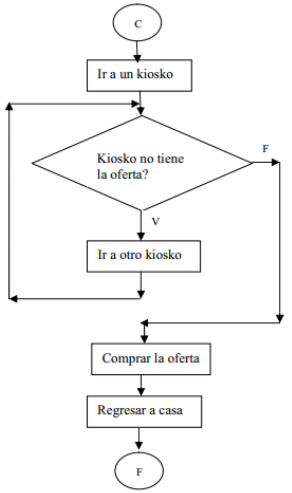
Representación **TEXTUAL**

Pseudocódigo

```
1   Proceso InformarPuedeVotar
2       Definir edad,FNac Como Entero
3       Escribir 'Ingrese anio de nacimiento '
4       Leer FNac
5       Mientras FNac<=0 Hacer
6           Escribir 'Ingrese anio de nacimiento '
7           Leer FNac
8       FinMientras
9       edad<-2024-FNac
10      Si edad>=16 Entonces
11          Escribir 'Puede votar'
12      Sino
13          Escribir 'No puede votar'
14      FinSi
15  FinProceso
```

Representación **GRÁFICA**

Diagrama de Flujo (DF)

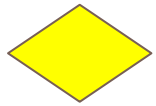


Representación de algoritmos: Diagrama de Flujo.

Un **Diagrama de Flujo de Datos** permite bosquejar gráficamente, el orden de las tareas involucradas en un algoritmo que da solución a un problema.

En esta materia se usan los siguientes símbolos para representar un DF:

inicio/fin 

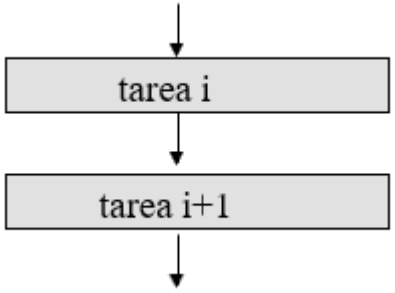
 condición

acción 

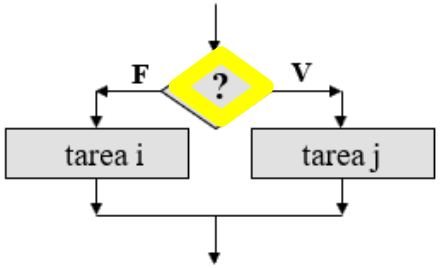
 flujo de ejecución

Algunas de las pautas para crearlos son:

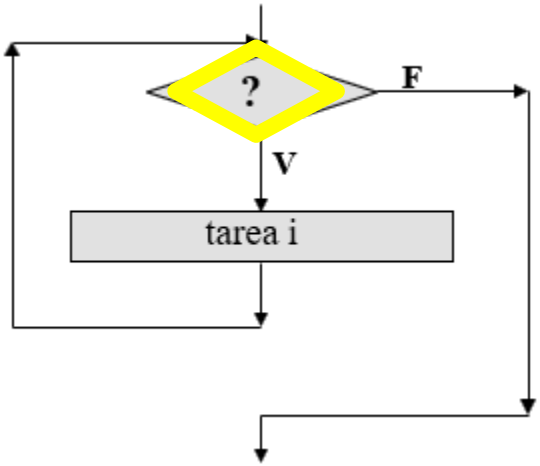
- Debe ser diseñado de forma que sea leído de arriba hacia abajo.
- Todo **DF** debe tener un **comienzo** y al menos un **final**.
- Para determinar el flujo de ejecución se deben usar **flechas** (NO líneas).



