Llegar más rápido: arreglos y grafos

Grado 8° Guía 2

TIC

Estudiantes

Apoya:

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}

Llegar más rápido: arreglos y grafos

Grado 8° Guía 2

TIC

Estudiantes

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}

MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Julián Molina Gómez
Ministro TIC

Luis Eduardo Aguiar Delgadillo
Viceministro (e) de Conectividad

Yeimi Carina Murcia Yela
Viceministra de Transformación Digital

Óscar Alexander Ballen Cifuentes
Director (e) de Apropiación de TIC

Alejandro Guzmán
Jefe de la Oficina Asesora de Prensa

Equipo Técnico
Lady Diana Mojica Bautista
Cristhiam Fernando Jácome Jiménez
Ricardo Cañón Moreno

Consultora experta
Heidy Esperanza Gordillo Bogota

BRITISH COUNCIL

Felipe Villar Stein
Director de país

Laura Barragán Montaña
Directora de programas de Educación, Inglés y Artes

Marianella Ortiz Montes
Jefe de Colegios

David Vallejo Acuña
Jefe de Implementación Colombia Programa

Equipo operativo
Juanita Camila Ruiz Díaz
Bárbara De Castro Nieto
Alexandra Ruiz Correa
Dayra Maritza Paz Calderón
Saúl F. Torres
Óscar Daniel Barrios Díaz
César Augusto Herrera Lozano
Paula Álvarez Peña

Equipo técnico
Alejandro Espinal Duque
Ana Lorena Molina Castro
Vanesa Abad Rendón
Raisa Marcela Ortiz Cardona
Juan Camilo Londoño Estrada

Edición y coautoría versiones finales
Alejandro Espinal Duque
Ana Lorena Molina Castro
Vanesa Abad Rendón
Raisa Marcela Ortiz Cardona

Edición
Juanita Camila Ruiz Díaz
Alexandra Ruiz Correa

British Computer Society – Consultoría internacional

Niel McLean
Jefe de Educación

Julia Adamson
Directora Ejecutiva de Educación

Claire Williams
Coordinadora de Alianzas

Asociación de facultades de ingeniería - ACOFI

Edición general
Mauricio Duque Escobar

Coordinación pedagógica
Margarita Gómez Sarmiento
Mariana Arboleda Flórez
Rafael Amador Rodríguez

Coordinación de producción
Harry Luque Camargo

Asesoría estrategia equidad
Paola González Valcárcel

Asesoría primera infancia
Juana Carrizosa Umaña

Autoría
Arlet Orozco Marbello
Harry Luque Camargo
Isabella Estrada Reyes
Lucio Chávez Mariño
Margarita Gómez Sarmiento
Mariana Arboleda Flórez
Mauricio Duque Escobar
Paola González Valcárcel
Rafael Amador Rodríguez
Rocío Cardona Gómez
Saray Piñerez Zambrano
Yimzay Molina Ramos

PUNTOAPARTE EDITORES

Diseño, diagramación, ilustración, y revisión de estilo

Impreso por Panamericana Formas e Impresos S.A., Colombia

Material producido para Colombia Programa, en el marco del convenio 1247 de 2023 entre el Ministerio de Tecnologías de la Información y las Comunicaciones y el British Council

Esta obra se encuentra bajo una Licencia Creative Commons Atribución-No Comercial 4.0 Internacional. https://creativecommons.org/licenses/by-nc/4.0/

CC BY-NC 4.0

"Esta guía corresponde a una versión preliminar en proceso de revisión y ajuste. La versión final actualizada estará disponible en formato digital y puede incluir modificaciones respecto a esta edición"

Prólogo

Estimados educadores, estudiantes y comunidad educativa:

En el Ministerio de Tecnologías de la Información y las Comunicaciones, creemos que la tecnología es una herramienta poderosa para incluir y transformar, mejorando la vida de todos los colombianos. Nos guia una visión de tecnología al servicio de la humanidad, ubicando siempre a las personas en el centro de la educación técnica.

Sabemos que no habrá progreso real si no garantizamos que los avances tecnológicos beneficien a todos, sin dejar a nadie atrás. Por eso, nos hemos propuesto una meta ambiciosa: formar a un millón de personas en habilidades que les permitan no solo adaptarse al futuro, sino construirlo con sus propias manos. Hoy damos un paso fundamental hacia este objetivo con la presentación de las guías de pensamiento computacional, un recurso diseñado para llevar a las aulas herramientas que fomenten la creatividad, el pensamiento crítico y la resolución de problemas.

Estas guías no son solo materiales educativos; son una invitación a imaginar, cuestionar y crear. En un mundo cada vez más impulsado por la inteligencia artificial, desarrollar habilidades como el pensamiento computacional se convierte en la base, en el primer acercamiento para que las y los ciudadanos aprendan a programar y solucionar problemas de forma lógica y estructurada.

Estas guías han sido diseñadas pensando en cada región del país, con actividades accesibles que se adaptan a diferentes contextos, incluyendo aquellos con limitaciones tecnológicas. Esta es una apuesta por la equidad, por cerrar las brechas y asegurar que nadie se quede atrás en la revolución digital. Quiero destacar, además, que son el resultado de un esfuerzo colectivo: más de 2.000 docentes colaboraron en su elaboración, compartiendo sus ideas y experiencias para que este material realmente se ajuste a las necesidades de nuestras aulas. Además, con el apoyo del British Council y su red de expertos internacionales, hemos integrado prácticas globales de excelencia adaptadas a nuestra realidad nacional.

Hoy presentamos un recurso innovador y de alta calidad, diseñado en línea con las orientaciones curriculares del Ministerio de Educación Nacional. Cada página de estas guias invita a transformar las aulas en espacios participativos, creativos y, sobre todo, en ambientes donde las y los estudiantes puedan desafiar estereotipos y explorar nuevas formas de pensar.

Trabajemos juntos para garantizar que cada estudiante, sin ·importar dónde se encuentre, tenga acceso a las herramientas necesarias para imaginar y construir un futuro en el que todos seamos protagonistas del cambio. Porque la tecnología debe ser un instrumento de justicia social, y estamos comprometidos a que las herramientas digitales ayuden a cerrar brechas sociales y económicas, garantizando oportunidades para todos.

Con estas guias, reafirmamos nuestro compromiso con la democratización de las tecnologías y el desarrollo rural, porque creemos en el potencial de cada región y en la capacidad de nuestras comunidades para liderar el cambio.

Julián Molina Gómez
Ministro de Tecnologías de la Información y las Comunicaciones
Gobierno de Colombia

Grado 8° Guía 2*

TIC

Guía de íconos

- Algoritmos, patrones, abstracción y descomposición
- Lógica, programación y depuración
- Prácticas de datos

Aprendizajes de la guía

Con las actividades de esta guía se espera que puedas avanzar en:

- Representar tareas visualmente por medio de grafos y diagramas.
- Implementar condicionales que utilizan variables, eventos, operadores de comparación y operadores lógicos (y, o, no) para controlar la ejecución de los programas.

Resumen de la guía

Esta guía propone 5 sesiones de trabajo orientadas a aprender sobre representaciones de datos en variables, arreglos y grafos. Se utilizan arreglos y variables para realizar búsquedas y cálculos.

Resumen de las sesiones

Sesión 1

En esta sesión se repasa el uso de arreglos y variables en la *micro:bit* para almacenar y utilizar información.

Sesión 2

En esta sesión se introduce el concepto de los grafos y su relación con los mapas de rutas. Se realiza un ejercicio con papel y lápiz para determinar las posibles rutas a partir de un grafo.

Sesión 3

Durante esta sesión, se lleva a cabo la prueba manual de un algoritmo utilizando papel y lápiz. Este algoritmo tiene como objetivo identificar el valor más pequeño dentro de un arreglo. Posteriormente, se utiliza *MakeCode* para crear una función que permita encontrar el valor mínimo en un arreglo.

* Esta guía es una adaptación de la ficha "Un algoritmo para ir a la escuela" desarrollada en 2022 por ACOFI para el programa Coding for Kids, en el marco del convenio 8/002 de 2022 suscrito entre el MEN, el Ministerio TIC y el British Council.

Aprendizajes de la guía

Usar diferentes tipos de variables incluyendo arreglos dentro del desarrollo de un programa, algoritmo o simulación y explicar cómo cambian sus valores durante la ejecución.

Si se requiere

- La Guía 2 de grado 7 introduce los temas de variables y arreglos y la guía 1 de grado 5 permite revisar aprendizajes iniciales sobre el uso de *MakeCode*.

Sesión 4

En esta sesión, se desarrolla un programa en *MakeCode* que tiene la capacidad de buscar en un arreglo y contar la cantidad de veces que aparece un valor específico entre los datos almacenados.

Sesión 5

En esta sesión se crea y utiliza la información almacenada en un arreglo sobre diferentes rutas, para encontrar la opción más corta. Para este ejercicio se usa *MakeCode*.

Conexión con otras áreas

Esta guía propone un reto que permite comprender cómo funcionan las tecnologías de mapeo y los sistemas de información geográfica, y los retos que enfrentan quienes los diseñan, al trabajar grandes volúmenes de datos. Con este fin, se revisan conceptos de estadística como el valor máximo y mínimo y la media de un conjunto de registros. Además, se pueden articular aprendizajes diversos como los descritos a continuación:

Matemáticas

- El uso de la media para presentar o resumir el comportamiento de un conjunto de datos, y el uso de grafos para representar y comprender las relaciones entre elementos de un conjunto.

Ciencias sociales

- El reconocimiento de la relación entre la geografía urbana y su impacto en las problemáticas actuales asociadas a la planificación y el tráfico en las grandes ciudades.

Ética y valores

- La reflexión sobre las habilidades personales con relación a la planificación y gestión del tiempo.

Grado 8° Guía 2

TIC

Sesión 1

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Definir y utilizar variables y arreglos para guardar información y realizar cálculos en un programa.
- Identificar la equivalencia decimal de un número binario.
- Explicar la relación que existe entre los elementos de memoria de un computador y la programación de variables y arreglos.

