# Algoritmos y Estructuras de Datos I - Unidad 01 - Actividad 01

* **Estudiante**: Lapenta Carlos Matías

## 📌 Descripción de la Actividad

* [Plataforma de Entrega](https://classroom.google.com/c/Njg4NzQ0NDQwNjQy/a/Njk4NjAyNDg2MzQ5/details) | [Instrucciones](https://drive.google.com/file/d/1P_PEGZIvEWyV3vZisp7nJWPkOYZP3_jA/view)
* Objetivos principales:
  + Responder el siguiente cuestionario en función de la bibliografía Obligatoria.
* Formato de entrega requerido: **.docx**

## 📝 Desarrollo

### 1. Definir los conceptos de Informática y computadora.

La **informática** es el conjunto de teorías y métodos sobre el *procesamiento automático de la información*. Este procesamiento puede darse *a través de sistemas digitales* como computadoras.

Una **computadora** es un objeto que calcula, un *dispositivo electrónico programable* capaz de *almacenar, recuperar y procesar datos*. Puede introducir, almacenar, recuperar, procesar y enviar datos. Compara valores y *realiza operaciones aritméticas*.

### 2. Definir Hardware y Software

**Hardware** se refiere a los *componentes físicos de una computadora*. Estos componentes son dispositivos electrónicos que se pueden clasificar en:

* **Unidad de Memoria**
* **Unidad Aritmética/Lógica** (ALU)
* **Unidad de Control** (CU)
* **Dispositivos de Entrada**
* **Dispositivos de Salida**
* **Dispositivos de Almacenamiento Auxiliares**

La UAL y la UC comprenden la “*Unidad Central de Procesamiento*” o *CPU* (Central Processing Unit), mientras que a los dispositivos de entrada/salida y almacenamiento auxiliar se los conoce como “*Periféricos*”.

El **Software** comprende el *conjunto de programas que permiten operar el hardware*. Los programas son series de instrucciones ordenadas y finitas que *se ejecutan en la computadora* e interactúan con el hardware.

### 3. Describir los roles de los Programadores (extender lo visto en clase)

Los programadores son los encargados de traducir algoritmos a secuencias de instrucciones para que la computadora pueda ejecutarlas. Desarrollan soluciones mediante lenguajes que el hardware entiende y puede procesar.

Pueden trabajar solos o en equipo y tienen la responsabilidad de entregar soluciones de calidad sin errores, independientemente del dominio en el que se encuentre trabajando y las herramientas que use.

### 4. Relacionar los conceptos de la Diapositiva 9

La **algoritmia** es la *resolución de problemas paso a paso*. Son métodos lógicos y eficientes, en el diseño e implementación de instrucciones secuenciales y ordenadas, que realizan un trabajo.

La **programación** es la *traducción de esas instrucciones* o pasos en *lenguajes que las computadoras puedan entender y ejecutar*. Un algoritmo digital es un programa que puede ser interpretado por la computadora y realiza una tarea.

### 5. Describir el Modelo de Von Neumann

El **Modelo de Von Neumann** plantea 4 componentes principales para un sistema computacional:

* **Dispositivos I/O** (Entrada/Salida): Permite la comunicación con el exterior (Periféricos), *envía y recibe información del CPU*
* **Unidad Central de Procesamiento** (CPU): Procesa los datos recibidos, decide cómo procesarlos, dónde almacenarlos (RAM o secundario) y cómo mostrarlos siguiendo instrucciones del software.
  + **Unidad Aritmético/Lógica** (ALU): Realiza los *cálculos y operaciones lógicas*
  + **Unidad de Control** (CU): Coordina las operaciones siguiendo instrucciones del programa
* **Memoria** (RAM): Almacena temporalmente *datos* e *instrucciones* (programas) en el mismo espacio
* **Buses**: Canales que permiten la *comunicación entre componentes*

La inmediata adición de **almacenamiento secundario** o almacenamiento permanente permitió al sistema de una computadora *guardar los programas y datos* almacenados, sin que se borraran al apagarse.

### 6. Qué es un Lenguaje de Programación, qué permite

Un **lenguaje de programación** es un *conjunto de reglas, símbolos y palabras especiales* que se utilizan para darle instrucciones a una computadora. Permiten al programador expresar algoritmos en un idioma que el dispositivo electrónico programable entiende y puede ejecutar.

### 7. Describir los lenguajes de Alto, Medio y Bajo Nivel, dar ejemplos

El **nivel de los lenguajes** está dado por la *abstracción* o distancia que posean del lenguaje binario que la CPU utiliza para realizar las operaciones aritméticas.