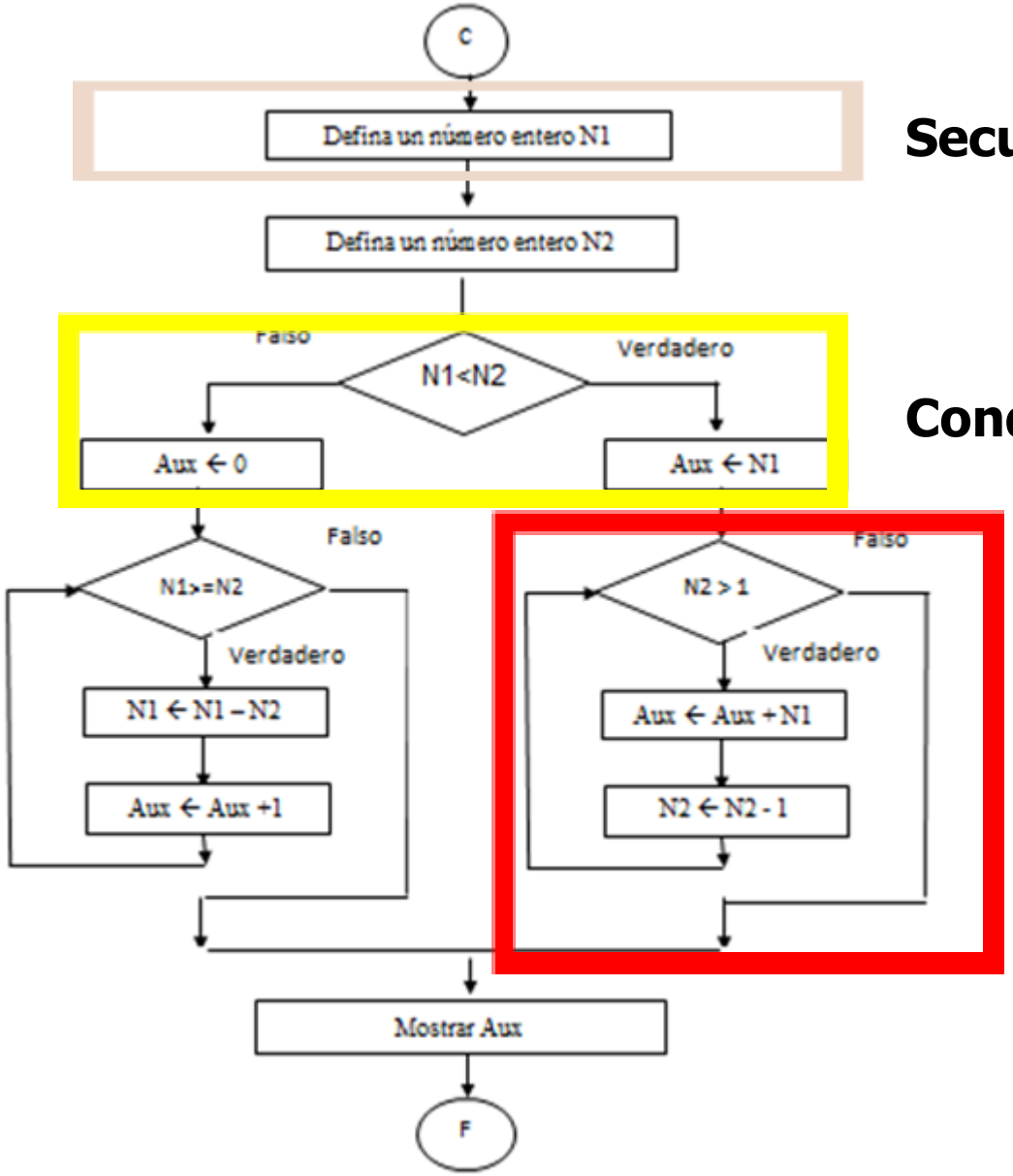
Secuencial: las tareas (acciones) se deben realizar en el orden en que se escriben, es decir, primero una, luego otra, desde la primera hasta la última (o de arriba hacia abajo).



Condicional (selección): las tareas (acciones) se realizarán dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.



Repetición (Iteración): una tarea o conjunto de tareas (acciones) se deben realizar en forma repetitiva dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.



