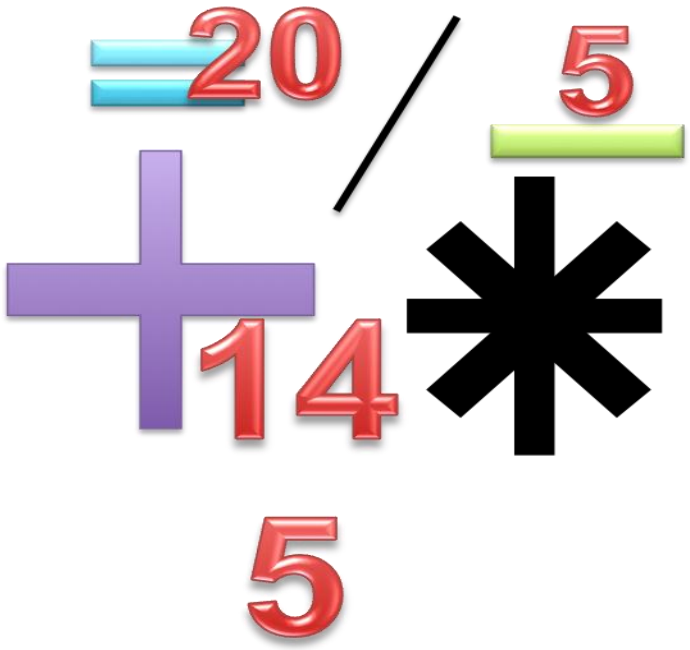
UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD III. DATOS Y ENTIDADES PRIMITIVAS

CONTENIDO:



UNIDAD III

DATOS Y ENTIDADES PRIMITIVAS

CONCEPTO Y DIFERENCIA ENTRE DATO E INFORMACIÓN

Dato: Es la expresión general que describe un objeto, hecho, situación o valor. Puede estar representado por letras del alfabeto, números, símbolos, entre otros. Ejemplo: el nombre de una persona, la calificación de un estudiante, el sueldo de un trabajador.

Información: Es una colección de datos relacionados que ofrecen un significado, idea, conocimiento o conclusiones.

Los datos por sí solos no tienen la capacidad de comunicar un significado, a diferencia de la información que se basa en un conjunto de datos que al procesarse proporcionan un significado, propósito y utilidad.

Los algoritmos y los programas correspondientes operan sobre datos. Para ello, el computador maneja varios tipos de datos que a continuación se detallan.

Tipos de datos: Los tipos de datos simples que maneja el computador son:

- Numéricos
- Lógicos
- Alfanuméricos o carácter

Datos numéricos: es el conjunto de valores numéricos. Pueden representarse de dos formas:

- **Numérico entero:** es un subconjunto finito de los números enteros, no tienen componentes fraccionarios o decimales y pueden ser positivos o negativos .

  Ejemplos:

  8 5 15 -9 2000 -543
- **Numérico real:** es un subconjunto de los números reales. Tienen una parte decimal y pueden ser positivos o negativos.

Ejemplos:

23435454 -35.75 0.00056 -6.86 8.0

Datos Lógicos: Es un dato que sólo puede tomar uno de dos valores: verdadero o falso. También es denominado booleano.

Datos alfanuméricos, carácter o de tipo cadena: Es el conjunto finito y ordenado de caracteres que la computadora reconoce.

Ejemplos:

- Caracteres alfabéticos: A,B, C, D….,Z, a,b,c,d,…..,z.
- Caracteres numéricos: 1,2,3,4,5,6,7,8,9,0.
- Caracteres especiales: Son todos los caracteres que no son letras ni números. Ejemplo: ¿,¡,<,>,+,*,/,(,),%,&,$,”,!,?,[,], ^…….

LOS OPERADORES: CONCEPTO Y TIPOS

Los operadores definen las operaciones que van a realizarse con los datos u operandos.

Tipos de operadores:

Existen varios tipos de operadores: aritméticos, relacionales, lógicos y de asignación.

Operadores Aritméticos: son análogos a los operadores matemáticos para realizar operaciones aritméticas como suma, resta, multiplicación, división y exponenciación, tal como se muestra en el cuadro 3.1.

