GUÍA DE APRENDIZAJE TECNOLOGÍA
GRADO 9º - TRIMESTRE III
COLEGIO CIUDADELA EDUCATIVA DE BOSA INSTITUCIÓN EDUCATIVA DISTRITAL

ALCALDIA MAYOR DE BOGOTA D.C.

DESEMPEÑO DE COMPRENSION:

- Los estudiantes implementan diagramas de bloques aplicando los conceptos de programación, para lograr la resolución de problemas cotidianos. Así mismo, plantea algoritmos mediante diagramas de flujo para identificar y comprender una secuencia lógica de un proceso industrial. El proceso se consolidará mediante el uso de makecode y microbit, como recursos dinámicos para la comprensión de la electrónica aplicada a problemas del contexto. Finalmente se aplicarán los conocimientos del año escolar en el desarrollo de un proyecto tecnológico, logrando socializar con la comunidad educativa la trascendencia de la electrónica en las dinámicas cotidianas.

PREGUNTA PROBLEMATIZADORA:

¿Cómo se pueden resolver problemas sencillos usando un lenguaje de programación?

TOPICOS GENERATIVOS:

- Fundamentos de lógica de programación

UNIDAD DIDÁCTICA 3 PERIODO[1]

ALGUNOS CONCEPTOS IMPORTANTES SOBRE PROGRAMACIÓN

Dato

Un dato es una representación simbólica (un número, letra, gráfico, entre otros) de una característica de un elemento u objeto. En este sentido, se afirma que un dato puede estar representado por una cifra, letra, palabra o conjunto de palabras que describen una característica (atributo o propiedad) del elemento. Por ejemplo, "Gabriela Herrera" y "20" pueden representar el nombre y la edad de una persona respectivamente, donde el nombre y la edad son las características de esa persona.

Es necesario aclarar: un solo dato no representa información, sino que la información está conformada por un conjunto de datos que, al ser procesados mediante algún mecanismo, constituyen un mensaje para incrementar el conocimiento de quien lo recibe. Esto significa que solo los seres humanos o sistemas de cómputo muy avanzados como los sistemas expertos procesan información.

Tipos de dato e identificadores

Un tipo de dato corresponde a una clasificación que se hace para poder tratar cada dato de la forma más adecuada, según lo que se requiera. El tipo de dato le indica al dispositivo de procesamiento cuanto espacio de memoria debe reservar para almacenar el dato, es decir, para determinar el tamaño del espacio de memoria. Los tipos de datos más comunes y que se trabajarán en este libro son: alfanuméricos, numéricos y lógicos.

Alfanuméricos

Estos datos se componen de la combinación de letras, dígitos y caracteres especiales. Se dividen en caracteres y cadenas

Carácter: Son datos que solo tienen un carácter, que puede ser una letra del alfabeto, un dígito del 0 al 9 o un carácter especial. Por lo general van encerrados en comillas simples.

Por ejemplo: 'a', 'R', '2','#','@'

Los dígitos que son tratados como caracteres son diferentes a los valores numéricos, por lo tanto, no se deben realizar operaciones aritméticas con ellos.

Cadena: Son datos que están compuestos por un conjunto de letras del alfabeto, dígitos y caracteres especiales. Se deben encerrar entre comillas dobles

---

[1] **Las actividades e información que se presentará en la siguiente unidad didáctica fueron tomados en su mayoría del texto: Introducción a la lógica de programación que fue escrito por Jorge Orlando Herrera Morales, Julián Esteban Gutiérrez Posada y Robinson Pulgarín Giraldo. Editorial EIIZCOM. EL texto se puede distribuir bajo licencia Creative Commons: Atribución-No comercial- Sin derivadas.**

Son ejemplos de datos de tipo cadena los siguientes: “Carrera 17 # 12 – 65”, “Gato”, “Jacobo Herrera”, “75478923”, “01800043433”.

Numéricos

Corresponden a los datos que están compuestos por solo números y signos (positivo y negativo), es decir, dígitos del 0 al 9, con los cuales se pueden realizar operaciones aritméticas. Estos tipos de datos se subdividen en: Enteros y Reales.

Entero: Son aquellos datos compuestos por números que no tienen punto decimal. Pueden ser números positivos, negativos o el cero.

Ejemplos de este tipo de datos pueden ser: 20, -5, 200, 1500000.

Real: Son datos con componente decimal, pueden ser positivos, negativos o cero.

Ejemplos: 1.75, 4.5, 1800000.00, -234.00.

Lógicos o booleanos

Son aquellos datos que toman solo uno de los dos posibles valores booleanos4: Verdadero o Falso. Estos valores son equivalentes a los dígitos del sistema binario: 1 corresponde a Verdadero y 0 a Falso

Un ejemplo es la necesidad de almacenar información sobre el contagio de un grupo de personas. Solo hay dos posibilidades: Contagio positivo o contagio negativo (verdadero o falso en contagio).

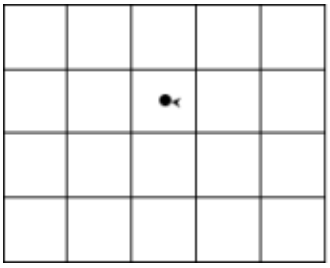
Identificadores

Un identificador es el nombre que se le asigna a las variables, constantes, funciones, procedimientos y al algoritmo; esto se hace para que el algoritmo pueda identificar claramente cada uno de estos elementos.

Variables y constantes

Una variable es una posición o espacio de memoria en el cual se almacena un dato. Su valor puede cambiar en cualquier momento de la ejecución del algoritmo, precisamente por eso recibe el nombre de variable.

