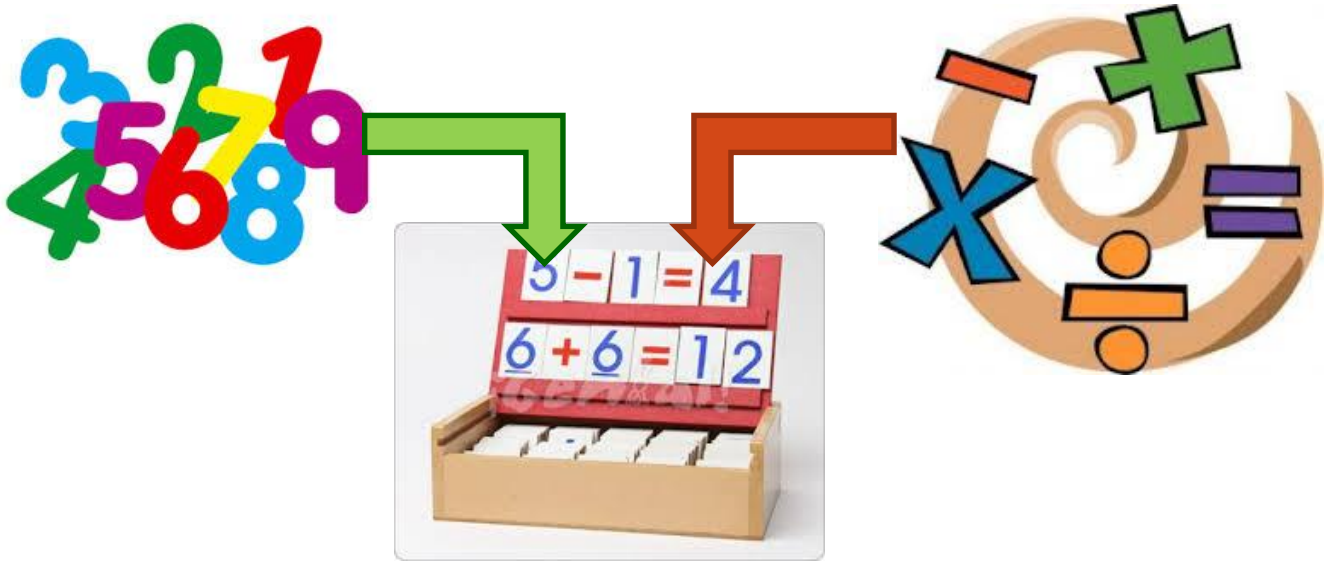
UNIDAD II: LOS DATOS SIMPLES Y LAS OPERACIONES BÁSICAS.



Un programa es un conjunto de instrucciones y datos que permiten solucionar problemas de forma eficiente y rápida, agilizando el trabajo de la personas. Los datos e información que procesa un programa, así como las operaciones que realiza tienen asociados una serie de conceptos que aquí se presentan.

Tipos de datos, tipos de datos abstractos y estructuras de datos

El término *tipo de datos* hace referencia a un conjunto de valores, mientras que el término *tipo de datos abstracto* (TDA) comprende tanto el conjunto de valores como las operaciones que pueden aplicárseles. Por ejemplo, sobre valores de tipo entero se pueden realizar tales como suma, resta, producto, cociente, etc.

Una *estructura de datos* se refiere a la implementación física de un *tipo de datos abstracto*, caracterizándolos por su organización y operaciones.

Los tipos de datos más utilizados en los diferentes lenguajes de programación son:

- Datos Simples
  - Estándar
    - Entero
    - Real
    - Carácter
    - Lógico
  - Definidos por el usuario
    - Subrango
    - Enumerado
- Datos Compuestos
  - Estáticos
    - Cadena
    - Conjunto
    - Arreglo
    - Registro
    - Archivos
  - Dinámicos
    - listas simples
    - listas dobles
    - Árboles
    - Grafos

- Datos Simples: Los tipos de datos simples caracterizan a las formas de presentación más sencillas de objetos de datos e información de un programa. Los lenguajes de programación permiten representar y manipular datos enteros, reales, caracteres y lógicos. Además es posible que el usuario defina tipos especiales, como subrango o enumerado, de acuerdo a los requerimientos que presente un problema en particular.
- Datos Compuestos: Los tipos de datos estructurados están construidos a partir de los tipos de datos simples y pueden ser estáticos o dinámicos. Las estructuras de datos estáticas son aquellas cuyo tamaño en memoria debe ser definido antes que el programa se ejecute y que no pueden modificar dicho tamaño durante su ejecución.

