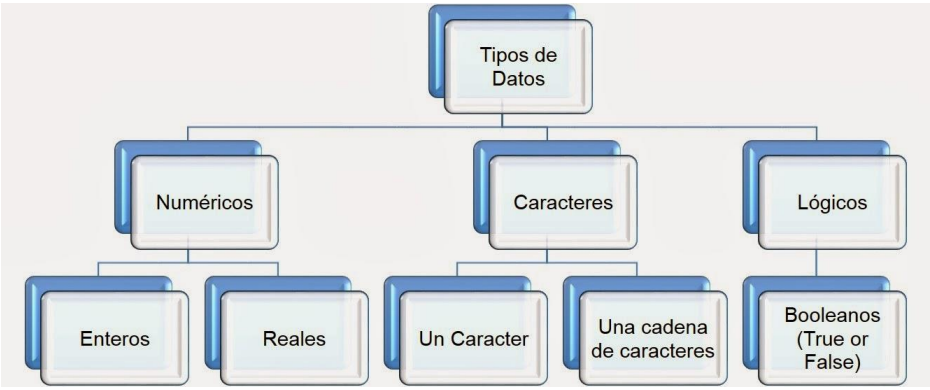
MODULO 2 – ELEMENTOS BÁSICOS DE UN LENGUAJE DE PROGRAMACIÓN

TIPOS DE DATOS

El primer objetivo de toda **computadora** es el **manejo de datos e información**. Estos **datos** pueden ser las cifras de ventas de un supermercado o las calificaciones obtenidas en una asignatura. La mayoría de las computadoras pueden trabajar con varios tipos (modos) de **datos**. Es por eso por lo que los **algoritmos** y **programas** correspondientes, operan sobre estos **tipos de datos**.

Un **dato** es la expresión general que describe los objetos con los cuales opera una **computadora.** Todos los **datos** tienen un tipo asociado con ellos. Un **dato** puede ser un simple carácter, tal como "b" o un valor entero tal como 35. El **tipo de dato** determina la naturaleza del conjunto de valores que puede tomar una **variable**.

Un **tipo de dato** es la propiedad de un valor que determina su dominio (qué valores puede tomar), qué operaciones se le pueden aplicar y cómo es representado internamente por el computador. Veamos los **tipos de datos**:



Dato Nulo: Existe un valor llamado **None** (en inglés, **«ninguno»)** que es utilizado para representar casos en que ningún valor es válido, o para indicar que una variable todavía no tiene un valor que tenga sentido. El valor **None** tiene su propio tipo, llamado **NoneType**, que es diferente al de todos los demás valores.

Datos Simples: ocupan sólo una casilla de memoria, por tanto, una variable simple hace referencia a un único valor a la vez. Hay tres tipos básicos de **datos simples**, los cuales son:

<table>
<tr><th>Datos Simples</th><th>Descripción</th></tr>
<tr><td>Numéricos</td><td>Permiten representar valores de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.
Entero (int-integer-Entero): es un subconjunto finito de los números enteros (negativos, cero, positivos)
Ejemplo: 5, 0, -10, 1234, -9876

Real (punto flotante - Float): es un subconjunto de los números reales (tienen punto decimal y pueden ser positivos o negativos)
Ejemplo: 0.05; 3.1416; -12.0; 3456.0; -2.71828.
Otra forma de representar los números reales es la notación exponencial o científica y se utiliza para números muy grandes o pequeños.
Ejemplo: $0.00045 = 0.45 \times 10^{-3} = 0.45E-3$
Los tipos de datos reales a su vez pueden clasificarse en:
<table>
<tr><th>Tipo</th><th>Tamaño (bits)</th><th>Intervalo aproximado</th></tr>
<tr><td>Simple</td><td>32</td><td>-3.4×10^{38} a $+3.4 \times 10^{38}$</td></tr>
<tr><td>Doble</td><td>64</td><td>$\pm 5,0 \times 10^{-324}$ a $\pm 1,7 \times 10^{308}$</td></tr>
</table>
</td></tr>
<tr><td>Lógicos (Booleanos)</td><td>Datos que pueden tener dos valores (cierto - true o falso - false) Representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).
Se utiliza para representar alternativas (si/no) a determinadas condiciones.</td></tr>
<tr><td>Alfanuméricos (Carácter) (Cadena)</td><td>Es una secuencia de caracteres (letras y números) que permiten representar valores identificables de forma descriptiva; esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

Carácter (Char): se representan en muchos lenguajes de programación comenzando y terminando con el apóstrofe (‘) de la siguiente manera: ‘A’, ‘7’, ‘$’. Contienen un solo carácter, que puede ser:
• Las letras del alfabeto de la A-a a la Z-z.
• Los dígitos del 0 al 9.
• Caracteres especiales como: +, -, *, /, %, etc.