Para almacenar los datos en un dispositivo de procesamiento de datos o computador, se utiliza la memoria de este, la cual se puede comparar con un conjunto de cuadritos que guardan valores. Cada “cuadrito” o celda representa una dirección física dentro de la memoria de la maquina a la cual se le puede asignar un nombre mediante un identificador.



Para trabajar con variables, se deben tener presentes los siguientes elementos:

- *Tipo*
- *Nombre o identificador*
- *Contenido*

El *tipo* se refiere al tipo de dato que va a almacenar. Puede ser uno de estos cinco: **Carácter, Cadena, Entero, Real o Lógico**. *El nombre o identificador* de la variable, corresponde al mecanismo con el que se referencia el espacio o posición de memoria en el cual se almacenará el dato. Mientras que el *contenido*, hace referencia al valor que almacena, el cual depende del tipo de dato que se haya definido.

Declaración de variables y asignación de datos

Cuando en un algoritmo se requiera utilizar una variable, esta debe ser declarada. Declarar una variable quiere decir que se va a reservar un espacio de memoria, el cual tendrá un nombre y un tipo de dato.

Almacenar un dato en una variable, se puede hacer de dos maneras:

La primera forma es leyendo el dato, proveniente desde el exterior del algoritmo, el cual lo proporciona el usuario; por ejemplo, cuando un cajero automático le solicita la clave de su tarjeta, el usuario le está asignando un valor a la variable donde se almacenará dicha clave, el valor será la clave que se digitó.

La segunda forma se hace a través de una expresión de asignación. Una expresión de asignación, es el mecanismo por medio del cual una variable o una constante toman un valor. Para realizar una asignación se utiliza el signo igual (=).

```
variable = valor
```

Un ejemplo:

```
cedula    = "41459822"
telefono  = "7454548"
salario   = 3000000.00
edad      = 24
esFumador = Verdadero
estratoSocioeconomico = '3'
```

Ahora… Algunas preguntas

1) Teniendo en cuenta que la forma general para declarar variables es la siguiente:

```
Tipo variable
```

¿Cómo debería declararse las siguientes variables? (escribe únicamente el tipo de variable: cadena, carácter, entero, real, booleano)

_____ cedula
_____ teléfono
_____ salario
_____ edad
_____ estrato Socioeconómico
_____ esFumador

2) Describe cómo se realizó el proceso de asignación en el ejercicio que se muestra a continuación. Describe cómo van cambiando los valores de las variables en cada línea. ¿Cuál es el valor final de la variable 'a'?

```
1  a = 5
2  b = a
3  c = 3 * b
4  a = c - b
```

3) ¿Qué valores quedan almacenados en las variables a, b y c?

```
a <-  10
b <-  20
c <-  5
a <-  a+3
b <-  b+4-a
c <-  a+b+c
a <-  a+c
b <-  4
c <-  c+3-b+2
```

4) ¿Qué valores quedan almacenados en las variables a, b, c y d?

```
a <-  5
b <-  18
c <-  15
d  <-   25
a <-  a+10
b <-  b+5-c
c <-  c+4+b
d <-  d+b+a
a <-  a+1
b <-  b+c
c <-  b+c
d <-  b+b
```

5) ¿Qué valores quedan almacenados en las variables a, b, c y d?

```
a <-  8
b <-  7
c <-  5
d <-  8
a <-  a+b-c+d
b <-  a+b-c+d
c <-  a+b-c+d
d <-  a+b-c+d
a <-  a+b-c+d
b <-  a+b-c+d
c <-  a+b-c+d
d <-   a+b-c+d
```

6) ¿Qué valores quedan almacenados en las variables a, b y c?

```
a <- 10
b <- 5
c <- 10
a  <-  a+b-5
b  <-  a+b-5
c  <-  a+b-5
a <- a+5*b/2
b <- a+5*b/2
c <- a+5*b/2
```

7) ¿Qué valores quedan almacenados en las variables a, b y c?

```
a <- 1
b <- 2
c <- 3
a  <-  a+2
b <- a+2+b
c <- a+2+c
a  <-  a/2
b  <-  b/2
c <- c/2
```

8) Con la información de la lectura de la sesión realiza un mapa conceptual (recuerda que un mapa conceptual no contiene párrafos, debe asociar los conceptos expresados en las palabras que sean más importantes).

¿Quieres saber más? Puedes visitar… http://mafebelltranpm.blogspot.com/p/ejercicios-de-asignacion.html

SESIÓN 2

Operadores y expresiones

Un operador es un símbolo que permite realizar una operación con números o con datos que se encuentran almacenados en las variables y constantes. En lógica de programación, existen 3 tipos de operadores: aritméticos, relacionales y lógicos.

Por su parte, una expresión es una instrucción que puede estar compuesta por operadores, variables, constantes y números, que generalmente produce un resultado, ya sea numérico o lógico.

Operadores Aritméticos

Se utilizan para realizar operaciones aritméticas entre datos de tipo entero o real, su resultado es de tipo numérico. Los operadores aritméticos son los siguientes:

+ Suma

- Resta

*** Multiplicación o producto**

/ División real o entera

% Módulo o Resto de la división entera

ˆ Potenciación, este símbolo tiene el nombre de circunflejo.

Operadores Relacionales

Estos operadores se utilizan para escribir expresiones relacionales o de comparación, las cuales producen un resultado lógico o booleano: Verdadero o Falso.

Los operadores relacionales son los siguientes:

< Menor que.

> Mayor que.

<= Menor o igual a.

>= Mayor o igual a.

! = Diferente de.

== Igual a.

