



APRENDE

C

GUÍA PRÁCTICA

A C A D E M I A X

## Contenido

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introducción</b>                                       | <b>4</b>  |
| 1.1      | Bienvenida . . . . .                                      | 4         |
| 1.1.1    | Libro vivo . . . . .                                      | 5         |
| 1.1.2    | Alcance . . . . .   | 5         |
| 1.2      | Prerequisitos . . . . .                                   | 5         |
| 1.3      | ¿Cómo evitar bloqueos? . . . . .                          | 6         |
| <b>2</b> | <b>Primeros pasos</b>                                     | <b>7</b>  |
| 2.1      | Visión general . . . . .                                  | 7         |
| 2.1.1    | ¿Qué es y por qué debes aprenderlo? . . . . .             | 7         |
| 2.1.2    | ¿En dónde se utiliza? . . . . .                           | 7         |
| 2.1.3    | ¿Qué trabajos puedes conseguir? . . . . .                 | 8         |
| 2.1.4    | ¿Cuánto puedes ganar? . . . . .                           | 8         |
| 2.1.5    | ¿Cuales son las preguntas más comunes? . . . . .          | 9         |
| 2.2      | Historia, evolución, y versiones . . . . .                | 9         |
| 2.3      | Características y ventajas . . . . .                      | 10        |
| 2.4      | Diferencias con otros lenguajes de programación . . . . . | 11        |
| 2.5      | Configuración . . . . .                                   | 11        |
| 2.5.1    | IDE . . . . .   | 11        |
| 2.5.2    | Entorno . . . . .   | 12        |
| 2.6      | Hola Mundo . . . . .                                      | 13        |
| <b>3</b> | <b>Gramática</b>  | <b>13</b> |
| 3.1      | Sintaxis . . . . .  | 13        |
| 3.2      | Comentarios . . . . .                                     | 14        |
| <b>4</b> | <b>Tipos y variables</b>                                  | <b>15</b> |
| 4.1      | Variables . . . . .                                       | 15        |
| 4.2      | Listas . . . . .  | 18        |

## Contenido

---

|           |                                     |           |
|-----------|-------------------------------------|-----------|
| 4.3       | Constantes . . . . .                | 20        |
| <b>5</b>  | <b>Operadores</b>                   | <b>20</b> |
| 5.1       | Operadores de aritméticos . . . . . | 20        |
| 5.2       | Operadores comparativos . . . . .   | 21        |
| 5.3       | Operadores lógicos . . . . .        | 22        |
| <b>6</b>  | <b>Condicionales</b>                | <b>23</b> |
| 6.1       | Condición única . . . . .           | 23        |
| 6.2       | Múltiples condiciones . . . . .     | 24        |
| <b>7</b>  | <b>Bucles</b>                       | <b>27</b> |
| 7.1       | Bucle for . . . . .                 | 27        |
| 7.2       | Bucle while . . . . .               | 28        |
| <b>8</b>  | <b>Funciones</b>                    | <b>29</b> |
| <b>9</b>  | <b>Estructuras</b>                  | <b>33</b> |
| <b>10</b> | <b>Módulos</b>                      | <b>35</b> |
| <b>11</b> | <b>Siguientes pasos</b>             | <b>36</b> |
| 11.1      | Herramientas . . . . .              | 36        |
| 11.2      | Recursos . . . . .                  | 37        |
| 11.3      | ¿Que viene después? . . . . .       | 38        |
| 11.4      | Preguntas de entrevista . . . . .   | 38        |

# 1 Introducción

## 1.1 Bienvenida

Bienvenid@ a esta guía de Academia X en donde aprenderás C práctico.

Hola, mi nombre es Xavier Reyes Ochoa y soy el autor de esta guía. He trabajado como consultor en proyectos para Nintendo, Google, entre otros proyectos top-tier, trabajé como líder de un equipo de desarrollo por 3 años, y soy Ingeniero Ex-Amazon. En mis redes me conocen como Programador X y comparto videos semanalmente en YouTube desde diversas locaciones del mundo con el objetivo de guiar y motivar a mis estudiantes mientras trabajo haciendo lo que más me gusta: la programación.

