MINISTERIO DE EDUCACIÓN
CENTRO EDUCATIVO BILINGÜE FEDERICO ZÚÑIGA FELIÚ
ASIGNATURA: PROGRAMACIÓN

11°

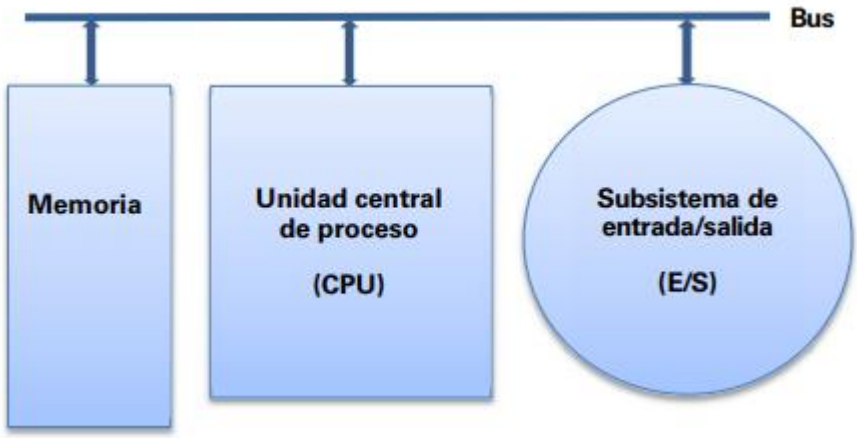
PROFESOR: CÉSAR A. DELGADO B.

@elprofecesard
@elprofecesard
@elprofecesard
@elprofecesard
@elprofecesard
Sitio Web elprofecesard.com

MÓDULO 1, U1 – ASPECTOS GENERALES DE LA PROGRAMACIÓN

Organización de un computador

Un computador es un dispositivo electrónico que permite **transmitir, almacenar y manipular información** mediante la ejecución de **programas.** La mayoría de los computadores siguen el modelo basado en la estructura de **John Von Neumann (1903-1957),** quien fue un prestigioso matemático estadounidense de origen húngaro que colaboró en el **proyecto ENIAC** para la construcción de la primera computadora de propósito general. **Von Neumann** realizó importantes aportaciones en el área de la organización y la estructura de los computadores que consta de tres elementos básicos: ***la unidad central de proceso, la memoria y el subsistema de entrada/salida.*** Estos tres elementos básicos están interconectados mediante un bus:



John Von Neumann

Un Bus es una serie de cables que funcionan cargando datos en la memoria para transportarlos a la **Unidad Central de Procesamiento o CPU.**

Términos básicos en programación

El estudiante de esta asignatura ha de tener muy claro que el computador no resuelve problemas por sí solo, sino que únicamente es capaz de realizar, con una gran potencia de cálculo, las operaciones necesarias para obtener la solución. ***Es el programador, mediante los programas, quien indica al computador los cálculos que ha de realizar para obtener la solución del problema.*** Por tanto, el programador debe realizar una serie de pasos ante el planteamiento y la solución de un problema:

Programación es aquella ciencia o arte que nos permite en un momento dado programar una computadora con el fin de resolver o encontrar la solución a un problema planteado.

Planteamiento de un Problema

1. **Buscar o pensar un método para resolverlo,** es decir, determinar las operaciones necesarias y el orden en que estas han de ser ejecutadas ***(algoritmo)***
2. **Escribir el algoritmo** en un lenguaje de programación determinado ***(programa)***
3. **Probar, depurar y finalmente ejecutar el programa** en el computador para obtener la solución al problema **(proceso de codificación y prueba)**

A continuación, definimos algunos términos básicos relacionados con la programación, explicamos las etapas en la elaboración de un programa y, finalmente, mostramos el proceso de codificación y prueba de este:

Algoritmo

Es un procedimiento de cálculo que ejecuta, de forma ordenada, una serie de instrucciones y conduce a la solución de un problema en un tiempo finito. Las características de un algoritmo son:

- **Es preciso:** indica exactamente el orden en que deben realizarse las operaciones.
- **Ha de estar bien definido (sin ambigüedad):** Si se sigue el algoritmo dos veces con los mismos datos de entrada, se obtiene el mismo resultado.
- **Es finito.** El algoritmo tiene que terminar en algún momento.

Para diseñar un algoritmo se debe comenzar por identificar las tareas más importantes para resolver el problema y disponerlas en el orden en el que han de ser ejecutadas. En un algoritmo se deben de considerar tres partes:

- **Entrada:** Información dada al algoritmo.
- **Proceso:** Operaciones o cálculos necesarios para encontrar la solución del problema.
- **Salida:** Respuestas dadas por el algoritmo o resultados finales de los procesos realizados.

Algoritmo es un procedimiento a seguir para resolver un problema en términos de:

1. Las acciones por ejecutar.

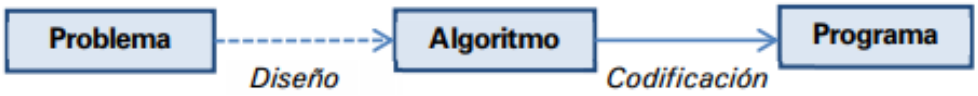
2. El orden en que dichas acciones deben ejecutarse.

Un **algoritmo** nace en respuesta a la aparición de un determinado problema y está compuesto por una serie finita de pasos que convergen en su solución.

En cada problema el **algoritmo** se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo.

Programa

Es la codificación de un algoritmo en un lenguaje de programación determinado. Así, dado el enunciado de un problema, primero hay que pensar el algoritmo que lo resuelve ***(fase de diseño)*** y, a continuación, codificar el algoritmo en un lenguaje de programación determinado ***(fase de codificación).***



La fase de diseño es un proceso creativo en que cuentan tanto la inteligencia como la intuición y la experiencia del programador. No existe un método fijo para encontrar el algoritmo que resuelve un problema determinado.