  Las estructuras de datos dinámicas pueden modificar el tamaño de memoria que ocupan mientras se ejecuta el programa, solicitando o liberando memoria según se requiera. Mediante el uso de un tipo de datos especial, denominado puntero, se pueden construir estructuras dinámicas soportadas por la mayoría de los lenguajes.

Es importante destacar que los tipos de datos simples tienen como característica común que cada objeto de datos representa a un elemento; mientras que los tipos compuestos tienen como característica común que un identificador (nombre de un objeto de datos) puede representar múltiples datos individuales, pudiendo referenciarse cada uno en forma independiente.

A continuación se describen los tipos de datos simples, definidos por el usuario y algunos tipos estructurados.

Datos Simples

Los diferentes objetos de información con los que trabaja un programa de computadora se conocen como datos. Todos los datos tienen un tipo asociado a ellos. El tipo de un dato determina el conjunto de valores que éste puede asumir. Los tipos de datos simples utilizados en casi todos los lenguajes de programación son:

Datos Numéricos

El tipo numérico es el conjunto de valores numéricos. Éstos pueden representarse en dos formas:

- Enteros: este tipo es un subconjunto de los números enteros, es decir, se trata de números sin parte decimal y que pueden ser positivos o negativos. Por ejemplo: -123, 0, 48, etc.
- Reales: este tipo es un subconjunto de los números reales, es decir, se trata de números con parte entera y parte decimal, que pueden ser positivos o negativos. Por ejemplo: -234.33, 0.0, 78.21, etc.

Datos Lógicos

El tipo lógico es un tipo ordinal, es decir, que tiene un número fijo de posibles valores y orden definido para éstos.

- Lógico: el tipo lógico o booleano puede tomar sólo 2 valores: Verdadero (V) o Falso (F). En general, este tipo se utiliza para representar la ocurrencia o no de un suceso o condición. Se considera que el valor Falso es menor que el valor Verdadero.

Datos Carácter

El tipo carácter representa una letra ('a', 'A'), un dígito ('0','9') o símbolo especial ('@', '&', '#'). Los datos de tipo carácter son ordinales.

- Caracteres: este tipo es el conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato tipo caracter contiene un solo símbolo, dígito o letra.

Subrango

El tipo *subrango* o intervalo se define de un tipo ordinal, especificando dos constantes de ese tipo, que actúan como límite inferior o superior del conjunto de datos. Por ejemplo, la siguiente definición especifica un subrango de caracteres cuyo límite inferior es '*a*' y cuyo límite superior es '*z*'. La variable letras declarada de tipo *minusculas* sólo podrá tomar valores comprendidos entre 'a' y 'z'.

```
PROGRAMA intervalo_letras
TIPOS
    minusculas='a'..'z'
VARIABLES
    letras:minusculas
```

Nota: recuerde que un dato es de tipo ordinal si sus elementos están dispuestos de modo que existe un primer y último elemento, cada elemento tiene un predecesor y un sucesor, salvo el primero y el último.

Enumerado

Un tipo enumerado se compone de un conjunto de valores referenciados por identificadores. Por ejemplo, la siguiente definición especifica el tipo enumerado vehiculos. La variable viaje declarada de tipo vehiculos sólo puede asumir los valores indicados en la definición.

```
PROGRAMA transportes
TIPOS
    vehiculos=(motocicleta, coche, camion, tren)
VARIABLES
    viaje:vehiculos
```

Características:

- Un tipo enumerado es un tipo ordinal cuyo orden se indica por la disposición de los valores en la definición.
- El número de orden de los elemento comienza en cero. Por ejemplo, considerando la definición anterior: 0 corresponde a motocicleta, 1 a coche, 2 a camion y 3 a tren.
- Las variables de tipo enumerado sólo pueden tomar valores de estos tipos. Por ejemplo, la asignación viaje←coche es correcta.
- Los únicos operadores que pueden utilizarse con los tipos enumerados son los de relación y de asignación. Por ejemplo, coche < tren (VERDADERO).
- No pueden aplicarse las operaciones LEER o ESCRIBIR sobre datos de tipo enumerado.
- Un mismo valor no puede aparecer en las definiciones de 2 tipos enumerados diferentes.

