MÉTODOS Y ELEMENTOS DE PROGRAMACIÓN

Índice

Bibliografía básica

Algarabel, S. y Sanmartin, J. (1990). *Métodos Informáticos aplicados a la Psicología*. Madrid: Pirámide.

Joyanes, L. (1994). *Programación en QuickBasic/Qbasic. 2ª Edición*. Madrid: McGraw-Hill.

1.Introducción

- Los **Lenguajes de programación** son aplicaciones específicas diseñadas para crear otras aplicaciones o programas. Son programas para crear programas.
- Se basan en un **sistema de instrucciones** preestablecidas que indican al ordenador lo que debe realizar
- Son **códigos integrados** compuestos por un **vocabulario**, con una **sintaxis** y una **semántica** que permite elaborar infinitas secuencias válidas de tareas e instrucciones

- Los lenguaje de programación permiten **crear programas específicos** que ofrezcan solución a programas particulares
- Para llevar a cabo cualquier tarea, el ordenador necesita tener **información** sobre la tarea y un **método** para ejecutarla
- **PROGRAMA:** ***conjunto de instrucciones convenientemente ordenadas*** *que indican al ordenador* ***qué procesos y tareas*** *debe seguir. Cada una de las instrucciones tiene un función específica y está escrita en un lenguaje que el ordenador entiende*

En resumen, los lenguajes de programación:

- ✓ Constituyen sistemas de palabras-órdenes (lengua o idioma), ya establecidos.
- ✓ Comprensibles tanto por el programador como por la máquina,
- ✓ Permiten desarrollar programas

Clasificación de los Lenguajes de Programación:

a) Lenguajes de Alto-Bajo nivel

b) Lenguajes Interpretados o Compilados

c) Lenguajes clásicos, visuales y de Internet

d) Por el objetivo

a) Lenguajes de Alto-Bajo nivel

El nivel de un lenguaje hace referencia a **su proximidad al lenguaje natural**, considerándose de más nivel cuanto más cercanos están a este y de menos nivel cuando más cerca están del lenguaje máquina

- ✓ El lenguaje de más **bajo nivel** o lenguaje máquina es el que utiliza el ordenador, el que la máquina entiende, basado en un sistema de 0 y 1. Son difíciles de aprender y manejar, ya que no resultan cercanos al ser humanos, pero son rápidos ya que evitan las traducciones intermedias. Fueron los primeros en aparecer.
- ✓ Los lenguajes de **alto nivel** son más fáciles de aprender y permiten despreocuparse de la arquitectura del ordenador. Ejemplos son: BASIC, PASCAL, FORTRAN, C (aunque este es intermedio)…

b) Lenguajes Interpretados o Compilados

Los LP deben traducirse (excepto el código máquina) para que sean interpretables (o inteligibles) por el ordenador. Esta traducción puede hacerse mediante:

- ✓ Los **Lenguajes interpretados**, se encargan de **realizar la traducción instrucción a instrucción** a la vez que se ejecuta el programa. Son más lentos, pero mejores cuando el proceso de traducción/ejecución se realiza en repetidas ocasiones, por lo que son más adecuados para principiantes.
- ✓ Los **Lenguajes compilados** traducen el **programa entero** y luego lo montan generando un **programa ejecutable** por si sólo. Una vez compilado el programa, el compilador no tiene porque estar presente, pudiéndose transportar el programa ejecutable a cualquier ordenador, sin necesidad de manejar el compilador.

c) Lenguajes clásicos, visuales y de Internet

✓ Los **Lenguajes clásicos** están **basados en un lenguaje** en el que se escribe el código necesario para realizar las operaciones que se requieren (posteriormente será traducido o compilado, generando un programa ejecutable). Los más conocidos son el BASIC, el PASCAL, el C, el COBOL y el CLIPPER