Operadores Lógicos

Estos operadores se utilizan para crear expresiones lógicas o booleanas cuyo resultado es de tipo lógico: **Verdadero o Falso.**

Los operadores lógicos son los siguientes:

- Y Conjunción.
- Disyunción.
- NO Negación.

Estos operadores funcionan con datos de tipo lógico; para obtener el resultado de la aplicación de estos operadores, es indispensable conocer cómo funcionan las tablas de verdad de cada uno de ellos.

Operador Y, denominado Conjunción. Es un operador binario, es decir, requiere de dos operando para producir un resultado. *El resultado es verdadero solo cuando ambos operando son verdaderos*

*Operando*1	*Operando*2	*Operando*1 **Y** *Operando*2
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

Operador O, denominado Disyunción. Igual que el anterior, es un operador binario. Su resultado ser ‘a verdadero cuando al menos uno de los dos operando tenga ese valor

*Operando*1	*Operando*2	*Operando*1 **O** *Operando*2
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Operador NO, denominado Negación. A diferencia de los dos anteriores, este es un operador unario, es decir, requiere de un solo operando para producir su resultado. Si el operando es Verdadero, cambia su estado a Falso, y viceversa. En conclusión, su función es cambiar el valor o estado lógico de su único operando

*Operando*1	**NO** *Operando*1
Verdadero	Falso
Falso	Verdadero

Expresiones aritméticas y notación algorítmica

En estas expresiones intervienen variables, constantes, números y operadores aritméticos, así como los paréntesis. La expresión entrega un resultado de tipo numérico luego de ser calculada. Estos son algunos ejemplos de expresiones aritméticas

Ejemplo	**Explicación**	**Resultado**
3 + 5	Suma 3 con 5	8
2 - 8	Resta 8 de 2	-6
7 * 6	Multiplica 7 por 6	42
20.0 / 3.0	Divide 20.0 entre 3.0 (Real)	6.33
20 / 3	Divide 20 entre 3 (Entero)	6
2^3	Eleva 2 a la potencia 3	8
20%3	Resto de 20 entre 3	2

Las expresiones aritméticas también pueden ser construidas con variables o en combinación con constantes.

Expresión	Explicación
lado1 + lado2	Se suman los valores almacenados en las variables lado1 y lado2.
a - b	Al valor de la variable *a* se le resta el valor de la variable b.
*base * altura*	Los valores almacenados en las variables base y altura son multiplicados entre sí.
a / b	El valor almacenado en la variable a es dividido entre el valor de la variable b. El resultado será entero, si ambas variables fueron declaradas de tipo Entero, en otro caso, será Real (con decimales).
x^2	El valor almacenado en la variable x es elevado a la 2.
25 % divisor	25 es dividido entre el valor de la variable divisor, el resultado será el resto de la división. La variable divisor deberá ser declarada de tipo Entero.

Cuando una expresión aritmética involucra varios operadores, es necesario realizar los cálculos respetando la precedencia de los mismos, es decir, se debe tener en cuenta lo que se denomina prioridad en las operaciones. La Tabla 1.6 muestra el orden de ejecución, la cual puede ser modificada con el uso de paréntesis, los cuales serían la máxima prioridad.

Orden	Operador
1	( )
2	^
3	*, /, %
4	+, -

Ejemplos:

Encuentra el resultado de la siguiente expresión aritmética

```
10 + 3 * 5
```

Esta expresión presenta dos operaciones: una suma y una multiplicación. La prioridad la tiene el operador, por lo tanto, debe ser la primera operación que se realiza:

```
10 + 15
```

Al realizar la suma, el resultado es 25

Si en el anterior ejemplo, no se hubiese tenido en cuenta el orden de ejecución, de acuerdo a la prioridad de operación, sino que se hubiese ejecutado en el orden como está escrita la expresión, se obtendría el siguiente resultado, que obviamente es un error

```
10 +  3 * 5
13 * 5
65
```

Si en una expresión se encuentran 2 o más operadores consecutivos de igual jerarquía o precedencia, las operaciones se van realizando de izquierda a derecha. Por ejemplo:

```
10 - 6 + 2
4 + 2
6
```

Cuando se requiera cambiar el orden de ejecución de algún operador, se deben usar los paréntesis. Por ejemplo, si en la anterior expresión lo que se buscaba era que primero se ejecutara la suma y luego que el resultado de esa suma le fuera restado al 10, la expresión tendría que haber sido escrita así:

```
10 - (6 + 2)
```

Al calcular dicha expresión se tiene entonces:

```
10 - (6 + 2)
10 - 8
2
```

Ahora observe el siguiente ejemplo, asumiendo que a = 17 y b = 20.

```
a % 3 + b
```

El valor de la variable a es dividido entre 3, el residuo de esta división se suma con el dato contenido en la variable b. Por lo tanto, 17 al ser dividido entre 3 dará como residuo 2, valor que es sumado con el dato que hay en b, es decir 20, obteniendo 22 como resultado final.

Ahora bien, si esta expresión se escribe de la siguiente forma:

```
a % (3 + b)
```

El resultado sería diferente. Observe:

```
17 % (3 + 20)
17 % 23
17
```

Cuando se tengan varias agrupaciones de paréntesis, se deben solucionar de izquierda a derecha, teniendo en cuenta que, si hay paréntesis internos, estos son de mayor prioridad. Analice la expresión que aparece enseguida:

```
((4 + 8) * 3) /  ( 7 ^ 2  %  ( 2 + 1 ) )
```

En esta expresión, la máxima prioridad será resolver los paréntesis antes de llevar a cabo la división que los separa; pero como hay dos juegos de paréntesis, se deben resolver de izquierda a derecha.