Material para la clase

- ◯ Dispositivo con acceso a *MakeCode*
- ◯ Copia del Anexo 1.1

Duración sugerida

40% 40% 20%

Anexo

Anexo 1.1

Lo que sabemos, lo que debemos saber

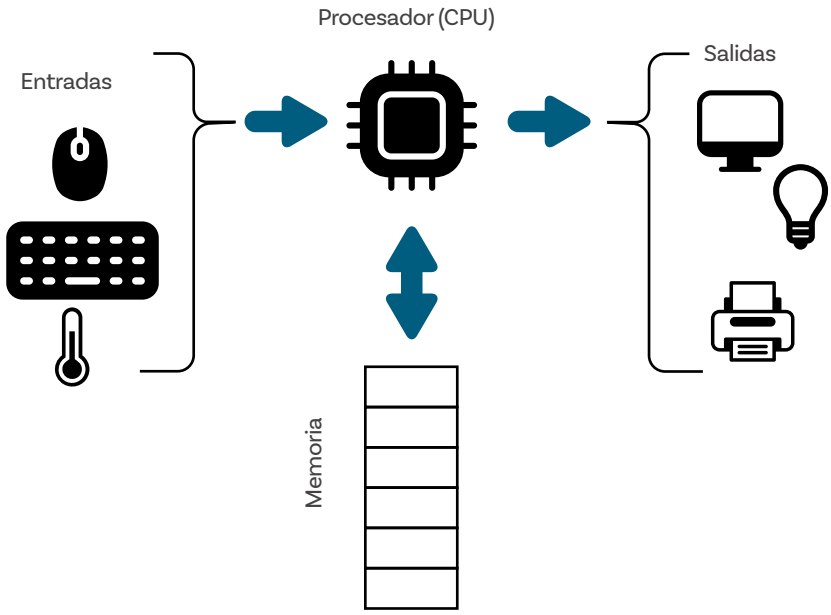
Esta sección corresponde al 40% de avance de la sesión

Comienza leyendo el reto que orienta las actividades a desarrollar a lo largo de esta guía, que se encuentra en el *Anexo 1.1*. Igualmente, ten en cuenta que debes contar con experiencia previa tanto en el manejo del editor *MakeCode*, como en el uso de variables, a fin de poder realizar lo que se propone. Si cumples con estas condiciones, entonces estás lista(o) para empezar.

Como seguramente recuerdas, los computadores tienen una memoria que les permite almacenar una gran cantidad de información, incluidos los programas que deben ejecutar y también los datos que utilizan y procesan.

Sabes, además que, todo computador está compuesto de muchos elementos, entre los que se destacan el procesador, la memoria, las entradas y las salidas. El siguiente diagrama te recuerda en qué consisten estos elementos. Obsérvalo atentamente y luego discútelo con una compañera o compañero. ¿Qué recuerdan sobre los elementos de la *Figura 1*?

Figura 1. Elementos de un sistema computacional



Como quizás surgió en la discusión sobre el diagrama, la memoria en un computador es una larga sucesión de espacios para almacenar datos en formato binario o en base 2, es decir, solo utilizando dos dígitos: 0 y 1. Cada espacio de memoria puede contener una instrucción, un dato numérico, un carácter de un texto o un elemento de una variable tipo arreglo.

En vista de que la memoria sirve para guardar todo, incluyendo instrucciones y datos, ¿cómo sabe el procesador qué está guardado en cada espacio?

Quien programa no debe preocuparse por esto, pues el lenguaje de programación utilizado es el que asigna los espacios en la memoria. Al transferir al computador las instrucciones, este agrega todo lo que se necesita para ir al lugar apropiado de la memoria donde se encuentra la información requerida e interpretar cada dato apropiadamente. Vas a entender esto mucho mejor con el siguiente ejemplo. Piensa:

¿?? *¿A qué corresponde el número binario 1100001?*

Ten presente que en el sistema en base 2, los números que se usan están construidos exclusivamente con los dígitos 0 y 1. Por tanto, si interpretas ese número binario como un número, corresponderá al número 97, pero si lo interpretas como una letra en código ASCII será la "a" minúscula.

Pero ¿cómo sabes que el número binario 1100001 es 97? Para ello, primero que todo debes realizar la conversión del número binario a decimal multiplicando cada dígito binario por la potencia de 2 que corresponda a su posición. La *Tabla 1* te muestra esa correspondencia.

Tabla 1. Conversión de binario a decimal

Dígito	1	1	0	0	0	0	1
Potencia	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Ahora que ya sabes qué valores deben multiplicarse, puedes plantear la operación. Deberás tener algo así:

$$(1x2^6) + (1x2^5) + (0x2^4) + (0x2^3) + (0x2^2) + (0x2^1) + (1x2^0)$$

Después, resuelves las operaciones en cada paréntesis. Si ya lo hiciste, seguro obtuviste lo siguiente:

$$64 + 32 + 0 + 0 + 0 + 0 + 1$$

Y finalmente, suma todos los valores obtenidos.

¿Obtuviste 97? Si es así, ya sabes por qué el número binario 1100001 corresponde al número decimal 97.

La *Tabla 2* muestra otra forma de visualizar este mismo ejemplo:

Tabla 2. Opción de conversión de binario a decimal

Binario	Base	Potencia	Valor
1	2	6	64
1	2	5	32
0	2	4	0
0	2	3	0
0	2	2	0
0	2	1	0
1	2	0	1
Total			97

Recuerda lo que has revisado hasta el momento. Ya sabes que la memoria de un computador es una sucesión de números almacenados en binario. Es decir que cada elemento de la memoria solo puede tener un valor 0 o un valor 1 y es el lenguaje de programación que usamos lo que interpreta adecuadamente cada valor.

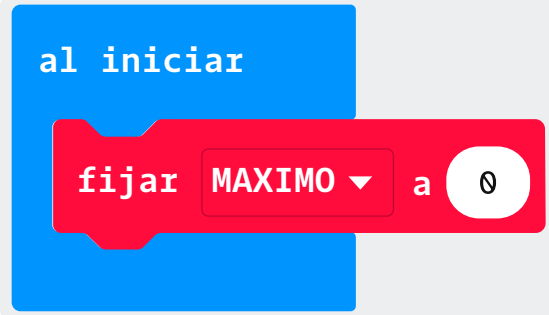
Como aprendiste en la guía anterior, estos elementos de la memoria que pueden almacenar solo ceros o unos, se conocen como bits. Un agrupamiento de 8 bits hace un **byte** y el agrupamiento de 4 u 8 bytes conforma una **palabra** o memoria de base que es capaz de trabajar el procesador de un computador.

Como ya viste, la información en un lugar de memoria (la palabra) es interpretada por el procesador como una letra, un número, una variable booleana o lógica, o incluso como un componente de una variable con varias posiciones (arreglo) según el lenguaje de programación que se utilice. Para quien programa en general esto se maneja de forma automática.

Cuando, al programar, declaras una variable, el lenguaje de programación que utilizas reserva un espacio en la memoria para esa variable y así se le indica al procesador de qué tipo de información se trata.

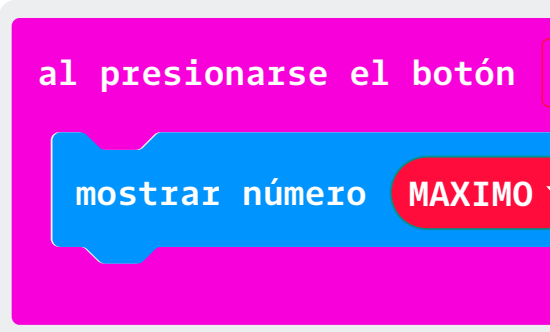
Examina el código en *MakeCode*, como se muestra en la *Figura 2*:

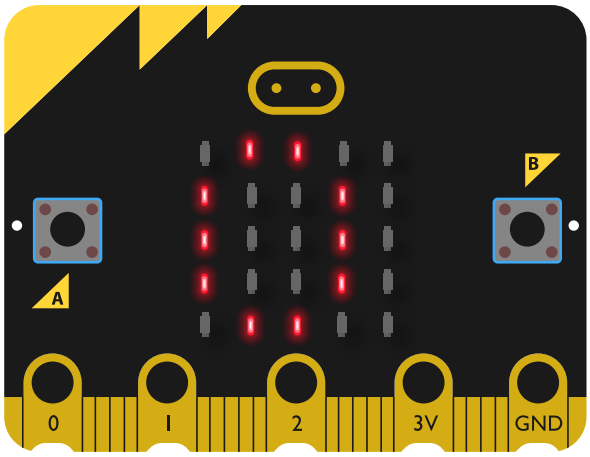
Figura 2. Creación de variable



Como seguro notaste en el código anterior, se ha creado la variable *MAXIMO*. Al crearla, el lenguaje *MakeCode* designó un lugar en la memoria del computador para guardar la información que contenga la variable *MAXIMO*. Cada que se haga referencia a esta variable en el programa, el computador se dirigirá al lugar de la memoria física donde está almacenada la información de esta variable y te permitirá visualizarla, por ejemplo, como sucede en el programa al presionar el botón A, como se muestra en la *Figura 3*.

Figura 3. Mostrar número de variable





Ahora recuerda lo que has aprendido sobre los arreglos. Estos se utilizan para referirse a múltiples datos agrupados con un solo nombre. Considera un ejemplo de la guía anterior. Imagina que quieres almacenar 10 datos de temperatura, por lo que creas un arreglo como este:

Tabla 3. Datos temperatura

TEMPERATURA	
TEMPERATURA(0)	28,7
TEMPERATURA(1)	32,5
TEMPERATURA(2)	33,4
TEMPERATURA(3)	31,2
TEMPERATURA(4)	30,0
TEMPERATURA(5)	28,7
TEMPERATURA(6)	25,2
TEMPERATURA(7)	22,8
TEMPERATURA(8)	23,5
TEMPERATURA(9)	25,1

Esta variable *TEMPERATURA* es un arreglo con 10 posiciones, rotuladas de 0 a 9, a las que puedes acceder para guardar o recuperar la información correspondiente. En las *Figuras 4* y *5* puedes observar los bloques usados para crear este arreglo.

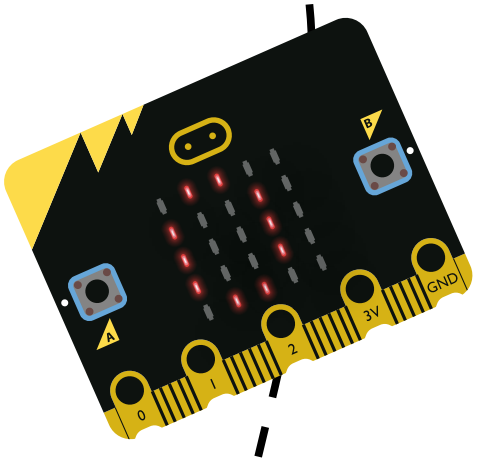
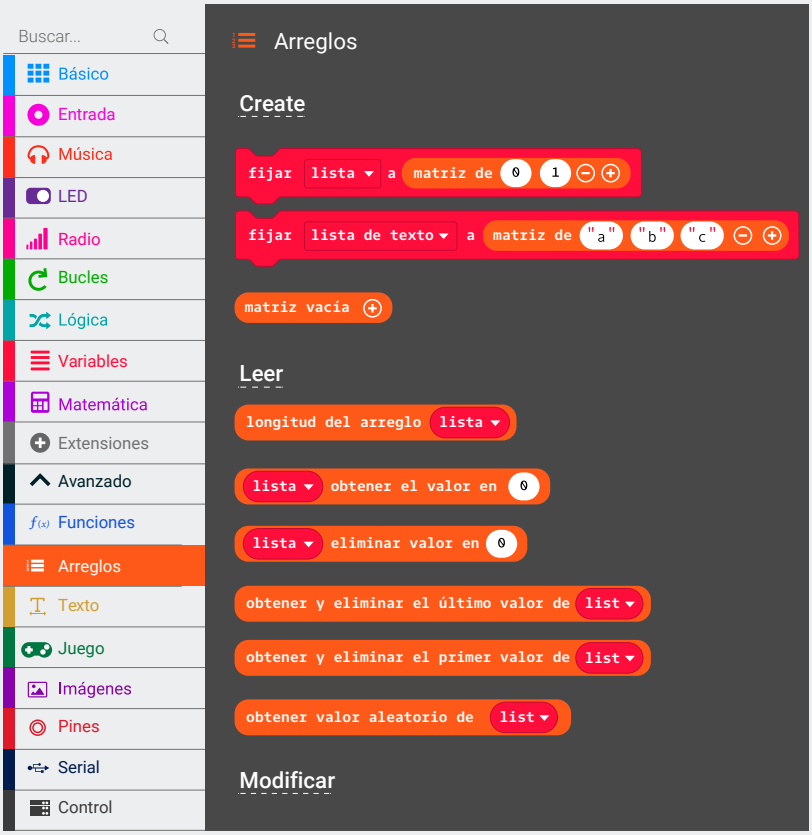


Figura 4. Bloques de Arreglos



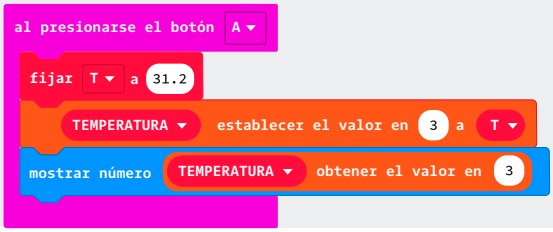
Si quieres escribir un valor en la casilla o posición 3 del arreglo, en pseudocódigo debes indicarlo así:

TEMPERATURA (3) =31,2

Si comparas el valor del ejemplo anterior, notarás que es justo el mismo que aparece almacenado en el arreglo en esa posición.

