# Unidad 00 - TP 00

**Date:** 2025-03-18  
**Student:** Lapenta Carlos Matías **Course:** Algoritmos y Estructuras de Datos I **Unit:** 0 **Assignment:** 0 **Topic:** Introducción **GDrive Link:** [TP - Unidad 1 Parte I.docx - Google Drive](https://drive.google.com/file/d/1P_PEGZIvEWyV3vZisp7nJWPkOYZP3_jA/view) **Local Link:** <TP - Unidad 0.docx>

## Answers

1. Un **programa** es una *secuencia lógica de instrucciones escritas* (**algoritmo**) *en un lenguaje de programación*. El objetivo de tal secuencia es realizar una tarea, ya sea controlar el hardware, hacer cálculos o procesar datos.
2. Un **algoritmo** es una *secuencia de pasos definidos, ordenados y finitos* que se siguen para resolver un problema. Es independiente del medio en el que se emplea y el formato. Un algoritmo puede ser la receta para hacer un café, los pasos para cambiar el aceite del auto, o las instrucciones de un robot para moverse.
3. Un **lenguaje de alto nivel** es un lenguaje de programación diseñado para ser fácil de leer y escribir por humanos. La sintaxis es similar al lenguaje natural, #Cpp, #CSharp y #C son algunos ejemplos. Estos lenguajes abstraen los detalles de la comunicación con el #hardware , precisando un intérprete o compilador que los traduzca a [codigo maquina](machine code).
   1. **Compilación:** El compilador traduce [codigo fuente](source code) de alto nivel a <codigo objeto> o [codigo maquina](machine code). En el proceso incluye, la revisión de sintaxis, expansión de macros y el anexo (o fusión) de las direcciones de las bibliotecas referenciadas por el código fuente, usando un [enlazador](linker). El producto final es un [archivo binario](binary code) o [ejecutable](#exe) como punto de entrada para la ejecución del programa.
   2. **Ejecución:** Con el entorno de ejecución adecuado, el archivo ejecutable se carga en memoria y la [CPU](Control Process Unit) ejecuta instrucciones secuencialmente, interactuando con el OS y con el hardware, dependiendo de cada instrucción.
   3. **1980**: <Bjarne Stroustrup> desarrolla **C++** en los laboratorios Bell, extendiendo el lenguaje **C** con características de [programación orientada a objetos](Object Oriented Programming) #OOP (como [clases](Classes) y [herencia](Inheritance)).
   4. **1985**: Primera versión comercial.
   5. **1998**: Se estandariza (*C++98*).
   6. Actualmente, sigue evolucionando (*C++11*, *C++17*, *C++20*), añadiendo funcionalidades modernas.
   7. **CPU**: Ejecuta instrucciones y procesa datos.
   8. **Memoria RAM**: Almacena temporalmente datos y programas en uso.
   9. **Disco Duro/SSD**: Almacenamiento permanente.
   10. **Tarjeta Madre**: Conecta todos los componentes.
   11. **Periféricos**: Entrada/Salida (teclado, monitor, etc.)
   12. **FUNCIONAMIENTO**: La [CPU](Control Process Unit) accede a datos de la [Memoria RAM](Random Access Memory) o [Disco Duro/SSD](Disc Drive), los procesa y envía resultados a [Periféricos](Peripherals), coordinado por él [sistema operativo](Operative System).
4. El conocimiento de la ética en la industria del software se ha vuelto fundamental para evitar las siguientes problemáticas en individuos y/o sociedades:
   1. Dilemas con la privacidad de los datos
   2. Problemas de seguridad cibernética / digital.
   3. Sesgo en los algoritmos entrenados.
   4. Distintos impactos del software en la sociedad.
5. Las **etapas básicas** para escribir un **programa** consisten en:
   1. *Fase de Resolución*:
      1. **Análisis**: Entender qué debe resolver el programa.
      2. **Diseño**: Planear la lógica (pseudocódigo/diagramas).
      3. **Verificación**: Comprobar si el algoritmo resuelve el problema.
   2. *Fase de implementación*:
      1. **Desarrollo**: Escribir el código en un lenguaje.
      2. **Prueba**: Verificar que el programa funcione correctamente.
   3. *Fase de Mantenimiento*:
      1. **Ejecución**: Utilización del programa en contexto real.
      2. **Mantenimiento**: Corregir errores o mejorar.
   4. **Documentación**: Es transversal a todas las fases de producción de un programa.
6. Un [compilador](Compiler) es un **programa que traduce** [codigo fuente](Source Code) a [codigo objeto](Object Code) o [codigo maquina](Machine Code) ([ejecutable](Executable)), verificando errores de sintaxis, optimizando el código y/o usando el [enlazador](Linker) del entorno de desarrollo para anexar o fusionar las *librerías referenciadas* en las instrucciones. El archivo generado un *binario* o ejecutable.
7. Los aspectos que pueden **diferenciar** él [software](Software) y [hardware](Hardware), pueden resumirse en:
   1. **Hardware**: Referido a **componentes físicos** ([CPU](Control Process Unit), [Disco Duro/SSD](Disc Drive), teclado, placa madre, etc.)
   2. **Software**: Todo lo referido a **programas e instrucciones** que controlan el hardware ([sistema operativo](Operative System), [aplicaciones](Application Program))
8. Un método para la **resolución de problemas** mediante **algoritmos** puede definirse en los siguientes pasos:
   1. **Análisis y Especificación**: Entender (*definir*) el problema a resolver y especificar lo que *DEBE* hacer la solución (que se espera que realice el *algoritmo*). Implica también definir las *entradas* del sistema y las *salidas* esperadas.
   2. **Solución General (Algoritmo)**: Desarrollo de la secuencia de pasos que resuelve dicho problema. En esta etapa se pueden usar distintos métodos, como dividir el problema en subproblemas (*dividir y vencer*), *resolución por analogía*, etc.
   3. **Verificación**: *Seguir* los pasos del *algoritmo*, de forma *ordenada y precisa*. Asegurarse de que el sistema *realiza correctamente la tarea esperada*. Se comprueba por cada entrada que las salidas sean las que se especificaron en la primera etapa.

##### Bibliografía

<Compiladores_e_interpretes_teoria_y_prac.pdf> <ingenieria de Software.pdf> <material_Complementario_Ing_soft.pdf> <Programacion-y-resolucion-de-problemas-con-c++.pdf> <Undad 0.1 Bjarne Stroustrup - El lenguaje de programacion C++.pdf>