En esta guía vas a aprender estos temas:

- Gramática
- Tipos y variables
- Operadores
- Condicionales
- Bucles
- Funciones
- Estructuras
- Módulos

La motivación de esta guía es darte todo el conocimiento técnico que necesitas para elevar la calidad de tus proyectos y al mismo tiempo puedas perseguir metas más grandes, ya sea utilizar esta tecnología para tus pasatiempos creativos, aumentar tus oportunidades laborales, o si tienes el espíritu emprendedor, incluso crear tu propio negocio en línea. Confío en que esta guía te dará los recursos que necesitas para que tengas éxito en este campo.

Empecemos!

## 1 INTRODUCCIÓN

---

### 1.1.1 Libro vivo

Esta publicación fue planeada, editada, y revisada manualmente por Xavier Reyes Ochoa. La fundación del contenido fue generada parcialmente por inteligencia artificial usando ChatGPT (Mar 14 Version) de OpenAI. Puedes ver más detalles en <https://openai.com/>

Esto es a lo que llamo un trabajo **VIVO**, esto quiere decir que será actualizado en el tiempo a medida que existan cambios en la tecnología. La versión actual es 1.0.0 editada el 24 de marzo de 2023. Si tienes correcciones importantes, puedes escribirnos a nuestra sección de contacto en <https://www.academia-x.com>

### 1.1.2 Alcance

El objetivo de esta guía es llenar el vacío que existe sobre esta tecnología en Español siguiendo el siguiente enfoque:

- Se revizan los temas con un enfoque práctico incluyendo ejemplos y retos.
- Se evita incluir material de relleno ya que no es eficiente.
- Se evita entrar en detalle en temas simples o avanzados no-prácticos.

Si deseas profundizar en algún tema, te dejo varios recursos populares y avanzados en la lección de recursos como el estándar de esta tecnología (que puede ser difícil de leer si recién estás empezando).

## 1.2 Prerequisitos

Antes de aprender C, necesitas lo siguiente:

1. Un computador: cualquier computador moderno tiene las capacidades de correr este lenguaje. Te recomiendo un monitor de escritorio o laptop ya que dispositivos móviles o ipads no son cómodos para programar.

## 1 INTRODUCCIÓN

---

2. Sistema operativo: conocimiento básico de cómo utilizar el sistema operativo en el que se ejecutará C (por ejemplo, Windows, MacOS, Linux). Te recomiendo tener el sistema operativo actualizado.
3. Conocimiento básico de la línea de comandos: se utiliza para ejecutar programas en C.
4. Un editor de texto: lo necesitas para escribir código de C. Los editores de texto más populares son Visual Studio Code, Sublime Text, Atom y Notepad ++.
5. Un navegador web y conexión al internet: es útil para investigar más sobre esta tecnología cuando tengas dudas. Los navegadores más populares son Google Chrome, Mozilla Firefox, Safari y Microsoft Edge. Se recomienda tener el navegador actualizado.

Si ya tienes estos requisitos, estarás en una buena posición para comenzar a aprender C y profundizar en sus características y aplicaciones.

Si todavía no tienes conocimiento sobre algunos de estos temas, te recomiendo buscar tutoriales básicos en blogs a través de Google, ver videos en YouTube, o hacer preguntas a ChatGPT. Alternativamente, puedes empezar ya e investigar en línea a medida que encuentres bloqueos enteniendo los conceptos en esta guía.

### 1.3 ¿Cómo evitar bloqueos?

Para sacarle el mayor provecho a esta guía:

1. No solo leas esta guía. La práctica es esencial en este campo. Practica todos los días y no pases de lección hasta que un concepto esté 100% claro.
2. No tienes que memorizarlo todo, solo tienes que saber donde están los temas para buscarlos rápidamente cuando tengas dudas.
3. Cuando tengas preguntas usa [Google](#), [StackOverFlow](#), y [ChatGPT](#)
4. Acepta que en esta carrera, mucho de tu tiempo lo vas utilizar investigando e innovando, no solo escribiendo código.