En *MakeCode* esta asignación se puede hacer así:

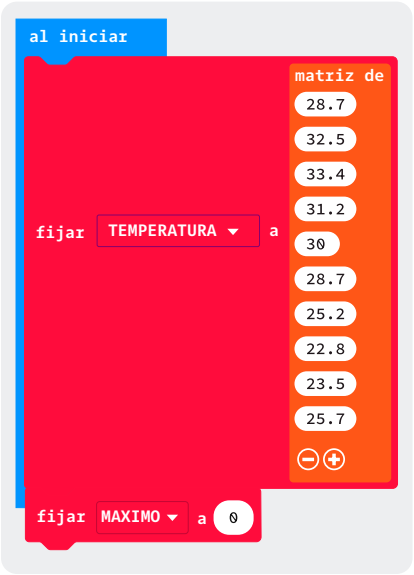
Figura 5. Asignación de valores a matriz



En esta instrucción se asigna el valor en la variable T (31.2) a la posición 3 del arreglo de temperatura.

Luego se muestra la posición 3 del arreglo de temperatura.

Figura 6. Fijar valores de matriz



Si quieres realizar la suma de la temperatura de la casilla 3 con la de la casilla 5, en pseudocódigo deberías escribir:

SUMA ← TEMPERATURA(3) + TEMPERATURA(5)

Esta operación en *MakeCode* puedes implementarla así:

Figura 7. Sumar valores

No dudes en probar, individualmente o en grupo, estos códigos en *MakeCode*.

A continuación, podrás utilizar todo lo que has aprendido y recordado hasta el momento resolviendo un reto que también te preparará para dar solución al reto de encontrar la ruta más rápida de un punto A hasta un punto B que se ha planteado en esta guía.

Glosario

Byte: Agrupación de 8 bits. Es la unidad básica más común, dado que rara vez existe interés en un solo bit, sino lo que representa el conjunto de 8 bits o el Byte. El número más grande que se puede almacenar en un Byte es 256. El tamaño de la memoria de un computador se mide en Bytes, por ejemplo 8 kB son más de ocho mil bytes y 8 Mb son 8 millones de bytes.

Palabra (word): Es un grupo de bytes. Se suele trabajar con grupos de bytes y no con bytes individuales pues estos son demasiado pequeños para ser útiles, excepto para representar letras, por ejemplo. Existen dos agrupaciones importantes, 4 bytes (32 bits) u 8 bytes (64 bits). Estas corresponden a la base del sistema de memoria de un computador, que el procesador puede manejar al mismo tiempo.

Manos a la obra

Conectadas

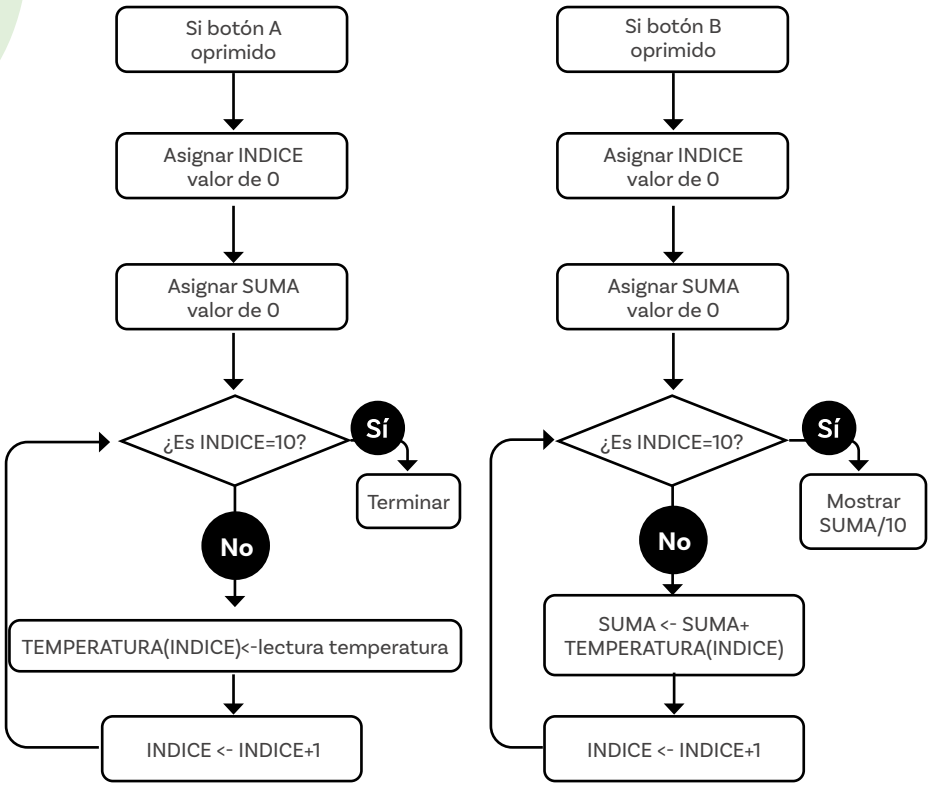
Esta sección corresponde al 80% de avance de la sesión

Recuerden que MakeCode cuenta con herramientas de accesibilidad como alto contraste y funciona con lectores en pantalla. Consulten a su docente si necesitan apoyo para usar alguna de estas opciones.

Trabaja en grupos de 2 a 3 estudiantes, según lo que te indique tu docente.

Analicen el siguiente algoritmo expresado como diagrama de flujo:

Figura 8. Diagrama de flujo - Asignación valores



Piensen individualmente en lo que creen que hace este algoritmo si se oprime primero el botón A y luego el B. ¿Qué creen que pasará si se oprimen los botones en diferente orden?

Como grupo, compartan sus respuestas a las preguntas anteriores y traten de llegar a una respuesta unificada que podrán validar con su docente.

Ahora es momento de programar. Diríjanse a *MakeCode*.

Su tarea consiste en crear un programa que simule la toma de temperatura durante 20 días y luego muestre la temperatura promedio de este rango de tiempo. Cada toma de temperatura equivaldrá al registro de un día. Este programa debe cumplir con los siguientes requerimientos:

- Al oprimir el botón A:
  - Hacer 20 lecturas de temperatura, una cada 0,5 segundos.
  - Almacenar estos valores en un arreglo que irá hasta la casilla 19.
- Al oprimir el botón B:
  - Sumar todos los valores almacenados en el arreglo.
  - Mostrar el promedio, dividiendo la suma en 20.

Pueden guiarse con el siguiente ejemplo de la *Figura 8* que muestra el programa requerido para solucionar lo que se pide cuando se oprima el botón *A*.

Figura 9. Arreglo para valores de temperatura

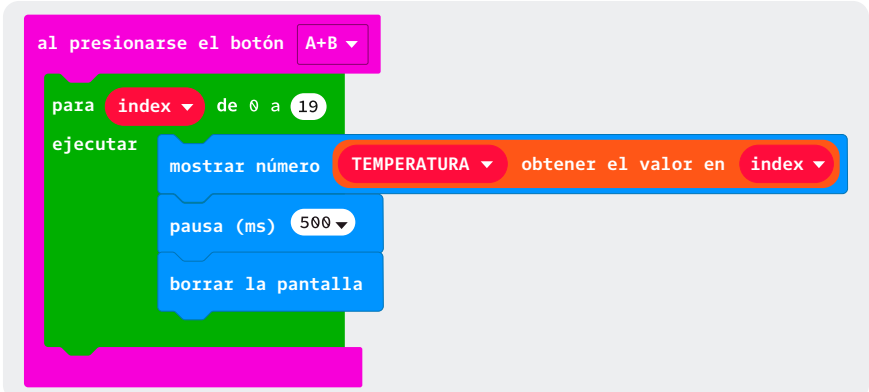
Es su turno de probar este programa.

Como notaron, el código programado no está mostrando el valor que ha quedado guardado.

¿Cómo podría mostrarse cada lugar de este arreglo una vez se tengan los valores registrados? Observen este otro programa de la *Figura 9* y luego codifíquenlo para que lo verifiquen.

Si tienen problemas para resolver este reto, pueden consultar con otras compañeras y compañeros de clase, o con su docente.

Figura 10. Arreglo para valores de temperatura



Fíjense que en ese segundo caso se usa la misma estructura del programa en bloques anterior, en que se recorre cada elemento del arreglo usando una variable llamada *index* que va incrementando en el bloque PARA de 0 a 19.

Es ahora su turno de crear el código para que al oprimir el botón B se calcule el promedio de las diferentes temperaturas registradas.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

¿Puedes definir y utilizar variables y arreglos para guardar información y realizar cálculos?

1. ¿Puedes definir y utilizar variables y arreglos para guardar información y realizar cálculos?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2. ¿Puedes determinar la equivalencia decimal de un número binario?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

3 ¿Puedes explicar la relación que existe entre los elementos de memoria de un computador y la programación de variables y arreglos?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", revisa nuevamente los contenidos y consulta las inquietudes que todavía tienes con tu docente.

Ahora, con tu grupo, siguiendo las instrucciones dadas por su docente, realicen al menos una de las siguientes actividades:

◯ Discutan estas preguntas y luego registren sus conclusiones.

Teniendo presente el reto propuesto, ¿para qué les sirve lo que han revisado en esta sesión?

_____
_____
_____

¿Qué ventajas ofrece trabajar con un arreglo como el de *TEMPERATURA*, en vez de crear muchas variables que guarden información similar (ej. T1, T2, T3, ..., etc.)?

_____
_____
_____

◯ Creen una rima o canción que les ayude a recordar algo de lo aprendido, por ejemplo, sobre los elementos de memoria de un computador.

_____
_____
_____
_____
_____

- Averigüen cómo convertir un número decimal en binario, y hagan un diagrama de flujo con el algoritmo correspondiente.

