Capítulo 2

Datos e información

Los datos constituyen la "materia prima" de la información. Un dato representa un número, una letra o un carácter que puede tener o no significado para quien lo recibe. Si tomamos un conjunto de datos aleatoriamente, es posible que no transmita ningún mensaje coherente o que no haya relación alguna entre sus elementos, es decir, los datos no tienen la capacidad de representar información por sí solos.

La información es la asociación coherente y con significado dentro de un contexto definido de datos individuales o de conjuntos de datos; por lo tanto, la información siempre tendrá significado para quien la recibe.

Para la computadora los datos tendrán significado si están codificados en el mismo lenguaje que estas utilizan. El lenguaje utilizado por las computadoras es el sistema numérico binario (*0 y 1*), representado por impulsos eléctricos (*apagado y encendido*). Por esto, los datos se almacenan en la memoria de la computadora en forma de un arreglo lineal de ceros y unos, como se muestra a continuación:

...	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	...

Cada una de las celdas del arreglo anterior se conoce como bit.

2.1 Bit y Byte

2.1.1 Bit

Es la contracción o acrónimo de dígito binario (***Bi***nary digi***t***). El bit puede ser considerado como la unidad más pequeña de información. Se utiliza fundamentalmente en la realización de las denominadas operaciones booleanas: AND, OR, NOT y XOR. Sin embargo, se necesita más de un bit para la representación de datos (*números, letras, símbolos*), por lo que se introduce el concepto de **byte**.

2.1.2 Byte

Se denomina byte u octeto a un conjunto de ocho bits, en los que se almacenan números binarios de ocho dígitos. El byte es la unidad básica de medida de la capacidad de la memoria, o cualquier dispositivo de almacenamiento de una computadora. En un byte se representa el código ASCII de un carácter . Para almacenar cadenas de caracteres se utilizan conjuntos de mayor tamaño, conocidos como múltiplos del byte, pero antes de definirlos resolveremos el siguiente interrogante.

¿Por qué un byte tiene ocho bits y no siete u once, por ejemplo?

La razón es simple. Cuando se diseñó la computadora era necesario asignar códigos a un conjunto de caracteres, alrededor de ciento cincuenta y cuatro, que fueron clasificados en **numéricos** (0, 1, ..., 9), **alfabéticos** (a, b, c, ..., x, y, z, A, B, C, ..., X, Y, Z), **especiales** ($\sharp, @, |, \%, ..., ", !$) y de **control**, *retorno de carro o enter, avance de línea, imprimir pantalla* y otros. Mediante estos códigos se debe suministrar la información a la máquina a través de un dispositivo de entrada, que generalmente es el teclado. Para generar estos códigos son necesarios los grupos de ocho bits, como se muestra a continuación.

Con un bit solo se pueden generar $2^1 = 2$ códigos distintos:
(0, 1).

Con dos bits se pueden generar $2^2 = 4$ códigos distintos:
(00, 01, 10, 11).

Con tres bits se pueden generar $2^3 = 8$ códigos distintos:
(000, 001, 010, 011, 100, 101, 110, 111).

Si continuamos razonando de esta forma, es fácil darse cuenta de que con siete bits pueden generarse $2^7 = 128$ códigos, que es inferior a los 154 que se necesitaban inicialmente.

Con ocho bits se pueden generar $2^8 = 256$ códigos; con estos era posible asignar códigos a los 154 caracteres, y además quedaban a disposición de los diseñadores 102 códigos para nuevos requerimientos o necesidades; esto fue aprovechado y se constituyó en un estándar que se conoce como códigos ASCII (American Standard Code for Information Interchange) .

Este estándar fue definido para 7 bits (128 códigos) y para 8 bits (256 códigos), los cuales se muestran en la tabla 1.2.

Los 32 primeros caracteres y el último son caracteres de control y no son imprimibles. Los siguientes son los significados de los caracteres de control:

NULL	Nulo	DC1	Control de dispositivo 1
SOH	Comienzo de cabeza	DC2	Control de dispositivo 2
STX	Comienzo de texto	DC3	Control de dispositivo 3
ETX	Final de texto	DC4	Control de dispositivo 4

EOT	Fin de transmisión	NAK	Acuse de recibo negativo
ENQ	Petición, consulta	SYN	Sincronización
ACK	Acuse de recibo	ETB	Final del bloque de transmisión
BEL	Pitido	CAN	Anulación
BS	Retroceso de un espacio	EM	Fin de soporte (de cinta, etc.)
HT	Tabulación horizontal	SUB	Sustituir
LF	Saltar a la línea siguiente	ESC	Escape
VT	Tabulación vertical	FS	Separador de archivo
FF	Alimentación de hoja	GS	Separador de grupo
CR	Retorno de carro	RS	Separador de registro
SO	Fuera de código	US	Separador de sub-registro
SI	Dentro de código	DEL	Borrar
DLE	Escape de enlace de datos. Carácter de control que cambia el significado del carácter que se le sigue.		

A este conjunto de caracteres se le denomina "código ASCII extendido". Los 32 primeros caracteres de control son los mismos del conjunto de caracteres ASCII de 7 bits.