Los tipos de datos cadena y conjunto, si bien no son simples, se utilizan con frecuencia en pequeños programas, por tanto se describen a continuación.

Datos Tipo Cadena de Caracteres

Una cadena de caracteres es un conjunto de caracteres (incluido el espacio en blanco) reconocidos por la computadora, que se almacenan en posiciones de memorias contiguas. La longitud de una cadena es el número de caracteres que ésta contiene. La cadena que no contiene ningún carácter se denomina vacía o nula.

La representación de las cadenas suele utilizar comillas simples o dobles. Por ejemplo, la cadena *'hola mundo'* está delimitada por comillas simples.

Una subcadena es un conjunto de caracteres extraído de una cadena de mayor longitud. Por ejemplo, si se considera la cadena *'Escuela de Minas'*, la cadena *'Escuela'* es subcadena de la primera.

Conjunto

Un conjunto es una colección ordenada de datos simples, todos del mismo tipo. Es decir, es una colección de elementos no repetidos de tipo ordinal.

La declaración de tipos y definición de variables tipo conjunto sigue el formato presentado a continuación:

```
PROGRAMA conjunto_letras
TIPOS
    alfabeto=conjunto de caracter
VARIABLES
    letras, simbolos:alfabeto
```

Constantes y Variables

Una *constante* es un objeto de datos cuyo valor no cambia durante la ejecución de un programa. Una *constante* recibe su valor al momento de la compilación del programa y este valor no será modificado durante la ejecución.

Una *variable* es un objeto de datos de programa cuyo valor puede cambiar durante la ejecución de un programa. Este cambio se producirá mediante sentencias ejecutables. Una variable es una posición de memoria con nombre. El nombre de la posición se llama nombre de variable, y el valor almacenado en la posición se llama valor de la variable.

Tanto constantes como variables se definen con un tipo específico (numérico, lógico, carácter).

Operadores

Los operadores permiten realizar acciones sobre los datos de acuerdo al tipo de éstos. A continuación se presentan los operadores más utilizados en los lenguajes de programación.

Tipo	Símbolo	Nombre	Función
Paréntesis	( )		Anida expresiones
Aritméticos	** ó ^ *, / +, - div, mod	Potencia Producto, división Suma, diferencia División entera, resto	Conectan objetos o campos numéricos
Alfanuméricos	+	Concatenación	Conectan campos alfanuméricos
Relacionales	= < <= > >= <>	Igual a Menor que Menor o igual que Mayor que Mayor o igual que Distinto a	Conectan objetos, campos o expresiones de cualquier tipo. Su evaluación da como resultado "Verdadero" o "Falso".
Lógicos	NOT AND OR	Negación Conjunción Disyunción	Conectan expresiones de tipo lógico. Su evaluación da como resultado "Verdadero" o "Falso".

Los operadores lógicos NOT, AND y OR se evalúan según tablas de verdad. Considerando las variables booleanas *a* y *b*, las siguientes tablas indican los valores que se obtienen al aplicarles los operadores indicados.

Operador NO (NOT)

A	*NO a*
Verdadero	Falso
Falso	Verdadero

Operador Y (AND)

a	*b*	*a Y b*
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

Operador O (OR)

A	*b*	*a O b*
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Cuando se combinan operadores, es necesario considerar el orden en que se resuelven para obtener un resultado correcto. La siguiente tabla indica la precedencia de operadores.

Operador	Prioridad
NO, ^ *, /, Y div, mod +, -, O <, <=, =, <>, >=, >	Más alta (se evalúa primero) Más baja (se evalúa al final)
Si se utilizan paréntesis, las expresiones encerradas se evalúan primero.	