5. No tienes que aprender inglés ahora pero considera aprenderlo en un futuro porque los recursos más actualizados están en inglés y también te dará mejores oportunidades laborales.
6. Si pierdas la motivación, recuerda tus objetivos. Ninguna carrera es fácil pero ya tienes los recursos para llegar muy lejos. Te deseo lo mejor en este campo!

## 2 Primeros pasos

### 2.1 Visión general

#### 2.1.1 ¿Qué es y por qué debes aprenderlo?

C es un lenguaje de programación de bajo nivel, estructurado y de propósito general. Es uno de los lenguajes de programación más populares y ampliamente utilizados en la actualidad.

Debes aprender C porque es un lenguaje de programación muy versátil y poderoso. Ofrece una gran cantidad de características que lo hacen ideal para la programación de sistemas, aplicaciones de escritorio, aplicaciones web, dispositivos móviles, etc. Además, es un lenguaje de programación estándar, lo que significa que hay una gran cantidad de recursos disponibles para aprenderlo. También es un lenguaje de programación muy eficiente, lo que significa que los programas escritos en C se ejecutan más rápido que los programas escritos en otros lenguajes.

#### 2.1.2 ¿En dónde se utiliza?

C es un lenguaje de programación multipropósito ampliamente utilizado para desarrollar sistemas operativos, controladores, aplicaciones de escritorio, aplicaciones

## 2 PRIMEROS PASOS

---

web, aplicaciones móviles, juegos, bases de datos, compiladores, intérpretes, herramientas de desarrollo, etc. También se utiliza en el desarrollo de firmware para dispositivos electrónicos, como microcontroladores, tarjetas de circuito impreso, etc.

### 2.1.3 ¿Qué trabajos puedes conseguir?

- Programador de software
- Desarrollador de aplicaciones
- Desarrollador de sistemas
- Desarrollador de juegos
- Desarrollador de dispositivos
- Desarrollador de software de seguridad
- Desarrollador de software de redes
- Desarrollador de software de bases de datos
- Desarrollador de software de inteligencia artificial
- Desarrollador de software de sistemas embebidos
- Desarrollador de software de sistemas operativos

### 2.1.4 ¿Cuánto puedes ganar?

El salario que puedes ganar usando C depende de muchos factores, como tu experiencia, el lugar en el que trabajes y el tipo de trabajo que estés realizando. Por ejemplo, un programador de C con experiencia puede ganar entre \$50.000 y \$100.000 al año, mientras que un ingeniero de software con experiencia puede ganar entre \$80.000 y \$150.000 al año.

Es importante tener en cuenta que estos son solo promedios y que el salario real que puedes ganar puede ser mayor o menor, dependiendo de los factores mencionados anteriormente. Además, siempre es una buena idea investigar y hacer preguntas sobre los salarios y las condiciones laborales antes de aceptar un trabajo.



### 2.1.5 ¿Cuales son las preguntas más comunes?

1. ¿Cómo puedo compilar un programa en C?
2. ¿Cómo puedo usar la función printf en C?
3. ¿Cómo puedo leer y escribir archivos en C?
4. ¿Cómo puedo usar punteros en C?
5. ¿Cómo puedo usar estructuras en C?
6. ¿Cómo puedo usar funciones en C?
7. ¿Cómo puedo usar bucles en C?
8. ¿Cómo puedo usar arreglos en C?
9. ¿Cómo puedo usar funciones recursivas en C?
10. ¿Cómo puedo usar bibliotecas en C?

Al finalizar este recurso, tendrás las habilidades necesarias para responder o encontrar las respuestas a estas preguntas.