Dentro del primer paréntesis hay una suma y un producto; la prioridad la tendría el producto, pero los paréntesis que encierran la suma deben resolverse primero, lo cual cambia el orden de operación, debiéndose desarrollar en primer lugar la suma:

```
((4  + 8)  * 3) / (7^2 % (2 + 1))

(12  * 3)  / (7^2 % (2 + 1))
```

Ahora si es posible llevar a cabo el producto dentro del paréntesis del lado izquierdo de la división, obteniendo el siguiente resultado:

```
36 / (7 ^ 2 % (2 + 1))
```

A continuación, se debe resolver la expresión que está entre el paréntesis del lado derecho de la división. Dentro de él se encuentra una operación de potencia, una división modular y una suma. Según la jerarquía de los operadores aritméticos (Tabla 1.6) primero debe resolverse la potencia, luego la división y por último la suma, pero los paréntesis internos cambian el orden de ejecución, por lo tanto, lo primero que se va a ejecutar es la suma:

```
36 / (7 ^ 2 % (2 + 1))
```

La expresión se reduce de la siguiente manera, donde la prioridad la tiene la potencia:

```
36 / (7 ^ 2 % 3)
```

Al resolverla, la expresión se presenta así:

```
36 / (49 % 3)
```

Aún hace falta resolver la división modular para eliminar los paréntesis:

```
36 / 1
```

Por último, se realiza la división, lo que da como resultado 36.

Conversión de fórmulas aritméticas en notación algorítmica

Cuando se va a trabajar con una fórmula dentro de un algoritmo, se debe convertir a una expresión aritmética que sea interpretada por los dispositivos de procesamiento de datos, esta conversión se le conoce como notación algorítmica. Generalmente las fórmulas son encontradas en forma de notación matemática:

$$a = \frac{2x + y}{x - y}$$

Pero para poder ser trabajada dentro de un algoritmo debe reescribirse en una sola línea, así:

```
a = (2 * x + y) / (x - y)
```

Para llegar a esta notación algorítmica, es necesario tener en cuenta las siguientes reglas:

Expresiones relacionales

En este tipo de expresiones intervienen los operadores relacionales, además de variables, números y constantes. Estas expresiones comparan el valor de sus operan dos y arrojan un resultado de tipo lógico: Verdadero o Falso.

Expresión	Resultado	Explicación
4 < 8	Verdadero	Compara si 4 es menor que 8.
3 > 7	Falso	Compara si 3 es mayor que 7.
6 <= 6	Verdadero	Compara si 6 es menor o igual a 6.
2 >= 9	Falso	Compara si 2 es mayor o igual a 9.
5! = 1	Verdadero	Compara si 5 es diferente de 1.
10 == 10	Verdadero	Compara si 10 es igual a 10.

Expresiones lógicas

También llamadas booleanas. Son aquellas que se utilizan para crear condiciones a partir de datos lógicos. Están compuestas por expresiones relacionales o lógicas, conectadas a través de operadores lógicos. Los resultados que arroja la evaluación de este tipo de expresiones solo pueden ser de dos formas: Verdadero o Falso.

Prioridad en el procesamiento de datos

Las expresiones lógicas también pueden combinar expresiones relacionales. En el momento de hacer su evaluación, al igual que con los operadores aritméticos, se debe respetar la precedencia y para ello se dará el orden de ejecución de todos los operadores, incluyendo los paréntesis.

Orden	Operador
1	( )
2	- (signo)
3	^
4	*, /, %
5	+, -
6	<, <=, >, >=
7	==, ! =
8	NO(Negación)
9	Y(Conjunción)
10	O(Disyunción)
11	= (asignación)

Obsérvese que se tienen dos juegos de paréntesis separados por el operador lógico O. Para que el resultado final sea verdadero, uno de los juegos de paréntesis deberá tener un resultado verdadero. Sin embargo, cada juego de paréntesis tiene en su interior un operador Y que implica que ambas expresiones dentro de cada paréntesis deben ser verdaderas para entregar un resultado verdadero. La evaluación, de acuerdo a la prioridad, es la siguiente:

Primero se evalúa el paréntesis de la izquierda:

```
(20 > 40 Y 2 <= 10) O (32 < 50 Y 20 <= 20)
```

```
(Falso Y 2 <= 10) O (32 < 50 Y 20 <= 20)
```

Dentro de él, la prioridad la tienen los operadores relaciones:

```
(Falso Y Verdadero) O (32 < 50 Y 20 <= 20)
```

Teniendo en cuenta que la expresión lógica está conectada por el operador Y, no hace falta la evaluación de la expresión relacional del lado derecho (2 <= 10), porque se sabe que el resultado será falso. Sin embargo, se realizará con fines didácticos:

```
(Falso Y Verdadero) O (32 < 50 Y 20 <= 20)
```

La primera expresión lógica arroja un resultado falso:

```
Falso O (32 < 50 Y 20 <= 20)
```

Ahora se procede con la parte derecha del operador O.

```
Falso O (32 < 50 Y 20 <= 20)
Falso O (Verdadero Y 20 <= 20)
Falso O (Verdadero Y Verdadero)
Falso O Verdadero
```

El resultado final de esta expresión es **verdadero.**

Las expresiones lógicas también son muy útiles para definir intervalos:

```
(edad >= 18) Y (edad <= 24)
```

Para que esta expresión lógica entregue un resultado verdadero, el dato almacenado en la variable edad deberá ser mayor o igual a 18 y menor o igual a 24.