ALGORITMO PARA CONVERTIR NÚMEROS DECIMALES EN NÚMEROS BINARIOS

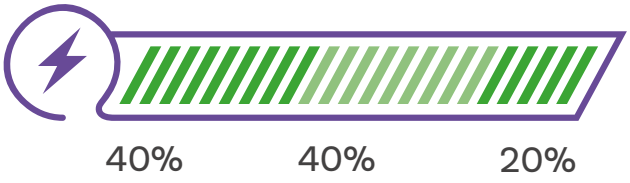
Sesión 2

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Utilizar un grafo para almacenar información de rutas en un mapa.
- Realizar cálculos con los datos de un grafo.

Duración sugerida



40% 40% 20%

Material para la clase

- ○ Anexo 2.1
- ○ Anexo 2.2
- ○ Anexo 2.3

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

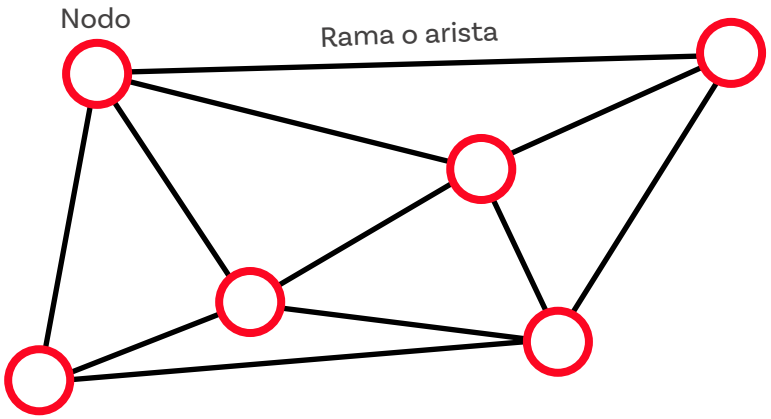
Es el momento de explorar una nueva estructura de datos muy importante para representar computacionalmente muchas situaciones o problemas, entre ellos, los mapas.

Identificar las mejores rutas partiendo de un lugar de origen hasta llegar a un lugar de destino es una de las tareas que hacen los computadores. Para ello deben encontrar todas las rutas de interés y encontrar la que menos tiempo requiere, lo cual implica buscar esta información entre un conjunto de muchísimos datos desordenados.

Ya trabajaste con los arreglos, que son una de las estructuras de datos utilizadas en computación. Ahora conocerás otra estructura llamada **grafo**.

Un grafo se utiliza en computación para múltiples propósitos. Uno de ellos es capturar la información de un mapa que contiene rutas. Un grafo está compuesto por nodos y por aristas o ramas, como puedes ver en la *Figura 1*.

Figura 1. Elementos de un grafo



Así, la información de un mapa puede ser abstraída de modo que los **nodos** representan lugares de interés, las intersecciones del

mapa representan lugares donde convergen rutas, por ejemplo, y las aristas representan las calles que conectan estas intersecciones. La información de cada **arista** puede ser el tiempo que tarda hacer el recorrido, los peajes o incluso los registros de accidentes.

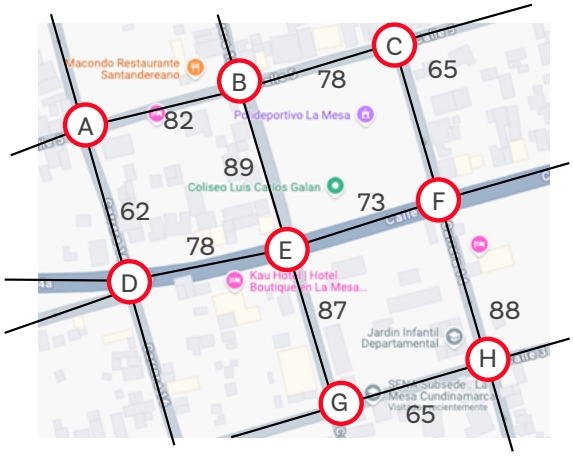
Examina esta pequeña sección de un mapa de una ciudad, tomado de Google Maps, como se muestra en la *Figura 2*.

Figura 2. Mapa de Ciudad

¿Cómo crees que sería el grafo correspondiente?

Recuerda que cada intersección es un nodo y cada unión entre ellas una arista. Por tanto, este pequeño mapa podría representarse en forma de grafo, así:

Figura 3. Representación de grafo sobre mapa



En las aristas se han colocado valores que corresponden al tiempo en segundos que implica este trayecto si se realiza caminando.

En un computador, los grafos se almacenan usando arreglos. Por ejemplo, en este caso, se podría almacenar la información en tres arreglos como los siguientes:

Tabla 1. Información de grafo

Inicio	Final	Tiempo
A	B	82
A	D	62
D	E	78
B	E	89
E	F	73
B	C	78
C	F	65
F	H	88
E	G	87
G	H	65

Glosario

- **Grafo:** Es una estructura matemática que representa relaciones entre pares de objetos. Estos objetos, llamados nodos o vértices, se conectan mediante aristas o enlaces.
- **Nodos:** Representan los objetos o entidades del sistema que se está modelando. Por ejemplo, cada intersección en un mapa. También se les conoce como vértices.
- **Aristas:** Ramas o enlaces que representan las conexiones o relaciones entre los nodos. Las aristas pueden ser dirigidas o no dirigidas, ponderadas o no ponderadas.

Anexo

Anexo 2.1

Manos a la obra

Desconectadas

Esta sección corresponde al 80% de avance de la sesión

Ahora vas a trabajar en grupos de 2 a 3 personas, según te indique tu docente.

La misión del grupo será ayudar a una de las personas que forman parte del grupo, a encontrar la ruta más corta en términos de tiempo, desde un punto de origen A, que corresponde a su casa, hacia un punto de destino llamado Z, que corresponde a la escuela.

Deben basarse en la abstracción del mapa que se presenta en forma de **grafo** en el *Anexo 2.1*, en el que cada número en las **aristas** corresponde al tiempo, en minutos, que toma ir de un **nodo** al otro.

Para calcular la ruta más rápida, necesitarán calcular la cantidad de tiempo en minutos que hay de un punto a otro, partiendo de A hacia Z. Con este fin, pueden usar tabla del *Anexo 2.2*. A continuación, verán un ejemplo de la *Figura 1* en la *Tabla 2* que muestra cómo diligenciar la información ahí:

Tabla 2. Registro de información entre nodos

Rutas	Paso 1		Paso 2		Paso 3		Paso 4		...		Total
	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma			
0	A	0	D	2	H	8	I	18			26
1											

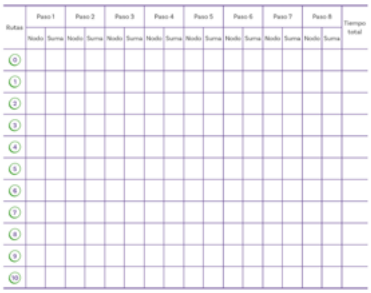
En esta tabla pueden observar el registro del comienzo de una posible ruta, la ruta 0.

Esta ruta, al igual que todas las rutas, inicia en el punto A, la casa de una persona del grupo. Como comienzan ahí, no les toma tiempo llegar a ese punto. Luego, en esta primera ruta, en el paso 2 se dirigen al nodo D, lo cual les tomaría 2 minutos.

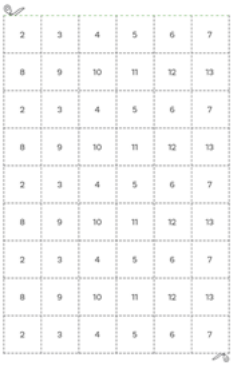
Para el paso 3 siguen, por ejemplo, al nodo H, y como llegar a este desde el nodo D les toma 6 minutos, los suman a la duración acumulada que llevan. Así deberán hacer hasta llegar a la escuela

Anexos

Anexo 2.2



Anexo 2.3



que está en el nodo Z. En el último paso tendrán el acumulado del tiempo empleado para completar todo el trayecto.

Escribe 5 posibles rutas, descúbranlas y calculen sus respectivas duraciones registrándolas en la tabla.

Luego, repitan el ejercicio con tiempos diferentes. Para esto, recorten los tiempos de la tabla del *Anexo 2.3* y colóquenlos al azar en las aristas del grafo del *Anexo 2.1.*

Continúen llenando la tabla del *Anexo 2.2*, usando las mismos 5 rutas posibles, pero con los nuevos valores de tiempos entre los nodos. Al finalizar tendrán un total de 10 registros.

¿? *¿Cambiaron las duraciones de estas últimas 5 rutas?*

Como se dieron cuenta, cuando se busca la mejor ruta, es decir, la más rápida de un lugar a otro, el número de rutas posibles es muy grande y crece muy rápidamente entre más nodos existan.

En el ejercicio anterior encontraron, al menos, 5 rutas posibles. Además, notaron que cuando los tiempos entre las intersecciones varían, es necesario volver a hacer el ejercicio completo.

Ahora les queda todavía un problema: determinar la ruta más rápida.

Para 10 datos, muy probablemente podrán determinar en algunos segundos y por inspección visual cuál es la mejor ruta.

Encuéntrenla en la tabla verificando los tiempos del *Anexo 2.1* y los tiempos asignados al azar. Ahora encuentren la ruta más lenta en ambos casos.

¿? *¿Son iguales los resultados usando los tiempos del* Anexo 2.1 *y los tiempos al azar?*

Piensen en lo siguiente. Los casos reales son muchos más complejos. Si tuvieran una lista de 2.000 posibles rutas, encontrar la mejor ruta sería un trabajo que podría tomarles varias horas.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes utilizar un grafo para almacenar información de rutas en un mapa?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2. ¿Puedes realizar cálculos con los datos de un grafo?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

> **Si tus respuestas fueron "Parcialmente" o "Aún no", puedes devolverte a revisar con atención las explicaciones y actividades de esta sesión, y consultar con tus compañeras y compañeros, así como con tu docente, las dudas que aún tienes.**

Ahora, nuevamente con tu grupo, según les indique su docente, discutan las siguientes preguntas o completen los siguientes retos:

Si regresan al reto propuesto en el *Anexo 1.1*, ¿para qué les serviría lo aprendido en esta sesión?

Dibujen o construyan grafos de 2, 3, 4, y 5 nodos, respectivamente. En cada caso, analicen cuántas rutas diferentes existen entre un punto A y un punto B que estén lo más alejado posible el uno del otro. ¿Qué encontraron?

Una pequeña ciudad tiene al menos 50 intersecciones entre sus calles. ¿Podría ayudarles un computador a optimizar los tiempos de desplazamiento?

Aprovecha este espacio final para hacer un esquema gráfico en el que resumas algo de lo que aprendiste, por ejemplo, colocando tus propias definiciones de palabras claves como grafos, nodos y aristas.

Grado 8º | Guía 2

TIC

Sesión 3

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Definir una función para encontrar el valor más pequeño en un arreglo.

Duración sugerida

40% 40% 20%

Material para la clase

- ○ Acceso a *MakeCode*

Si se requiere

Consulta la *Guía 3* de grado 7 para revisar aprendizajes previos sobre las funciones.