2.2 Múltiplos y submúltiplos del byte

Nibble
Un nibble es un conjunto de cuatro bits, conocido también como "cuarteto o semioctecto", que permite representar un número binario de cuatro dígitos. Estos números van desde el 0000 (0_{16}) al 1111 (F_{16}) y corresponden a un carácter del sistema numérico hexadecimal. Los únicos submúltiplos del byte son el nibble y el bit.

Word, Half Word, Double Word
El Word (Palabra) , el Half Word (Media Palabra) y el Double Word (Doble Palabra) son múltiplos del byte. Un Word es el número máximo de bits con los que trabaja un procesador de manera simultánea. Existen procesadores de 8, 16, 32 hasta 64 bits, y el diseño de los sistemas operativos está restringido a la capacidad de los procesadores.

El Half Word y el Double Word son, entonces, la mitad y el doble de un Word, respectivamente. La longitud o el número de bits de estos múltiplos varía de acuerdo con el tamaño de la palabra; en este caso definiremos la longitud de una palabra como 32 bits para establecer la similitud con la mayoría de procesadores que se fabrican hoy en día.

Los múltiplos del byte más conocidos por su uso frecuente en la especificación de

la capacidad de los dispositivos de almacenamiento son: kilobyte (kB[1]), megabyte (MB), gigabyte (GB) y terabyte (TB). Estos prefijos (*kilo*, *mega*, *giga*, *tera*) están definidos en el Sistema Internacional de Medidas (*SI*).

Tabla 2.1 Tabla de múltiplos y submúltiplos del byte

Múltiplos/Submúltiplos	n° de bytes	n° de bits
Bit (b)	2^{-3}	2^0
Nibble	2^{-1}	2^2
Byte (B)	2^0	2^3
Half Word / Media Palabra	2^1	2^4
Word / Palabra (P)	2^2	2^5
Double Word / Doble Palabra	2^3	2^6
Kilobyte (KB)	2^{10}	2^{13}
Megabyte (MB)	2^{20}	2^{23}
Gigabyte (GB)	2^{30}	2^{33}
Terabyte (TB)	2^{40}	2^{43}
Petabyte (PB)	2^{50}	2^{53}
Exabyte (EB)	2^{60}	2^{63}
Zettabyte (ZB)	2^{70}	2^{73}
Yottabyte (YB)	2^{80}	2^{83}

Se plantea el siguiente interrogante:

¿Por qué un kilobyte corresponde a $2^{10} = 1024$ bytes y no a 1000 bytes, como lo sugiere el prefijo *kilo*, en el Sistema Internacional de Medidas?

En el sistema numérico decimal, de base $b = 10$, el prefijo *kilo* es equivalente a $10^3 = 1000$; por ejemplo, 1 kilogramo equivale a 1000 gramos, un kilómetro a 1000 metros. Los diseñadores de las computadoras intentaron darle el mismo significado, sin embargo, la máquina trabaja con el sistema numérico binario, de base $b = 2$.

Así surge la siguiente ecuación: $2^x = 10^3$, donde x debe ser un número entero. Las posibles soluciones fueron $x = 9$ o $x = 10$, con lo que $2^9 = 512$ y $2^{10} = 1024$; la segunda solución, $x = 10$, producía un resultado más cercano a 1000, y fue la escogida. Esta situación es un poco desafortunada, ya que profesionales de otras áreas cuestionan el uso del prefijo *kilo* en este contexto; algunos incluso proponen que se utilice el prefijo *kiwi*.

[1]Tenga en cuenta que en minúsculas 'k' es el símbolo de la unidad adecuada para el prefijo kilo. 'K' mayúscula es propiamente el símbolo de la unidad para la unidad de temperatura termodinámica Kelvin. Sin embargo, es muy común dentro de la industria de la computación binaria al indicar la capacidad, especialmente en la literatura de marketing y envasado del producto, el uso de K mayúscula y sin espacio (1 KB), pero '1 KB' no es incorrecto, pero a menudo se considera más adecuado en lenguaje técnico.

2.2.1 Ejemplos

- **¿Cuántos caracteres pueden almacenarse en un área de memoria de 128 kilobytes?**

  Asumiendo 1 byte $\equiv$ 1 carácter, se tiene que

$$128\,\mathrm{KB} = 2^7\,\mathrm{KB} \cdot \frac{2^{10}\,\mathrm{B}}{1\,\mathrm{KB}} = 2^{17}\,\mathrm{B}$$

- **¿Cuántas palabras hay en 48 gigabytes?**

$$48\,\mathrm{GB} = 3 \cdot 2^4\,\mathrm{GB} \cdot \frac{2^{30}\,\mathrm{B}}{1\,\mathrm{GB}} \cdot \frac{1\,\mathrm{P}}{2^5\,\mathrm{B}} = \frac{3 \cdot 2^{34}\,\mathrm{P}}{2^5} = 3 \cdot 2^{29}\,\mathrm{P}$$

- **¿Cuántos bits hay en 12 megapalabras?**

$$12\,\mathrm{MP} = 12\,\mathrm{MP} \cdot \frac{2^{20}\,\mathrm{P}}{1\,\mathrm{MP}} \cdot \frac{2^5\,\mathrm{B}}{1\,\mathrm{P}} = 12 \cdot 2^{25}\,\mathrm{B} = 402\,653\,184\,\mathrm{B} = 48\,\mathrm{MB}$$