Por el contrario, ***la fase de codificación*** es mecánica y automática. Simplemente, es necesario conocer las reglas sintácticas del lenguaje de programación que se esté usando y codificar el algoritmo en ese lenguaje.

La **sintaxis** de un lenguaje de programación especifica las **reglas** para crear un programa apropiado en ese lenguaje.

En la actualidad encontramos una gran cantidad de programas y/o aplicaciones desarrolladas para mejorar la calidad de vida de los seres humanos.

Criterios para determinar un buen programa:

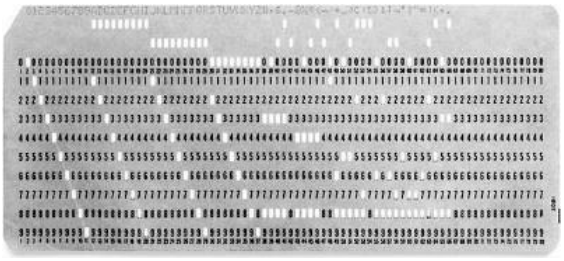
Corrección: Han de funcionar correctamente y resolver el problema planteado.

Claridad: Han de ser fáciles de leer, entender y modificar.

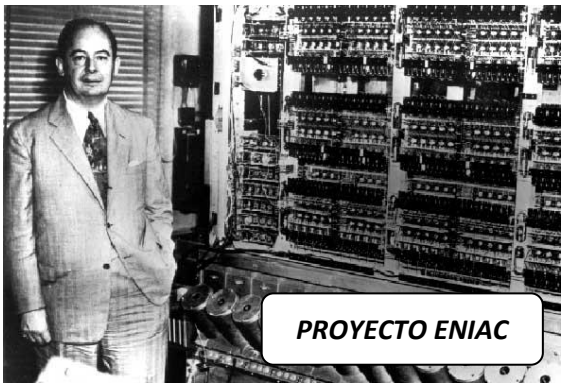
La eficiencia: Han de llegar a la solución de un problema en el menor tiempo posible.

Historia de los Lenguajes de Programación

Los primeros lenguajes de programación surgieron en el siglo XIX de la idea de ***Charles Babagge,*** un profesor matemático de la universidad de Cambridge e inventor inglés, que predijo muchas de las teorías en que se basan los actuales ordenadores. Consistía en lo que él denominaba la maquina analítica, pero que por motivos técnicos no pudo construirse hasta mediados del siglo XX. Con él colaboró **Ada Lovelace**, la cual es considerada como la primera programadora de la historia, pues realizo programas para aquélla supuesta máquina de **Babagge**, en tarjetas perforadas. Como la maquina no llego nunca a construirse, los programas de **Ada**, lógicamente, tampoco llegaron a ejecutarse, pero si suponen un punto de partida de la programación, sobre todo si observamos que en cuanto se empezó a programar, los programadores utilizaron las técnicas diseñadas por **Charles Babagge**, y **Ada**, que consistían entre otras, en la programación mediante ***tarjetas perforadas.***

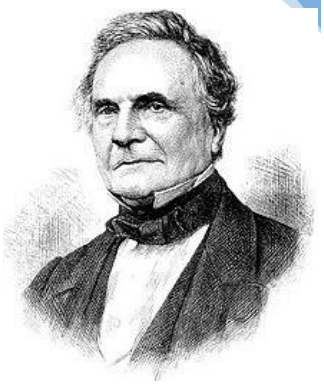


Charles Babbage, conocido como el **"padre de la informática"** no pudo completar en aquella época la construcción del computador que había soñado, dado que faltaba algo fundamental: **la electrónica.**



PROYECTO ENIAC

El camino señalado de **Babbage**, no fue nunca abandonado y siguiéndolo, se construyeron las primeras computadoras. Al desarrollarse las primeras computadoras electrónicas, se vio la necesidad de **programarlas**, es decir, de almacenar en memoria la información sobre la tarea que iban a ejecutar. Las primeras computadoras se usaban como calculadoras simples; se les indicaban los pasos de cálculo, uno por uno.



Charles Babagge
Padre de la Informática

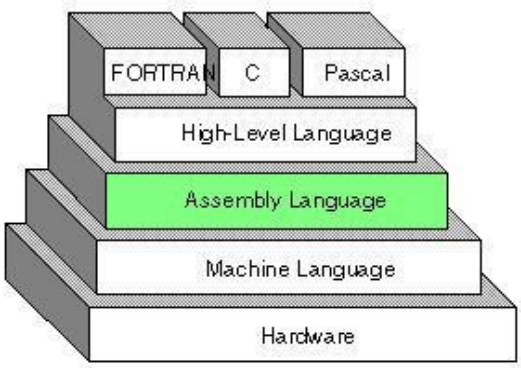
Ada Lovelace
Primera Programadora

La **tarjeta perforada** es una lámina hecha de cartulina que contiene información en forma de perforaciones según un **código binario.** Fueron los primeros medios utilizados para ingresar información e instrucciones a una computadora en los años 1960 y 1970.

Han sido reemplazadas por medios magnéticos y ópticos de ingreso de información. Sin embargo, muchos dispositivos actuales, como por ejemplo el CD-ROM se basan en un método similar al usado por las tarjetas perforadas,

John Von Neumann desarrolló el modelo que lleva su nombre, para describir el concepto de **"programa almacenado".** En este modelo, se tiene una abstracción de la memoria como un conjunto de celdas, que almacenan simplemente números. Estos números pueden representar dos cosas: ***los datos sobre los que va a trabajar el programa; o bien, el programa en sí.***

La programación en esos momentos era sumamente tediosa, pues el programador tenía que **"bajarse"** al nivel de la máquina y decirle, paso a paso, cada punto de la tarea que tenía que realizar. Además, debía expresarlo en forma numérica; y por supuesto, este proceso era propenso a errores, con lo que la productividad del programador era muy limitada. Sin embargo, hay que recordar que, en ese momento, simplemente aún no existía alternativa.