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

Encontrar valores máximos, mínimos y calcular promedios en un conjunto de datos suelen ser tareas frecuentes, particularmente cuando existe un nivel de incertidumbre. Por ejemplo, en el reto que se plantea en esta guía, una vez encontradas todas las posibles rutas de un punto a otra, que pueden ser muchas, se requiere determinar, entre otras, cuál es la más rápida.

Como has aprendido, cuando una tarea se realiza con frecuencia, es común utilizar funciones, dado que estas se pueden reutilizar en diferentes contextos.

En esta sesión tu reto es crear una función que pueda encontrar el valor mínimo en un arreglo de cualquier tamaño, lo cual te será útil, por ejemplo, para encontrar la ruta más corta en tiempo entre un conjunto de rutas posibles.

Examina el siguiente algoritmo de la *Figura 1*.

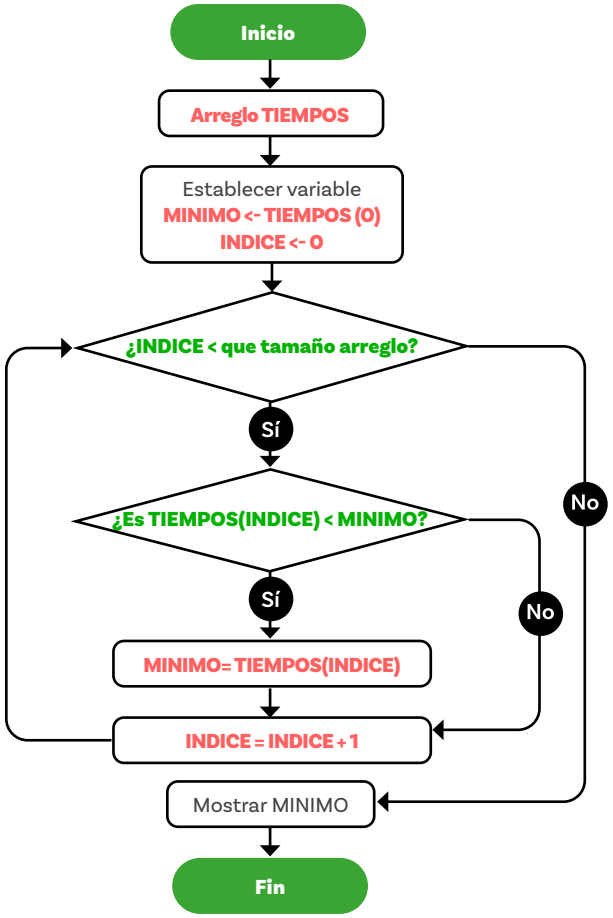
Nota

Seguir a mano un algoritmo es una forma de comprender su funcionamiento y encontrar eventuales errores.

Tabla 1. Casos de prueba - Duración de rutas en minutos

Tiempo
20
18
35
42
15
21
29
31

Figura 1. Algoritmo para encontrar ruta más corta



Ahora, sigue el algoritmo usando papel y lápiz. Para ello considera los valores que se presentan en el arreglo de la *Tabla 1*.

Sobre una hoja de papel crea una tabla como la *Tabla 2* y registra los valores de las variables, según la ejecución del algoritmo.

Tabla 2. Registro de valores de variables

Variable	Valor
MINIMO	
INDICE	

¿Encontraste que el valor mínimo es 15?

Ahora, prepárate. Es hora de programar una función.

Nota

Recuerden que este bloque

permite encontrar en un arreglo el valor almacenado en la posición *index*.

Además, este bloque permite identificar el tamaño de un arreglo. Recuerden que las posiciones van desde la posición 0 hasta la posición N-1. Por tanto, si el arreglo es de 10 posiciones, la posición final será la 9.

Manos a la obra

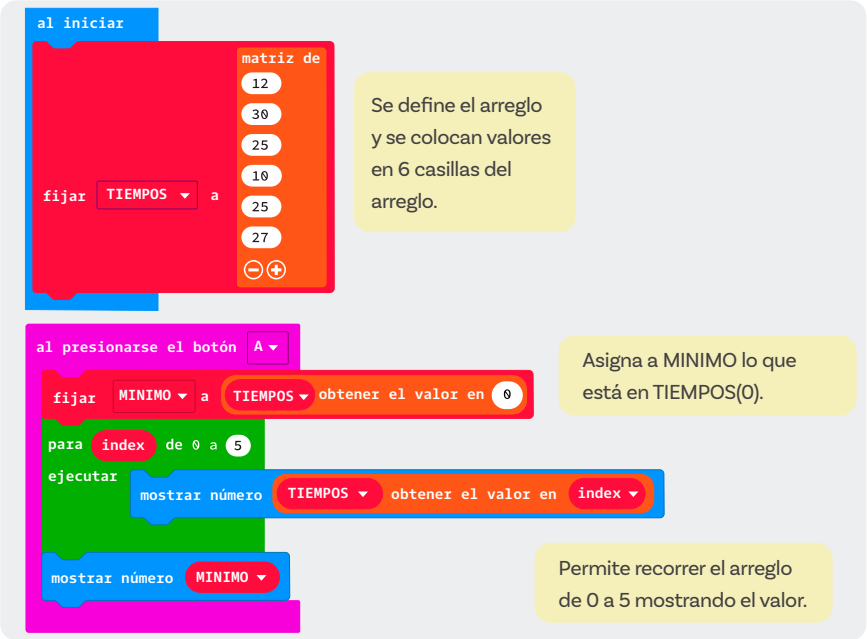
Conectadas

Esta sección corresponde al 80% de avance de la sesión

Organízate en parejas, siguiendo las indicaciones de tu docente. Lo primero que van a hacer es crear un programa en *MakeCode* que les permita encontrar el valor mínimo que le permita a una persona determinar la ruta más corta de un punto a otro según los tiempos de las posibles rutas. Es decir, un programa que encuentre el valor mínimo entre un conjunto de valores almacenados en un arreglo llamado *TIEMPOS*.

Examinen el programa en la *Figura 2* y compárenlo con el algoritmo expresado en el diagrama de flujo anterior:

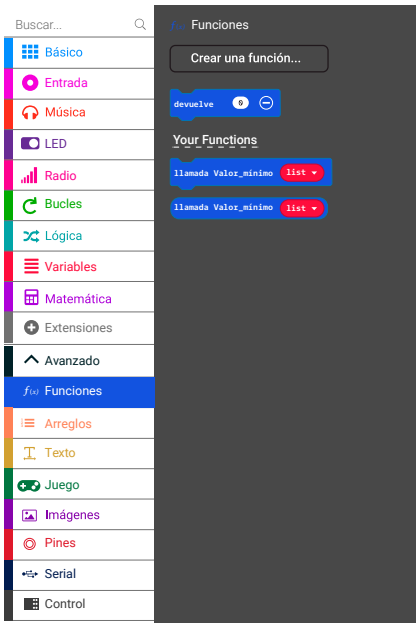
Figura 2. Programa para encontrar ruta más corta



Su tarea es completar el programa para que, en lugar de mostrar cada valor del arreglo, lo compare con el valor que está guardado en la variable *MINIMO* y si el nuevo valor es menor que lo que hay en *MINIMO*, reemplazarlo por este nuevo valor.

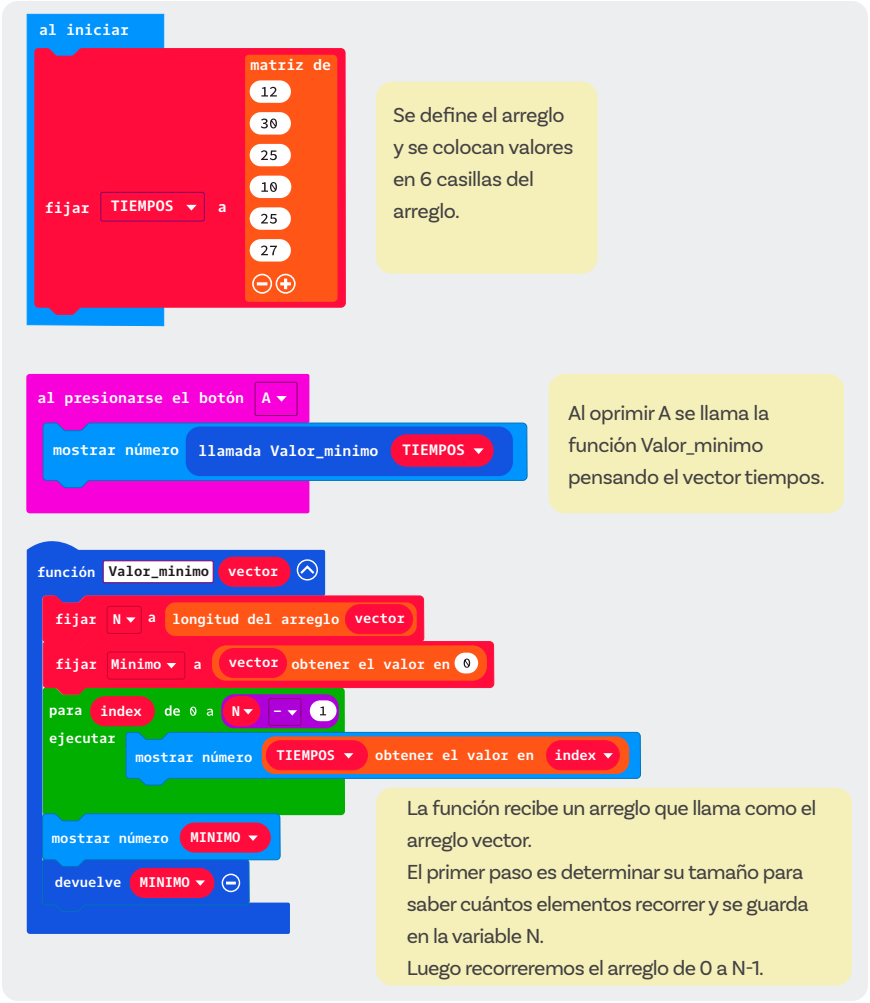
Una vez tengan listo su programa, verifíquenlo presionando el botón A. Si funciona correctamente, su código debería mostrar que el valor más pequeño almacenado en este arreglo es 10.

Figura 3. Menú de funciones en MakeCode



Cuando hayan creado esta función para encontrar el valor mínimo en un arreglo, podrán usarla con el mismo objetivo en otros programas, independientemente del tamaño de los arreglos.

Ahora conviertan en una función el código que tienen en el bloque "al presionar el botón A". La imagen siguiente muestra un ejemplo de función que solo muestra los valores del arreglo anterior. Este modelo les puede servir de inspiración para crear su función.



Si editan en *MakeCode* este programa encontrarán que hace lo mismo que el propuesto inicialmente:

> Presenta cada uno de los valores en el arreglo *TIEMPOS* y al final muestra el valor almacenado en la variable *MINIMO* que corresponde al primer valor del arreglo.

La *Figura 3* muestra el menú de funciones en el editor de *MakeCode*.

