TEMA 1.
PROGRAMACIÓN DE UN COMPUTADOR

✓ PROGRAMA

- Para que un computador realice procesos útiles necesita tener almacenadas en memoria instrucciones precisas y exactas que le indiquen las operaciones que debe hacer.
- Un programa es el conjunto de instrucciones necesarias para resolver un problema en un computador, ordenadas en una secuencia adecuada.
- El conjunto de programas de un computador se denomina *software*, en contraposición al *hardware*, que se refiere a la parte física.
- El software de un computador se puede dividir en:
  - software básico (sistema operativo)
  - software de usuario

Lenguaje C

2. Lenguajes de programación

- ✓ **Los programas de un computador se realizan utilizando *lenguajes de programación.***
  - × **Notación o conjunto de símbolos y caracteres combinados entre sí de acuerdo con una sintaxis definida, para posibilitar la transmisión de instrucciones a la CPU.**
  - × **En última instancia, estos símbolos y caracteres son traducidos a un conjunto de señales eléctricas representadas en código binario (0 y 1)**
- ✓ **Lenguajes de bajo nivel:**
  - × **Lenguaje máquina**
    - Es el lenguaje usado directamente por el computador y compuesto de instrucciones codificadas en binario.
    - En este lenguaje una instrucción es una cadena de unos y ceros que permite a la unidad de control reconocer una operación elemental y ejecutarla.
    - Cada procesador tiene su lenguaje máquina propio, no entendible por otro tipo de procesadores.
  - × **Lenguaje ensamblador**
    - Es un lenguaje de bajo nivel (cercano a la máquina) en el que se utilizan nemotécnicos para representar las instrucciones del lenguaje máquina para un computador concreto.
    - Precisa de un alto conocimiento sobre la estructura y funcionamiento interno de un ordenador y manejo hábil de los códigos binario y hexadecimal.
    - Un *ensamblador* es un programa que lee, como datos de entrada, un programa escrito en lenguaje ensamblador y produce, como resultado, un programa en lenguaje máquina.

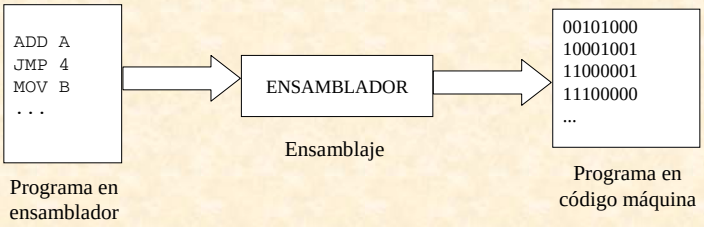
Lenguaje C

2. Lenguajes de programación

- ✓ **Ejemplo de instrucciones en ensamblador en un microprocesador Motorola.**

MOVE.L #$35,D2	**Copia el número 35 en hexadecimal al registro D2.**
MOVE.B (A0)+,D1	**Copia en D1 el contenido de la posición de memoria direccionada por A0 incrementando el contenido de A0 en una unidad.**
ADD.L D1,D2	**Suma el contenido del registro D1 al contenido de registro D2 y guarda el resultado en el registro D2.**

- ✓ **Proceso de ensamblaje**



Programa en ensamblador — Ensamblaje — Programa en código máquina

2. Lenguajes de programación

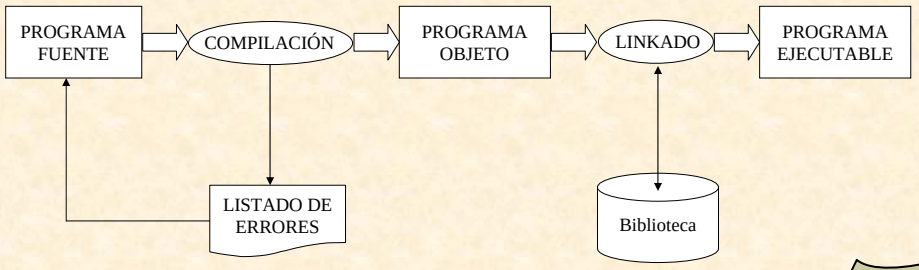
- El desarrollo de los ensambladores supuso un paso importante, pero los programadores estaban aún forzados a pensar en términos de instrucciones máquina individuales.
- El lenguaje máquina y el lenguaje ensamblador son "lenguajes orientados a la máquina" y se denominan ***lenguajes de bajo nivel***, porque son dependientes de la arquitectura del computador que los soporta.