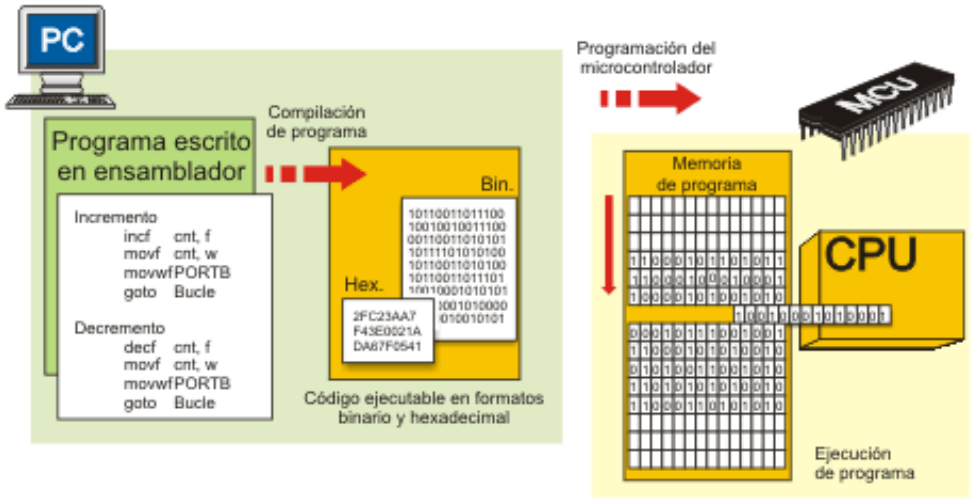


El primer gran avance, fue la abstracción dada por el **Lenguaje Ensamblador,** y con él, el nacimiento de las primeras herramientas automáticas para generar el **código máquina**. Esto redujo los errores triviales, como podía ser el número que correspondía a una operación, que son sumamente engorrosos y difíciles de detectar, pero fáciles de cometer. Sin embargo, aun así, es fácil para el programador perderse y cometer errores de lógica, pues debe bajar al nivel de la forma en que trabaja el CPU, y entender bien todo lo que sucede dentro de él. Con el desarrollo en los años 50s y 60s de algoritmos de más elevado nivel, y el aumento de poder del hardware, empezaron a entrar al uso de computadoras científicos de otras ramas; ellos conocían mucho de Física, Química y otras ramas similares, pero no de Computación, y por supuesto, les era sumamente complicado trabajar con lenguaje Ensamblador en vez de fórmulas.

Así nació el concepto de **Lenguaje de Alto Nivel**, con el primer compilador de **FORTRAN** (**FOR**mula **TRAN**slation), que, como su nombre indica, inició como un **"simple"** esfuerzo de **traducir** un lenguaje de fórmulas, ***al lenguaje ensamblador y por consiguiente al lenguaje de máquina.***

El **lenguaje ensamblador** es el lenguaje de programación utilizado para escribir programas informáticos de bajo nivel, y constituye la representación más directa del **código máquina** específico para cada arquitectura de computadoras legible por un programador.

El **lenguaje máquina o conocido** como **código máquina** es el único que entiende directamente la computadora, utiliza el **alfabeto binario** que consta de los dos únicos símbolos 0 y 1, denominados bits (abreviatura inglesa de dígitos binarios). Fue el primer lenguaje utilizado en la Programación de computadoras, pero dejó de utilizarse por su dificultad y complicación, siendo sustituido por otros lenguajes más fáciles de aprender y utilizar, que además reducen la posibilidad de cometer errores.



A partir de **FORTRAN**, se han desarrollado innumerables lenguajes, que siguen el mismo concepto: ***buscar la mayor abstracción posible, y facilitar la vida al programador, aumentando la productividad, encargándose los compiladores o intérpretes de traducir el lenguaje de alto nivel, al lenguaje de computadora.*** Hay que notar la existencia de lenguajes que combinan características de los de alto nivel y los de bajo nivel (es decir, **Ensamblador**). El ejemplo más apropiado podría ser el **lenguaje C** ya que puede acceder a los registros del sistema, trabajar con direcciones de memoria, todas ellas características de lenguajes de bajo nivel y a la vez realizar operaciones de alto nivel.

Concepto de Lenguaje de Programación

Un **lenguaje de programación** es un conjunto de símbolos y caracteres que se combinan entre sí siguiendo una serie de **reglas sintácticas** predefinidas por el lenguaje. Es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama **programación.**

El lenguaje C o **"Lenguaje de programación de sistemas"** fue desarrollado en el año 1972 por Dennis Ritchie para UNIX como un sistema operativo multiplataforma, ya que a pesar de que es un lenguaje de alto nivel (estructurado con sentencias y funciones que simplifican su funcionamiento) tenemos la posibilidad de programar a bajo nivel (como en el Assembler tocando los registros y memoria.

Lenguaje de programación es cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por una computadora.