Cuadro 3.1 Operadores aritméticos

Operador	Significado	Tipos de operandos
^, **	Exponenciación	Entero o real
+	Suma	Entero o real
-	Resta	Entero o real
*	Multiplicación	Entero o real
/	División	Real
Div	División Entera	Entero
Mod	Módulo (resto)	Entero

Nota: *no todos los lenguajes de programación tienen los operadores de exponenciación, división entera y resto.*

Ejemplos de operadores en representación matemática y algorítmica

Representación matemática	Representación algorítmica
6 x 5	6 * 5
10 ÷ 2 $\frac{10}{2}$	10 / 2
2^5	2 ^ 5
2 + 3	2 + 3
5 – 6	5 – 6

Operadores de Relación: Permiten realizar comparaciones de valores de tipo numérico o carácter. Los operadores de relación sirven para expresar condiciones en los algoritmos. En el cuadro 3.2 se presenta su símbolo, significado y un ejemplo.

Cuadro 3.2 Operadores de relación

Operador	Significado	Ejemplo
>	Mayor que	a > b
<	Menor que	b < a
>=	Mayor o igual que	a >= b
<=	Menor o igual que	b <= a
=	Igual que	a = b
< >	Distinto de	a < > b

Operadores lógicos o booleanos: Corresponden a los operadores de negación, disyunción y conjunción, tal como se muestra en el cuadro 3.3

Cuadro 3.3 Operadores lógicos

Operador lógico	Expresión lógica	Significado
no (not)	no p (not p)	Negación de p
y (and)	p y q (p and q)	Conjunción de p y q
o (or)	p o q (p or q)	Disyunción de p y q

Nota: *p y q pueden ser expresiones con operadores relacionales. Los operadores de la conjunción y la disyunción requieren dos expresiones con operadores relacionales mientras que la negación solo una.*

Ejemplos:

a) 3>5 and 4<6
b) 5<=5 or 9>8
c) 7 < > 9
d) not 2 = 2

LA EXPRESIÓN: CONCEPTO, TIPOS Y EVALUACIÓN DE EXPRESIONES.

Una expresión consta de operandos y operadores. Se clasifican en:

- Expresiones aritméticas

- Expresiones lógicas (booleanas)
- Expresiones alfanuméricas o de carácter.

Expresiones Aritméticas: dan como resultado un valor numérico y está formada por operadores aritméticos. A continuación se presentan algunos ejemplos:

Ejemplo de expresión aritmética	Resultado
10/4	2.5
3 * 5	15
10 div 4	2
2 ^ 4	16
5 mod 2	1
10 mod 2	0
5 + 3	8
5 – 8	-3

Las expresiones que tienen dos o más operandos requieren unas reglas matemáticas que permiten determinar el orden de las operaciones. Estas reglas se denominan reglas de prioridad y se muestran en el cuadro 3.4.

Cuadro 3.4 Prioridad de los operadores aritméticos

Prioridad	Operador
1	( )
2	^ , **
3	*, /
4	div , mod
5	+ , -

Como puede observarse en el cuadro anterior, primero se evalúan las operaciones que están encerradas entre paréntesis. Si existen paréntesis anidados, es decir, unos dentro de otros, se evalúan primero las expresiones de los paréntesis más internos. El orden de prioridad de las operaciones son las exponenciaciones, multiplicaciones y divisiones, divisiones enteras y restos, y por último las sumas y restas. En caso de coincidir varios operadores de igual prioridad en una expresión, el orden de prioridad en este caso es de izquierda a derecha.

Ejemplos:

a) 5 + 2 * 7 - 3

5 + 2 * 7 – 3

5 + 14 – 3

19 – 3

16

b) 3 + 9 * 2 – 4 * 5

3 + 9 * 2 – 4 * 5

3 + 18 – 4 * 5

3 + 18 – 20

21 – 20

1

c) 1 – 5 * 2 ^ 3 * 2 / 8

1 - 5 * 2 ^ 3 * 2 / 8

1 – 5 * 8 * 2 / 8

1 – 40 * 2 / 8

1 – 80 / 8

1 – 10

– 9

d) 3 + 5 * 4 / 2 * (5 – 3)

3 + 5 * 4 / 2 * (5 – 3)