Ahora… Unos ejercicios

1. Describa, de acuerdo a la precedencia de los operadores, en qué orden se ejecutan las siguientes expresiones:

a) `resultado = PI * radio^2`

b) `resultado = 2 * a + 3 * b - c`

c) `resultado = 2 * (a + 3) * b - c`

d) `resultado = a^2 - b * 36 ^ (1/2)`

2. Resuelva paso a paso las siguientes expresiones, teniendo en cuenta la prioridad de ejecución de los operadores.

a) `3 * (3 + 4) * (5 - 2)`

b) `(3 ^ (2 + 3) - 1) ^ (1.0/2.0)`

c) `5.0/2.0 * 3 + 4 ^ 3.0 * 2/ (5.0 + 2.0) - 2`

d) `10 * (7 + 7)% (9 + 2)/10`

e) `3% 2 - (2 + 2) / 1 * (3 + 1) + 3`

f) `(5 + 8*2 - 3.0/5.0) / (4*1 - 2.0/3.0 + 8/2)`

3. Suponga que se requieren las variables: a, b, c, d de tipo real.

a) Declare estas variables.

b) Asígnele un número cualquiera entre 10 y 20 a cada variable.

c) Evalué las siguientes expresiones, teniendo en cuenta los valores

asignados:

```
a = 3 * b + d%5
d = (4 * a/2 - 3*c) / (4 + b%3)c = 3 * b^2 - 5 * c/3
d = 2 * a + 3 * b + 4 * c/da = 3 * a
b = 7 + a
c = b - a
d = c + a - b
```

4. Se tienen las variables: a, b, c y d de tipo entero. A partir de las asignaciones que se dan enseguida, determine si las siguientes expresiones entregan como resultado un valor verdadero o falso:

```
a = 5
b = 4
c = 7
d = 3
```

a) `3 * a <= 15`

b) `c - d == a + 2`

c) `(5 * a + 3 * b) >= (c ^ d - 35)`

d) `(5% c + 3 * b/d) < (c^d/3)`

e) `(32% (4 * c) + 3 * (b + d)) < (c ^(1 + a% 2))`

f) `(5 * d + b != 4^d)` **O** `(c + d >= a/b)`

g) `(c * a + 10 > b * d - 6)` **Y** `((b + c)% 5 <a)`

h) `(c*(a + 10) > b*(8*d-6))` **Y** `((b + c)% 5 < a)`

i) `((a + b + c) / d != 12)` **O** `(4*c + 5 != 3 * a -d)`

j) `(c*a+7 >b * d - 3) Y ( ( b + c )^(1/2) <a^(3/2))`

k) **NO** `(a == b)`

l) **NO** `(a * b + c != d)`

5. Escriba en notación algorítmica las siguientes expresiones matemáticas:

a) $x = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$

b) $x = \dfrac{3a + 4b}{5 + c} + \dfrac{8c - 10}{3a + b}$

c) $x = \dfrac{3a\sqrt{4b + 8}}{\frac{4b+7}{10+\sqrt{4c}}} + \sqrt[3]{8d}$

d) $x = \dfrac{5ab^7}{3b + a} + 2ac^{3/5}$

e) $x = \dfrac{3a + \frac{2b+4}{3c4^3}}{\frac{a+b+c}{\sqrt{a+c+7}}}$

SESIÓN 3

ALGORITMOS Y PSEUDOCÓDIGO

Un algoritmo es un conjunto de acciones o pasos finitos, ordenados de forma lógica y que se utilizan para resolver un problema o para obtener un resultado. Los algoritmos deben ser:

Ordenado: el orden de ejecución de sus pasos o instrucciones debe ser riguroso, algunos tendrán que ser ejecutados antes de otros, de manera lógica, por ejemplo, no se podrá imprimir un archivo, si previamente no se ha encendido la impresora y no se podrá encender la impresora si previamente no se tiene una. Cada uno de ellos debe ser lo suficientemente claro para que determine con exactitud lo que se debe hacer.

Definido: si el algoritmo se ejecuta en repetidas ocasiones, usando los mismos datos, debe producir siempre el mismo resultado.

Finito: todo algoritmo posee un inicio, de igual forma debe tener un final; la ejecución de sus instrucciones debe terminar una vez procese los datos y entregue resultados.

Adicionalmente, el algoritmo debe plantear soluciones generales, es decir, que puedan ser utilizadas en varios problemas que tengan las mismas características.

Clasificación de los algoritmos

Algoritmos matemáticos: mediante un conjunto de pasos, describen como se realiza una operación matemática.

Algoritmo informal: son aquellos que son ejecutados por el ser humano. Por ejemplo, cepillarse los dientes o preparar un alimento. Hace algunos años conducir un vehículo era solamente una tarea propiamente humana, hoy en día existen vehículos autónomos que circulan por algunas grandes ciudades del mundo; de igual forma la búsqueda de una dirección no era algo que pudiera realizar un computador o un dispositivo de procesamiento de datos, en la actualidad existen los GPS o también las aplicaciones que son instaladas en los Smartphone que desarrollan esa tarea y permiten guiar a una persona hacia cualquier lugar.

Algoritmos computacionales: son aquellos que se diseñan para que luego puedan ser ejecutados por un computador.