## 2.2 Historia, evolución, y versiones

C es un lenguaje de programación de alto nivel creado por Dennis Ritchie en los años 70. Fue diseñado para reemplazar el lenguaje de programación B, que era un lenguaje de programación más antiguo. C fue diseñado para ser un lenguaje de programación portátil, lo que significa que podía ser compilado en diferentes sistemas operativos y arquitecturas de computadoras.

La primera versión de C fue lanzada en 1972 y fue llamada C de K&R (por sus autores, Brian Kernighan y Dennis Ritchie). Esta versión de C fue la base para todas las versiones posteriores. Desde entonces, ha habido varias versiones de C, cada una con sus propias características y mejoras.

La versión más reciente de C es C17, que fue lanzada en 2018. Esta versión incluye características como el soporte para el lenguaje de programación concurrencia, mejoras en la seguridad de la memoria, y mejoras en la sintaxis.

Además de C17, hay varias versiones de C que se usan aún hoy en día. Estas incluyen C89, C99, y C++. C89 fue la primera versión estándar de C, mientras que C99 fue una versión posterior que agregó características como el soporte para el lenguaje de programación concurrencia. C++ es un lenguaje de programación basado en C que agrega características como la programación orientada a objetos.

### 2.3 Características y ventajas

Características:

- Es un lenguaje de programación de bajo nivel, lo que significa que no es tan fácil de leer y escribir.
- Es un lenguaje estructurado, lo que significa que se pueden crear estructuras de datos y estructuras de control de flujo.
- Es un lenguaje de propósito general, lo que significa que se puede usar para crear cualquier tipo de programa.
- Es un lenguaje compilado, lo que significa que los programas escritos en C se compilan antes de ser ejecutados.

Ventajas:

- Es un lenguaje de programación eficiente, lo que significa que los programas escritos en C se ejecutan más rápido que los programas escritos en otros lenguajes.
- Es un lenguaje de programación flexible, lo que significa que se pueden crear programas con una gran variedad de funcionalidades.
- Es un lenguaje de programación portátil, lo que significa que los programas escritos en C se pueden ejecutar en diferentes sistemas operativos sin necesidad de cambiar el código.
- Es un lenguaje de programación estándar, lo que significa que hay un conjunto de reglas y estándares que deben seguirse al escribir código en C. Esto hace que sea más fácil para los programadores comprender y compartir el código.

### 2.4 Diferencias con otros lenguajes de programación

C es un lenguaje de programación de bajo nivel, estructurado, imperativo y compilado. Está diseñado para ser un lenguaje de programación eficiente y de bajo nivel. Esto significa que es un lenguaje de programación que se usa para escribir código que se ejecuta directamente en el hardware. Esto lo hace ideal para escribir sistemas operativos, controladores de dispositivos y aplicaciones de bajo nivel.

C es un lenguaje de programación muy diferente de otros lenguajes de programación como Java, Python y JavaScript. Estos lenguajes son lenguajes de programación de alto nivel, interpretados y orientados a objetos. Estos lenguajes se usan para escribir aplicaciones de alto nivel, como aplicaciones web y aplicaciones de escritorio. Estos lenguajes también tienen una sintaxis más fácil de entender que C.

Otra diferencia entre C y otros lenguajes de programación es que C no es un lenguaje orientado a objetos. Esto significa que no hay clases, objetos o herencia. Esto lo hace un lenguaje más simple y fácil de entender, pero también significa que no se pueden crear aplicaciones de alto nivel con C.

### 2.5 Configuración

#### 2.5.1 IDE

Los archivos de C son archivos de texto. Puedes editarlos con **editores de texto** como Notepad en Windows o Notes en MacOS pero es recomendado utilizar un **IDE** (Integrated Development Environment) que es una aplicación de edición de código más avanzado que le da colores a tu código para que sea más fácil de leer y tengas funciones de autocompletado, entre otras. Algunos IDEs populares son [Brackets](#), [Atom](#), [Sublime Text](#), [Vim](#), y [Visual Studio Code](#).