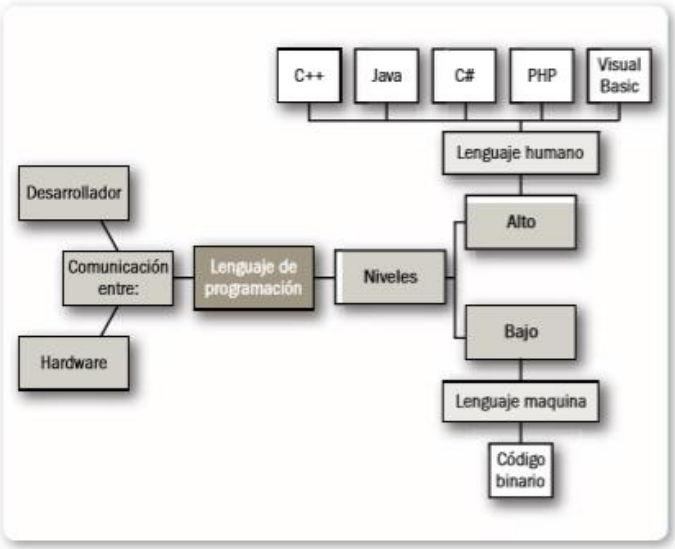
También la palabra **programación** se define como ***el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos***, a través de los siguientes pasos:

- El desarrollo lógico del programa para resolver un problema en particular.
- Escritura de la lógica del programa empleando un lenguaje de programación específico (codificación del programa)
- Ensamblaje o compilación del programa hasta convertirlo en lenguaje de máquina.
- Prueba y depuración del programa.
- Desarrollo de la documentación.

La creación de un **programa de computadora** requiere de procedimientos ordenados para solucionar un problema a través del computador.

Clasificación de los Lenguajes de Programación

Existen diversas clasificaciones de los lenguajes de programación: según su ***evolución histórica, el propósito, el paradigma de la programación, el nivel de abstracción,*** etc.



Un **paradigma** de **programación** es un modelo básico de diseño y desarrollo de programas producidos con normas específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad.

Abstracción es algo que está en el universo de las ideas, los pensamientos, pero que no se puede concretar en algo material, que se pueda tocar.

Nosotros nos centramos en la clasificación según el **nivel de abstracción.**

Lenguajes de alto nivel

Los lenguajes de **programación de alto nivel** son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los **lenguajes humanos** – en general **lenguaje inglés** – lo que facilita la escritura y comprensión por parte del programador. Son los lenguajes de programación que, por sus características, ***se parecen más al lenguaje humano***. La característica principal de estos lenguajes es **que son portables o independientes de la arquitectura.** Esto quiere decir que un programa escrito

en un lenguaje de **alto nivel** se puede ejecutar en distintos computadores sin necesidad de modificarlo.

Una propiedad de los lenguajes de **alto nivel** es que **son independientes de la máquina**, esto es, las sentencias del programa no dependen del diseño de hardware de una computadora específica.

Los programas escritos en lenguajes de **alto nivel**, al igual que los escritos en lenguajes de **bajo nivel**, no son entendibles directamente por la máquina, sino que necesitan ser traducidos a instrucciones en lenguaje máquina que entiendan las computadoras. **Los programas** que realizan esta traducción se llaman **Compiladores**, y **los programas** escritos en un lenguaje de **alto nivel** se llaman **Programas Fuente**. El compilador traduce el Programa Fuente en un programa llamado Programa Objeto. El proceso de traducción de un programa fuente a un programa objeto se denomina Compilación.

Los lenguajes de alto nivel, a su vez, se pueden clasificar en función del estilo de programación empleado (paradigma de programación), y así nos encontramos, entre otros, con los tipos siguientes:

Lenguajes de programación procedurales

La programación consiste en dividir el problema inicial en partes más pequeñas. Los subproblemas son resueltos por subprogramas (subrutinas, funciones, procedimientos), que se llaman entre sí para llegar a la solución completa del problema. Los lenguajes C y Pascal, por ejemplo, son lenguajes de programación procedurales.

Lenguajes de programación orientados a objetos

La programación consiste en crear clases y objetos, y especificar la interacción entre ellos. Los lenguajes C++ y Java son lenguajes de programación orientados a objetos.

Lenguajes de bajo nivel

Son lenguajes específicos de la arquitectura, es decir, los programas escritos en lenguajes de bajo nivel no son portables, por lo que solo se pueden ejecutar en el computador para el cual fueron diseñados. Se incluyen en esta categoría el lenguaje máquina y el lenguaje ensamblador.

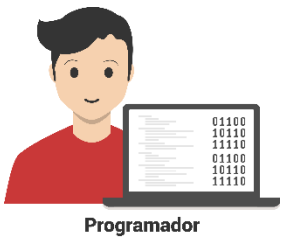
Ejemplo de Lenguajes de Alto Nivel

Estructurados
Basic, C, Pascal

Orientados a Objetos
C#, Visual Basic.NET, C++, Java

Declarativos
Lisp, Prolog

Funcionales
AML, CAML



El **lenguaje máquina** se caracteriza por ser el único que el computador puede interpretar directamente y se basa en la combinación de solo dos símbolos (el 0 y el 1). Los **lenguajes de máquina** son aquellos cuyas instrucciones son directamente entendibles por la computadora, y no necesitan traducción posterior para que el **CPU** pueda comprender y ejecutar el programa. La programación en lenguaje de máquina es difícil, porque implica escribir directamente en un sistema binario (ceros y unos), por eso se necesitan lenguajes que permitan simplificar ese proceso.