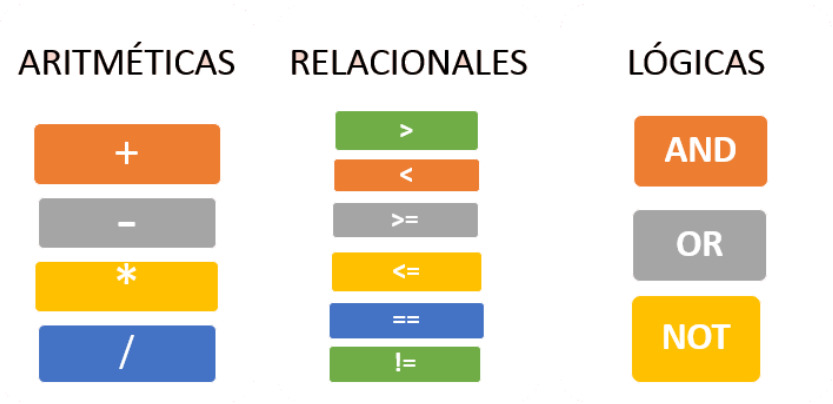
Cadena de caracteres (string): es una sucesión de caracteres que se encuentran delimitados por un apóstrofe o por comillas dobles, según el tipo de lenguaje de programación. La longitud de una cadena de caracteres es el número de ellos comprendidos entre los separadores o limitadores. Ejemplos: “DIOS”; “SER COMO EL AGUILA.”; “ ”; “”.</td></tr>
</table>

Datos Estructurados: datos que ocupan más de una casilla de memoria, entre ellos tenemos: Arreglos (Vectores y Matrices), Registros, Archivos o ficheros, Punteros.

EXPRESIONES

Las **expresiones** son combinaciones de *constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales.* Por ejemplo: a+(b + 3)/c.

Cada **expresión** toma un valor que se determina tomando los valores de las **variables** y **constantes** implicadas y la ejecución de las operaciones indicadas. Una **expresión** consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican en: **Aritméticas, Relaciónales, Lógicas.**



Una **expresión** es una combinación de valores y operaciones que, al ser evaluados, entregan un valor. Algunos elementos que pueden formar parte de una expresión son:
valores **literales** (como 2, "hola" o 5.7), **variables**, **operadores** y **llamadas a funciones**.

Por ejemplo, la **expresión** 4 * 3 - 2 entrega el valor 10 al ser evaluada por el intérprete:

```
>>> 4 * 3 – 2 = 10
```

El valor de la siguiente **expresión** depende del valor que tiene la variable n en el momento de la evaluación:

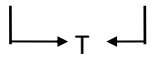
```
>>> n / 7 + 5
```

Una **expresión** está compuesta de otras expresiones, que son evaluadas recursivamente hasta llegar a sus componentes más simples, que son los literales y las variables.

En algunos casos es indispensable utilizar paréntesis para evitar la ambigüedad en las expresiones. Por ejemplo, la **expresión** 10 – 4 – 2 puede ser interpretada de dos maneras, cada una con un resultado distinto: 10 – (4 – 2) = 8, o también (10 – 4) – 2 = 4. Es buena idea usar siempre paréntesis en las expresiones, para estar seguros de que la interpretación del computador es la que nosotros necesitamos.

OPERADORES Y OPERANDOS

Los operadores son elementos que relacionan de forma diferente, los valores de una o más variables y/o constantes. Es decir, los operadores nos permiten manipular valores.
Un **operador** es un símbolo o palabra que significa que se ha de realizar cierta acción entre dos o más valores, llamados **operandos.** A continuación, se muestran los tipos de operadores:

Operadores	Descripción	Operadores
Aritméticos	• Permiten la realización de operaciones matemáticas con los valores (variables y constantes) según orden y/o reglas de prioridad. • Se utilizan con tipos de datos enteros o reales. • Si ambos son enteros, el resultado es entero; si alguno de ellos es real, el resultado es real. Ejemplo: 3 + 5 * (10 - 6) = 3 + 5 * 4 = 3 + 20 = 23 **Observación:** **Div** representa la división entera. **Mod** representa el resto o residuo de la división entera, aunque hay lenguajes que utilizan otro símbolo como %.	^ Potencia/exponente * Multiplicación / División Div División entera Mod Módulo + Suma - Resta
Relacionales	• Se utilizan para establecer una relación entre dos valores. • Compara estos valores entre si y esta comparación produce un resultado de certeza o falsedad (verdadero o falso). • Los operadores relaciónales comparan valores del mismo tipo (numéricos o cadenas) • Tienen el mismo nivel de prioridad en su evaluación. Ejemplo: Si a = 10 b = 20 c = 30 a + b > c Falso (F es False) a - b < c Verdadero (T es True)	> Mayor que < Menor que > = Mayor o igual que < = Menor o igual que < > Diferente = Igual
Alfanuméricos	• Se utilizan para establecer relaciones entre valores lógicos. • Estos valores pueden ser resultado de una expresión relacional. Ejemplo: Si a = 10 b = 20 c = 30 (a < b) and (b < c) (10<20) and (20<30) T and T 	And Y Or O Not Negación

Precedencia de los Operadores

Es importante destacar que antes de evaluar una expresión se debe tener presente que estas se deben sujetar a una **jerarquía de operadores**; esto evita que se produzcan ambigüedades en la ejecución de los cálculos. Con cada operador se asocia una prioridad; los operadores que tienen mayor prioridad se procesan primero y si existen varios operadores con la misma prioridad se continúa procesando de izquierda a derecha.

La **precedencia de operadores** es un conjunto de reglas que especifica el orden que deben ser evaluadas las operaciones de una expresión. La prioridad de los operadores a la hora de evaluar una expresión es:

Paréntesis->Potencias->Productos-divisiones->Sumas-restas->Relacionales->Lógicos.

COMENTARIOS

Un **comentario en programación** es la línea de texto en nuestro código fuente que el compilador ignora. El **código fuente** se compone de instrucciones, y el compilador traduce esas instrucciones del lenguaje de programación que estamos usando a lenguaje máquina. Cuando el compilador se encuentra con un comentario, esa línea, o varias de ellas, no las traduce, y continúa buscando en la instrucción siguiente.

La mayoría de los lenguajes de programación usan la sintaxis del **comentario en C**, que es la siguiente: */* Esto es un comentario */* Con la sintaxis /* */, podemos escribir en varias líneas:

/ Esto es un*

comentario/*

En el caso del **lenguaje C++,** también se añadió la opción de hacer un comentario en una línea:

//Comentario de tipo C++.

Importante: Java, C++, C, PHP y muchos lenguajes aceptan los dos tipos de comentario.

En otros lenguajes, los comentarios se hacen de otra forma. Veamos algunos de ellos:

Lenguaje	Descripción	Comentario
Visual Basic	Usa el símbolo '	'Comentario en 'visual basic
Pascal y Delphi	Usan (* *)	(* Comentario en Pascal *)
Python	Usa # para iniciar un comentario en línea comentarios multilíneas """ """	#Comentario en Python """Comentario multilínea. Segunda línea """
HTML	Se usa <!-- -->	<!-- Comentario HTML -->

IDENTIFICADORES

Los identificadores son los nombres que se les asignan a los **objetos,** los cuales se pueden considerar como **variables o constantes,** éstos intervienen en los procesos que se realizan para la solución de un problema, por consiguiente, es necesario establecer qué características tienen.

Para establecer los nombres de los identificadores se deben respetar ciertas reglas que establecen cada uno de los lenguajes de programación, para el caso que nos ocupa se establecen de forma indistinta según el problema que se esté abordando, sin seguir regla alguna, generalmente se utilizará la letra, o las letras, con la que inicia el nombre de la variable que representa el objeto que se va a identificar.

REGLAS PARA FORMAR UN IDENTIFICADOR

- El primer carácter de un identificador debe ser un carácter alfabético (A hasta la Z, en mayúsculas o minúsculas, o un carácter de subrayado (_). Debido a que los identificadores de algunos lenguajes como C++ distinguen entre mayúsculas y minúsculas, nombreArchivo es diferente de nombrearchivo.
- Las letras pueden ser minúsculas o mayúsculas del alfabeto inglés. Así pues, no está permitido el uso de las letras 'ñ' y 'Ñ'.
- No deben contener espacios en blanco.
- Los demás caracteres pueden ser letras, dígitos o el carácter especial **de** subrayado (_)
- Las letras no deben ser tildadas "No usar tildes".
- No deberá coincidir con palabras reservadas del lenguaje algorítmico. (Ejemplo: Var, Const, Entero, Real, etc.)
- La longitud de los identificadores puede ser de hasta 8 caracteres. Este valor dependerá del lenguaje de programación que esté utilizando.
- Deben ser nombres significativos al programa que se esté realizando.
- Indicar su tipo (Entero, Real, cadena, carácter o booleano)
- No pueden existir dos identificadores iguales que hagan referencia a dos elementos de distinta naturaleza en un programa. Lo cual no quiere decir que un identificador no pueda aparecer más de una vez en un programa.
- No pueden existir dos identificadores iguales que hagan referencia a dos elementos de distinta naturaleza en un programa. Lo cual no quiere decir que un identificador no pueda aparecer más de una vez en un programa.