Secuencial

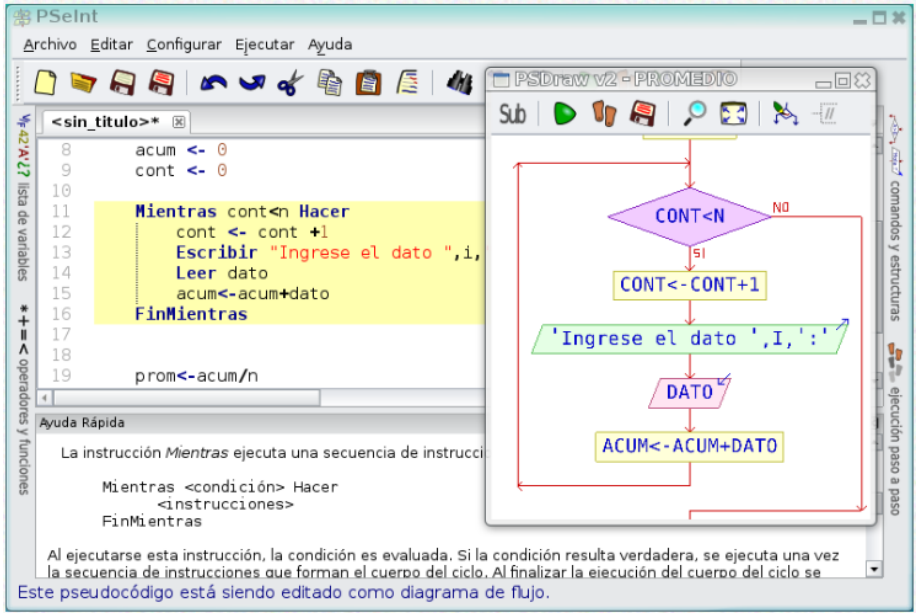
Condicional

Iteración

¿Qué es PSeInt?

https://pseint.sourceforge.net/

Es un editor e intérprete de algoritmos escritos en pseudocódigo que, además, genera su respectivo DF.



¿Por qué usar PSeInt?

- Software libre, sin necesidad de gastar dinero o violando los derechos de autor.
- Posee una extensa ayuda, que favorece el aprendizaje del lenguaje y la utilización.
- Posee previsualización y exportación a C, C++ y otros lenguajes, lo que ayuda a aprender estos y otros lenguajes.
- Se trata de un compilador que compila automáticamente cuando el usuario pulsa Ejecutar.
- Multiplataforma.

¿Por qué usar PSeInt?

Ejemplos de algoritmos en PSeInt

```
// Calcula el promedio de una lista de N datos

Proceso Promedio

    Definir i,N como Entero
    Definir acum,dato,prom como Reales
    Escribir "Ingrese la cantidad de datos:"
    Leer n

    acum<-0

    Para i<-1 Hasta n Hacer
        Escribir "Ingrese el dato ",i,":"
        Leer dato
        acum<-acum+dato
    FinPara

    prom<-acum/n

    Escribir "El promedio es: ",prom

FinProceso
```


```
// este es el ejemplo más simple de esta ayuda,
// toma dos numeros, los suma y muestra el resultado

Proceso Suma

    Definir A,B,C como Reales

    // para cargar un dato, se le muestra un mensaje al usuario
    // con la instrucción Escribir, y luego se lee el dato en
    // una variable (A para el primero, B para el segundo) con
    // la instrucción Leer

    Escribir "Ingrese el primer numero:"
    Leer A

    Escribir "Ingrese el segundo numero:"
    Leer B


    // ahora se calcula la suma y se guarda el resultado en la
    // variable C mediante la asignación (<-)

    C <- A+B


    // finalmente, se muestra el resultado, precedido de un
    // mensaje para avisar al usuario, todo en una sola
    // instrucción Escribir

    Escribir "El resultado es: ",C

FinProceso
```

Ejemplo completo

Enunciado: dado el año de nacimiento de una persona, informarle si puede o no votar.

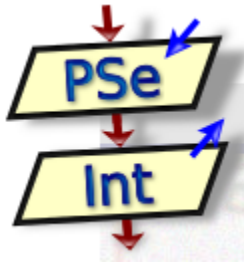
Versión 1

T1: Declarar los objetos necesarios.

T2: Ingresar el año de nacimiento de una persona controlando que sea un número positivo.

T3: Calcular la edad con el año de nacimiento ingresado.

T4: Informar si puede votar o no teniendo en cuenta la edad calculada en T3.



Versión 1

T1: Declarar los objetos necesarios.

T2: Ingresar el año de nacimiento de una persona controlando que sea un número positivo.

T3: Calcular la edad con el año de nacimiento ingresado.

T4: Informar si puede votar o no teniendo en cuenta la edad calculada en T3.

Versión 2

T1.1: Declarar Edad de tipo entero.

T1.2: Declarar Fnac de tipo entero.

T2.1: Ingresar el año de nacimiento de una persona en Fnac.

Mientras no sea positivo el número

    T2.2: Ingresar nuevamente el año de nacimiento.

T3: Calcular la edad restando el año actual con el año de nacimiento ingresado.

Si edad calculada en T3 es mayor o igual a 16

    T4.1: Informar que puede votar.

de lo contrario

    T4.2: Informar que no puede votar.

Versión 2

T1.1: Declarar Edad de tipo entero.

T1.2: Declarar Fnac de tipo entero.

T2.1: Ingresar el año de nacimiento de una persona en Fnac.

Mientras no sea positivo el número

T2.2: Ingresar nuevamente la fecha de nacimiento.

T3: Calcular la edad restando el año actual con el año de nacimiento ingresado.