## 2 PRIMEROS PASOS

---

El editor recomendado para practicar el código que vamos a ver es Visual Studio Code (o VSCode) que puedes bajar desde <https://code.visualstudio.com/>

### 2.5.2 Entorno

Para correr código de C necesitamos instalar un compilador que compilará el código a un programa ejecutable. Después podemos correr el ejecutable en nuestro sistema operativo.

1. Instalar un compilador de C: Es posible que tu sistema operativo ya tenga un compilador de C como GCC o Clang. Puedes verificar en macOS si tienes Clang corriendo el comando **cc -v** en la terminal y obteniendo la versión instalada. Si no lo tienes, puedes correr **xcode-select --install** para instalarlo. Puedes ver otras formas de instalar el compilador en estos recursos:
  - [GCC](#)
  - [Clang](#)
2. Configurar el compilador: Si ya lo tienes instalado, puedes saltar este paso.
  - [Guía de configuración de GCC](#)
  - [Guía de configuración de Clang](#)
3. Verificar la instalación: Puedes verificar en macOS si tienes Clang corriendo el comando **cc -v** en la terminal y obteniendo la versión instalada.
  - [Verificar la instalación de GCC](#)
  - [Verificar la instalación de Clang](#)
4. Correr archivos de C: Para compilar tus archivos con la extensión **.c**, puedes correr **cc nombre\_de\_archivo.c** en la terminal. Esto creará un ejecutable con el nombre de **a.out**. Puedes correr el ejecutable corriendo **./a.out** en la terminal. Si deseas compilar y ejecutar en un solo paso, puedes correr el ejecutable de justo después de compilar con este comando **cc nombre\_de\_archivo.c; ./a.out**

### 2.6 Hola Mundo

“Hola Mundo” es un ejemplo clásico que se utiliza para mostrar el funcionamiento básico de un lenguaje de programación.

Ejemplo:

En este ejemplo, se imprime el texto “Hola Mundo” en la consola.

A screenshot of a code editor window with a dark background and a red-to-orange gradient border. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The code inside is written in a light green monospace font.

```
#include <stdio.h>

int main() {
    printf("Hola Mundo\n");

    return 0;
}
```

También podrías modificar este código con tu nombre. Si es “Juan”, debería imprimir en la consola “Hola, Juan” .

Reto:

Modifica el ejemplo anterior para imprimir “Hola Universo” en la consola.

## 3 Gramática

### 3.1 Sintaxis

La sintaxis en C está compuesta por palabras clave, identificadores, operadores, constantes, declaraciones y comentarios. Los identificadores son nombres asigna-

## 3 GRAMÁTICA

---

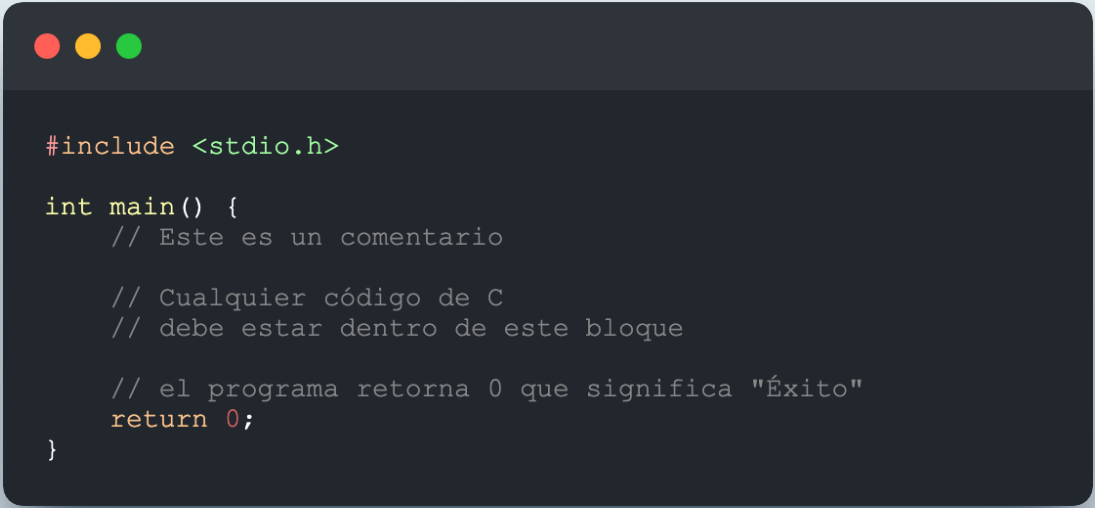
dos a variables, constantes y funciones. Las declaraciones son instrucciones que le dicen al compilador qué hacer. Los comentarios son líneas de texto que se ignoran por el compilador. La indentación es importante para mantener el código organizado y legible. El conjunto de caracteres utilizado en C es ASCII. La sensibilidad a mayúsculas y minúsculas es importante, ya que los identificadores y palabras clave deben escribirse con exactitud.