✓ Lenguajes de alto nivel

- Son lenguajes de programación más cercanos a los lenguajes naturales, tales como el inglés, y no tan dependientes de las características de los computadores.
- Ejemplos: C, C++, Pascal, Basic, COBOL, FORTRAN, ...
- Son independientes de la arquitectura del computador utilizado, los programas desarrollados con estos lenguajes pueden ser ejecutados en ordenadores con distinto microprocesador.
- Los programas escritos en estos lenguajes se traducen a instrucciones en lenguaje máquina mediante programas especiales llamados *compiladores* e *intérpretes*.

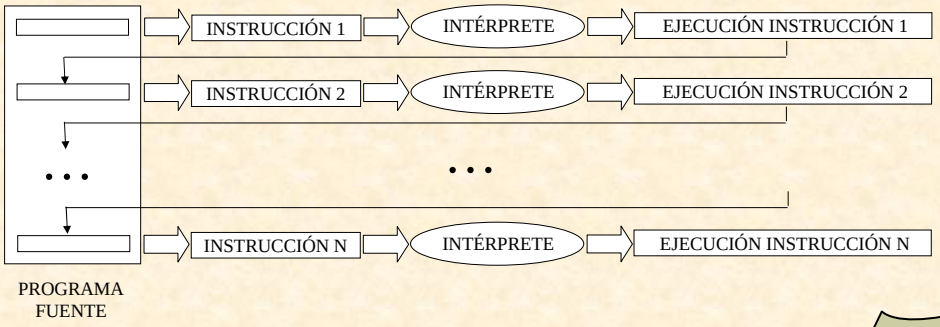
2. Lenguajes de programación

- Un ***compilador*** analiza el programa, comprobando su sintaxis e indicando los errores, si los hay, y luego genera el programa en lenguaje máquina.
  - Programa fuente: es el programa en lenguaje de alto nivel.
  - Programa objeto: es el programa en lenguaje máquina generado por el compilador a partir del programa fuente.
  - El programa objeto necesita otro proceso además de la compilación, el enlazado ("linkado").



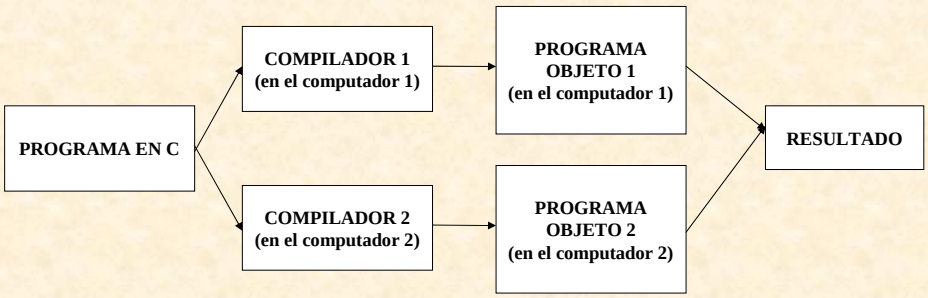
2. Lenguajes de programación

- **Un *intérprete* es un programa encargado de procesar y traducir cada instrucción o sentencia de un programa escrito en lenguaje de alto nivel a código máquina y después ejecutarla.**
  - Más lento que un compilador en ejecuciones sucesivas del programa.



2. Lenguajes de programación

- **Si escribimos un programa en un lenguaje de alto nivel, podemos ejecutarlo en cualquier computador que tenga el compilador apropiado.**



- **En las prácticas de Fundamentos de Informática vamos a utilizar un entorno de programación llamado *Dev C++*, que incluye un compilador de C y C++.**

Lenguaje C

3. Algoritmos

- ✓ **Todo programa está compuesto por un conjunto de órdenes o instrucciones capaces de manipular un conjunto de datos.**

- ✓ **Entrada de datos:**
  - × **Instrucciones que toman datos de un dispositivo o periférico externo, depositándolos en memoria principal para ser procesados.**
- ✓ **Proceso o Algoritmo:**
  - × **Instrucciones que procesan los datos, realizando una serie de cálculos con los mismos, dando lugar a los resultados, que también son depositados en memoria principal.**
- ✓ **Salida de datos.**
  - × **Los resultados del programa se muestran a través de algún dispositivo de salida.**