✓Los **Lenguajes visuales** son más avanzados y están **basados en objetos**. Cada entidad del programa (eventos, acciones..) es un objeto sobre el que se definen operaciones. Estos permiten almacenar los objetos (con todo su código) en una serie de librerías. Son lenguajes muy intuitivos que sustituyen las líneas de código por directas representaciones gráficas. P.ej.: Visual Basic

✓Los **Lenguajes de Internet** son lenguajes específicos diseñados para la **creación de páginas Web** y realizar su programación (motores de búsqueda, seguridad, establecimiento de comunicaciones…). Son la última generación de lenguajes. Existen distintos tipos dependiendo del grado de especialización. P ej.: JAVA, HTML

d) Por el Objetivo

Los programas pueden clasificarse por el objetivo para el que fueron creados:

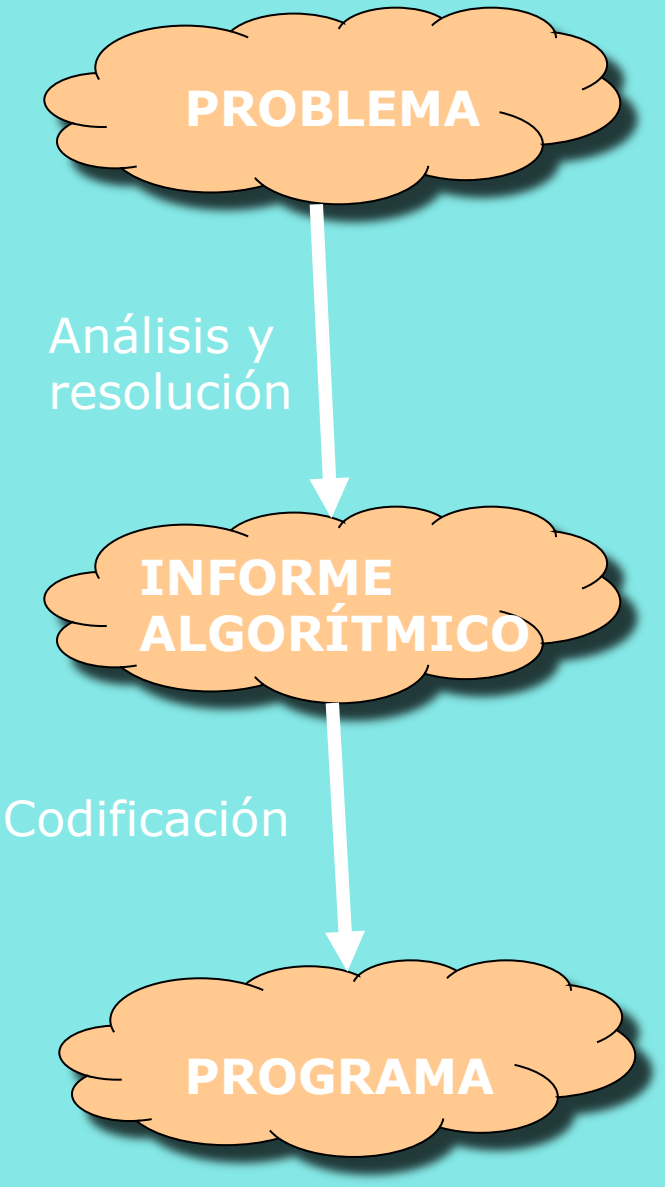
- ✓ **BASIC, PASCAL**: aprendizaje de programación
- ✓ **C y C++**: programación de sistemas
- ✓ **COBOL, RPG, Natural**: gestión de empresas
- ✓ **FORTRAN**: cálculo numérico
- ✓ **CLIPPER, ACESS, Dbase, Delphi, SQL**: bases de datos
- ✓ **Visual BASIC, Visual C**: programación en Windows
- ✓ **HTLM, JAVA, PERL**: Internet (páginas Web)
- ✓ **Lingo**: programas multimedia
- ✓ **Prolog, Lisp**: Inteligencia Artificial