### 3.2 Comentarios

Los comentarios en C son líneas de texto que no son interpretadas como parte del código. Se usan para proporcionar información adicional sobre el código, como explicaciones, notas, o instrucciones para el desarrollador. Los comentarios también se pueden usar para deshabilitar código temporalmente, sin tener que eliminarlo completamente.

Ejemplo:

Este código es un comentario.



```
#include <stdio.h>

int main() {
    // Este es un comentario

    // Cualquier código de C
    // debe estar dentro de este bloque

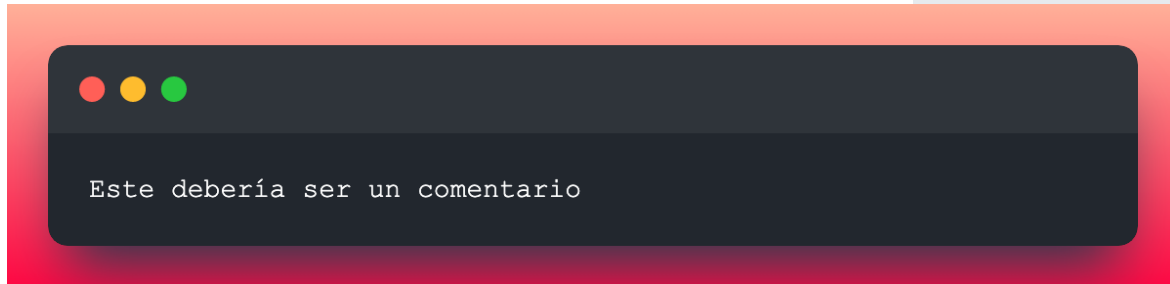
    // el programa retorna 0 que significa "Éxito"
    return 0;
}
```

## 4 TIPOS Y VARIABLES

---

Reto:

Comenta esta línea para que no cause errores y se lea como comentario:



## 4 Tipos y variables

### 4.1 Variables

La manipulación de datos es una tarea fundamental de un lenguaje de programación. Para trabajar con datos necesitamos guardarlos en variables.

Una variable es un contenedor para almacenar datos que retiene su nombre y puede cambiar su valor a lo largo del tiempo. En los siguientes ejemplos vamos a ver varios tipos de datos que puedes guardar en variables.

Ejemplo:

Este código declara una variable llamada “libro” y le asigna el valor de una cadena de texto que contiene el título de un libro. Luego, imprime el valor de la variable “libro” en la consola.

## 4 TIPOS Y VARIABLES

```
#include <stdio.h>

int main() {
    char libro[] = "El Programador Pragmático";
    // %s indica que se reemplace con el texto
    // \n es un salto de línea
    printf("%s\n", libro);

    return 0;
}
```

Texto es un tipo de dato útil para guardar información como números telefónicos y colores, entre otros. Este código asigna dos variables, una llamada telf que contiene un número de teléfono como una cadena de caracteres, y otra llamada color que contiene el color amarillo como una cadena de caracteres.

```
#include <stdio.h>

int main() {
    // char es un caracter
    // [] es una lista
    // texto es una lista de caracteres
    char telf[] = "406-234 2342";
    char color[] = "amarillo";


    return 0;
}
```