Los lenguajes de bajo nivel fueron diseñados con ese fin. Éstos son dependientes de la arquitectura física de la computadora y de un conjunto específico de instrucciones para el **CPU**, y los programas escritos en ellos deben ser traducidos a lenguaje máquina para ser ejecutados. Un lenguaje típico de bajo nivel es el lenguaje **ensamblador (Assembler)**

Lenguaje C

```
main()
{
  int area,lado=9;
  area=lado*lado;
}
```

Lenguaje ensamblador

```
.bss
.comm area, 4, 4
.data
lado: .long 9
.text
.global main
main:
    movl lado, %eax
    imull %eax, %eax
    movl %eax, area
    movl $0, %ebx
    movl $1, %eax
    int $0x80
```

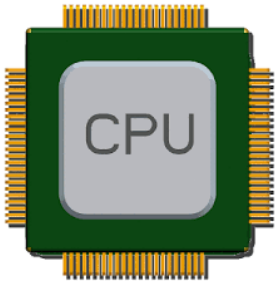
Lenguaje máquina

```
10100001 00100000
10010101 00000100
00001000 00001111
10101111 11000000
10100011 00101000
10010101 00000100
00001000 10111011
00000000 00000000
00000000 00000000
10111000 00000001
00000000 00000000
00000000 11001101
10000000
```

En la figura anterior, se muestra un programa que calcula el área de un cuadrado de 9 cm de lado, escrito en tres lenguajes de programación diferentes.

Obsérvese cómo los programas escritos en lenguajes de **bajo nivel (lenguaje ensamblador y lenguaje máquina)** son más difíciles de entender que los escritos utilizando **lenguajes de alto nivel (lenguaje C).**

CPU se refiere a la Unidad Central de Procesamiento.



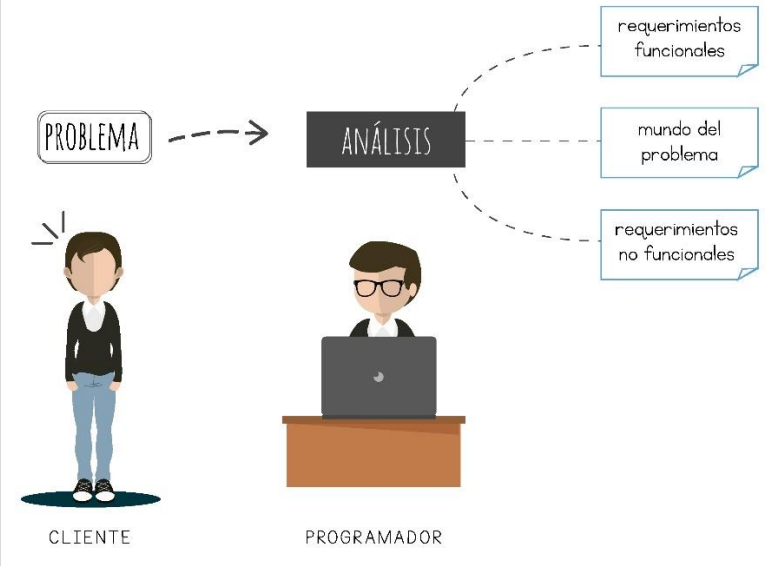
El **sistema binario** es un sistema de numeración en el que los números se representan utilizando solamente dos cifras: cero y uno (0 y 1).

Las personas que desarrollan programas se les conoce como **programadores.**

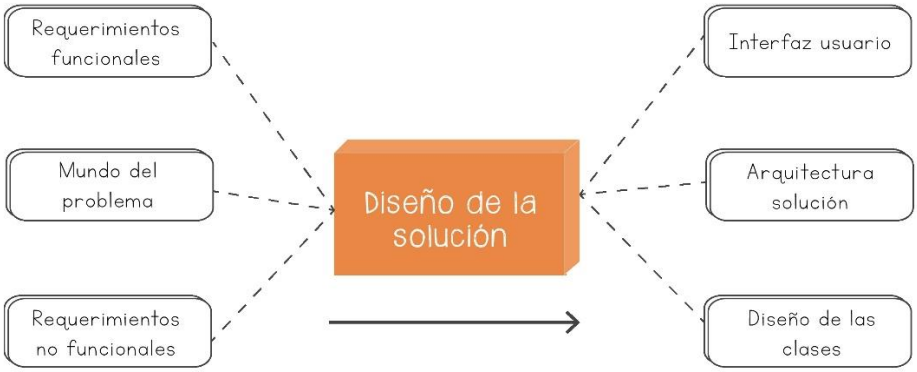
Etapas en la elaboración de un programa

La metodología que vamos a seguir es la siguiente:

PASO 1 - Análisis: Se define bien el problema, dejando muy claro cuál es la entrada y cuál ha de ser la salida del programa; también se especifican, a muy alto nivel, las estructuras de datos que se utilizarán y las líneas generales para resolver el problema.



PASO 2 - Diseño: Se desarrollan los algoritmos para resolver el problema planteado mediante la resolución de problemas más sencillos. El problema inicial se descompone en subproblemas autocontenidos y de dificultad menor. Estos subproblemas, a su vez, también se dividen en subproblemas todavía más simples, hasta llegar a subproblemas fáciles de codificar.



Una **Metodología** se define como el grupo de mecanismos o procedimientos racionales, empleados para el logro de un objetivo, o serie de objetivos que dirige una investigación científica. La metodología para elaborar de programas se fundamenta en una serie de pasos hasta solucionar un problema.

Un **programador** es la persona que escribe, depura y mantiene el código fuente de un programa informático, es decir, el conjunto de instrucciones que ejecuta el hardware de una **computadora**, para realizar una tarea determinada.

