Laboratorio de Algoritmos y Estructuras de Datos

En esta materia, conoceremos como hacer programas a partir de la experimentación, aprendiendo mediante el hacer. Por eso decimos que es un **LABORATORIO.**

Algoritmo

Secuencia de pasos para resolver problemas

> ORDEN
> PRECISO
> FINITO

Estructura de Datos

Orden y distribución de una cosa

>BÁSICA >Bit
>Byte
>Palabra

>SIMPLE >Numéricos enteros
>Numéricos reales
>Alfanuméricos
>Lógicos
>Punteros

>SIMPLE DERIVADA > Variables
> Constantes

>COMPLEJAS > Vectores, Array o arreglo unidimensionales
> Arreglos bidimensionales o matrices
> Árboles
> Grafos
> Bases de datos

Pasos para hacer un programa

1. Análisis del problema

En esta etapa, debemos identificar y definir que debe hacer el programa y cuál es resultado deseado.

Ejemplo: Realizar un programa para calcular la edad aproximada de una persona, leyendo el año de nacimiento.

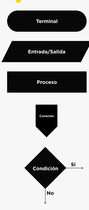
ENTRADA	PROCESO	SALIDA
anio_nac	edad<-anio_act- anio_nac	edad

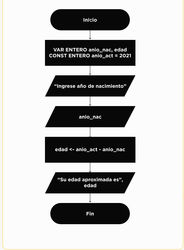
2. Diseño del Algoritmo

En el análisis del problema, definimos el objetivo del programa, que tiene que hacer. En esta etapa plantearemos como va a ralizar el programa la tarea deseada, planteando una secuencia de pasos (algoritmo) aritmético-lógica, utilizando las herramientas de programacion. Ellas son el **diagrama de flujo** y el **pseudocódigo**.

Diagrama de Flujo

Es una representación gráfica del algoritmo. Los símbolos utilizados han sido normalizados por el Instituto Norteamericano de Normalización (ANSI). Los más usados son los siguientes:

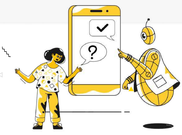




Paseudocódigo

Es una herramienta de programación en la que las instrucciones se escriben en palabras similares al inglés o español.
Se dice que es un *lenguaje de especificación de algoritmos*, ya que no existen reglas para la escritura del pseudocódigo, pero las sentencias que detallan como va a hacer el programa la tarea solicitada son similares a la traducción en español de las instrucciones en un lenguaje de programación.

```
INICIO
VAR ENTERO anio_nac, edad
CONT ENTERO anio_act = 2021
IMPRIMIR "Ingrese año de nacimiento"
LEER anio_nac
edad <- anio_act - anio_nac
IMPRIMIR "Su edad aproximada es", edad
FIN
```



3. Codificación

Traducción de un algoritmo a un **lenguaje de programación.**

Los lenguajes de programación nos permiten a los programadores (mediante reglas gramaticales) dar instrucciones a la computadora. Los podemos clasificar en tres tipos:

Alto Nivel

Se aproxima más al lenguaje natural humano que al lenguaje binario de las computadoras. Un lenguaje de alto nivel permite al programador escribir las instrucciones de un programa utilizando palabras o expresiones sintácticas muy similares al inglés. Por ejemplo, C, Java, Javascript, Phyton son lenguajes de alto nivel.

Ejemplos:

PRINTF imprimir
SCANF leer

Bajo Nivel

Assembler o ensamblador. Se basa en reglas mnemotécnicas. Necesita de un compilador para que la computadora los pueda entender.

Ejemplos:

ADD suma
DIV dividir
STO almacenar

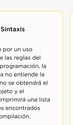
Lenguaje Máquina

Codigo binario (ceros y unos). Es directamente inteligible por la máquina.

Ejemplos:

20 10100
77 1001101

Nosotros vamos a programar en el lenguaje C. El algoritmo traducido a un lenguaje de programación se denomina **código fuente.**



```
#include <stdio.h>
#define anio_act 2021
int anio_nac, edad;
main(){
   printf ("Ingrese su año de nacimiento: ");
   scanf ("%d", &anio_nac );
   edad=anio_act-anio_nac;
   printf ("Su edad aproximada es %d", edad);
}
```

4. Compilación y Ejecución

En esta etapa, mediante un compilador, el programa fuente es traducido a lenguaje máquina. Si el programa no tiene errores, se obtendrá como resultado el código objeto, que se podrá ejecutar.
Si el código fuente presentara errores, estos se deberán identificar y depurar.

5. Depuración y Verificación

La depuración es el proceso de encontrar los errores del programa y corregir o eliminar dichos errores.
Los errores en programación pueden ser de tres tipos:

Errores de Sintaxis

Se producen por un uso incorrecto de las reglas del lenguaje de programación, la computadora no entiende la instrucción, no se obtendrá el programa objeto y el compilador imprimirá una lista de los errores encontrados durante la compilación.

Errores Lógicos

Se producen en la etapa de diseño del algoritmo. Son errores en el planteo de la lógica del programa. Se pueden identificar mediante la prueba de escritorio.

Errores en Tiempo de Ejecución

Se deben a instrucciones que la computadora entiende pero que no puede realizar, por ejemplo división por cero. Estos errores producen que el programa interrumpa su ejecución normal.

La verificación es el proceso de ejecución del programa en donde se le suministrarán una amplia variedad de datos de entrada, llamados datos de test o prueba, para verificar que devuelve el programa en función a las entradas dadas, y evaluar si hay que rectificar algo.

6. Documentación y Mantenimiento

La documentación de un programa consta de las descripciones de los pasos para dar en el proceso de resolución de un problema. La importancia de la documentacion debe ser destacada por su decisiva influencia en el producto final. Programas pobremente documentados son difíciles de leer, difíciles de depurar y casi imposibles de mantener y modificar.

La documentación de un programa puede ser interna y externa. La documentación interna es la contenida entre las línea de comentarios. La documentación externa incluye el análisis, diagramas de flujo y/o pseudocódigos, manuales de usuarios con instrucciones para ejecutar el programa y para interpretar los resultados.

La documentación es vital cuando se desea corregir posibles errores futuros o bien cambiar el programa. Tales cambios se denominan mantenimiento del programa. Después de cada cambio la documentación debe ser actualizada para facilitar cambios posteriores. Es práctica numerar las sucesivas veriones de los programas 1.0, 1.1, 2.0, 2.1.