*A mayor semejanza con el lenguaje natural, mayor abstracción* y mayor es el nivel del lenguaje. Se puede decir que *cuanto más “cerca del metal”* se encuentra el lenguaje, *menor abstracción* y menor nivel tiene.

**Niveles**:

* **Alto**: Son lenguajes que *se asemejan al idioma inglés* y a otros lenguajes naturales (C++, Java, Fortran), no permiten el control de los recursos (hardware) de manera directa.
* **Medio**: Son considerados los lenguajes que, si bien implementan sintaxis de lenguaje natural, *todavía permiten control sobre el hardware* de manera directa (C y Pascal serían considerados de nivel medio)
* **Bajo**: Son lenguajes que no se asemejan al lenguaje natural, tienen poco nivel de abstracción y permiten el *control total del hardware de manera directa*. (Código Máquina y Ensamblador)

### 8. Qué es un Compilador, qué función cumple

Un **compilador** es un **programa que traduce** **código fuente** a **código objeto** o **código máquina** (**ejecutable**). Realiza análisis morfológico, sintáctico y semántico antes de producir el código objeto.

1. **Análisis morfológico**: Divide el programa en unidades sintácticas (símbolos, palabras clave, literales, todo componente de una expresión es separado en “tokens”)
2. **Análisis sintáctico** (parser): Revisa la estructura gramatical. (Analiza las expresiones de las instrucciones)
3. **Análisis semántico**: Comprueba la corrección del significado de las instrucciones dentro de una rutina.

El código objetivo generado se puede guardar y ejecutar repetidamente sin recompilar.

### 9. Qué es un Intérprete, cómo funciona

Un **intérprete** es un *traductor que analiza y ejecuta directamente* un programa escrito en un código de alto nivel, procesa y ejecuta instrucción por instrucción en *tiempo de ejecución*.

Tiene un funcionamiento similar al compilador, con los respectivos *análisis morfológicos, sintácticos y semánticos*, pero no genera código objeto, en su lugar, *analiza y ejecuta secuencialmente* cada instrucción.

### 10. Relacionar Java Virtual Machine, .Net Framework con JIT (ampliar lo visto)

**Java Virtual Machine** (JVM) y **Common Language Runtime** (CLR, .NET virtual machine) son *máquinas virtuales, entornos de ejecución*. *Traducen “bytecode” o código intermedio* a lenguaje máquina *y ejecutan instrucciones* en la computadora.

El *código intermedio* (**bytecode** para Java y **CIL** para .NET) es *generado previamente por compiladores* de los respectivos entornos de desarrollo (JDK o .NET SDK).

**Just-In-Time compilation** es una estrategia de *compilación dinámica en tiempo de ejecución*. Partes del bytecode se compilan a lenguaje máquina en tiempo de ejecución, para aprovechar la eficiencia de la ejecución de código nativo.

### 11. Relacionar Problema, Algoritmo y Programa (armar un mapa conceptual o similar, trabajarlo en grupo)



Mapa Conceptual Unidad 01 Actividad 01

### 12. Describir el concepto que afirma “SOMMERVILLE” en su libro Ingeniería de Software

Según Sommerville, un **modelo del proceso de software** es una *representación abstracta o simplificada del proceso de desarrollo de software*.

Cada representación ofrece una visión parcial del proceso de desarrollo. Un *modelo de actividad del proceso* puede mostrar actividades y su secuencia, pero no detallar roles o personas involucradas.

Los modelos no son descripciones definitivas, son abstracciones que explican diferentes enfoques del proceso de desarrollo. Son considerados “*marcos de trabajo*” (frameworks).

### 13. Describir el Modelo Iterativo e Incremental

En el **modelo iterativo e incremental**, el software se *desarrolla a través de múltiples ciclos* (iteraciones), en *cada iteración* se produce un incremento funcional del software, *añadiendo nuevas funcionalidades* y agregando valor con cada ciclo.

### 14. Relacionar UML, Ingeniería de Software y modelos

El **Lenguaje de Modelo Unificado** (UML) es un *lenguaje gráfico estándar* que *permite la creación de modelos*. Estos **modelos** son *representaciones abstractas, que permiten comprender, diseñar, comunicar y documentar* un sistema de software en su ciclo de desarrollo.

La **Ingeniería de Software** se ocupa de todos los aspectos de la producción de software (especificación, diseño, desarrollo, validación, etc.), y *utiliza modelos (a menudo con UML) para gestionar la complejidad del software*.

### 📚 Material de Referencia

* <Unidad 1 Parte 1.pptx>

### 📂 Archivos Adjuntos

* <lapenta-carlos-matias-unidad-01-actividad-01.docx>