## 4 TIPOS Y VARIABLES

---

También podemos guardar datos como números enteros y decimales. Estos datos se usan para realizar operaciones matemáticas y representar valores de peso, dinero, entre otros. Este código asigna dos variables, una con un valor entero (100) y otra con un valor decimal (1.967857).




```
#include <stdio.h>

int main() {
    int entero = 100;
    // %d indica que se reemplace con entero
    printf("%d\n", entero);

    float decimal = 1.967857;
    // %f indica que se reemplace con decimal
    printf("%f\n", decimal);

    return 0;
}
```

El tipo de dato booleano representa los valores de verdadero y falso. Este tipo de datos es útil, por ejemplo, para indicar si un usuario está autorizado a acceder a una app o no, entre varios usos. Este código crea una variable llamada “autorizado” que es verdadera y otra variable llamada “seleccionado” que es falsa.



```
#include <stdio.h>

int main() {
    int autorizado = 1;
    int seleccionado = 0;

    return 0;
}
```

Es importante saber que, en el mundo del código binario, el número 1 representa verdadero y 0 representa falso.

Reto:

Crea una variable “a” con 33 como texto e imprímela a la consola.

### 4.2 Listas

Las listas en C son una estructura de datos que nos permite almacenar una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo, desde números hasta sublistas. También los vas a encontrar con otros nombres como arreglos, matrices, arrays, etc.

Ejemplo:

Este código está imprimiendo el primer elemento de una matriz de números a la consola. En este caso, el primer elemento es 23.

## 4 TIPOS Y VARIABLES

---

```
#include <stdio.h>

int main() {
    int numeros[] = {23, 45, 16, 37, 3, 99, 22};
    printf("%d\n", numeros[0]); // 23

    return 0;
}
```

También existen listas de texto. Este código crea una variable llamada “animales” que contiene una lista de elementos, en este caso, tres animales: perro, gato y tigre.

```
#include <stdio.h>

int main() {
    // 3 es el número máximo de elementos
    char *animales[3] = {"perro", "gato", "tigre"};
    printf("%s\n", animales[0]);

    return 0;
}
```

Reto:


Crea una lista en C que contenga los nombres de los meses del año. Accede al 3er índice e imprímelo en la consola.

### 4.3 Constantes

Las constantes son variables que no pueden ser reasignadas. Esto significa que una vez que se asigna un valor a una constante, este no puede ser cambiado.

Ejemplo:

Esta línea de código establece una constante llamada “pi” con un valor de 3.14159265359. Esta constante se puede usar para almacenar un valor numérico que no cambiará a lo largo del programa.



```
#include <stdio.h>

int main() {
    const float pi = 3.14159265359;

    return 0;
}
```

Reto:

Crea una constante llamada ‘SALUDO’ y asígnale el valor ‘Hola Planeta’ . Luego, imprime el valor de la constante en la consola.