2. Metodología de la programación: aspectos básicos

La programación puede ser entendida como un **PROCESO DE SOLUCIÓN DE PROBLEMAS** que tiene lugar en dos etapas:

1. Obtener la **solución del problema**. Comprende:
   a. un **análisis** detallado de la situación
   b. y, la búsqueda de **tácticas** que conduzcan a la solución
2. **Codificar** esta información en un lenguaje informático concreto

El método general de programación es:



1. **PROBLEMA**: actividad que no sabemos cómo llevar a cabo

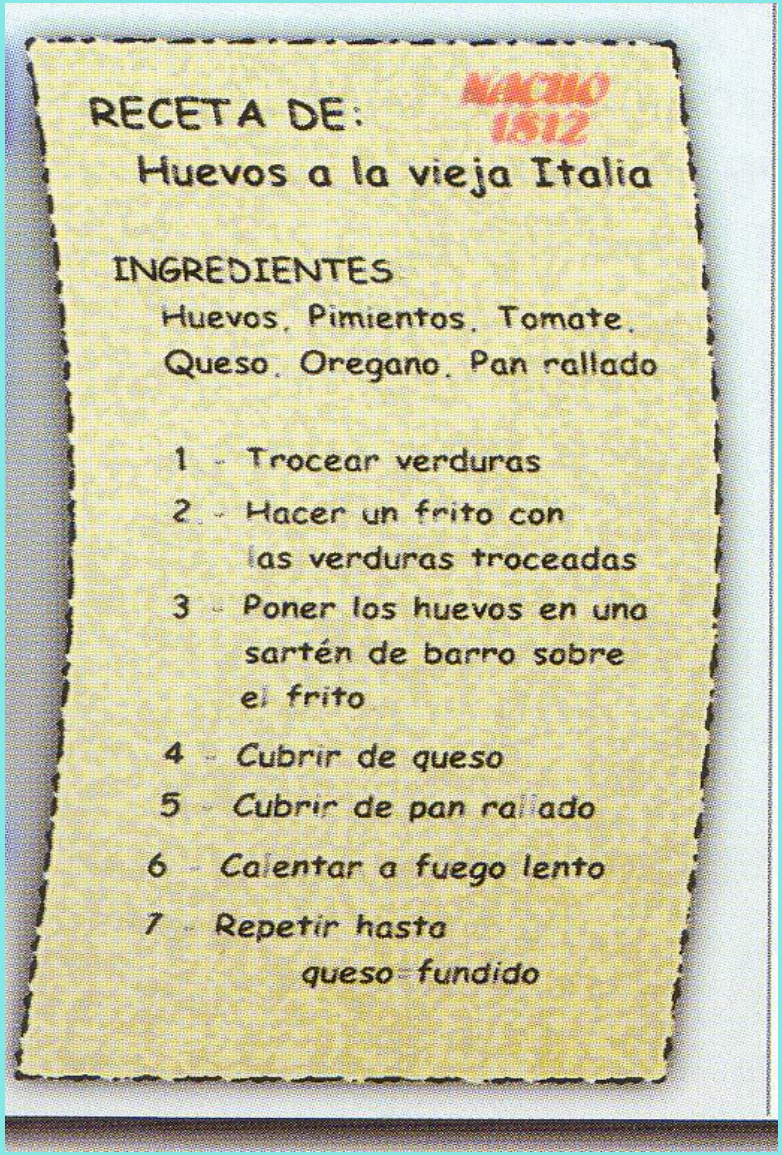
2. **INFORME ALGORITMICO**: la actividad se analiza en busca de la forma de resolución. El resultado se plasma en un informe que contiene:
   a. La descripción de la tarea y la enumeración de los objetivos a conseguir
   b. El procedimiento empleado
   c. Los recursos y elementos necesarios
   d. El algoritmo (la secuencia en la que hay que realizar cada una de las operaciones)

3. **PROGRAMA**: el algoritmo, traducido a un lenguaje de programación específico, se convierte en un programa que el ordenador puede ejecutar

¿Qué es un algoritmo?