Ahora hagan los cambios requeridos para que el programa muestre realmente el valor *MINIMO* almacenado en el arreglo *TIEMPOS*.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes definir una función para encontrar el valor más pequeño en un arreglo?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

> **Si tu respuesta fue "Parcialmente" o "Aún no", no olvides volver a revisar los contenidos e intentar por ti misma(o) las actividades propuestas. Pide aclaraciones o apoyo de tu docente cuando lo requieras.**

Con tu grupo, analicen las siguientes preguntas y luego registren sus conclusiones:

¿Qué pasaría si un lenguaje de programación no tuviese la posibilidad de crear funciones?

¿Cómo les sirve lo aprendido en esta sesión, para dar respuesta parcial al reto propuesto en el *Anexo 1.1*?

Aprovechen este espacio final para hacer un gráfico que integre imágenes y texto, e indique los pasos a seguir en *MakeCode* para crear una función que identifique el valor más grande, el máximo, almacenado en un arreglo.

Grado 8º | Guía 2

TIC

Sesión 4

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

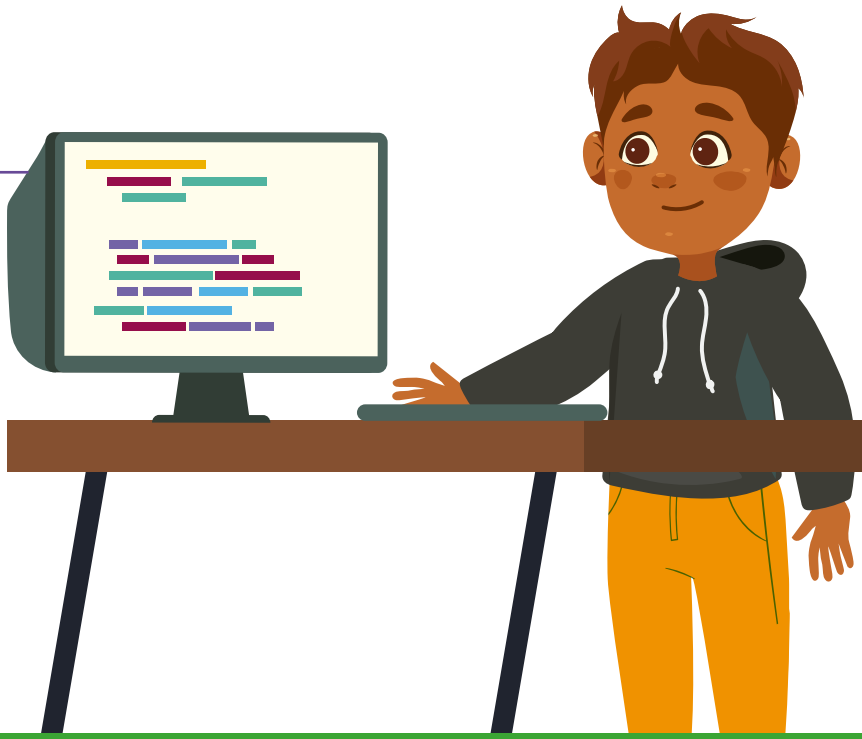
- Recorrer arreglos para poder realizar diferentes tipos de operaciones con sus valores.

Duración sugerida

15% 70% 15%

Material para la clase

- ○ Acceso a *MakeCode*



Lo que sabemos, lo que debemos saber

Esta sección corresponde al 15% de avance de la sesión

Sobre los arreglos a menudo debemos realizar muchos tipos de operaciones, entre ellas las siguientes:

- Encontrar el valor promedio.
- Encontrar el valor máximo.
- Encontrar el valor mínimo.
- Encontrar el valor más frecuente (la moda).
- Verificar si un valor dado se encuentra presente o no.

Cuando un aplicativo de navegación y tráfico trata de encontrar la ruta más rápida, la más corta o la menos costosa en peajes, por ejemplo, implica que debe buscar entre cientos y a veces miles de rutas, hasta hallar la que cumple con el criterio de búsqueda.

En esta sesión realizarás un nuevo tipo de operación sobre un arreglo: encontrar cuántas veces está presente un valor dado en un arreglo de datos.

¿Para qué crees que puede servir hacer este tipo de búsqueda?

Si pensaste, por ejemplo, en que a una o un docente le podría servir usar esta operación para determinar con un arreglo de notas cuántos de sus estudiantes han obtenido cierta calificación; o en que un programa de manejo de registro de ventas de una tienda puede, mediante un arreglo, consultar cuántas unidades han vendido de cierto artículo en particular, estás en lo correcto. En estos casos se requiere justo eso, consultar cuántas veces se encuentra un valor específico dentro del conjunto de datos almacenado.

Ahora que sabes lo importante que puede ser esta operación, deberás partir de lo que ya sabes hacer y de los programas desarrollados previamente para poder darle solución.

Nota

Recuerda que con este bloque puedes encontrar el valor que almacenado en una posición específica de un arreglo. En este caso, en la posición que corresponde al número de la variable *index*.

De igual forma, recuerda que este es el bloque que puedes usar para determinar el número total de posiciones de almacenamiento que tiene un arreglo. Recuerda que la primera posición del arreglo es la 0 y que la última corresponde a N-1, es decir, el tamaño total del arreglo menos 1.

Glosario

- **Tendencia central:** Se refiere a los valores que representan el centro de un conjunto de datos.
- **Moda:** Valor más frecuente en un conjunto de datos. Se trata de una medida estadística de tendencia central.
- **Promedio:** Resulta de sumar todos los valores y dividir este resultado entre el número de valores. Se trata de una medida de tendencia central muy utilizada.
- **Mediana:** Si se organizan los datos, por ejemplo, de menor a mayor, representa el valor que se encuentra en el medio.

Manos a la obra

Conectadas

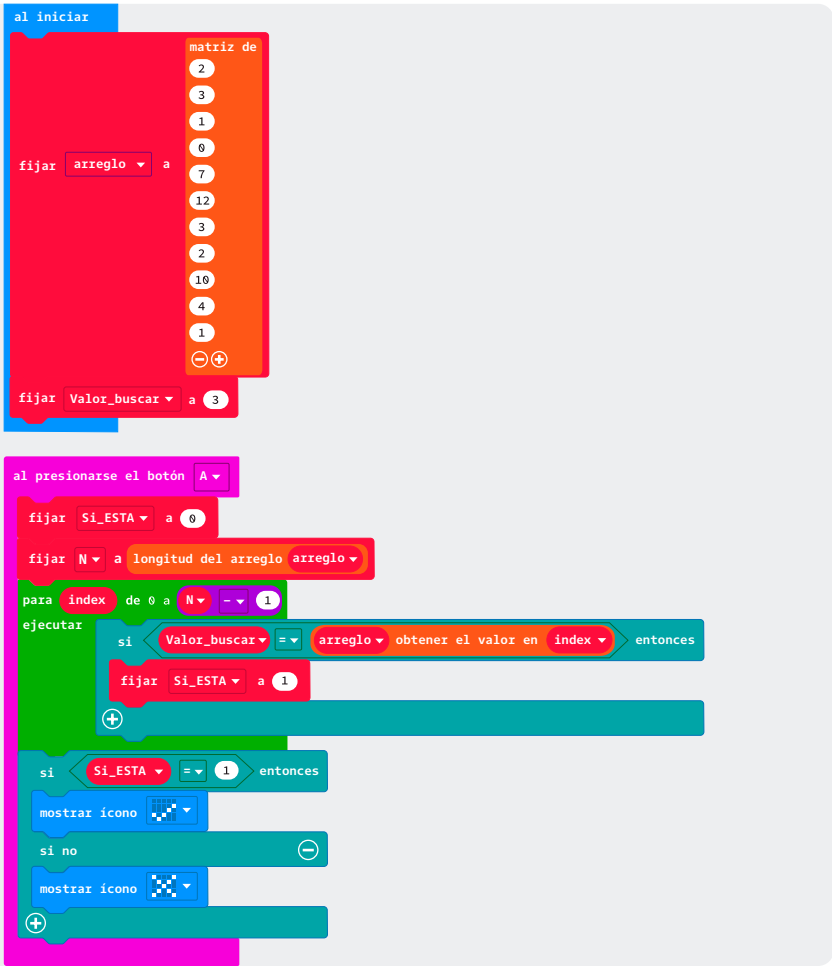
Esta sección corresponde al 85% de avance de la sesión

En esta actividad vas a trabajar en grupos de 2 a 3 personas, siguiendo las instrucciones de tu docente.

Ingresen a *MakeCode* y codifiquen el siguiente programa de ejemplo en la *Figura 1*, el cual verifica si un valor dado, almacenado en la variable *valor_buscar*, está o no dentro de un arreglo.

Usen diferentes casos de prueba para verificar el funcionamiento de su programa.

Figura 1. Programa de verificación de valor dentro de arreglo



Como observan en el bloque al iniciar, el programa se está probando con un caso de prueba pues en *Valor_buscar* se asignó el número 3.

Prueben el programa con diferentes valores existentes y no existentes en el arreglo. Noten que, en el código la variable *Si_ESTA* se asigna inicialmente a cero y cuando aparece el valor que buscamos este valor se cambia a uno.

Su misión, entonces, es modificar este programa para que no solo indique si el valor está presente o no, sino que además indique cuántas veces se encuentra. Por tanto, si el valor no está presente, su programa deberá devolver 0, es decir indicar que está cero veces.

Una forma de solucionar esto es reemplazar la variable de *Si_ESTA* por una llamada *Num_Veces_ESTA* que igualmente comience en cero,

pero que cada vez que encuentre el valor en el arreglo, se incremente en uno, y al final muestre cuántas veces está el valor buscado.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes recorrer arreglos para poder realizar diferentes tipos de operaciones con sus valores?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Si tu respuesta a la pregunta anterior fue "Parcialmente" o "Aún no", devuélvete al programa de ejemplo que se presentó en la sección anterior. Trata de poner en tus palabras las acciones que realiza cada bloque. Puedes hacer esto, agregando comentarios al código, si así lo deseas. Luego, ejecuta el programa paso a paso y ve comprobando si realiza lo que esperabas que hiciera. No olvides hacerle a tu docente las preguntas que te surjan durante ese ejercicio.

Una vez aclaradas tus dudas, trata resolver el siguiente reto:

Se está planeando una salida de la institución para las y los estudiantes que participarán en los juegos intercolegiados. Te han pedido apoyar al equipo organizador, creando un programa que registre los grados a los que pertenece la delegación del colegio. La directora de grupo te ha pedido que este programa indique específicamente cuántos son de grado octavo.

Ten en cuenta que para dar solución a este reto necesitarás crear un programa con un arreglo que contenga datos de estudiantes de varios grados, incluyendo estudiantes de octavo, que es el valor que se desea buscar y contabilizar.

Luego, con una compañera o compañero de grupo discutan las siguientes preguntas:

Si regresan nuevamente al reto propuesto en el *Anexo 1.1*, ¿para qué les sirve lo aprendido hasta ahora?

Si tienen una lista de tiempos de desplazamiento de una misma ruta, en diferentes momentos del día, ¿para qué les podría servir conocer el valor máximo, el mínimo y el promedio de estos datos?