3 + 5 * 4 / 2 * 2

3 + 20 / 2 * 2

3 + 10 * 2

3 + 20

23

e) 3 + 5 * (4 / 2) * 5 – 3

$3 + 5 * (4 / 2) * 5 - 3$

$3 + 5 * 2 * 5 - 3$

$3 + 10 * 5 - 3$

$3 + 50 - 3$

$53 - 3$

50

f) 5 + ((8 – 2 – 4) / 2) ^ 5 – 12

$5 + ((8 - 2 - 4) / 2) \wedge 5 - 12$

$5 + (2 / 2) \wedge 5 - 12$

$5 + 1 \wedge 5 - 12$

$5 + 1 - 12$

$6 - 12$

-6

Expresiones lógicas: Son expresiones que dan como resultado un valor lógico: verdadero o falso. Se forman combinando operadores relacionales y operadores lógicos.

Ejemplos:

Expresión lógica	Resultado
5 >= 6	Falso
8 < 10	Verdadero
7 <= 7	Verdadero
9 < > 9	Falso
12 > 10	Verdadero
6 = 6	Verdadero

Para evaluar operadores lógicos en una expresión, se debe utilizar las tablas de verdad que se presentan a continuación.

Cuadro 3.5 Tablas de verdad

Tabla de la conjunción		
a	**b**	**a and b**
V	**F**	**F**
F	**V**	**F**
V	**V**	**V**
F	**F**	**F**

Tabla de la disyunción		
a	**b**	**a or b**
V	**F**	**V**
F	**V**	**V**
V	**V**	**V**
F	**F**	**F**

Tabla de la negación	
a	**not a**
V	**F**
F	**V**

Ejemplos:

a) 3 > 4 **and** 5 <= 5
F **and** V
V

b) 15 < > 51 **or** -3 > -2
V **or** F
V

c) 7 = 8 **or** 3 <=0
F **or** F
F

d) **not** 6 = 6

not V

F

Si la expresión tiene varios operadores lógicos se debe cumplir con el siguiente orden de prioridad:

Cuadro 3.6 Prioridad de los operadores lógicos

Prioridad	Operador
1	**not**
2	**and**
3	**or**

Ejemplos:

a) 1 > = 5 or 3 = 8 and 6 < 9 or 7 > 7
F or F and V or F
F or F or F
F or F
F

b) 2>=2 or not 4<12 and 5 < > 6 and 8<=10
V or not V and V and V
V or F and V and V
V or F and V
V or F
V

Resumiendo en una sola tabla las prioridades de todos los operadores, quedaría de la siguiente manera:

Cuadro 3.7 Prioridad de los operadores

Prioridad	Operador
1	( )
2	^, **
3	*, /
4	div, mod
5	+, -
6	<, >, <=, >=, =, < >
7	not
8	and
9	or

Ejemplos:

a) 5 – 3 ^ 3 > 0 or 5 < > 6 and not 7>=7
5 – 27 > 0 or 5 < > 6 and not 7>=7
5 – 27 > 0 or 5 < > 6 and not 7>=7
-22>0 or 5 < > 6 and not 7>=7
F or V and not V
F or V and F
F or F
F

b) 7 + 8 / 2 = 11 and (5< 5 or 6>=6)
7 + 8 / 2 = 11 and (F or V)
7 + 8 / 2 = 11 and V
7 + 4 = 11 and V
11=11 and V
V and V
V

LOS IDENTIFICADORES O VARIABLES

Una variable es una localización de la memoria del computador donde se almacena un dato cuyo valor puede variar. El nombre de la variable es un nombre simbólico (identificador) que se debe relacionar con su contenido. Por ejemplo si la variable guarda la edad de una persona, el identificador podría ser *edad.* De esta manera facilita la comprensión de los algoritmos y programas.

Reglas de escritura de los identificadores:

Las reglas de escritura para los nombres de las variables depende del lenguaje de programación a utilizar, sin embargo, a continuación se presentan las reglas básicas de la mayoría de los lenguajes:

- Debe comenzar por una letra
- Puede tener letras y números
- No puede tener espacios en blanco
- No puede tener caracteres especiales (caracteres que no son alfabéticos ni numéricos), a excepción del subrayado (_). Las vocales acentuadas y la letra ñ son consideradas caracteres especiales.