- **¿Cuántos bits son necesarios para representar un texto que contiene 458 megabytes?**

$$458\,\mathrm{MB} = 458\,\mathrm{MB} \cdot \frac{2^{23}\,\mathrm{B}}{1\,\mathrm{MB}} = 3\,841\,982\,464\,\mathrm{B}$$

- **¿Cuántos terabytes hay en 0,048 yottabytes?**

  1 terabyte $\equiv 2^{40}$ bytes
  1 yottabyte $\equiv 2^{80}$ bytes

$$0,048\,\mathrm{YB} = 0,048\,\mathrm{YB} \cdot \frac{2^{80}\,\mathrm{B}}{1\,\mathrm{YB}} \cdot \frac{1\,\mathrm{TB}}{2^{40}\,\mathrm{B}} = 0.048 \cdot 2^{40}\,\mathrm{TB}$$

- **¿Cuántos nibbles pueden almacenarse en un dispositivo de almacenamiento extraíble de 0,0001 terabytes?**
  1 terabyte $\equiv 2^{40}$ bytes
  1 nibble $\equiv 2^{-1}$ bytes

$$\begin{aligned} 0,0001\,\mathrm{TB} &= 0,0001\,\mathrm{TB} \cdot \frac{2^{40}\,\mathrm{B}}{1\,\mathrm{TB}} = 109\,951\,162.8\,\mathrm{B} \cdot \frac{1\ \mathrm{nibble}}{2^{-1}\,\mathrm{B}} \\ &= 219\,902\,325.6\ \mathrm{nibbles} \end{aligned}$$

- **¿Cuántos exabytes hay en 458 697 doblepalabras?**
  1 Exa Byte $\equiv 2^{60}$ bytes
  1 Doble Palabra $\equiv 2^3$ bytes

$$458.697\,\mathrm{DP} = 458.697\,\mathrm{DP} \cdot \frac{2^3\,\mathrm{B}}{1\,\mathrm{DP}} \cdot \frac{1\,\mathrm{EB}}{2^{60}\,\mathrm{B}} = 458\,697 \cdot 2^{-57}\,\mathrm{EB}$$

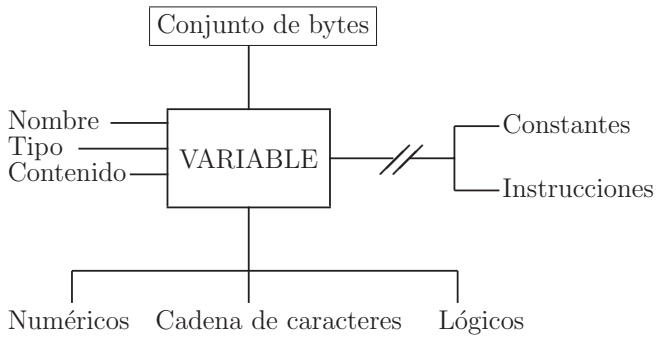
2.3 Variables y tipos predefinidos

2.3.1 Variables

Una *variable* es un conjunto de bytes en la memoria, referenciado por un nombre, donde se almacena el valor correspondiente a un dato. Dicho valor puede modificarse cuando un programa lo requiera. El nombre de una variable se construye con una o más letras seguidas de un número y/o más letras. No se permiten caracteres especiales ($\sharp, @, |, \%, ..., ", !$) en los nombres de las variables. Este nombre es elegido por el usuario cumpliendo el criterio anterior. Para que estén correctamente definidas las variables hay que especificar:

- Su nombre.
- El tipo de dato: numérico, carácter, cadena de caracteres o booleano.
- Su valor inicial, el cual es opcional, porque al no inicializar la variable, esta tomará el valor previamente almacenado en el conjunto de bytes que le corresponda.

Cuando una variable se ha declarado de un cierto tipo de dato, solamente puede asignársele datos del mismo tipo. Por ejemplo, una variable que ha sido declarada de tipo alfabético no puede almacenar datos de tipo numérico. Si se intenta asignar a una variable un valor de un tipo de dato que no corresponde con su declaración, se produce un error de tipo.



Hay otras características de las variables (la dirección y el tamaño) que trataremos posteriormente.

2.3.2 Tipos de datos

Datos numéricos

Los *datos numéricos* están contenidos en el conjunto de los números. Existen dos clases de datos numéricos: número entero y número real.

- *Entero*: este tipo de variable admite todos los valores de los números enteros, negativos y positivos. Son los números de valor completo, no se permiten partes fraccionarias o decimales. Se les llama también números de coma fija.

  Por ejemplo: $5, -6, 2000, 3456, -981$.

- *Real*: este tipo de variable admite todos los valores de los números reales, negativos y positivos. Estos números se conocen como compuestos, constan de una parte entera y una parte fraccionaria o decimal. Los números reales pueden representarse en notación científica o en coma flotante.

  Los números reales contienen a los números enteros, por lo tanto, un dato de tipo entero puede ser presentado también como un dato de tipo real con parte decimal nula.

  Por ejemplo: $5.98, -67.21, 0.0201, -3456.00, -981.6788$.