Un método para resolver un problema mediante una serie de pasos precisos, definidos, finitos

- ✓ **precisos**: indicar el orden de presentación de cada paso
- ✓ **definidos**: si se siguen dos veces se obtiene igual resultado
- ✓ **finitos**: tiene un número determinado de pasos



Una receta de cocina puede resultar similar (en cuanto a concepto) a un algoritmo o programa

3. BASIC: Estructura general de un programa

Ejemplo de un programa en BASIC

```
10 CLS
20 PRINT “CALCULO DEL CONSUMO DE GASOLINA"
30 PRINT "POR FAVOR, INTRODUZCA LOS LITROS"
40 INPUT LITROS
50 PRINT "INTRODUZCA LOS KILOMETROS"
60 INPUT KILOMETROS
70 LET CONSUMO = LITROS / KILOMETROS * 100
80 PRINT  “El CONSUMO ES:”;  CONSUMO
```

Si lo ejecutamos..

```
CALCULO DEL CONSUMO DE GASOLINA
POR FAVOR, INTRODUZCA LOS LITROS
?450
INTRODUZCA LOS KILÓMETROS
?1000
EL CONSUMO ES: 45
```

- Una **sentencia o instrucción** escrita en lenguaje BASIC indica las tareas que debe realizar la computadora.
- Las sentencias constan de **una o varias palabras reservadas** (INPUT, REM, etc.) y una **sintaxis** (operadores, signos de puntuación, etc.) asociada que constituyen el formato de la misma y cuyo conocimiento es imprescindible para escribir programas.

- Las sentencias pueden ser:
  - **Ejecutables**: avanzan el flujo de control de la lógica del programa (leer una entrada, escribir una salida, realizar una operación, etc.)
  - **No ejecutables**: no avanzan en el flujo del programa (introducen comentarios, asignan variables, etc).

NORMAS GENERALES

El programa estará compuesto por una serie de instrucciones que han de ejecutarse siguiendo un orden determinado. De esta forma tres son las reglas generales:

- ✓ Cada instrucción se numera
- ✓ Las instrucciones sucesivas van en orden creciente
- ✓ Cada número va seguido de una instrucción

4. Sentencias o instrucciones en BASIC

REM

REM es una sentencia no ejecutable y permite introducir comentarios en los programas. A esta práctica se la denomina documentación.

FORMATO: **REM texto del comentario**

Reglas de funcionamiento:

- ✓ Se puede situar en cualquier parte del programa
- ✓ Se pueden poner tantas líneas REM con observaciones cómo y donde queramos

Ejemplo:

```
10 REM Esto es un programa de prueba
```

CLS

La sentencia CLS permite dejar la pantalla en blanco. Durante la escritura de programas se necesita diseñar, en puntos específicos, instrucciones de limpieza o borrado de la pantalla, a fin de dar claridad a los resultados o mensajes que aparecen en la pantalla

FORMATO: **CLS**

Reglas de funcionamiento:

- ✓ Se puede situar en cualquier parte del programa

Ejemplo:

```
10 CLS
```

PRINT

Visualiza datos por pantalla. El formato contiene una lista de expresiones que pueden ser constantes o variables.

FORMATO: **PRINT [lista de expresiones] [,][;]**

Reglas de funcionamiento:

- ✓ Una coma (separa un tabulador) o punto y coma (enlaza los mensajes) deben separar cada expresión
- ✓ Una sentencia PRINT sola imprime por pantalla una línea en blanco
- ✓ Las expresiones deben ir entre comillas. Las variables no.