Ejemplos de identificadores válidos:

Nombre

Edad

Sueldo

Calificacion

Precio

impuesto

fecha_de_nacimiento

Ejemplos de identificadores no válidos:

Teléfono *tiene una vocal acentuada*

Fecha de nacimiento *espacios en blanco*

Nºtelefono *tiene el carácter especial º*

1er_nombre *comienza por un número*

Años_servicio *tiene el carácter especial ñ*

Impuesto% *tiene el carácter especial %*

Clasificación de variables según su contenido:

De acuerdo al contenido o valor que guardan las variables, éstas se clasifican en:

- **Numéricas:** almacenan valores numéricos: enteros o reales. **Ejemplo:** edad, sueldo, pago, num_hijos, cantidad_habitantes.
- **Lógicas:** almacenan un valor lógico: verdadero o falso.
- **Alfanuméricas:** almacenan letras, números y/o caracteres especiales. **Ejemplo:** nombre, direccion, correo, cedula, fecha_nacimiento.

Asignación de valor a una variable:

Para asignarle o atribuirle un valor a una variable se utiliza el operador de asignación que tenga el lenguaje de programación. Algunos de estos operadores son: ← , **=**, **:=** y el orden es como a continuación se presenta:

Variable ← valor

Variable = valor

Variable := valor

Para los ejemplos se utilizará el operador **=**

Ejemplos:

Edad= 18

Monto=5000

dia=”lunes”

Nota: *para las variables alfanuméricas se coloca el valor a asignar entre comillas simples o dobles.*

Clasificación de las variables según su uso:

Las variables se pueden utilizar como un contador o un acumulador:

Contador: Es una variable cuyo valor se incrementa o decrementa en una cantidad fija cada vez que se ejecuta una determinada acción. Se utiliza para contar el número de veces que se ejecuta un suceso o acción.

Representación:

Nombre_del_contador=Nombre_del_contador + valor_fijo

Ejemplos de contadores:

contador=contador + 1

x=x + 5

cont=cont – 1

Acumulador: Es una variable que se incrementa en cantidades que varían. Se utiliza para sumar cantidades que están almacenadas en una variable y cuyo valor varía en cada suceso o acción.

Representación:

Nombre_del_acumulador=Nombre_del_acumulador + nombre_de_la_variable

Ejemplos:

a) acumsueldos=acumsueldos+sueldos

b) sumaedad=sumaedad+edad

c) total=total + monto

Constantes: a diferencia de las variables, su valor nunca varía, ni puede ser modificado a lo largo del programa o algoritmo, por lo tanto, su valor hay que definirlo y no puede cambiar a lo largo de la ejecución.

Ejemplos:

a) pi=3.14

b) N=10

c) iva=12

EJERCICIOS RESUELTOS

Evaluación de expresiones aritméticas:

a) K=3+5*2^3/10-3

K=3+5*8/10-3

K=3+40/10-3

K=3+4-3

K=7-3

K=4

b) L= 5*4/2*3/5*8

L= 20/2*3/5*8

L= 10*3/5*8

L= 30/5*8

L= 6*8

L= 48

c) M= 2*9+ 2^(4*2-5) - 30

M= 2*9+ 2^(8-5) - 30

M= 2*9+ 2^3 - 30

M= 2*9+ 8 - 30

M= 18+ 8 - 30

M= 26 - 30

M= -4

Evaluación de expresiones relacionales y lógicas:

d) 5>=5 or 6<6 and -3>-1

V or F and F

V or F

V

e) 8< >9 and (15<15 or not 3>=3)

8< >9 and (F or not V)

8< >9 and (F or F)

8< >9 and F

V and F

F

f) (-1)^70 = 1 and not 40^0 =1 or 3*4/6 < 2

1 = 1 and not 40^0 =1 or 3*4/6 < 2

1 = 1 and not 1=1 or 3*4/6 < 2

1 = 1 and not 1=1 or 12/6 < 2

1 = 1 and not 1=1 or 2 < 2

1 = 1 and not 1=1 or 2 < 2

1 = 1 and not 1=1 or 2 < 2

V and not V or F

V and F or F

F or F

F

REFERENCIAS BIBLIOGRÁFICAS

Bassard, G y Bratley, P. (2010). Fundamentos de algoritmia. Prentice-Hall.

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.