Datos carácter y cadena de caracteres

Los datos del tipo *carácter* están contenidos en el conjunto finito de caracteres: alfabéticos, numéricos y especiales. Cuando una variable es declarada de este tipo, admitirá un solo carácter del conjunto de caracteres establecido para la computadora. Por lo tanto, la longitud de una variable tipo carácter siempre será un byte.

Por ejemplo: ‘a’,‘4’,‘G’,‘Z’,‘#’,‘@’.

Los datos del tipo *cadena de caracteres* son sucesiones de caracteres con longitud finita. La longitud de una variable de este tipo es el número de caracteres que la componen.

Por ejemplo: “casa”, “7850”, “GABT”, “Zapato”, “(definición)”, “@jut”, “911”.

Datos lógicos

Los datos tipo lógico pueden tomar uno de los dos valores booleanos: VERDADERO (*true*) o FALSO (*false*). Estos dos valores tienen correspondencia con los dígitos del sistema binario: *0* para indicar falso y *1* para indicar verdadero.

2.3.3 Tipos Predefinidos

Mediante los lenguajes de alto nivel los programadores pueden definir sus propios tipos de datos, y se conocen como tipos predefinidos. Para los lenguajes de programación *C++* y *Java*, los tipos predefinidos más utilizados son:

- Entero:
  - int (*Variable entera en C++ y Java*)
  - Integer (*Objeto de tipo entero en Java*)
- Real:
  - float (*Variable real de precisión simple en C++ y Java*)
  - double (*Variable real de precisión doble en C++ y Java*)
  - Float (*Objeto de tipo real de precisión simple en Java*)
  - Double (*Objeto de tipo real de precisión doble en Java*)
- Carácter:
  - char (*Variable de tipo carácter en C++ y Java*)
- Cadena de Caracteres:
  - string (*Variable de tipo cadena de caracteres en C++*)
  - String (*Objeto de tipo cadena de caracteres en Java*)
- Lógico o booleano:
  - bool (*Variable de tipo booleano en C++*)
  - boolean (*Variable de tipo booleano en Java*)
  - Boolean (*Objeto de tipo booleano en Java*)
- Valores Predefinidos (*C++ y Java*) :
  - Verdadero: **true**
  - Falso: **false**
  - Valor nulo: **null**

Las variables se representan en grupos de bits de diferente tamaño; pueden ser de 8, 16, 32 y hasta 64 bits. Debido a que los recursos de las máquinas son limitados, se ha establecido para cada tipo de variable un número máximo de bits para representar su valor. Por ejemplo, en la mayoría de las computadoras, una variable de tipo *int* dispone máximo de 16 bits para representar su valor numérico; por ende, en este

caso el máximo número entero representado es $2^{16} - 1 = 65\,535$, y corresponde al número binario que contiene 16 unos: 1111111111111111.

Además, con el objetivo de aprovechar al máximo los recursos disponibles de la máquina se han definido modificadores de longitud para el tamaño de las variables. Existen dos modificadores de longitud: *short* y *long*, los cuales reducen a la mitad o duplican, respectivamente, el número de bits en memoria reservados para la variable. Por ejemplo, si agregamos el modificador *long* a una variable de tipo *int*, tendríamos disponibles 32 bits para representar su valor.

Por ejemplo: *long int*, *short float*, *long double*.

En el sistema numérico binario, para representar números positivos o negativos se utiliza el bit más a la izquierda del número. Si el bit más a la izquierda es *0*, el número es positivo; si es *1*, el número es negativo. Por esto existen dos modificadores adicionales : *signed* (*con signo*) y *unsigned* (*sin signo*). Si queremos representar una variable de tipo *int con signo*, el número mínimo que se puede representar sería -32 768 y el número máximo sería 32 767. Si contamos cuántos números hay en este intervalo, obtendríamos $2^{16} - 1$ números.

Tabla 2.2 Tipos de Datos (*Cota Mínima y Máxima*)

Tipo de dato	n° de bits	Valor mínimo	Valor máximo
short int	8	-128	127
long int	32	$-2\,147\,483\,647$	$2\,147\,483\,647$
signed int	16	$-32\,768$	$32\,767$
int (unsigned int)	16	0	$65\,535$
float	32	$-3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$
double	64	$-1.7 \cdot 10^{308}$	$1.7 \cdot 10^{308}$
long double	80	$-3.4 \cdot 10^{4932}$	$3.4 \cdot 10^{4932}$

2.3.4 Ejemplos de declaraciones de variables en C++ y Java

- `Entero w`
  Nombre: w
  Tipo: Entero
  Contenido: No asignado
  `int` w; *Declaración en C++ y Java*

- `Real saldo_mensual`
  Nombre: saldo_mensual
  Tipo: Real
  Contenido: No asignado
  `float` $saldo\_mensual$; *Declaración en C++ y Java, Real de precisión simple*
  `double` $saldo\_mensual$; *Declaración en C++ y Java, Real de doble precisión*

- `Entero` $x \longleftarrow 5$

Nombre: x
Tipo: Entero
Contenido: 5
`int` $x = 5$; *Declaración en C++ y Java*

- `Real` $x \longleftarrow 9$
  Nombre: x
  Tipo: Real
  Contenido: 9
  `float` $x = 9$; *Declaración en C++ y Java, Real de precisión simple*
  `double` $x = 9$; *Declaración en C++ y Java, Real de doble precisión*