Ejemplos:

```
PRINT "BASIC es un lenguaje de programación"
PRINT valor
PRINT "hola", "adios"
PRINT 3 + 5
```

Ejemplo

```
10 REM Esto es una prueba
20 CLS
30 PRINT "****************************"
40 PRINT "estoy aprendiendo a programar"
50 PRINT "****************************"
```

Por pantalla saldría:

```
****************************
estoy aprendiendo a programar
****************************
```

Variables y Constantes

Constante: es un valor que no cambia. Una constante conserva el mismo valor cada vez que se hace un cálculo o se ejecuta un programa. Ejemplo: Pi (3.1416).

a) **Constante numérica**: es un número que contiene un máximo de 8 caracteres

b) **Constante alfanumérica**: es una cadena de caracteres que se colocan entre comilllas. Ejemplo : "Hola"

Variable: Un valor que fluctúa. En su memoria central el ordenador reserva ciertas casillas para determinados valores que no se fijan. Pueden adquirir valores provisionales que sirven en un momento determinado y pueden sustituirse por otros (generalmente para identificarlos se hace con una letra)

a) **Variables numéricas**: (sólo pueden contener cifras) y se identifican por letras o palabras (A, B, C, num,..).

b) **Variables alfanuméricas**: (pueden contener cifras y letras). Se identifican por letras o palabras seguidas del símbolo dólar (A $, B$, expr$, ...).

Ejemplo variables numéricas

```
10 CLS
20 PRINT "CONSUMO DE GASOLINA"
30 PRINT "POR FAVOR, INTRODUZCA LOS LITROS"
40 INPUT LITROS (variable)
50 PRINT "INTRODUZCA LOS KILOMETROS"
60 INPUT KILOMETROS (variable)
70 CONSUMO = LITROS / KILOMETROS * 100 (variable)
80 PRINT CONSUMO (variable)
```

```
10 CLS
20 PRINT "CONSUMO DE GASOLINA"
30 PRINT "POR FAVOR, INTRODUZCA LOS LITROS"
40 INPUT A (variable)
50 PRINT "INTRODUZCA LOS KILOMETROS"
60 INPUT B (variable)
70 C =  A / B * 100 (variable)
80 PRINT C (variable)
```

Ejemplo variables alfanuméricas

```
10 CLS
20 PRINT "INTRODUZCA SU
   NOMBRE"
30 INPUT NOMBRE$ (variable)
40 PRINT "INTRODUZCA SU
   APELLIDO"
50 INPUT APELLIDO$ (variable)
60 NOMBREYAPELL$ = NOMBRE
   $ + APELLIDO$ (variable)
70 PRINT  "BUENAS TARDES" ;
80 NOMBREYAPELL$ (variable)
```

```
10 CLS
20 PRINT "INTRODUZCA SU
   NOMBRE"
30 INPUT A$ (variable)
40 PRINT "INTRODUZCA SU
   APELLIDO"
50 INPUT B$ (variable)
60 C$ = A$ + B$ (variable)
70 PRINT  "BUENAS TARDES" ;
80 C$ (variable)
```

LET

Asigna un valor a una variable. Esto es, almacena un determinado valor en una posición de memoria

FORMATO: **LET [Nombre + Valor de la variable]**

Reglas de funcionamiento:

- ✓ **El nombre es opcional y en la práctica se omite**

Ejemplos:

```
10 LET A = 17
20 LET B = A + 1
```

```
10 A= 17
20 B= A + 1
```

Ejemplo:

```
10 LET A = 17
20 LET B = 33
30 LET C = A + B
40 LET D = A - B
50 LET E = A * B
60 PRINT A, B, C, D, E
```

…Si lo ejecutamos (RUN)

```
17      33      50      -16     531
```

INPUT

Solicita al usuario valores para asignarlos a una o más variables. Si en nuestro programa se coloca la instrucción INPUT la ejecución del programa se detendrá, presentará un signo de interrogación, y esperará la respuesta del usuario que debe introducir los datos que desee utilizar. La instrucción lo que hace es asignar el valor que introduce el usuario a una variable

INPUT A (Espera una variable numérica)

INPUT A$ (Espera una variable alfanumérica)

FORMATO: **INPUT [Nombre de la variable]**

INPUT [Expresión entrecomillada][;][Nombre de la variable]

Reglas de funcionamiento:

✓INPUT puede actuar como la sentencia PRINT colocando una cadena de caracteres entrecomillados para que esta cadena aparezca en pantalla al mismo tiempo que el signo de interrogación. Es indispensable aquí colocar un punto y coma después de la cadena de caracteres.