Expresiones

Las expresiones son combinaciones de constantes, variables, símbolos de operación y nombres de funciones especiales. Una expresión consta de *operandos* y *operadores*. De acuerdo al tipo de datos que manipulan las expresiones, éstas se clasifican en *aritméticas*, *alfanuméricas* y *lógicas*.

Las expresiones aritméticas son análogas a fórmulas matemáticas, en donde se utilizan operadores aritméticos y variables y constantes numéricas (reales o enteras).

Las expresiones alfanuméricas son aquellas en las que se utilizan operadores alfanuméricos y se producen resultados de tipo alfanumérico.

Las expresiones lógicas utilizan tanto operadores lógicos como relacionales para representar alguna condición. Al evaluar esta condición se obtiene un valor VERDADERO o FALSO.

Expresión	Tipo	Observación
3 + 9	Expresión numérica (entera)	
2.5 * 6.8 + 2	Expresión numérica (real)	
A + B * 2	Expresión numérica	A y B deben ser variables o de tipo numérico.
Falso Y Verdadero	Expresión lógica	
23 >= 30	Expresión lógica	
(A < B) Y (C = Falso)	Expresión lógica	A y B deben ser variables del mismo tipo y C debe ser variable de tipo lógico.

Escritura de Expresiones Algorítmicas

Las expresiones que tienen 2 o más operandos requieren de reglas matemáticas que permitan determinar el orden de las operaciones, estas reglas se denominan de prioridad o precedencia y son:

- las operaciones encerradas entre paréntesis se resuelven primero. Si existen paréntesis anidados se resuelven primero los más interiores.
- las operaciones aritméticas dentro de una expresión siguen el orden de precedencia indicado en la tabla Operador/Prioridad presentada anteriormente.

En caso de coincidir varios operadores de igual prioridad en una expresión o subexpresión encerrada entre paréntesis, el orden de prioridad es de izquierda a derecha.

Considerando las reglas de prioridad y los operadores aritméticos definidos, las expresiones matemáticas regulares pueden traducirse a expresiones aritméticas algorítmicas equivalentes que utilizan los lenguajes de programación. Por ejemplo, dadas las siguientes expresiones matemáticas sus equivalentes expresiones aritméticas algorítmicas son:

Expresión Matemática	Expresión Aritmética Algorítmica
$a \times x^2 + b \times x + c$	a * x ^ 2 + b * x + c
$\frac{-b + \sqrt{b^2 - 4 \times a \times c}}{2 \times a}$	(-b + (b ^ 2 – 4 * a * c) ^ (1 / 2))/(2 * a)
$\frac{a+2}{b} - \frac{c-4}{d \times 2}$	(a + 2) / b - ((c - 4) / (d * 2))

Asignación

La operación de *asignación* es el modo de darle valores a una variable. La operación de asignación se representa con el símbolo u operador ←. La operación de asignación se conoce como instrucción o sentencia de asignación cuando se refiere a un lenguaje de programación.

El formato general de la operación asignación es:

Nombre_Variable ← Expresión

La asignación es destructiva, esto significa que cuando se le asigna un valor a una variable se pierde el valor anterior. Por ejemplo, si la variable **suma** tiene valor 5, y luego se le asigna 12, éste será el valor final de la variable.

Las acciones de asignación se clasifican según el tipo de expresiones en: aritméticas, lógicas y de caracteres.

Asignación	Descripción	Ejemplos
Aritméticas	Las expresiones en las operaciones de asignación son aritméticas.	cantidad ← 100 total ← suma * 2 + 30
Lógicas	La expresión que se evalúa en la operación de asignación es lógica.	mayor ← 23 > 4 estado ← verdadero y (16 = 2 * valor)
Caracteres	La expresión que se evalúa es de tipo carácter o cadena de caracteres.	letra ← 'a' palabra ← 'programación'

Entrada y Salida de Información

Los cálculos que realizan las computadoras requieren, para ser útiles, de la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, la salida. Las operaciones de entrada permiten leer determinados valores y asignarlos a variables específicas. Esta entrada se conoce como operación de *Lectura*. Los datos se introducen al procesador mediante dispositivos de entrada (teclado, unidades de disco, etc.). La salida puede aparecer en un dispositivo de salida (pantalla, impresora, etc.). La operación de salida se denomina *Escritura*.