- `Real` $salario\_minimo \longleftarrow 408000$
  Nombre: salario_minimo
  Tipo: Real
  Contenido: 408000
  `float` $salario\_minimo = 408000$; *Declaración en C++ y Java, Real de precisión simple*
  `double` $salario\_minimo = 408000$; *Declaración en C++ y Java, Real de doble precisión*

- `Cadena` *linea*
  Nombre: linea
  Tipo: Cadena
  Contenido: No asignado
  `string` *linea*; *Declaración en C++, variable string*
  `String` *linea*; *Declaración en Java, objeto String*

- `Cadena` *cad* $\longleftarrow$ "*cazador*"
  Nombre: cad
  Tipo: Cadena
  Contenido: "*cazador*"
  `string` *cad* = "*cazador*"; *Declaración en C++, variable string*
  `String` *cad* = "*cazador*"; *Declaración en Java, objeto String*

- `Cadena` $info \longleftarrow$ `''Hoy es miércoles 24''`
  Nombre: info
  Tipo: Cadena
  Contenido: *"Hoy es miércoles 24"*
  `string` $info =$ *"Hoy es miércoles 24"*; *Declaración en C++, variable string*
  `String` $info =$ *"Hoy es miércoles 24"*; *Declaración en Java, objeto String*

- `Booleano sw`
  Nombre: sw
  Tipo: Booleano
  Contenido: No asignado
  `bool sw;` *Declaración en C++*
  `boolean sw;` *Declaración en Java*

- `Booleano` $sw \longleftarrow false$
  Nombre: sw
  Tipo: Booleano
  Contenido: false
  `bool sw = false;` *Declaración en C++*
  `boolean sw = false;` *Declaración en Java*

2.4 Operadores

Todos los símbolos que representan enlaces entre cada uno de los argumentos que intervienen en una operación se llaman operadores, y se utilizan para construir expresiones. Los operadores se clasifican en aritméticos, relacionales y lógicos.

2.4.1 Aritméticos

Se utilizan para formar expresiones cuyo resultado será un valor numérico. Junto con las variables de tipo numérico dan lugar a las expresiones aritméticas.

- $+$ Suma
- $-$ Resta, Negación
- $\cdot$ Multiplicación
- $\wedge$ Potenciación
- / División real
- DIV División entera
- MOD Residuo de la división entera

Estos operadores son binarios, es decir, admiten dos operandos: uno a la izquierda y otro a la derecha, y tienen un único resultado. Excepto en el caso del operador "-", es binario cuando indica resta $(X - Y)$ y es unario cuando indica la negación $(-X)$.

Los operadores DIV y MOD no están definidos para todos los lenguajes de programación. En el caso de los lenguajes C++ y Java, estos operadores se representan de la siguiente manera:

- número DIV divisor: `int(numero / divisor)`
- número MOD divisor: `numero % divisor`

2.4.2 Relacionales o Condicionales

Se utilizan para formar expresiones booleanas, es decir, expresiones que al ser evaluadas producen un valor booleano : VERDADERO o FALSO.

- $<$ Menor que
- $=$ Igual
- $>$ Mayor que
- $<=$ Menor o igual que
- $>=$ Mayor o igual que
- $\neq$ ó $<>$ Distinto de

Cuando se comparan caracteres alfanuméricos se hace uno a uno, de izquierda a derecha. Si las variables son de diferente longitud pero exactamente iguales hasta el último carácter del más corto, entonces se considera que el más corto es el menor. Solo son iguales dos datos alfanuméricos si son iguales su longitud y sus componentes. Debido al valor numérico en el sistema binario, las letras minúsculas tienen mayor valor que las mayúsculas. Los operadores relacionales son binarios también.

2.4.3 Lógicos o booleanos

Combinan sus operandos (*proposiciones simples o compuestas*) de acuerdo con las reglas del álgebra de Boole. El objetivo es producir un nuevo valor (FALSO o VERDADERO) que se convierta en el valor de la expresión. Las condiciones para control de flujo son expresiones de este tipo. Los operadores booleanos que utilizaremos básicamente son:

- **OR** ($\vee$) Suma lógica
- **AND** ($\wedge$) Producto lógico
- **NOT** ($\sim$) Negación

Operador OR u o
Es un operador binario, son necesarios dos operandos para producir un resultado. La expresión que se forma producirá un valor VERDADERO cuando al menos uno de sus operandos tenga este mismo valor; de otra forma, el valor de la expresión será FALSO. Es el operador lógico de disyunción.

p	q	*p or q*
verdadero	verdadero	verdadero
verdadero	falso	verdadero
falso	verdadero	verdadero
falso	falso	falso

Operador AND o y
Es también un operador binario . La expresión formada tendrá valor VERDADERO cuando ambos operandos tengan este mismo valor; en caso contrario, la expresión tendrá valor FALSO. Es el operador lógico de conjunción.

p	q	*p and q*
verdadero	verdadero	verdadero
verdadero	falso	falso
falso	verdadero	falso
falso	falso	falso

Operador NOT o negación
Es un operador unario, es decir, solo afecta a un operando. Afecta a una variable o a una expresión cambiando su estado lógico: si es VERDADERO, lo transforma en FALSO, y viceversa.

p	$\sim p$
verdadero	falso
falso	verdadero

2.4.4 Operador de Asignación

Mediante este operador, "←", se almacenan valores en una variable. Estos valores pueden ser constantes o ser resultado de una expresión. La sintaxis de la operación de asignación es

$$Variable \longleftarrow Expresion$$

Recordemos que el tipo de dato de *Variable* debe ser el mismo que el de *Expresión*; en caso contrario se producirá un error de tipo.