Ejemplo:

```
10 CLS
20 REM programa para el cálculo de la longitud de una circunferencia
30 INPUT "CUAL ES SU NOMBRE" ; A$
40 INPUT "CUAL ES EL RADIO DE SU CIRCUNFERENCIA" ; R
50 LET C= 2  * 3.1416 * R
60 PRINT "QUERIDO " ;
70 PRINT  A$
80 PRINT "LA LONGITUD DE SU CIRCUNFERENCIA ES: ";
90 PRINT C
```

GOTO

Provoca un salto de línea en el flujo general del programa.

FORMATO: GOTO Número de línea

Reglas de funcionamiento:

- No es recomendable utilizarla con mucha frecuencia
- Según se aprendan otras instrucciones y operadores debe evitarse su uso

Ejemplos:

GOTO 30

IF THEN ELSE

Permite introducir condiciones para ejecutar una instrucción o instrucciones. Puede tener varios formatos

FORMATO: IF expresión lógica THEN sentencia(s) [:]

FORMATO:

IF expresión lógica THEN sentencia(s) ELSE sentencia(S)

FORMATO:

IF expresión lógica THEN
sentencia(s)
ELSE
Sentencia(s)
END IF

Reglas de funcionamiento:

- ✓ La expresión lógica puede ser una expresión lógica o relacional en la que se incluyan operadores lógicos o de comparación
- ✓ Las sentencias de la cláusula THEN sólo se ejecutan si la expresión lógica es verdadera.
- ✓ Si la condición es falsa, se ejecutan las instrucciones de la de la cláusula ELSE, si existe, o se sigue la siguiente instrucción del programa.

Ejemplos:

```
10 IF A>B THEN GOTO 30
60 IF A$="S"  THEN GOTO 10 ELSE GOTO 90
```

Operadores de comparación

OPERADORES	FUNCIÓN LÓGICA	FUNCIÓN ALFANUMERICA
=	Igual a	Igual a
>	Mayor que	Sigue
<	Menor que	Precede
>=	Mayor o igual que	Sigue o es igual
<=	Menor o igual que	Precede o es igual
<>	Distinto de	Distinto de

Operadores lógicos

OPERADOR	FUNCIÓN
AND	Y
OR	O
NOT	NO

FOR TO NEXT

Permite repetir un conjunto de instrucciones un número fijo de veces. Se denomina bucle controlado por contador, debido a que se necesita una variable numérica que actúa como contador y que se incrementa o disminuye en un valor constante a partir de un valor inicial, después de cada iteración del bucle

FORMATO:

FOR v=v_i TO v_f [STEP x]

sentencia(s)

NEXT v

Reglas de funcionamiento:

- ✓ Una sentencia FOR debe existir siempre con una sentencia NEXT y viceversa.
- ✓ La variable v controladora del bucle actúa de contador.
- ✓ v_i es el valor inicial de la variable v, v_f es el valor final y X es el incremento/decremento del contador; cuando se omite la sentencia STEP, el valor por defecto es el incremento en 1.
- ✓ La sentencia NEXT marca el final del bucle y punto en el que la variable contador se incrementa/decrementa.
- ✓ El bucle se termina cuando v toma un valor mayor que v_f

Ejemplo 1:

```
10 PRINT "N", "CUADRADO"
20 FOR N=1 TO 3
30   PRINT N, N*N
40 NEXT N
```

Ejemplo 2:

```
10 FOR N=1 TO 3
20     PRINT "N", "CUADRADO"
30     PRINT N, N*N
40 NEXT N
```

Salida ej 1:

```
N       CUADRADO
1             1
2             4
3             9
```