Por último, aprovechen este espacio para escribir algunas recomendaciones que les puedan ayudar a ustedes o a sus compañeras y compañeros de clase a crear programas en *MakeCode* que realicen operaciones con los datos guardados en un arreglo. Se dan unos inicios de frase que pueden usar para dar recomendaciones del tipo qué hacer y que no hacer.

✓	✗
Al trabajar con arreglos en *MakeCode*,...	Evita...

Grado 8° **Guía 2**

TIC

Sesión 5

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

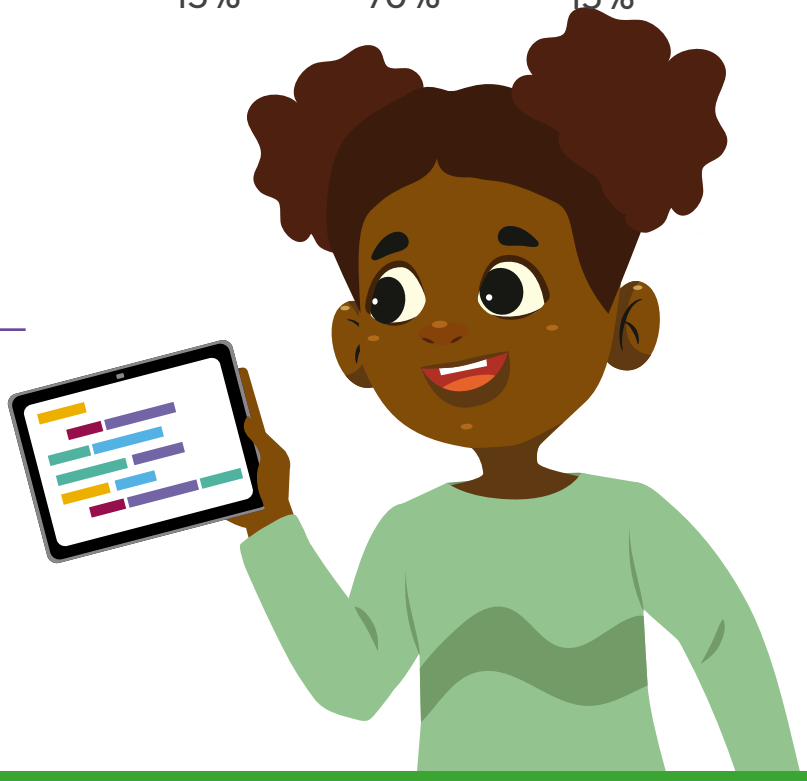
- Crear un programa que almacene, recupere y realice operaciones con datos en un arreglo.

Duración sugerida

15% 70% 15%

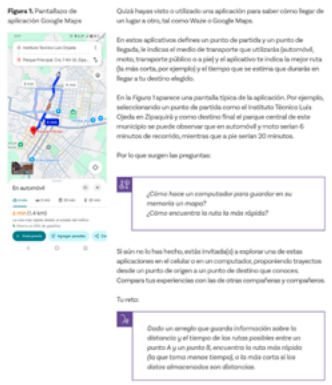
Material para la clase

- ○ Acceso a *MakeCode*
- ○ Anexo 1.1



Anexo

Anexo 1.1



Lo que sabemos, lo que debemos saber

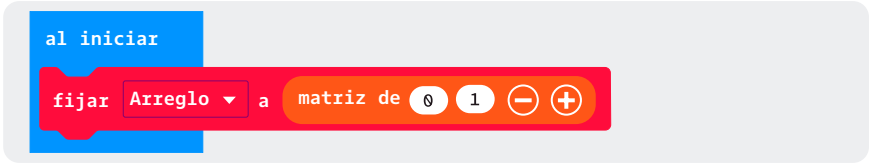
Esta sección corresponde al 15% de avance de la sesión

En este punto ya tienes lo necesario para resolver el reto que se planteó desde el comienzo y que se encuentra en el *Anexo 1.1*. Recuerda de qué se trata:

> *Dado un arreglo que guarda información sobre la distancia y el tiempo de las rutas posibles entre un punto A y un punto B, encuentra la ruta más rápida (la que toma menos tiempo), o la más corta si los datos almacenados son distancias.*

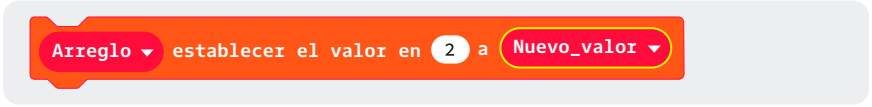
Recuerda que ya sabes:

- Definir un arreglo.



- Leer un valor específico en una posición de un arreglo.

- Escribir un valor específico en una posición del arreglo.



Así que, para darle solución al reto se debe crear un programa que, dado un arreglo, encuentre tanto el menor valor almacenado, como el lugar o posición de este valor en el arreglo, es decir, la trayectoria o ruta a la que corresponde.

Manos a la obra

Conectadas

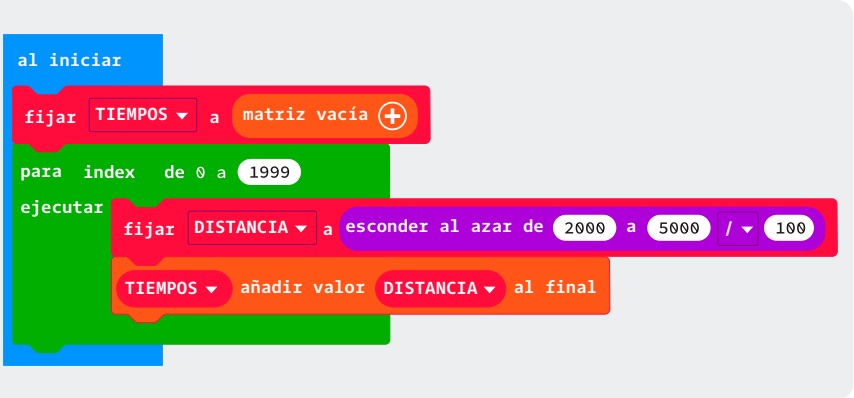
Esta sección corresponde al 85% de avance de la sesión

Organízate en grupos siguiendo las indicaciones de tu docente.

Ingresen a *MakeCode*.

Se espera que su programa pueda buscar en un arreglo de 1.000 rutas el tiempo requerido para ir del punto A al punto B. Por tanto, deberán crear un caso de prueba con arreglo grande, de 1.000 datos, que no podrá ser creado como hasta ahora lo han hecho. Para resolver el problema de la creación del caso de prueba, pueden utilizar el siguiente código.

Figura 1. Programa caso de prueba



Este programa creará un caso de prueba consistente en un arreglo de 2.000 elementos, ubicados entre las posiciones 0 y 1.999, con valores al azar entre 20 y 50, que representan minutos, con 2 decimales. Es probable que ningún valor autogenerado con este programa sea igual a otro.

Ahora sí, es su turno crear el programa que muestre en pantalla el número de ruta más rápido, es decir, la posición de 0 a 1999 que tenga almacenado el valor más pequeño, y su correspondiente valor.

Si tienen dificultades durante el desarrollo de esta tarea, pueden enriquecer sus ideas observando el trabajo de otros grupos, o pueden también consultar a su docente.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes crear un programa para almacenar, recuperar y realizar operaciones con datos en un arreglo?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

> **Si tu respuesta a la pregunta anterior fue "Parcialmente" o "Aún no", lee nuevamente el reto y prepara un algoritmo en palabras o un diagrama de flujo que pueda darle solución. Haz una prueba a papel y lápiz a tu algoritmo. Puedes usar un conjunto de máximo 10 datos diferentes para tal fin. Luego, revisa el código que tu equipo y tú propusieron como respuesta al reto. Verifica las diferencias que hay entre este programa y el de tu algoritmo en palabras o tu diagrama de flujo.**

Si aún sientes que lo requieres, pide a algunas(os) de tus compañeras y compañeros que te compartan las recomendaciones que escribieron al cierre de la sesión anterior. Verifica si pasaste por alto alguna de estas recomendaciones, y haz los ajustes pertinentes. En última instancia, recuerda que puedes volver a pedir apoyo a tu docente.

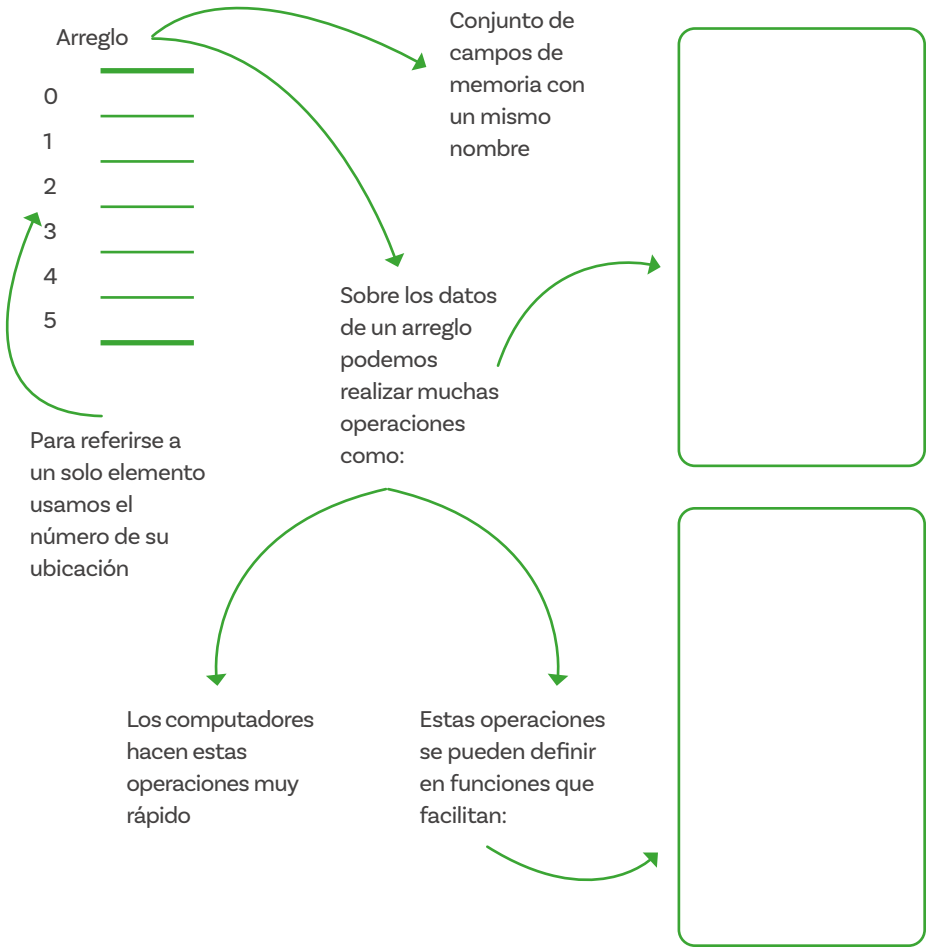
Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

Con tu grupo, discutan las siguientes preguntas y registren sus conclusiones:

Si se quisiera encontrar la ruta más rápida entre dos puntos, en un escenario en que el número de rutas posible es muy grande, por ejemplo, en las grandes ciudades, ¿podría hacerse este un trabajo a mano?