Si edad calculada en T3 es mayor o igual a 16

T4.1: Informar que puede votar.

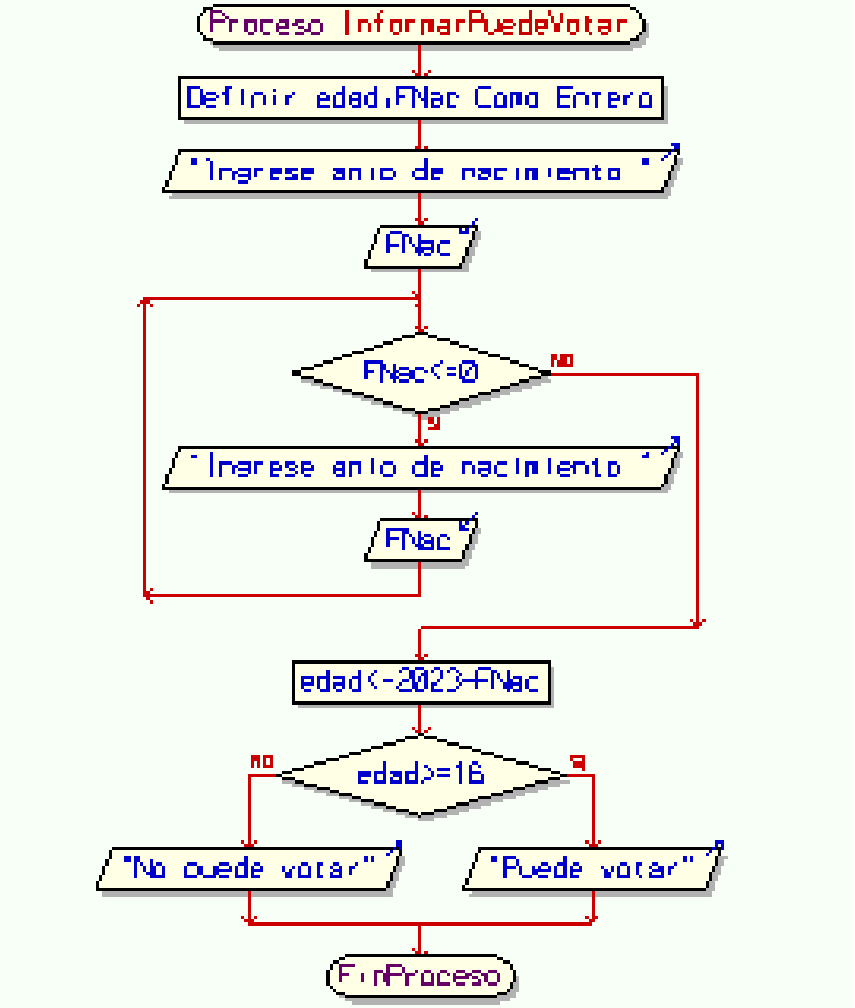
de lo contrario

T4.2: Informar que no puede votar.

Versión Final

```
Proceso InformarPuedeVotar
    Definir edad,FNac Como Entero
    Escribir 'Ingrese anio de nacimiento '
    Leer FNac
    Mientras FNac<=0 Hacer
        Escribir 'Ingrese anio de nacimiento '
        Leer FNac
    FinMientras
    edad<-2023-FNac
    Si edad>=16 Entonces
        Escribir 'Puede votar'
    Sino
        Escribir 'No puede votar'
    FinSi
FinProceso
```

```
Proceso InformarPuedeVotar
    Definir edad,FNac Como Entero
    Escribir 'Ingrese anio de nacimiento '
    Leer FNac
    Mientras FNac<=0 Hacer
        Escribir 'Ingrese anio de nacimiento '
        Leer FNac
    FinMientras
    edad<-2023-FNac
    Si edad>=16 Entonces
        Escribir 'Puede votar'
    Sino
        Escribir 'No puede votar'
    FinSi
FinProceso
```



Teoría
Pseudocódigo y Diagrama de Flujo

- ✓ **Representación de algoritmos: Pseudocódigo.**
  - Representación de datos (objetos).
  - Expresiones (aritméticas/lógicas/relacionales).
  - Asignación.
  - Entrada y salida de datos.
  - Estructuras de control: secuencial, condicional y de repetición.
- ✓ **Representación de algoritmos: Diagrama de Flujo.**
  - Simbología.
  - Ejemplo completo.
- ✓ **Pseudocódigo con PSeInt.**

Teoría

PSeInt y DF

¡Ya podemos comenzar con el Práctico!