Lenguaje C

3. Algoritmos

- ✓ **Un *algoritmo* es una descripción concisa (no ambigua) y ordenada de la secuencia de un número finito de instrucciones necesarias para la resolución de un problema.**

- ✓ **Ejemplos:**
  - × **El procedimiento que utilizamos para ordenar una lista de números.**
  - × **El procedimiento que utilizamos para calcular el factorial de un número.**
  - × **El procedimiento que utilizamos para calcular el m.c.d de dos números.**

- ✓ **Para que un computador resuelva un problema, debemos indicarle los pasos sucesivos que debe seguir para resolverlo.**

- ✓ **El algoritmo se traducirá posteriormente a un lenguaje de alto nivel y se compilará, obteniendo finalmente el programa objeto que será ejecutado para resolver el problema.**

3. Algoritmos

- ✓ **Representación de un algoritmo**
  - × **Para diseñar un algoritmo independientemente del lenguaje de programación se suelen utilizar distintas técnicas de representación, entre ellas el *lenguaje algorítmico* o *pseudocódigo* y los *diagramas de flujo* o *flujogramas*.**
  - × **Lenguaje algorítmico o pseudocódigo**
    - Se utiliza un vocabulario restringido y unas reglas para construir las frases.
    - Lenguaje intermedio entre el lenguaje natural y el lenguaje de programación.
    - En el cuerpo del programa se deben definir las variables que se van a utilizar durante el programa y las instrucciones que se deben ejecutar.
    - La secuencia de operaciones se expresa mediante la enumeración en líneas sucesivas.

```
ALGORITMO Nombre
    VARIABLES
        Tipo Var 1    Nombre Var 1
        Tipo Var 2    Nombre Var 2
        .....
    INSTRUCCIONES
        Instrucción 1
        Instrucción 2
        .....
FIN
```

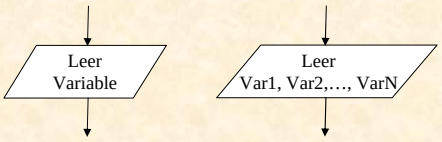

3. Algoritmos

- ✓ **Diagramas de flujo**
  - × **Símbolos principales del tratamiento**

	**PROCESO. Operaciones de tratamiento**		**Inicio, Fin del algoritmo**
	**Operación de Entrada / Salida**		**Conector**
	**Comparación. Toma de decisiones.**		**Línea de flujo**

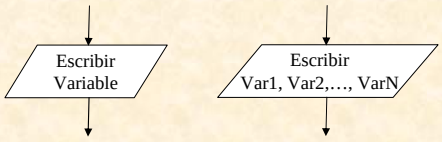
4. Tipos de instrucciones

- **Instrucciones primitivas:**
  - **Instrucciones de entrada**



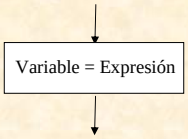
Leer Variable — **Leer** Var1, Var2, VarN

  - **Instrucciones de salida**



Escribir Variable — **Escribir** Var1, Var2, VarN

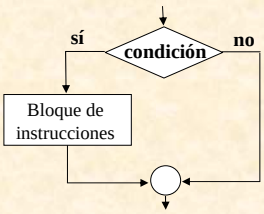
  - **Instrucciones de asignación**



Variable = Expresión

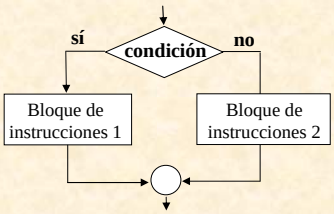
4. Tipos de instrucciones

- **Instrucciones condicionales:**
  - **Alternativa simple**



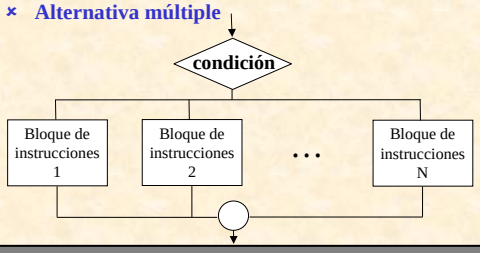
Si Condición
  Instrucción 1
  Instrucción 2
  …
  Instrucción N
Fin Si

