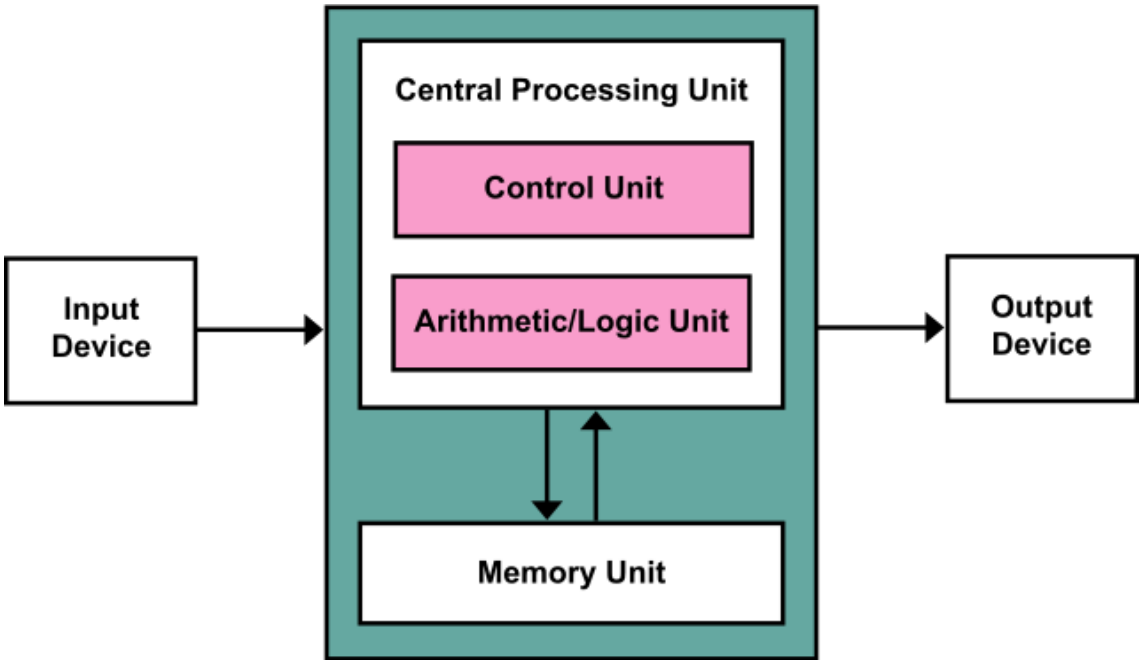
## 1. ARQUITECTURA DE VON NEUMANN

Los primeros intentos de resolver problemas de forma automática se llevaron a cabo a través de procesos mecánicos mediante el uso de elementos hardware y de información soportada en tarjetas perforadas.

Posteriormente, John Von Neumann desarrolló la idea de programa interno, ubicando las instrucciones y los datos en una memoria interna. De esta forma, las instrucciones podían modificarse y con ello modificar el tratamiento a realizar sobre los datos. La resolución de problemas de forma automática se convirtió en un proceso software.




- Para profundizar más en la arquitectura de Von Neumann, puedes visitar el siguiente link:

https://www.genbeta.com/desarrollo/como-funciona-la-computacion-actual-funcionaiento-de-la-arquitectura-de-von-neumann

- Resolver el siguiente quiz: https://quizizz.com/admin/quiz/56429f89a201fec602a01a9b/von-neumann-architecture
- experimentar con el siguiente simulador:

http://vnsimulator.altervista.org/

No obstante, en la máquina de Von Neumann y todas sus sucesoras, cualquier dato o instrucción había de estar en lenguaje máquina (ceros y unos), el cual era muy difícil de comprender para el ser humano. Para solucionar este problema de comunicación entre el ser humano y la máquina surgieron los denominados lenguajes de programación.

La principal razón para que las personas aprendan lenguajes y técnicas de programación es utilizar el ordenador para resolver problemas. Esta resolución exige el siguiente proceso:

- Definición y análisis del problema.
- Diseño del algoritmo.
- Transformación del algoritmo en un programa.
- Ejecución y validación del programa.