Las operaciones de *Lectura* y *Escritura* presentan el siguiente formato general:

LEER lista de variables de entrada

ESCRIBIR lista de expresiones de salida

Por ejemplo,

- la operación LEER *a*, *b*, *c* representa la lectura de 3 valores que se asignan a las variables *a*, *b* y *c*.
- la operación ESCRIBIR *'Hola Mundo'* visualiza en el dispositivo de salida elegido el mensaje "Hola Mundo"

Funciones Internas

Las operaciones que realizan los programas exigen en numerosas ocasiones, además de las operaciones básicas, un número determinado de operaciones especiales que se denominan funciones internas, incorporadas o estándar. Los lenguajes de programación incorporan estas funciones para facilitar el trabajo del programador. La siguiente tabla presenta algunas de las funciones internas incluidas en los lenguajes de programación.

Función	Descripción	Tipo de argumento	Resultado
abs(x)	valor absoluto de x	entero o real	igual que el argumento
arctan(x)	arco tangente de x	entero o real	Real
cos(x)	coseno de x	entero o real	Real
exp(x)	exponencial de x	entero o real	Real
ln(x)	logaritmo neperiano de x	entero o real	Real
log10(x)	logaritmo decimal de x	entero o real	Real
redondeo(x)	redondeo de x	Real	Entero
sen(x)	seno de x	entero o real	Real
cuadrado(x)	cuadrado de x	entero o real	igual que el argumento
raíz2(x)	raíz cuadrada de x	entero o real	Real

Operaciones con Cadenas

El tratamiento de cadenas contempla básicamente las siguientes operaciones:

- Cálculo de longitud
- Comparación
- Concatenación
- Extracción de subcadenas

Calculo de Longitud de una Cadena

La longitud de una cadena es el número de caracteres de la cadena. Por ejemplo, **'Hola Mundo!!!'** es una cadena de que tiene 13 caracteres (incluido el espacio en blanco).

La operación para determinar la longitud de una cadena se representará por la función longitud, cuyo formato es:

Longitud(cadena de caracteres**)**

Esta función, cuyo argumento es una cadena, retorna un valor entero que indica el número de caracteres de la cadena.

Comparación de Cadenas

La comparación de cadenas (igualdad y desigualdad) se basa en el orden numérico del código o juego de caracteres (por ejemplo, código ASCII) que admite una computadora.

Dos cadenas *a* y *b* de longitudes *m* y *n* son iguales si:

- El número de caracteres de *a* es igual al número de caracteres de *b* (*m*=*n*)
- Cada carácter de *a* es igual a su correspondiente de *b*; si $a=a_1a_2a_3...a_m$ y $b=b_1b_2b_3...b_n$, se debe verificar que $a_i=b_i$.

Por ejemplo, las siguientes expresiones comprueban la igualdad de cadenas:

'hola mundo' = 'hola mundo'
'informática'='computadora'
'Escuela de Minas' = 'escuela de minas'

La primera expresión es VERDADERA, la segunda es evidentemente FALSA, y la tercera expresión a primera vista puede parecer VERDADERA, sin embargo, los caracteres 'E' y 'e' son diferentes (el código binario que identifica a cada uno es distinto) por cuanto la expresión es FALSA.

Para comprobar la desigualdad de cadenas, en general, se utilizan los operadores relacionales (>, <, >=, <=, <>) y se ajustan a una comparación sucesiva de caracteres correspondientes en ambas cadenas hasta conseguir dos caracteres diferentes.

Por ejemplo, las siguientes expresiones comprueban la desigualdad de cadenas:

'zapato'<'avestruz'
'CARAMELO'>'golosina'
'amarillo'>='Amarillo'

La primera expresión es FALSA, al comparar el primer carácter de ambas cadenas (en realidad, sus correspondientes códigos binarios) se comprueba que 'z' no es menor que 'a' (considerando los caracteres ASCII). La segunda expresión es FALSA, porque las mayúsculas tienen códigos binarios menores a los de las minúsculas (código ASCII). Finalmente la tercera expresión es VERDADERA.