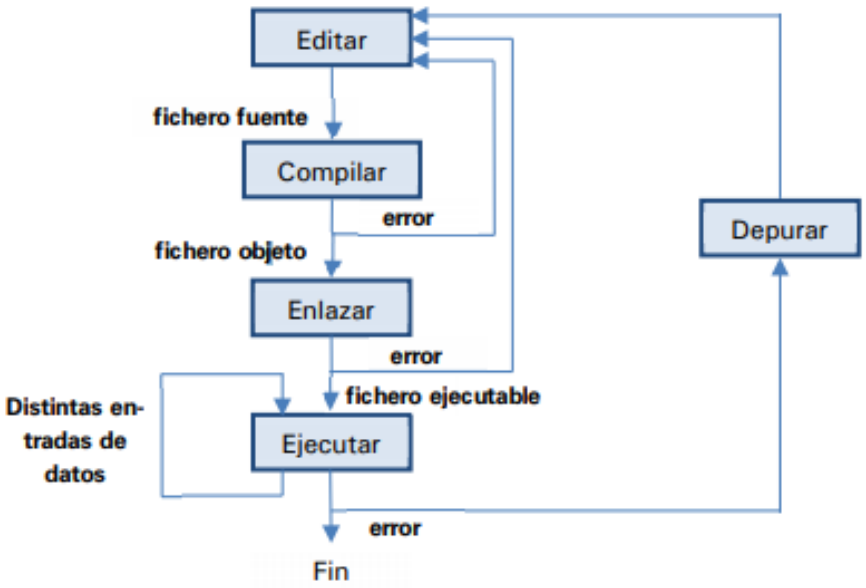
PASO 3 - Codificación: Consiste en escribir, en el lenguaje de programación elegido, cada uno de los algoritmos diseñados en la fase anterior. Por regla general, cada uno de los algoritmos ha de ser codificado y probado independientemente.

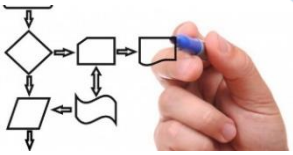
PASO 4 - Prueba: Consiste en ejecutar los programas en el entorno definido. El programa ha de probarse con diferentes datos de entrada, sin olvidarse de los casos especiales o menos frecuentes.

PASO 5 - Mantenimiento: En la fase anterior solo se detecta la presencia de errores, pero nunca se puede confirmar la ausencia total de estos. Es por ello por lo que la última etapa en la elaboración de un programa siempre es el mantenimiento. En esta etapa, se corrigen los errores hallados posteriormente y se añaden nuevas funcionalidades al programa.

Proceso de codificación (3) y prueba de un programa (4)

Independientemente del entorno de programación que se escoja para desarrollar los programas, el proceso de codificación y prueba de cada uno de los algoritmos diseñados en la fase de diseño es el que muestra a continuación:





Un **proceso** es una secuencia de pasos dispuesta con algún tipo de lógica que se enfoca en lograr algún resultado específico.

Los **procesos** son mecanismos de comportamiento que diseñan los hombres para mejorar la productividad de algo, para establecer un orden o eliminar algún tipo de problema.

Un **programador** es la persona que escribe, depura y mantiene el código fuente de un programa informático, es decir, el conjunto de instrucciones que ejecuta el hardware de una **computadora**, para realizar una tarea determinada

Edición: Se codifica el algoritmo siguiendo las reglas sintácticas definidas por el lenguaje de programación. Podemos utilizar cualquier editor de texto y el resultado será un fichero de texto con el código del programa (fichero fuente). La extensión de este fichero indica el lenguaje de programación utilizado (.c para códigos en C, .f para Fortran, .cpp para C++, etc.)

Compilación: Herramienta que detecta los errores sintácticos de los programas, e indica el tipo de error y la línea del código donde se encuentra. Así, una vez escrito el código en el fichero fuente, hay que **"compilar"** el programa, es decir, determinar si hay errores sintácticos en el código y corregirlos (ya no hay errores sintácticos), generando un fichero con un código objeto. Este fichero tiene la extensión .o o .obj y es necesario para poder generar posteriormente un fichero ejecutable.

Enlace: Cuando un programa consta de varios ficheros fuente o utiliza funciones propias del lenguaje de programación, el enlazador verifica que solamente haya un programa principal, que todas las funciones estén definidas, que las llamadas a las funciones correspondan con sus definiciones, etc. Si todo es correcto, el enlazador genera un fichero ejecutable que normalmente tiene la extensión .out, .exe o no tiene extensión.

Ejecución: Se ejecuta el programa para determinar la presencia de errores semánticos o de ejecución que determinan si el programa funciona correctamente. En esta etapa, se ha de ejecutar el programa varias veces con diferentes entradas de datos y probar todos los posibles casos del programa. Es muy importante probar exhaustivamente los programas con diferentes datos de entrada, sin olvidar las entradas especiales o menos frecuentes.

Depuración: En la etapa anterior se puede detectar si el programa es incorrecto, pero no tenemos pistas sobre qué parte del código ha causado el problema. El depurador permite observar el comportamiento de un programa paso a paso para detectar errores de ejecución. Cuando se detecta un error de ejecución, hay que ir a la etapa de edición para corregir el programa y nuevamente volver a compilar, enlazar y ejecutar. Al depurar, si no hay errores de ejecución (semánticos) y el programa funciona correctamente, se puede dar por finalizado el proceso de codificación y prueba.

La sintaxis o reglas sintácticas en programación son las reglas que indican cómo realizar las construcciones del **lenguaje de programación** para construir programas válidos. La sintaxis ayuda a la especificación y estructura de un **lenguaje de programación**.

```
#include <conio.h>
#include "stdio.h"
#define N 100
void main()
{
int num=1;
While (num<N)
 {
  printf("%d",num);
  num+=2;
  }
getch();
}
```

La sintaxis es similar a la ortografía de un **lenguaje** natural.