## 2. CONCEPTO DE ALGORITMO

La palabra algoritmo proviene del nombre de un matemático árabe llamado Mohammed al-Khowarizmi, cuyo apellido, al ser traducido al latín, derivó en la palabra algoritmo.

Un algoritmo se puede definir como un conjunto ordenado y finito de pasos u operaciones a realizar para resolver un problema, teniendo en cuenta que la salida o resultado del algoritmo dependerá de la entrada proporcionada.

Además, un algoritmo debe ser:

- **Preciso**: debe indicar el orden de realización de cada paso.
- **Determinista**: debe obtener el mismo resultado, siempre que se siga el algoritmo habiéndole proporcionado la misma entrada.
- **Finito**: debe terminar en algún momento.

Dado que los algoritmos permiten resolver los problemas de forma mecánica, está claro que resulta muy interesante compartirlos con otras personas, surgiendo así el problema de describir los algoritmos de forma fácilmente comprensible. Para dicha descripción se utilizan las siguientes técnicas.

## 2.1 Lenguaje natural

La forma más sencilla de describir un algoritmo es empleando el lenguaje natural. Por ejemplo, el algoritmo para describir un juego de dados que termina cuando la suma de los dos dados que se lanzan es seis, se describe de la siguiente manera:

1. Lanzar los dados.
2. Verificar si la suma de los dos dados es igual a seis.
3. Si es falso nuevamente ir al paso 1.
4. Si verdadero, concluye el juego.

La principal ventaja de este método es que cualquier persona que lea dicho algoritmo podría entenderlo.

Sin embargo, son varios los inconvenientes que plantea describir un algoritmo de esta forma:

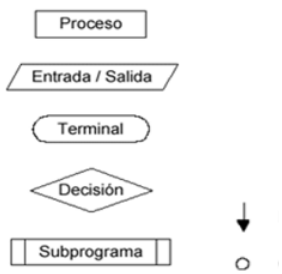
- **El lenguaje natural no es universal**. El algoritmo anterior es completamente inútil para cualquier persona que no entendiese el castellano.
- Las **descripciones** en lenguaje natural tienden a ser **extensas y ambiguas**.

Por estos motivos, se buscan otras formas de descripción de algoritmos que empleen un lenguaje universal y no ambiguo.

## 2.2 Ordinogramas (diagramas de flujo)

Representan gráficamente la secuencia lógica de las operaciones involucradas en la resolución de un problema.

Los símbolos gráficos más utilizados son:



Deben seguir una serie de normas:

- Los símbolos de inicio y fin deberán aparecer 1 única vez.
- El comienzo del programa figurará en la parte superior.
- El flujo de las operaciones será, a ser posible, de arriba abajo y de izquierda a derecha.
- Todos los símbolos empleados deben estar conectados por medio de líneas de flujo.
- Están prohibidos los cruces de líneas de flujo.

Son usados para representar algoritmos pequeños, ya que abarcan mucho espacio y su construcción es laboriosa.

## 2.3 Pseudocódigo

Técnica para expresar un algoritmo en un lenguaje intermedio entre el lenguaje natural y el lenguaje de programación seleccionado para su implementación. **Permite que el programador se centre en los aspectos lógicos del algoritmo, evitando las reglas de sintaxis de los lenguajes de programación**, pero manteniendo en buena parte su precisión.

Ejemplos de estructuras básicas:

SECUENCIAL

```
instrucción 1
instrucción 2
. . .
instrucción n
```

ALTERNATIVA
Exclusiva doble

```
SI condición ENTONCES
   instrucción 1
   . . .
   instrucción n
SINO
   instrucción 1
   . . .
   instrucción n
FIN-SI
```

ITERATIVA
Mientras

```
MIENTRAS condición HACER
   instrucción 1
   . . .
   instrucción n
FIN-MIENTRAS
```

Con nuestros algoritmos buscaremos siempre **dos objetivos** a conseguir con respecto al programa resultante:

- Que sea fácil de entender, codificar, depurar y mantener.
- Que sea eficiente en la utilización de recursos de la máquina (memoria y CPU).