  - **Alternativa doble**



Si Condición
  Instrucción 1A
  Instrucción 1B
  …
  Instrucción 1N
Si No
  Instrucción 2B
  Instrucción 1B
  …
  Instrucción 1N
Fin Si

  - **Alternativa múltiple**

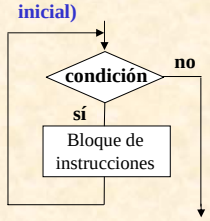


Según_Valor Expresión
  Valor 1:
    Bloque de instrucciones 1
  Valor 2:
    Bloque de instrucciones 2
  …
  Valor N:
    Bloque de instrucciones N
  Otros:
    …
Fin Según_Valor

4. Tipos de instrucciones

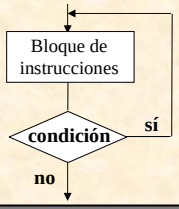
✓ Repeticiones o bucles:

× Estructura Mientras (bucle con condición inicial)



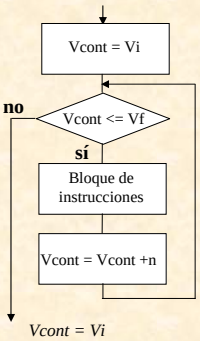
Mientras Condición
Instrucción 1
Instrucción 2
…
Instrucción N
Fin Mientras

× Estructura Repetir-Mientras (bucle con condición final).

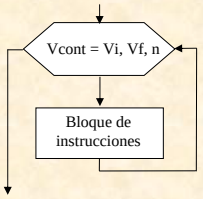


Repetir
Instrucción 1
Instrucción 2
…
Instrucción N
Mientras Condición

× Estructura Para



Vcont = Vi
Mientras *Vcont* <= *Vf*
Instrucción 1
Instrucción 2
…
Instrucción N
Vcont = Vcont+n
Fin Mientras



Para *Vcont* de *Vi* a *Vf* con *Inc* = *n*
Instrucción 1
Instrucción 2
…
Instrucción N
Fin Para

4. Tipos de instrucciones

✓ Contadores, acumuladores e interruptores

× Contadores:

- Variable destinada a contener un valor que se irá incrementando o decrementando en una cantidad fija y constante.
- Se utilizan para el control de procesos repetitivos. Contabilizan un conjunto de sucesos que se repiten en un programa mediante el uso de bucles (Mientras, Repetir-Mientras y Para).
- Todo contador debe tomar un valor inicial antes de ser utilizado.

× Acumuladores:

- Variable destinada a almacenar resultados de operaciones previas de manera sucesiva, lo que permitirá obtener el total acumulado de dichas cantidades.
- En aquellos casos en los que se pretende obtener el total como suma de distintas cantidades, el acumulador debe ser inicializado a 0.
- En aquellos casos en los que se pretende obtener el total como producto de distintas cantidades, el acumulador debe ser inicializado a 1.

× Interruptores (switches):

- Son variables que únicamente pueden tomar dos valores (0 o 1).
- Su objetivo es:
  - Recordar en un determinado lugar del programa si ha ocurrido un determinado suceso con antelación.
  - Hacer que dos acciones diferentes se ejecuten alternativamente en un proceso repetitivo..
- Deben ser inicializados a uno de los dos posibles valores.

5. Ejemplos de algoritmos

- ✓ **Ejemplo 1: Escribir un algoritmo que, dado un número real, obtenga su valor absoluto**
  - × Para almacenar el número necesitamos lo que en programación se denomina una *variable*, que en este caso será de tipo real.
  - × También necesitamos otra variable para almacenar el resultado, que también será de tipo real.
  - × El algoritmo en pseudocódigo puede ser:

```
ALGORITMO absoluto

    VARIABLES
    variables reales x, xAbs

    INSTRUCCIONES
    Leer x
    Si (x >= 0)
        xAbs = x
    Si no
        xAbs = -x
    Fin si
    Escribir xAbs
FIN
```


5. Ejemplos de algoritmos

- × Otra versión de este algoritmo que utiliza una variable:

```
ALGORITMO absolutoBis

    VARIABLES
    variables reales x

    INSTRUCCIONES
    Leer x
    Si (x < 0)
        x = -x
    Fin si
    Escribir x

FIN
```

4. Ejemplos de algoritmos

✓ **Diagrama de flujo de la segunda versión:**