De acuerdo con el tipo de dato de *Expresión* hay distintos tipos de asignación.

Operación de Asignación Aritmética
Este tipo de asignación corresponde con el esquema

$$Variable \longleftarrow Expresión\ Aritmética$$

El valor de *Expresión Aritmética* puede ser constante, una variable o el resultado de una operación aritmética entre valores constantes o variables. Por ejemplo:

$minuendo \longleftarrow 6$
$sustraendo \longleftarrow (5 \cdot 4) - 7$
$incremento \longleftarrow minuendo$
$incremento \longleftarrow (minuendo + 5)$
$diferencia \longleftarrow (minuendo - sustraendo) + incremento$

A la variable *minuendo* se le asigna un valor constante. La variable *sustraendo* es el resultado de una operación aritmética cuyo resultado es *13*. La variable *incremento* toma el valor de *minuendo* más cinco. El valor de la variable *diferencia* es el resultado de restar *sustraendo* de *minuendo* y sumarle *incremento*.

Las operaciones anteriores se pueden abreviar así:

$diferencia \longleftarrow (6 - ((5 \cdot 4) - 7)) + (6 + 5)$

Una variable puede tomar su valor previo, modificarlo y asignárselo nuevamente:

$diferencia \longleftarrow diferencia + 1$
$diferencia \longleftarrow diferencia * minuendo$

Operación de Asignación Lógica

Este tipo de asignación corresponde con el esquema

$$Variable \longleftarrow Expresión\ Relacional$$
$$Variable \longleftarrow Expresión\ Lógica$$

El valor asignado a *Variable* es un valor lógico; puede ser *true*, *false* o el resultado de evaluar una expresión relacional o lógica. Por ejemplo:

$SW \longleftarrow (4 \geq 3)$
$SW2 \longleftarrow 7 \neq 8$
$X \longleftarrow (SW \vee SW2)$
$Y \longleftarrow 9 = 9$

Las variables *SW*, *SW2* y X toman valor Verdadero y la variable Y toma valor Falso.

2.5 Expresiones

En términos generales, una expresión es una relación entre variables y operadores relacionales, aritméticos y/o lógicos.

El valor de una expresión está determinado por el tipo de operadores y operandos que combina. Además, este valor se verá modificado por el orden en el que se realicen la(s) operacion(es) implicadas en la expresión. Del mismo modo que los operadores, las expresiones se clasifican también en aritméticas, relacionales y lógicas.

Operadores matemáticos		Operadores lógicos	
$+$	Suma	*or*	o
$-$	Resta	*and*	y
$*$	Producto	$\sim$	negación
$/$	División	$>$	mayor que
$\%$	Residuo	$<$	menor que
$\subset$	Subconjunto	$=$	igual
$\subseteq$	Subconjunto o igual	$\leq$	menor o igual
$\wedge$	Potenciación	$\geq$	mayor o igual
( )	Paréntesis	$\neq$	diferente
[ ]	Corchetes	$\Longleftrightarrow$	Si sólo si
{ }	Llaves	$\Longrightarrow$	Si entonces

2.5.1 Expresiones aritméticas

Se componen de operadores y funciones aritméticas y sus operandos tienen valor de tipo numérico (*variables* y *constantes*); su resultado también es numérico. En algoritmos son utilizadas básicamente para la asignación de valores a las variables.

Por ejemplo:

Operando 1	Operando 2	Expresión aritmética	Resultado
3	4	$3 + 4$	7
4,78	3,91	$4,78 * 3,91$	18,6898
10	6	10 MOD 6	4
81	-	$\sqrt{81}$	9
45	8	45 DIV 8	5
6	-8,4	$6 - 8,4$	$-2,4$

2.5.2 Expresiones relacionales o condicionales

Una expresión relacional es utilizada para establecer comparaciones entre operandos lógicos (*variables* y *constantes lógicas*) mediante los operadores relacionales o condicionales. Su resultado es un valor lógico.

Por ejemplo:

Operando 1	Operando 2	Expresión relacional	Resultado
12,01	10,53	$12,01 > 10,531$	verdadero
45	8	$45 < 8$	falso
7	8	$7 = 8$	falso
3,6	4	$3 \geq 4$	falso
6	6	$6 \leq 6$	verdadero
5	8,9	$5 \neq 8,9$	verdadero

En la comparación de valores reales por medio del operador =, ya sean *float* o *double*, debido a las limitaciones de la computadora con respecto a la precisión, es posible obtener un valor falso incluso si los dos valores reales son exactamente iguales. En algunos lenguajes de programación, aun si los valores almacenados son los mismos, la comparación de igualdad entre ellos dará como resultado un valor falso.