## 3. TIPOS DE SOFTWARE

Es de sobra conocido que el ordenador se compone de dos partes bien diferenciadas: hardware y software.

El software es el conjunto de programas informáticos que actúan sobre el hardware para ejecutar lo que el usuario desee.

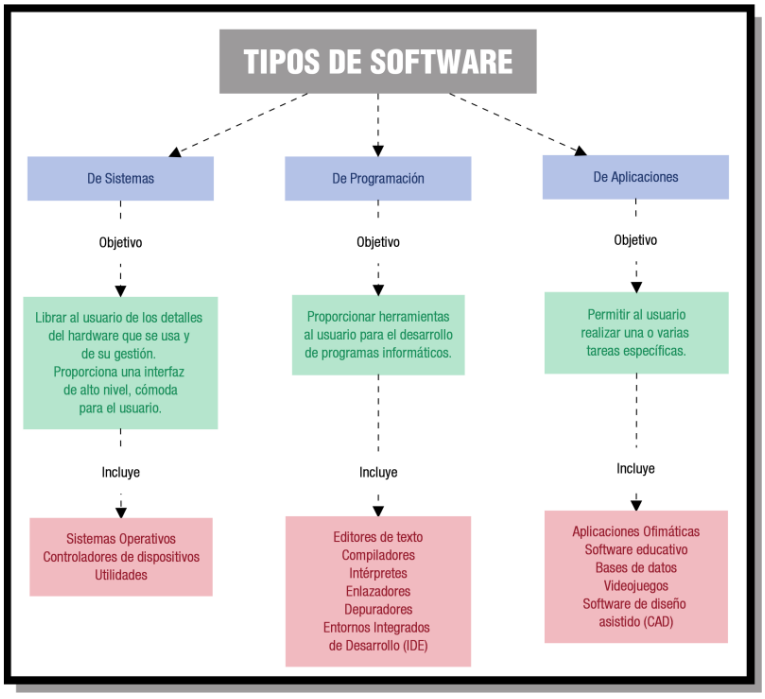
Según su función se distinguen **tres tipos de software**: sistema operativo, software de programación y aplicaciones.

El **sistema operativo** es el software base que ha de estar instalado y configurado en nuestro ordenador para que las aplicaciones puedan ejecutarse y funcionar. Son ejemplos de sistemas operativos: Windows, Linux, Mac OS X ...

El **software de programación** es el conjunto de herramientas que nos permiten desarrollar programas informáticos, y las **aplicaciones informáticas** son un conjunto de programas que tienen una finalidad más o menos concreta. Son ejemplos de aplicaciones: un procesador de textos, una hoja de cálculo, el software para reproducir música, un videojuego, etc.

A su vez, un programa es un conjunto de instrucciones escritas en un lenguaje de programación.

En definitiva, distinguimos los siguientes tipos de software:



En este tema, nuestro interés se centra en las aplicaciones informáticas: cómo se desarrollan y cuáles son las fases por las que necesariamente han de pasar.

A lo largo de esta primera unidad vas a aprender los conceptos fundamentales de software, las fases de desarrollo de una aplicación informática y los roles involucrados durante este proceso.

También aprenderás a distinguir los diferentes lenguajes de programación, los procesos que ocurren hasta que el programa funciona y realiza la acción deseada.

Por últomo comprenderás el proceso de traducción de un programa desde el lenguaje utilizado para escribirlo hasta la transformación del mismo en código binario (1s y 0s) que comprenden los ordenadores.

- Para profundizar más en los tipos de software existentes, puedes visitar la web:
  http://www.tiposdesoftware.com/
- Pregunta. Existen diversos S.O. en el mercado: Linux, Windows, Mac OS X etc. El más conocido es Windows. A pesar de eso, ¿por qué utilizamos cada vez más Linux?