GUÍA DE TRABAJO 1

ESTUDIANTE: _____ **CÉDULA:** _____ **NÚMERO** _____ **GRADO 11º** _____

Actividad #1 - Presencial - Individual **Fecha de entrega es el viernes 9/4/2021.**

MAPA CONCEPTUAL: CONCEPTOS BÁSICOS DE PROGRAMACIÓN **Valor 50 puntos.**

- Trabaje individualmente para diseñar un mapa conceptual utilizando el software Cmaptools, PowerPoint o de manera escrita en cartulina u hojas blancas.
- Utilice el contenido del **MÓDULO 1, U1 – ASPECTOS GENERALES DE LA PROGRAMACIÓN** para diseñar el mapa conceptual.
- El mapa conceptual debe tener coherencia, unión de conceptos por jerarquía, utilizar proposiciones y conexiones entre conceptos claves. Junto con esta guía, se encuentra información sobre la creación de mapas conceptuales.
- Al final, haga captura del mapa conceptual como imagen y envíe en formato PDF. En el caso de hacer el mapa escrito, tome una foto con su celular y envíe en formato .jpg o .png. El mapa debe ser presentado en una sola imagen o canva.
- Entregue los archivos donde le indique el docente y esta guía de trabajo para poder ser evaluado.

- Criterios de Evaluación (50 Puntos)

5: Excelente 4: Satisfactorio 3: Requiere Mejoras 2: Cumple muy poco 1: Cumple mínimo 0: No cumple (no entregó)

Aspectos para evaluar	Descripción – Evaluación para Nota Diaria	5	4	3	2	1
1. Orden de los conceptos	Los conceptos presentan un orden jerárquico (según su importancia) y se encuentran ubicados correctamente en la representación visual del mapa.					
2. Uso de figuras geométricas	Todos los conceptos están encerrados en figuras geométricas.					
3. Palabras de enlace o conectores	En las conexiones selecciona palabras de enlace o conectores que dan coherencia a la relación de conceptos.					
4. Líneas o flechas	Las líneas y flechas orientan al lector, facilitando la comprensión del tema.					
5. Nivel de ejemplos	Incluye ejemplos en las terminaciones del último nivel, en los cuales corrobora que se entendió claramente la información conceptual.					
6. Lectura del mapa	Establece relación entre conceptos principales: lectura lógica, de izquierda a derecha y de arriba hacia abajo.					
7. Uso de software	Uso correcto del software, que favorece el impacto visual hacia los lectores.					
8. Información	Identifica el nombre del autor, fecha, referencias bibliográficas, actividad de aprendizaje					
9. Fuentes	El tamaño y la forma de las fuentes utilizadas permiten una fácil lectura.					
10. Normas ortográficas	Escribe con ortografía.					
	NOTA DIARIA: Total de Puntos (50)					

Aspectos para evaluar	Descripción – Evaluación para Nota de Apreciación	5.0	4.0–4.9	3.0–3.9	2.0–2.9	1.0–1.9
Interés en la clase	Se esfuerza y muestra interés en conocer sobre el tema.					
Responsabilidad	Hace entrega de un producto de calidad y en tiempo oportuno					
Competencias	Demuestra destrezas en la creación de mapas conceptuales..					

Observación: el plagio dará lugar a colocar la mínima calificación (1.0) en todos los aspectos evaluados (en notas diarias y de apreciación)

¿SABES COMO HACER UN MAPA CONCETUAL?

Los mapas conceptuales nos permiten organizar y comprender ideas de manera significativa y rápida, a través de una representación gráfica en donde se esquematizan conceptos que hacen parte de un tema central. Se fundamentan en la jerarquía de los conceptos (en óvalos o recuadros), conectados entre sí y unidos con palabras de enlace o conectores. **Un mapa conceptual debe estar conformado por:**

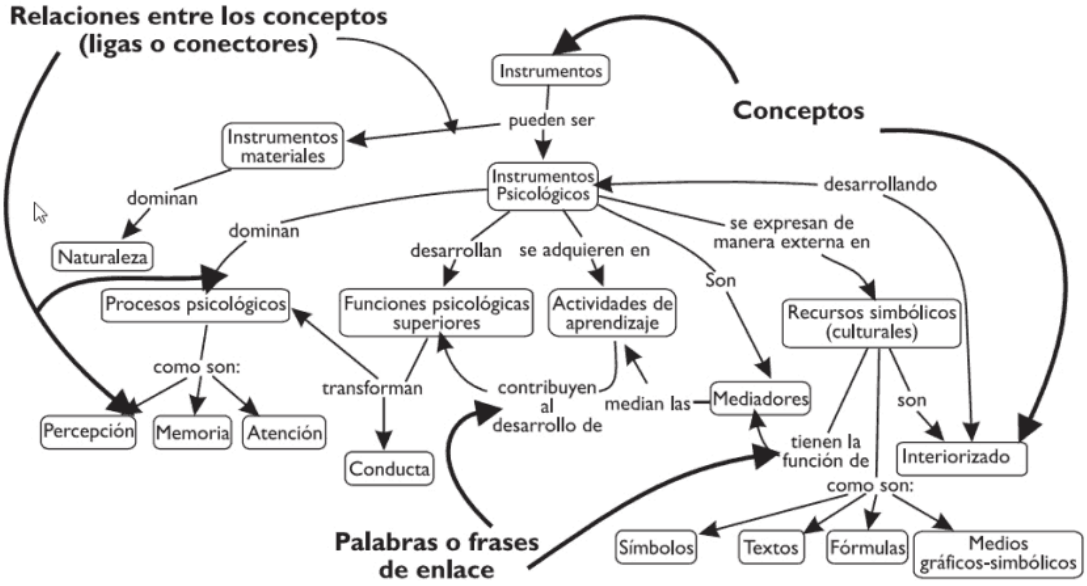
Cmaptools, software para crear mapas conceptuales

Los conceptos: se refieren a eventos, objetos, situaciones o hechos representados dentro de círculos o figuras geométricas que reciben el nombre de nodos. Los conceptos deben estar conectados con aquellos que tiene relación, más no con todos porque ello generaría una incomprensión. Además, en los conceptos no se deben colocar verbos ni oraciones. Ejemplos: agua, átomo, célula, silla, lluvia, mesa, Democracia, Estado, frío.