2.5.3 Expresiones lógicas o booleanas

Las expresiones lógicas, al igual que las expresiones relacionales, dan como resultado un valor lógico. Son utilizadas en la especificación de condiciones para control de flujo en los algoritmos. Estas expresiones generalmente están compuestas de otras expresiones más simples, cuya evaluación arroja valores lógicos también.

Por ejemplo:

Operando 1	Operando 2	Expresión lógica	R1	R2	Resultado
$(4 > 3)$	$(4 \neq 2)$	$((4 > 3) \wedge (4 \neq 2))$	V	V	V
$(7 \leq 8)$	$(2 > 3)$	$((7 \leq 8) \wedge (2 > 3))$	V	F	F
$\sim (11 \geq 9)$	-	$\sim (11 \geq 9)$	F	-	F
$\sim (61 \leq 43)$	$(76 = 3)$	$(\sim (61 \leq 43) \vee (76 = 3))$	V	F	V
$(9 = 1)$	$(9 > 9)$	$((9 = 1) \vee (9 > 9))$	F	F	F

2.5.4 Evaluación de expresiones

Para la evaluación de expresiones es necesario definir primero la precedencia de operadores aritméticos y lógicos, la cual consiste en la prioridad y orden en que se realizarán las operaciones contenidas en la expresión. A continuación se especifican:

Operadores matemáticos	Operadores lógicos
( ), [ ], { }	$\sim$
$\wedge$	$> < \geq \leq$
$\cdot, /, \%$	$= \neq$
$+, -$	*and*, *or*

La tabla anterior indica que las operaciones delimitadas por corchetes, paréntesis o llaves se realizarán primero que las que no están contenidas en estos operadores. El circunflejo (*operador de la potencia*) junto con las expresiones que tengan operadores relacionales tienen la siguiente prioridad en la tabla. Luego las operaciones de multiplicación y división, seguidas de los operadores de comparación de igualdad o desigualdad. Los operadores suma y resta tienen la prioridad más baja en los operadores matemáticos. Finalmente, las expresiones con los operadores lógicos *and* y *or* son evaluadas en último lugar.

Ejemplos

Si los valores de las variables a, b y c son 5, 3 y 1, respectivamente, y la variable sw tiene valor inicial falso, entonces

- $(a \leq b) \wedge (b \geq c)$
  $(5 \leq 3) \wedge (3 \geq 1)$
  $(falso) \wedge (verdadero)$
  $falso$

  Comentarios: De acuerdo con el valor de las variables, en la evaluación de condiciones se obtiene *falso* para la primera condición y *verdadero* para la segunda condición. Teniendo en cuenta los valores de la tabla de verdad del operador de conjunción, al evaluar el valor resultante es falso.

- $sw2 \longleftarrow (c \leq b)$
  $sw2 \longleftarrow (1 \leq 3)$
  $sw2 \longleftarrow verdadero$

  Comentarios: La variable $sw2$ ha sido declarada de tipo *booleano* y se le asigna el valor resultante de la operación lógica entre c y b con el operador menor o igual. De acuerdo con el valor de las variables, en la evaluación de condiciones se obtiene *verdadero* para esta condición.

- $(2b = a + c)$
  $(2 \cdot 3 = 5 + 1)$
  $(6 = 6)$
  $verdadero$

  Comentarios: De acuerdo con el valor de las variables, en la evaluación de condiciones, en esta operación lógica con operador de igualdad se obtiene *verdadero* luego de la evaluación de la condición.

- $\sim sw$
  $\sim falso$
  $verdadero$

Comentarios: En este ejemplo se niega el valor de la variable sw. De acuerdo con el valor ya definido de la variable, luego de la evaluación de la condición, se obtiene *verdadero*.

- $sw2 \longleftarrow (\sim sw) \wedge (5 > 3)$
  $sw2 \longleftarrow (\sim falso) \wedge (verdadero)$
  $sw2 \longleftarrow verdadero \wedge verdadero$
  $sw2 \longleftarrow verdadero$

  Comentarios: A la variable $sw2$ se le está asignando el resultado de la operación lógica de la conjunción que involucra a la negación de la variable sw y a la comparación con el operador mayor. De acuerdo con el valor de las variables, en la evaluación de condiciones se obtiene *verdadero* para la ambas condiciones, por lo tanto, el valor resultante es verdadero.

- $sw2 \longleftarrow (a > b)$
  $sw2 \longleftarrow (5 > 3)$
  $sw2 \longleftarrow verdadero$

  Comentarios: A la variable $sw2$ se le asigna el resultado de la operación lógica de la comparación con el operador mayor de a y b. De acuerdo con el valor de las variables, en la evaluación de condiciones se obtiene *verdadero* para esta condición:

- $sw2 \longleftarrow \sim ((a > b) \vee (b > c))$
  $sw2 \longleftarrow \sim ((5 > 3) \vee (3 > 1))$
  $sw2 \longleftarrow \sim ((verdadero) \vee (verdadero))$
  $sw2 \longleftarrow \sim (verdadero)$
  $sw2 \longleftarrow falso$

  Comentarios: A la variable $sw2$ se le asigna el resultado de la operación lógica de la negación de la disyunción de dos comparaciones: la primera entre a y b con el operador mayor, y la segunda entre b y c con el operador mayor. De acuerdo con el valor de las variables, en la evaluación de condiciones se obtiene *verdadero*; luego de la negación el valor asignado a la variable $sw2$ es *falso*.