Concatenación

La concatenación es la operación de reunir varias cadenas de caracteres en una sola, pero conservando el orden de los caracteres de cada una de ellas.

El símbolo que representa la concatenación varía de un lenguaje a otro. Los más utilizados son: +, // y &.

Por ejemplo, la siguiente operación de concatenación combina las cadenas almacenadas en las variables A y B en la variable C.

A← 'escuela'
B← 'de minas'
C←A + B

El contenido de la variable C tras realizar la concatenación es 'escuelade minas'.

Subcadenas

La operación de extraer una parte específica de una cadena se representa por la función *subcadena*. Esta función presenta el siguiente formato:

subcadena(cadena**,** p_inicial**,** p_final**)**

Como puede observarse la función utiliza 3 parámetros, siendo opcional el último. El primer parámetro especifica la cadena original, el segundo indica la posición del primer carácter de la subcadena y el tercero, si se especifica, indica la posición del carácter final de la subcadena.

Por ejemplo:

palabra←subcadena('domingo',3,5)

La variable palabra (tipo cadena de caracteres) almacena cadena 'min'.

frase← 'hola mundo'
subfrase←subcadena(frase,6)

La variable subfrase (tipo cadena de caracteres) almacena la cadena 'mundo'. Obsérvese que si el último parámetro de la función subcadena no se especifica, los caracteres se extraen desde la posición inicial hasta el último carácter de la cadena original.

Operaciones sobre Conjuntos

Se pueden realizar las siguientes operaciones sobre datos de tipo conjunto:

- Inicialización
- Asignación
- Pertenencia
- Unión
- Intersección
- Diferencia

La inicialización de una variable conjunto se referencia como *conjunto vacío*. Por ejemplo, la variable *simbolos* definida en el ejemplo de declaración de conjunto, se inicializa en la siguiente instrucción:

simbolos← { }

La operación de *asignación* permite cargar elementos en un conjunto. Por ejemplo, considerando la variable *letras* (de tipo conjunto de caracteres) se puede realizar la siguiente asignación:

letras←{'a', 'm', 'q', 'z'}

Esta operación almacena en letras los elementos '*a*', '*m*', '*q*' y '*z*'.

La operación *pertenencia*, representada por la función lógica *pertenece*, determina si un elemento en particular está contenido en un dato tipo conjunto. La función pertenece tiene el siguiente formato:

pertenece(nombre_conjunto, elemento**)**

La función utiliza 2 argumentos, el primero indica el nombre de la variable tipo conjunto y el segundo especifica el elemento cuya pertenencia al conjunto se quiere verificar. La función retorna un valor lógico VERDADERO si el elemento especificado pertenece al conjunto indicado, de lo contrario devuelve FALSO.

La unión de dos conjuntos es un conjunto compuesto de los elementos de ambos conjuntos. La operación se denota con el signo +. Por ejemplo:

simbolos←{'j', 'k', 'l'}
letras←{'a', 'm', 'q', 'z'}
todos←simbolos + letras

La variable *todos*, de tipo conjunto, resulta de la unión de los conjuntos *simbolos* y *letras*. El conjunto *todos* contiene los elementos '*j*', '*k*', '*l*', '*a*', '*m*', '*q*' y '*z*'.

La *intersección* de dos conjuntos es el conjunto formado por los elementos comunes a ambos conjuntos. El operador intersección es *. Por ejemplo:

simbolos← {'b', 'm', 'q'}
letras← {'a', 'm', 'q', 'z'}
interconj← simbolos * letras

La variable *interconj*, de tipo conjunto, resulta de la intersección de los conjuntos *simbolos* y *letras*. El conjunto *interconj* contiene los elementos '*m*' y '*q*'.

La *diferencia de conjuntos* es el conjunto formado por los elementos del primero que no pertenecen al segundo. El operador diferencia es -. Por ejemplo:

simbolos← {'b', 'm', 'q'}
letras← {'a', 'm', 'q', 'z'}
difconj← simbolos - letras

La variable *difconj*, de tipo conjunto, resulta de la diferencia de los conjuntos *simbolos* y *letras*. El conjunto *difconj* contiene el elemento '*b*'.