Salida ej 2:

```
N       CUADRADO
1             1
N       CUADRADO
2             4
N       CUADRADO
3             9
```

5. Resolución de problemas con el ordenador

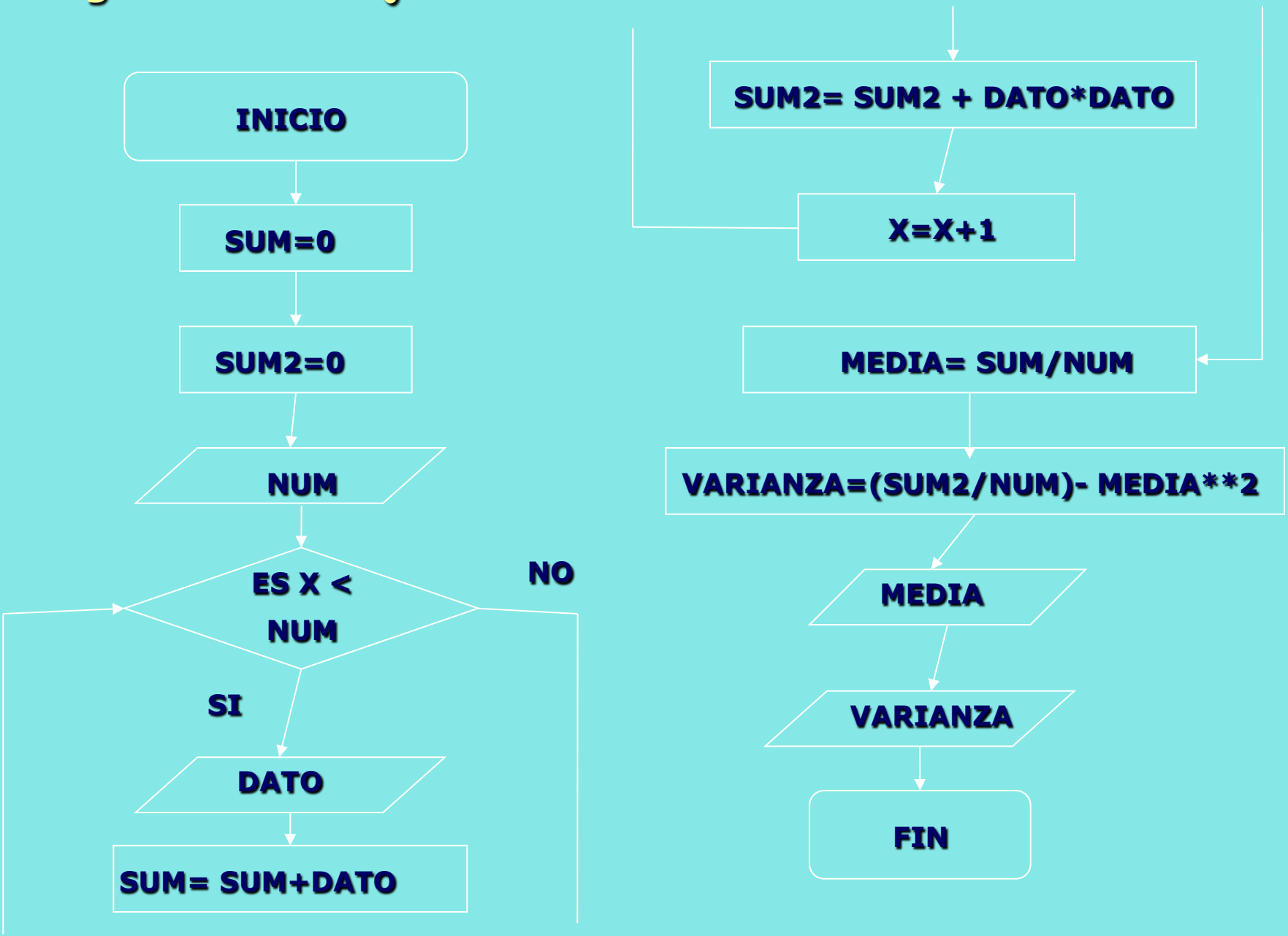
Pasos comunes en la creación de un programa entendido como un proceso de resolución de problemas:

1. **Análisis del problema**: definición del problema.
2. **Diseño del algoritmo**: proceso que convierte los resultados del análisis del problema en un diseño modular con refinamientos sucesivos que permita una posterior traducción a un lenguaje

Las herramientas más utilizadas para diseñar algoritmos son:

a. **Diagramas de flujo**: representación gráfica del algoritmo
b. **Pseudocódigo**: las instrucciones se escriben en palabra que facilita tanto la escritura como la lectura de programas

Diagramas de flujo



Pseudocódigo

```
INICIO
    {poner variables a cero}
    hacer SUMATORIO = 0
    hacer SUMATORIO2 = 0
    {entrada de datos}
    borrar la pantalla
    introducir NUMERO
    para X de 1 a NUMERO
            introducir DATO
            hacer SUMATORIO = SUMATORIO + DATO
            hacer SUMATORIO2 = SUMATORIO2 + DATO * DATO
    fin para
    {calculo de media y varianza}
    hacer MEDIA = SUMATORIO / NUMERO
    hacer VARIANZA=(SUMATORIO2/NUMERO) - MEDIA * MEDIA
    {presentacion de resultados}
    visualizar MEDIA
    visualizar VARIANZA
FIN
```

```
10  REM *********************************
20  REM **CALCULO DE MEDIA Y VARIANZA**
30  REM *********************************
40  REM *INICIALIZACION DE VARIABLES*
50  SUMATORIO = 0
60  SUMATORIO2 = 0
70  REM *ENTRADA DE DATOS*
80  CLS
90  PRINT "ESTE PROGRAMA CALCULA LA MEDIA Y VARIANZA"
100 INPUT "¨CUANTOS DATOS VA A INTRODUCIR"; NUMERO
110 FOR X = 1 TO NUMERO
120     INPUT "INTRODUZCA UN DATO ", DATO
130     SUMATORIO = SUMATORIO + DATO
140     SUMATORIO2 = SUMATORIO2 + DATO * DATO
150 NEXT X
160 REM *CALCULAR*
170 MEDIA = SUMATORIO / NUMERO
180 VARIANZA = (SUMATORIO2 / NUMERO) - (MEDIA * MEDIA)
190 REM *SALIDA RESULTADOS*
200 PRINT
210 PRINT
220 PRINT "MEDIA="; MEDIA
230 PRINT "VARIANZA="; VARIANZA
240 END
```

3. **Codificación de un programa**: traducción del algoritmo en un lenguaje de programación
4. **Compilación y ejecución**: traducirlo al lenguaje máquina. Este proceso se realiza mediante el compilador y el sistema operativo
5. **Verificación y depuración**: ejecutar el programa con una amplia variedad de datos de prueba para determinar si el programa tiene errores. La depuración es el proceso de encontrar los errores del programa y corregirlos

a. **Errores de sintaxis**: uso incorrecto de las reglas del lenguaje de programación
b. **Errores de ejecución**: se producen por instrucciones que el ordenador puede comprender pero no ejecutar (por ejemplo, dividir por cero)
c. **Errores lógicos**: se producen en la lógica del programa. La fuente de error suele ser el diseño del algoritmo

6. **Documentación y mantenimiento**: descripciones de los pasos a dar en el proceso de resolución un problema

Puede ser interna o externa. La documentación interna es la contenida en la línea de comandos (por ejemplo, líneas rem). La documentación externa incluye el análisis, los diagramas de flujo y/o pseudocódigos, manuales de usuario con instrucciones para ejecutar el programa y analizar los resultados…

La documentación es muy importante cuando se desea corregir posibles errores futuros o cambiar el programa. Tales cambios se denominan mantenimiento del programa