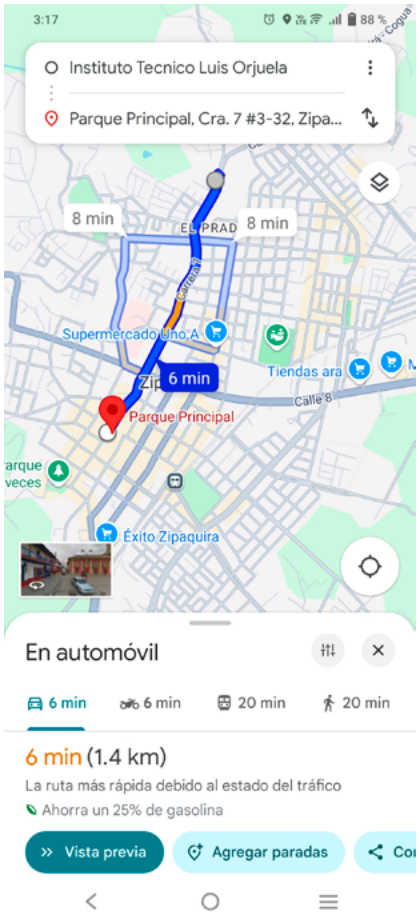
¿Qué problemas tendría hacerlo a mano?

Finalmente, examinen el siguiente esquema y complétenlo con lo aprendido.



Anexo 1.1 Reto

Figura 1. Pantallazo de aplicación Google Maps



Quizá hayas visto o utilizado una aplicación para saber cómo llegar de un lugar a otro, tal como Waze o Google Maps.

En estos aplicativos defines un punto de partida y un punto de llegada, le indicas el medio de transporte que utilizarás (automóvil, moto, transporte público o a pie) y el aplicativo te indica la mejor ruta (la más corta, por ejemplo) y el tiempo que se estima que durarás en llegar a tu destino elegido.

En la *Figura 1* aparece una pantalla típica de la aplicación. Por ejemplo, seleccionando un punto de partida como el Instituto Técnico Luis Ojeda en Zipaquirá y como destino final el parque central de este municipio se puede observar que en automóvil y moto serían 6 minutos de recorrido, mientras que a pie serían 20 minutos.

Por lo que surgen las preguntas:

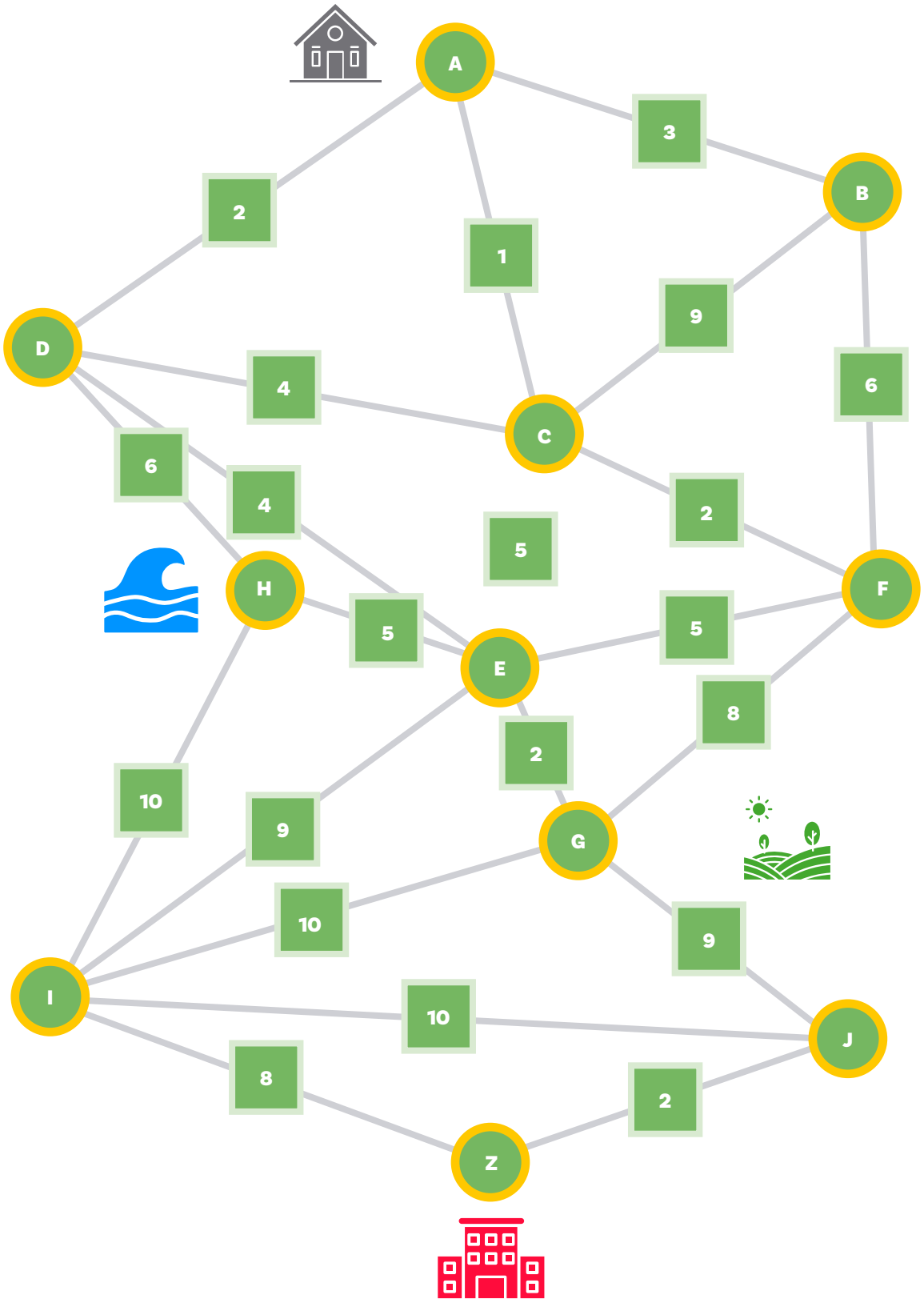
¿Cómo hace un computador para guardar en su memoria un mapa?
¿Cómo encuentra la ruta la más rápida?

Si aún no lo has hecho, estás invitada(o) a explorar una de estas aplicaciones en el celular o en un computador, proponiendo trayectos desde un punto de origen a un punto de destino que conoces. Compara tus experiencias con las de otras compañeras y compañeros.

Tu reto:

Dado un arreglo que guarda información sobre la distancia y el tiempo de las rutas posibles entre un punto A y un punto B, encuentra la ruta más rápida (la que toma menos tiempo), o la más corta si los datos almacenados son distancias.

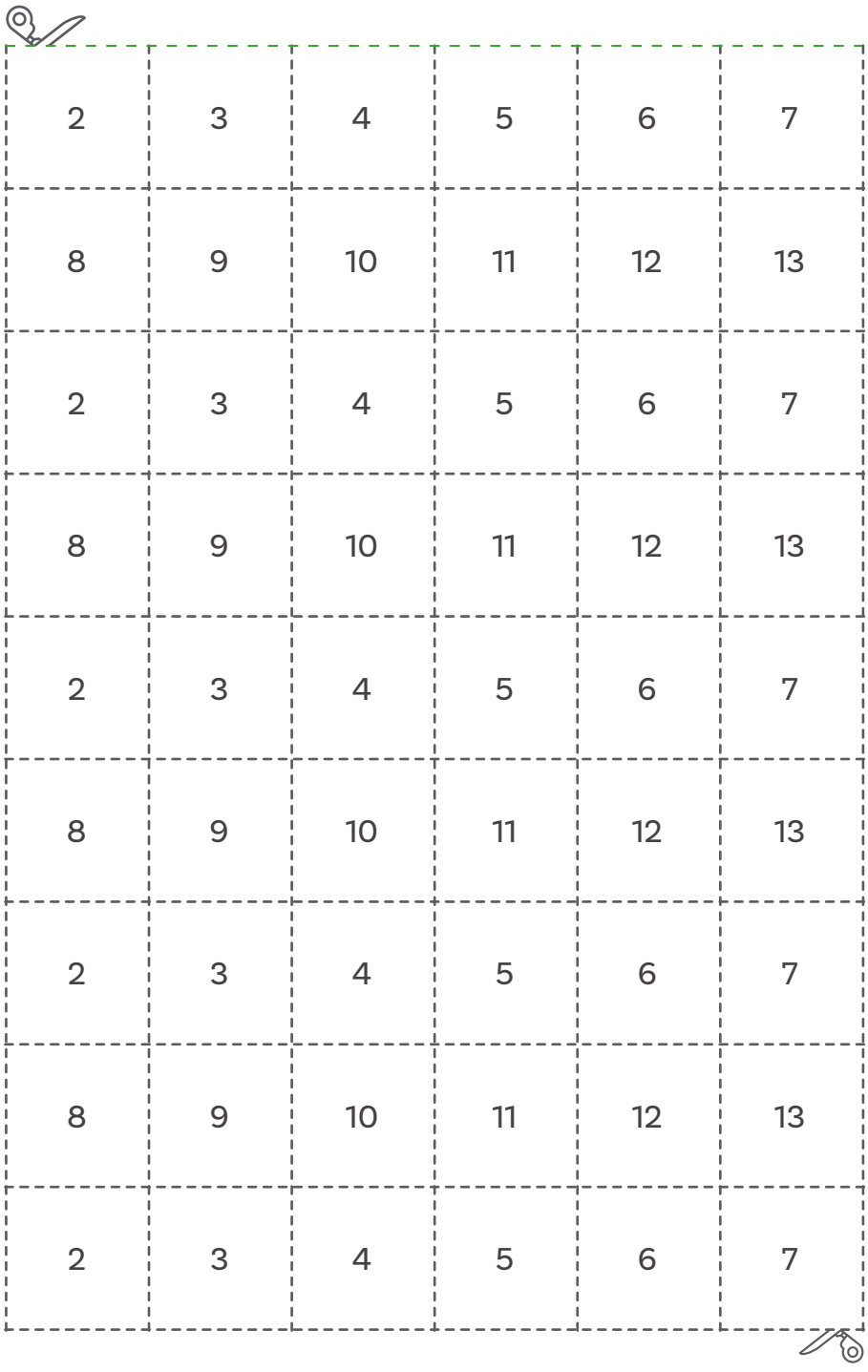
Anexo 2.1 Grafo con información de un mapa



Anexo 2.2 Arreglo resumen de rutas

Rutas	Paso 1		Paso 2		Paso 3		Paso 4		Paso 5		Paso 6		Paso 7		Paso 8		Tiempo total
	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma	Nodo	Suma	
0																	
1																	
2																	
3																	
4																	
5																	
6																	
7																	
8																	
9																	
10																	

Anexo 2.3 Valores al azar



2	3	4	5	6	7
8	9	10	11	12	13
2	3	4	5	6	7
8	9	10	11	12	13
2	3	4	5	6	7
8	9	10	11	12	13
2	3	4	5	6	7
8	9	10	11	12	13
2	3	4	5	6	7

TIC

Apoya:

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}