Ejemplo de Identificadores válidos	Ejemplos identificadores no válidos
CostoArticulo Nuevo_sueldo Direccion Nombre Horas_Trab	Nombre[1] #alumnos profesión 2categoría Sueldo Neto Nombre-Apellido $Área_2$

VARIABLES

Los identificadores de tipo variable son todos aquellos objetos cuyo valor cambia durante la ejecución o proceso de solución del problema. Por ejemplo, el sueldo, el pago, el descuento, etcétera, que se deben calcular con un algoritmo determinado, o en su caso, contar con el largo (L) y ancho (A) de un rectángulo que servirán para calcular y obtener su área. Como se puede ver, tanto L como A son variables que se proporcionan para que el algoritmo pueda funcionar, y no necesariamente se calculen dentro del proceso de solución.

Una variable es donde se guarda (y se recupera) datos que se utilizan en un programa. Son zonas de memoria cuyo contenido cambia durante la fase de procesamiento de información.

Todas las variables deben ser de un tipo de datos, ya sea un ***dato de tipo primitivo*****,** como un **número** o **texto**, o un ***dato abstracto*****,** como un **objeto** que se ha creado. Así que básicamente podemos decir que una variable es **Tipo *de dato → identificador variable → valor almacenado.***

TIPOS DE VARIABLES POR SU CONTENIDO		
Numéricas	**Alfanuméricas**	**Lógicas (booleanas)**
Enteras y reales Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal. Ejemplo: 0.15 2500 -5425	***Caracteres Alfabéticos*** ***Dígitos, Caracteres especiales*** Está formada por caracteres alfanuméricos (letras, números y caracteres especiales). Ejemplo: Letra='a' apellido='lopez' direccion='Av. Libertad #190'	**Valor verdadero o 1** **Valor falso o 0** Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.
TIPOS DE VARIABLES POR SU USO		
De Trabajo	**Contadores**	**Acumuladores**
Reciben el resultado de una operación matemática completa. Se usan normalmente dentro de un programa. Ejemplo: a=c+b*2/4	Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con los incrementos generalmente de uno en uno. Ejemplo: a=a+1	Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente. Ejemplo: a=a+b

¿Cómo se declara una Variable?

Si lo vemos desde el punto de vista de un **algoritmo**, **declarar** una o varias **variables** es un proceso que consiste en **listar** al principio del algoritmo todas las **variables** que se usarán; además de colocar el **nombre de la variable**, se debe decir qué tipo de variable es.

Variable	Algoritmo	Lenguaje C	Declarada como
Contador	ENTERO	Int	Tipo entero
Edad, I	ENTERO	Int	Tipo entero
Direccion	CADENA DE CARACTERES	Char [9]	Cadena de caracteres
Sueldo_Base	REAL	float	Tipo real – décimal
Opcion	CARACTER	char	Tipo carácter

En el momento de declarar **constantes** debe indicarse que lo es y colocarse su respectivo valor.

CONSTANTE Pi 3.14159
CONSTANTE Msg “Presione una tecla y continue”
CONSTANTE ALTURA 40

Cuando se trabaja con **algoritmos** por lo general no se acostumbra a **declarar las variables** ni tampoco **constantes** debido a razones de simplicidad, es decir, no es camisa de fuerza **declarar las variables.** Sin embargo, al hacer la **declaración de variables** podemos hacer el **algoritmo** más **entendible** y **organizado** y de paso, permite acostumbrarnos a declararlas, ya que la mayoría de los **lenguajes de programación** (*entre ellos C y C++)* requieren que necesariamente se declaren las **variables** que se van a usar en los **programas.**

Pseudocódigo

```
INICIO
   Num1, Num2, Suma ENTERO
   Escribir “Diga dos números:”;
   Leer Num1, Num2;
   Suma=Num1+Num2;
   Escribir ”La suma es:”, Suma;
FIN
```