- $x_1 \longleftarrow \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$
  $x_1 \longleftarrow \dfrac{-5 + \sqrt{5^2 - 4 \cdot 5 \cdot 1}}{2 \cdot 5}$
  $x_1 \longleftarrow \dfrac{-5 + \sqrt{25 - 20}}{10}$
  $x_1 \longleftarrow \dfrac{-5 + \sqrt{5}}{10}$

  Comentarios: A la variable x_1 se le asigna el resultado de la operación matemática correspondiente a la fórmula general para hallar las raíces soluciones

de una ecuación de segundo grado. De acuerdo con la precedencia de operadores, se efectúan las operaciones y se verifica que el discriminante es positivo y, por lo tanto, la raíz que se calculó es real.

- $x_1 \longleftarrow \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$

  $x_1 \longleftarrow \dfrac{-5 - \sqrt{5^2 - 4 \cdot 5 \cdot 1}}{2 \cdot 5}$

  $x_1 \longleftarrow \dfrac{-5 - \sqrt{25 - 20}}{10}$

  $x_1 \longleftarrow \dfrac{-5 - \sqrt{5}}{10}$

  Comentarios: A la variable x_1 se le asigna el resultado de una operación matemática en la que se incluye una raíz cuadrada. Se evalúa el contenido de la raíz, y su valor es positivo, por lo tanto queda una división de b entre un número real.

Evalúe las dos últimas expresiones aritméticas con los siguientes conjuntos de valores:

- $(a = 2, b = 4, c = 9)$
- $(a = 5, b = 4, c = 10)$

Establezca las diferencias de las siguientes expresiones con la expresión aritmética usada en el punto anterior:

- $x_1 \longleftarrow \dfrac{-b + \sqrt{(b^2 - 4) \cdot ac}}{2a}$
- $x_1 \longleftarrow \dfrac{-b}{2a} + \frac{\sqrt{b^2 - 4ac}}{2a}$

2.6 Ejercicios propuestos

1. Realice las conversiones necesarias aplicando las equivalencias de múltiplos y submúltiplos del Byte, según se indique:
   a) ¿Cuántos bytes hay en 2378 gigabytes?
   b) ¿Cuántas doblepalabras hay en 48 terabytes?
   c) ¿Cuántos nibbles hay en 67 kilopalabras?
   d) ¿Cuántos discos compactos (CD) son necesarios para guardar 15 gigabytes? ¿Cuánto espacio no se utiliza luego de guardar la información?
   e) ¿Cuántos kilobytes hay en 65 900 megapalabras? ¿Cuántos megabytes?

f) Se tienen dos discos duros, uno con capacidad de almacenamiento de 80 GB y otro con capacidad de almacenamiento de 0.078125 TB, ¿cuál de los dos almacenará más cantidad de palabras?

g) ¿Cuántos bits tiene un conjunto de memorias RAM (memoria principal) si su capacidad conjunta es de 4.5 GB?¿Cuántas megapalabras hay en este conjunto?

h) En una compañía se desea almacenar 870 400 MB de información para uso posterior. Se dispone de los siguientes dispositivos de almacenamiento:

- 4 discos duros de 250 GB de capacidad cada uno.
- 50 CD-ROM de 700 MB cada uno.
- 25 DVD-ROM de 4.7 GB cada uno.
- 100 disquetes de 1.44 MB cada uno.

Debe escoger la mejor combinación (*no desaprovechar capacidad de almacenamiento*) de dispositivos para guardar la información. No se tendrá en cuenta la confiabilidad ni el tiempo de vida útil de los dispositivos.

2. Evalúe paso a paso las siguientes expresiones, dando valores aleatorios a cada una de las variables. En caso de expresiones lógicas deberá obtener FALSO o VERDADERO. Tenga en cuenta la precedencia de los operadores y establezca diferencias donde se indique:

a) $S^2 = \sqrt{\dfrac{(X-m)^2}{n-1}}$

b) $S^2 = \sqrt{\dfrac{X^2-m^2}{n-1}}$ Compare con la expresión anterior.

c) $P = \dfrac{-(y^3-1)}{(y+1)-\sqrt{y+1}}$

d) $Z = \dfrac{x(x^2+1)^3}{\sqrt{2x}+1}$

e) $T = 1 - \dfrac{\sqrt[n]{x-2}}{x^3}$

f) $sw \longleftarrow ((x \neq y) \wedge (x \leq y))$

g) $sw \longleftarrow ((a \geq b) \vee (b \geq c))$

h) Para cada caso indique los errores que existen en la declaración y/o asignación de variables:

1) **Booleano** : $sw \longleftarrow verdadero$
2) **Real** : $valor \longleftarrow verdadero + 267,4$
3) **Real** : $nota \longleftarrow 3,2$
4) **Entero** : $cont2 \longleftarrow 3,2 + 1,5$
5) **Entero** : 6754626
6) **Entero** : $166numero$
7) **Real** : $sum@3445$
8) **Entero** : $7filcol9$
9) **Entero** : $suma \longleftarrow A$
10) **Real** : $promedio \longleftarrow 98.54331$