Las palabras de enlace: están conformadas por verbos y expresan la relación que existe entre dos o varios conceptos para que sean los más explícito posibles, estos se representan mediante líneas conectoras. Se hace necesario utilizar palabras conectoras o de enlace para dar mayor claridad y entendimiento. Ejemplo: “Es parte de”, “se clasifican en”, “es”, “depende de”, “para”, “contribuyen a”, “son”, “donde”, “como”, entre otras.

Las proposiciones: están compuestas por la unión de uno o varios conceptos o términos que se relacionan entre sí, a través de una palabra de enlace. Estas deben formar oraciones con sentido propio y no deben necesitar de otras proposiciones para tener coherencia. Ejemplo: "El ser humano necesita agua"

Líneas conectoras o de unión: se utilizan para unir los conceptos y para acompañar las palabras de enlace. Las líneas conectoras ayudan a dar mejor significado a los conceptos uniéndolos entre sí. A continuación, un ejemplo:



PROGRAMACIÓN DE COMPUTADORAS – UNDECIMO GRADO

GUÍA DE TRABAJO 2

ESTUDIANTE: **CÉDULA:** **NÚMERO** **GRADO 11°**

LECTURA-GLOSARIO DE TÉRMINOS: MÓDULO 1, U2 ASPECTOS GENERALES DE LA PROGRAMACIÓN **Valor 50 puntos**

Actividad #1 - Presencial – Individual **Fecha de entrega: viernes 9/4/2020.**

- Trabaje individualmente. Dé lectura al **MÓDULO 1, U2 ASPECTOS GENERALES DE LA PROGRAMACIÓN** y busque por lo menos 50 conceptos que considere importantes para el aprendizaje de la asignatura Programación.
- Presente el listado de conceptos encontrados tal como aparece en esta guía de trabajo (escrito a mano o a computadora). Entregue su archivo en PDF en la plataforma Teams y el enlace de Google que le suministrará el docente.

Criterios Evaluación	E = Excelente	B = Bueno	R = Regular	PM = Por Mejorar	NO = Mínimo observado

GLOSARIO DE TÉRMINOS: CRITERIOS A EVALUAR	E	B	R	PM	MO
Escala numérica (50 conceptos encontrados = 5.0)	**5.0**	**4.0-4.9**	**3.0-3.9**	**2.0-2.9**	**1.0-19**
Entrega preliminar de conceptos. **(ND)**					
Muestra interés en buscar conceptos para comprender el tema. (**NA**)					

1.	19.	37.
2.	20.	38.
3.	21.	39.
4.	22.	40.
5.	23.	41.
6.	24.	42.
7.	25.	43.
8.	26.	44.
9.	27.	45.
10.	28.	46.
11.	29.	47.
12.	30.	48.
13.	31.	49.
14.	32.	50.
15.	33.	51.
16.	34.	52.
17.	35.	53.
18.	36.	54.

PROGRAMACIÓN DE COMPUTADORAS – UNDECIMO GRADO

GUÍA DE TRABAJO 2

ESTUDIANTE: _____ **CÉDULA:** _____ **NÚMERO** _____ **GRADO 11°** _____

LECTURA-GLOSARIO DE TÉRMINOS: MÓDULO 1, U2 ASPECTOS GENERALES DE LA PROGRAMACIÓN **Valor 40 puntos**

Actividad #2 – Individual - Semi Presencial – Virtual. **Fecha de entrega: viernes 30/4/2020.**

- Trabaje individualmente para elaborar un glosario de términos (vocabulario)
- Revise los conceptos aprobados, verifique que no estén repetidos, ordene alfabéticamente e intégrelos en un solo documento.
- El documento debe ser consensuado y contar con un mínimo de 50 conceptos con sus respectivas definiciones.
- Presente el glosario escrito a mano y en hoja blanca o a computadora. El documento debe ser confeccionado por el estudiante.
- Evite escribir en cierre mayúscula. Cuide la ortografía. Sea creativo, ya que se evalúa su creatividad en la entrega.
- Entregue la actividad con la guía como portada en la fecha acordada. Recuerde colocar su número en la guía en orden ascendente.

Criterios Evaluación	E = Excelente	B = Bueno	R = Regular	PM = Por Mejorar	NO = Mínimo observado

GLOSARIO DE TÉRMINOS: CRITERIOS A EVALUAR (ND)	E	B	R	PM	MO
40 puntos / 10 valor máximo	10	7	5	3	1
Elabora correctamente el glosario (Conceptos y sus definiciones)					
Muestra orden y aseo en la asignación					
Muestra creatividad en la asignación					
Escribe con letra legible – Ortografía					
ND Total de Puntos (40)					
GLOSARIO DE TÉRMINOS: CRITERIOS A EVALUAR (NA)	**E**	**B**	**R**	**PM**	**MO**
Escala numérica (de 1.0 a 5.0)	**5.0**	**4.0-4.9**	**3.0-3.9**	**2.0-2.9**	**1.0-19**

Aspectos para evaluar	Descripción – Evaluación para Nota de Apreciación	5.0	4.0–4.9	3.0–3.9	2.0–2.9	1.0–1.9
Interés en la clase	Se esfuerza y muestra interés en conocer sobre el tema.					
Responsabilidad	Hace entrega de un producto de calidad y en tiempo oportuno					

Observación: el plagio dará lugar a colocar la mínima calificación (1.0) en todos los aspectos evaluados (en notas diarias y de apreciación)