Representación de un algoritmo

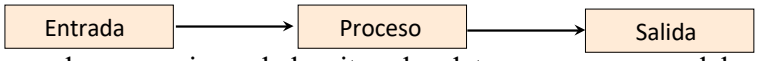
Existen tres formas de representar un algoritmo, siempre buscando que sea lo más simple posible, clara y que no genere ambigüedades:

- Descripción narrada
- Diagramas de flujo
- Pseudocódigo

Descripción narrada: Los pasos o instrucciones se describen mediante un lenguaje natural, usando palabras o frases normales y corrientes. Su uso principal se da en el disen˜o de algoritmos informales. Son ejemplos de ellos:

- Indicar como se llega a una dirección.
- Preparar una deliciosa receta de cocina.
- Registrarse en Netflix para disfrutar de su contenido

Antes de entrar en detalle con los diagramas de flujo y pseudoc´odigos, que son las formas de más uso para la representación de algoritmos computacionales, se debe establecer que este tipo de algoritmos siguen un patrón de ejecución en 3 etapas (Ver Figura 1.2). Figura 1.2: Patrón de ejecución de un algoritmo

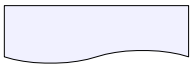
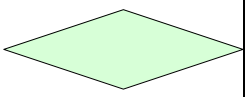


Entrada: en esta etapa se le proporciona al algoritmo los datos que se poseen del problema y que son necesarios para su solución.

Proceso*:* hace referencia a los pasos, actividades, instrucciones o cálculos que realiza el algoritmo para solucionar el problema o encontrar un resultado. Generalmente, en esta etapa se transforman los datos de entrada en resultados de salida.

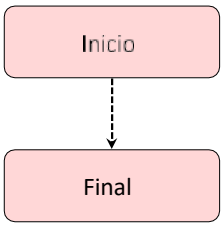
Salida*:* es la entrega de resultados o la respuesta dada por el algoritmo.

Diagrama de flujo

Símbolo	Nombre	Explicación
	**Terminal**	Representa el inicio y elfinal del algoritmo. Se rotula con la palabra Inicio o la palabra Final. En cada algoritmo solo puede estar presente un inicio y un final.
	**Entrada**	Permite la interacción con el entorno, a través de este símbolo el algoritmo recibe datos. Indica lectura de datos.
	**Proceso**	Se usa para indicar la ejecución de una acción.
	**Salida**	Permite la interacción con el entorno, a través de este símbolo muestra resultados. Indica escritura.
	**Decisión**	Indica una toma de decisiones. Se rotula con una expresión relacional o lógica, dependiendo de su resultado se toma un camino de ejecución. Se usa en decisiones, condiciones de las instrucciones `Mientras-FinMientras` y `Haga-MientrasQue`.

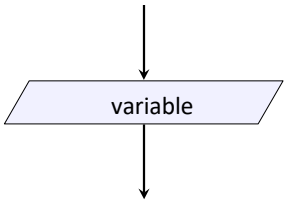
Terminal

En un diagrama deben existir solamente dos de estos símbolos, uno rotulado con la palabra Inicio y otro con la palabra Final (Ver Figuras 1.3 y 1.4).



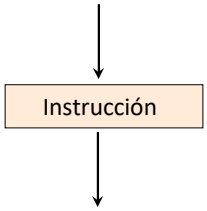
Entrada

A este símbolo entra y sale una única línea de flujo (Ver Figura 1.5). Se rotula con el identificador de la variable que recibir ‘a el valor que proporcione el usuario del algoritmo. La variable debe ser uno de los datos disponibles para la solución del problema.



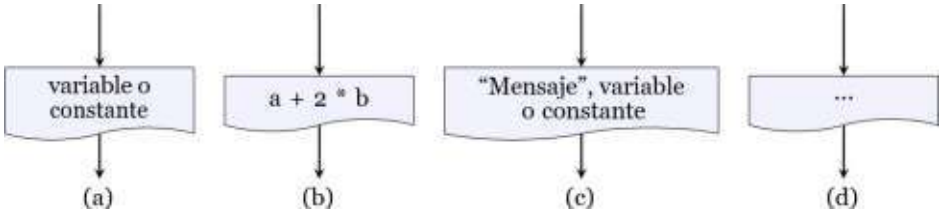
Proceso

A este símbolo entra y sale una única línea de flujo (Ver Figura 1.6). Se rotula con la instrucción que se vaya a ejecutar, puede ser, por ejemplo, una instrucción de asignación.



Salida

A este símbolo entra y sale una u ‘nica línea de flujo. Se utiliza para mostrar resultados o mensajes.



Pseudocódigo

El pseudocódigo está compuesto por un conjunto de palabras en español, inglés o cualquier otro idioma (códigos) que representan una instrucción que es entendida de una manera específica por el algoritmo en la solución de un problema. Estos códigos son fácilmente traducidos a un lenguaje de programación para que puedan ser interpretados y ejecutados por un computador.

Las palabras o códigos que se utilizan en el Pseudoc'odigo no tienen un estándar, por lo tanto, pueden variar de una fuente de información a otra. Sin embargo, conservan mucha similitud a las características de las palabras reservadas y estructuras de los lenguajes de programación. Palabras reservadas

Los códigos o palabras reservadas cumplen tareas específicas dentro de los algoritmos y no pueden usarse para propósitos diferentes. En este texto se trabajarán las siguientes:

`Algoritmo` – `Fin Algoritmo`. Indica el comienzo y final del algoritmo, respectivamente. Cada algoritmo tendrá solo un inicio y un solo final.

`Entero`, `Real`, `Cadena`, `Caracter`, `Logico`. Se usan para declarar el tipo de dato de las variables que se requieren en la solución del problema.

`Constante`. Este código se emplea en el momento de declarar constantes.

`Si`, `Entonces`, `SiNo`, `FinSi`. Se usan para la toma de decisiones simples y compuestas.

`Segun`, `Caso`, `FinCaso`, `EnOtroCaso`, `FinSegun`. Utilizadas para la toma de decisiones múltiples.

`Para`, `Hasta`, `Incremento`, `Decremento`, `FinPara`. Especifican una estructura repetitiva condicionada al comienzo.