En el caso del **lenguaje C o C++,** hay que definir las **variables** con un **tipo** y **un identificador** (es decir, *un nombre para la variable*) Por ejemplo: int numero; dónde se define una **variable de tipo entero** que se llama **numero.** Seguidamente hay que añadir un punto y coma a la **expresión**. El punto y coma transforma la expresión en una instrucción que se ejecutará en el **programa.** Cuando se llegue a ella, el **programa** reservará espacio de memoria según el **tipo de variable** para poder escribir y leer valores. Por lo tanto, la fórmula general es: **<tipo> <identificador> <;>** que es lo que llamamos una **declaración de variable.**
En **C y C++** todas las **variables** se deben declarar antes de su uso. Si no, se producirá un **error de compilación**.

```
int main()
{
char caracter;
short valor;
int numero;
long numeroMasLargo;
float numeroRealFloat;
double numeroRealDoble;
// fin
return 0;
}
```

CONSTANTES

Un identificador se clasifica como constante cuando el valor que se le asigna a este identificador no cambia durante la ejecución o proceso de solución del problema. Por ejemplo, en problemas donde se utiliza el valor de PI, si el lenguaje que se utiliza para codificar el programa y ejecutarlo en la computadora no lo tiene definido, entonces se puede establecer de forma constante estableciendo un identificador llamado PI y asignarle el valor correspondiente de la siguiente manera: PI = 3.1416. De igual forma, se puede asignar valores constantes para otro identificadores según las necesidades del algoritmo que se esté diseñando.
Una constante es un dato numérico o alfanumérico que no cambia durante todo el desarrollo del algoritmo o durante la ejecución del programa. Es un objeto de valor invariable. Para expresar una constante se escribe explícitamente su valor.
Tipos de Constantes: Numéricas (Enteras y Reales), Alfanuméricas, Lógicas (Boolenas)

Constantes sin nombre	Constantes con nombre
Es una expresión numérica donde se puede utilizar directamente el valor.	Se hace una reserva de memoria en la cual se guarda el valor que será utilizado como constante. Ejemplo: a) PI = 3.1416 b) E = 2.718228 c) Iva = 0.13 d) itbm = 0.07

PALABRAS RESERVADAS

Los lenguajes de programación usan internamente algunas palabras, denominadas palabras clave o **palabras reservadas que no podemos usar como identificadores,** excepto si llevan delante el carácter @ como prefijo. Así, por ejemplo @for es un identificador válido.

LISTADO DE PALABRAS RESERVADAS						
LENGUAJE C			LENGUAJE C++			
auto	extern	sizeof	asm	else	operator	template
break	float	static	auto	enum	private	this
case	for	struct	bool	explicit	protected	throw
char	goto	switch	break	extern	public	true
const	if	typedef	case	false	register	try
continue	int	union	catch	float	reinterpret_cast	typedef
default	long	unsigned	char	for	return	typeid
do	register	void	class	friend	short	typename
double	return	volatile	const	goto	signed	union
else	short	while	const_cast	if	sizeof	unsigned
enum	signed		continue	inline	static	using
			default	int	static_cast	virtual
			delete	long	struct	void
			do	mutable	switch	volatile
			double	namespace		while
			dynamic_cast	new		

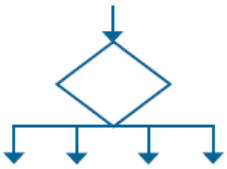
DIAGRAMAS DE FLUJO

Los diagramas de flujo son una herramienta que permite representar visualmente qué operaciones se requieren y en qué secuencia se deben efectuar para solucionar un problema dado. Por consiguiente, un diagrama de flujo es la representación gráfica mediante símbolos especiales, de los pasos o procedimientos de manera secuencial y lógica que se deben realizar para solucionar un problema dado.

Los diagramas de flujo desempeñan un papel vital en la programación de un problema, ya que facilitan la comprensión de problemas complicados y sobre todo aquellos en que sus procesos son muy largos;2 generalmente, los diagramas de flujo se dibujan antes de comenzar a programar el código fuente, que se ingresará posteriormente a la computadora.

Los diagramas de flujo facilitan la comunicación entre los programadores y los usuarios, además de que permiten de una manera más rápida detectar los posibles errores de lógica que se presenten al implementar el algoritmo. En la imagen se muestran algunos de los principales símbolos utilizados para construir un diagrama de flujo.

Dentro de los diagramas de flujo se pueden utilizar los símbolos que se presentan en la imagen, con los cuales se indican las operaciones que se efectuarán a los datos con el fin de producir un resultado.

Símbolo	Significado
	Terminal /Inicio.
	Entrada de datos.
	Proceso.
	Decisión.
	Decisión múltiple.
	Imprimir resultados.
	Flujo de datos.
	Conectores.