`Mientras`, `FinMientras`. Códigos para el manejo de una estructura repetitiva condicionada al comienzo.

`Haga`, `MientrasQue`. Palabras usadas para diseñar una estructura repetitiva condicionada al final.

`Procedimiento`, `FinProcedimiento`. Palabras reservadas empleadas para dividir el algoritmo en módulos independientes, que realizan una tarea indispensable en el resultado del mismo

Entre las funciones que serán trabajadas en los ejemplos y ejercicios de los siguientes capítulos se encuentran:

```
 Entero longitud( Cadena )
Real   abs( Real o Entero )
    Real   cos( Real )

 Real   convertirANumero( Cadena )
Real   raizCuadrada( Real o Entero )
         Real   ln( Real )

 Real log( Real )
 Real exp( Real )
 Real sen( Real )
 Real  tan( Real )
```

Una de las palabras reservadas más utilizadas en los algoritmos en Pseudoc'odigo, es la instrucción imprimir, por ello, tenga en cuenta estas consideraciones en su uso:

`imprimir`. Es una instrucci´on de salida, se usa para mostrar datos o informaci´on. Su forma general es la siguiente:

`imprimir` (parámetro)

Donde `parámetro` puede tomar cualquiera de los siguientes formatos:

`imprimir( variable )` o `imprimir( CONSTANTE )`

Muestra el valor almacenado en una variable o en una constante, previamente definidas y a las cuales se les asign´o un valor.

`imprimir( Operación )`

Donde `Operación` es una expresión matemática, que puede estar conformada por valores constantes o por variables, por ejemplo:

`imprimir (5 * 2)`, muestra como resultado de salida un 10.

`imprimir( "Mensaje" )`

La salida que muestra, es el `Mensaje` que se escriba entre las comillasdobles.

Los anteriores formatos pueden combinarse de diferentes formas:

- `imprimir( "Mensaje", variable )`
- `imprimir( "Mensaje", variable, CONSTANTE )`
- `imprimir( "Mensaje", operación )`
- `imprimir( variable, "Mensaje", operación )`
- Entre otros.

Ahora unos ejercicios:

Resuelve el siguiente ejercicio usando una descripción narrada, diagrama de bloques y pseudicódigo

El profesor de Física, desea que cada uno de sus estudiantes puedan calcular, mediante un algoritmo, la velocidad con que se desplazan de su casa al colegio. Todos los estudiantes deben medir la distancia que recorren y tomar el tiempo que invierten en él.

INTRODUCCIÓN A LA MICRO:BIT

La micro:bit es una computadora de bolsillo que te presenta cómo el software y el hardware funcionan juntos. Tiene una pantalla de luz LED, botones, sensores y muchas características de entrada/salida que, al programarse, le permiten interactuar contigo y con tu mundo.

La nueva micro:bit con sonido agrega micrófono y altavoz incorporados, así como un botón de entrada táctil extra y un botón de encendido. Descubre más en este video: https://www.youtube.com/watch?v=lwqgAUG8pms

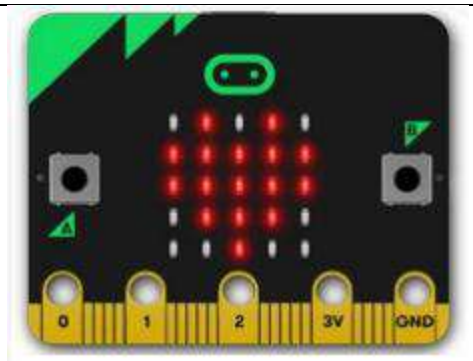
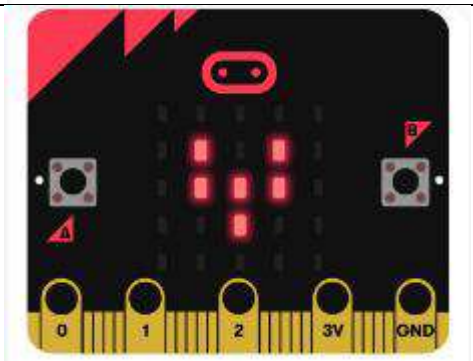
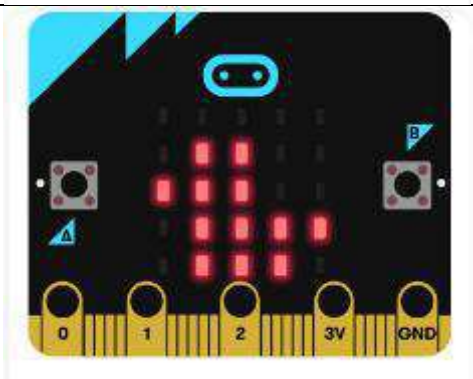
¿Qué necesitas?

- Una micro:bit y un porta pilas con 2 pilas AAA
- Una computadora, teléfono o tableta con acceso a internet para cargar los editores de código Microsoft MakeCode o Python
- Si estás usando un ordenador, un cable USB para conectar tu micro:bit
- Para construir y crear proyectos con tu micro:bit, algunos elementos adicionales que son excelentes para tener incluyen auriculares, cables de pinzas cocodrilo y materiales conductores como papel aluminio y clips para papel.

A continuación, encontraras grupos de proyectos que están diseñados para que vayas adquiriendo compresión y confianza progresivamente, por lo que ¡tú puedes trabajar sobre todos o elegir simplemente uno para empezar!

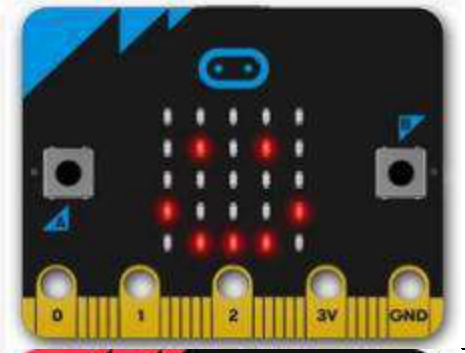
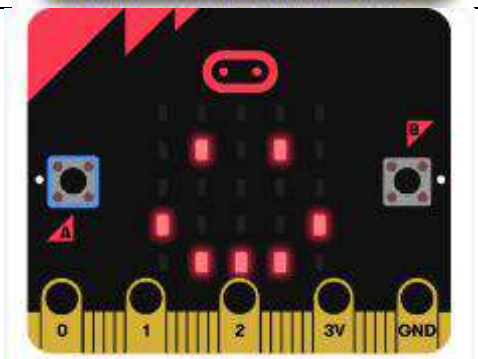
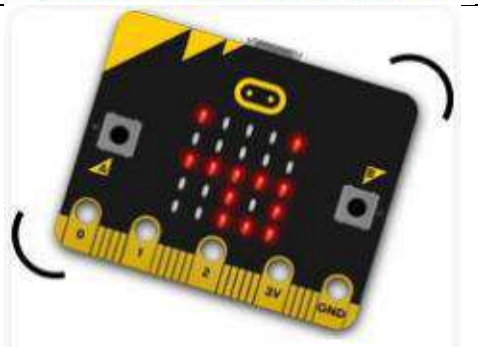
Taller 1: íconos y animales

Siguiendo esta secuencia de proyectos, aprenderás cómo crear diferentes imágenes en los LED del micro:bit mediante la secuenciación de instrucciones y el uso de botones. Luego, darás vida a tus creaciones mediante la animación y los bucles.

NIVEL	MICRO:BIT	PROYECTO
Principiante		*Corazón* Llena tu micro:bit de amor mostrando un corazón
Principiante		*Corazón palpitante* Haz latir el corazón de tu micro:bit usando bucles
Principiante		*Animales animados* Anima tus propios animales en la pantalla del micro:bit

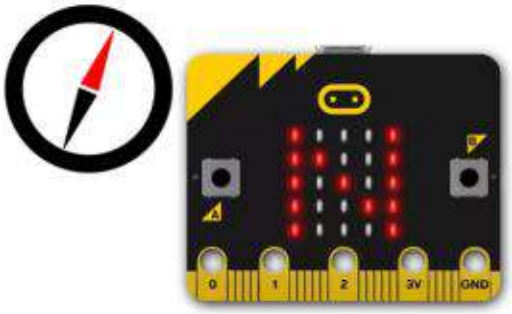
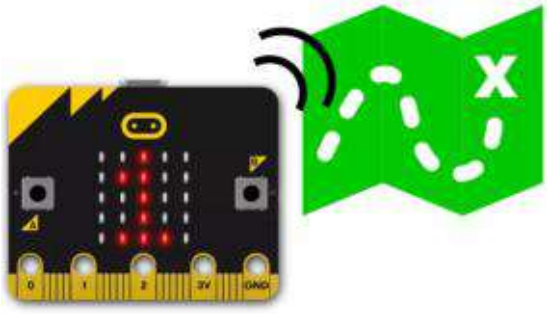
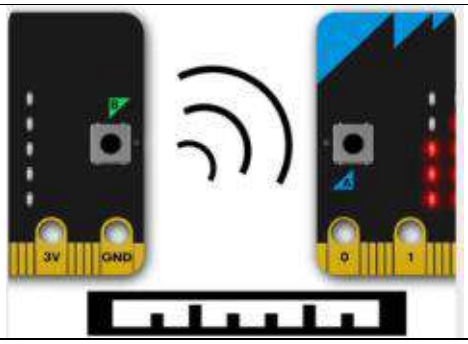
Taller 2: insignia de emociones

Sigue esta secuencia de proyectos para crear una insignia de emoción usando los LEDs, los botones y el acelerómetro para que otros puedan saber cómo te sientes. En primer lugar, programarás tu micro:bit para que muestre caras tristes y felices antes de que aparezcan y luego para que muestre más emociones al sacudir tu micro:bit.

NIVEL	*MICRO:BIT*	*PROYECTO*
Principiante		*<u>Insignia de emoción</u>* Usa tu micro:bit para expresar cómo te sientes
Principiante		*<u>Expresar emociones usando flash</u>* Haz llamativas caras felices y tristes
Principiante		*<u>Haz el tonto</u>* Agita tu micro:bit para mostrar una cara jocosa

Taller 3:

En esta secuencia de proyectos crearás diversas aplicaciones de nivel intermedio y avanzado, para aplicar conocimientos más amplios que requieren integrar sensores y una mayor programación.

NIVEL	MICRO:BIT	PROYECTO
Intermedio		Un contador de pasos puedes hacer más preciso personalizándolo según tu propio estilo de caminar.
Intermedio		Esta simple brújula te mostrará en qué dirección está el norte.
Intermedio		Usa varios micro:bits para jugar a la búsqueda del tesoro usando comunicaciones por radio.
Intermedio		*Programa tu propia mascota electrónica y personalízala para que sea la tuya. El nuevo altavoz integrado del micro:bit lo hace aún más divertido con los nuevos sonidos expresivos.*
Avanzado		*Usa la radio para detectar cómo de cerca está otro micro:bit y luego haz un juego de búsqueda del tesoro o úsalo para ayudar a la gente a saber que está a una distancia social segura*