## 5 Operadores

### 5.1 Operadores de aritméticos

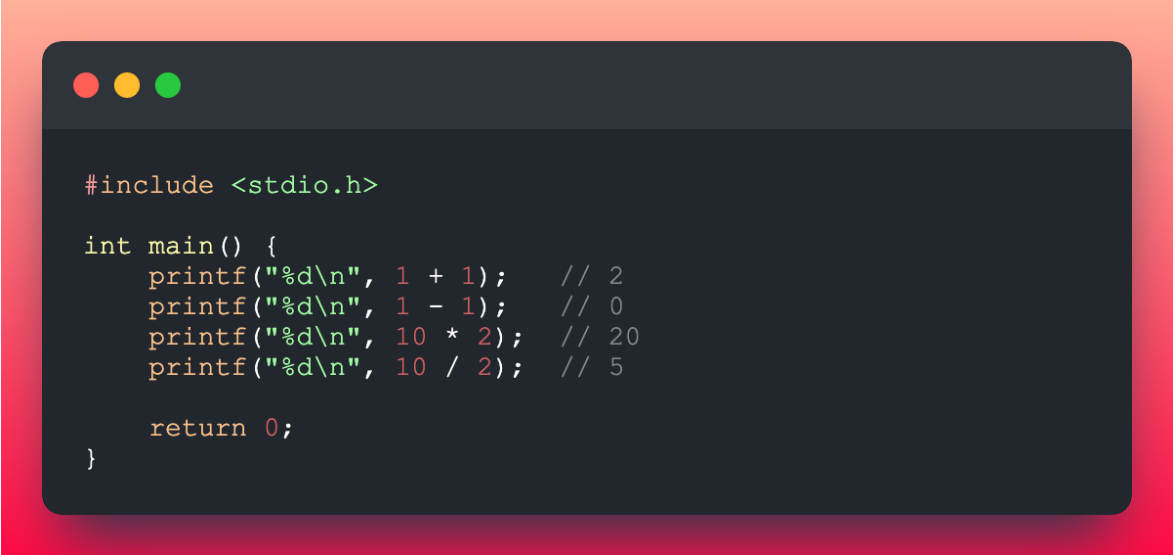
Los operadores aritméticos en C son usados para realizar operaciones matemáticas básicas. Estos operadores incluyen sumar, restar, multiplicar, dividir, entre otros.

## 5 OPERADORES

---

Ejemplo:

Este código realiza operaciones matemáticas básicas, imprimiendo los resultados en la consola. Estas operaciones incluyen suma, resta, multiplicación y división.



```
#include <stdio.h>

int main() {
    printf("%d\n", 1 + 1);    // 2
    printf("%d\n", 1 - 1);    // 0
    printf("%d\n", 10 * 2);   // 20
    printf("%d\n", 10 / 2);   // 5

    return 0;
}
```

Reto:


Usa los operadores aritméticos para calcular el área de un círculo con radio 5.

### 5.2 Operadores comparativos

Los operadores comparativos en C son usados para comparar dos valores y determinar si son iguales o diferentes. Estos operadores son útiles para determinar si dos valores son iguales, si dos valores no son iguales, si un valor es mayor o menor que otro, entre otros.

Ejemplo:

Este código muestra cómo se usan los operadores para comparar valores.



```
#include <stdio.h>

int main() {
    printf("%d\n", 4 == 4);    // 1
    printf("%d\n", 4 == '4'); // 0
    printf("%d\n", 4 != 5);   // 1
    printf("%d\n", 4 < 5);    // 1
    printf("%d\n", 4 >= 5);   // 0

    return 0;
}
```

Reto:

Escribe un programa que compare dos números 'a' y 'b' y determina si 'a' con el valor de 4 es mayor, menor o igual a 'b' con un valor de 2.

### 5.3 Operadores lógicos

Los operadores lógicos en C se usan para realizar comparaciones entre valores y devolver un valor booleano (verdadero o falso). Estos operadores puede ser útiles, por ejemplo, si deseamos saber si un 'animal' es un gato y es un mamífero (al mismo tiempo).

Ejemplo:

Este código muestra el uso de los operadores lógicos 'y' y 'o' para evaluar expresiones booleanas. El operador 'y' devuelve verdadero si ambas expresiones son verdaderas, de lo contrario devuelve falso. El operador 'o' devuelve verdadero si al menos una de las expresiones es verdadera, de lo contrario devuelve falso.

```
#include <stdio.h>

int main() {
    printf("%d\n", 1 && 1); // 1
    printf("%d\n", 1 && 0); // 0
    printf("%d\n", 0 && 1); // 0
    printf("%d\n", 0 && 0); // 0

    printf("%d\n", 1 || 1); // 1
    printf("%d\n", 1 || 0); // 1
    printf("%d\n", 0 || 1); // 1
    printf("%d\n", 0 || 0); // 0

    return 0;
}
```

Reto:

Escribe una línea de código que devuelva verdadero si 'x' es mayor que 0 y 'y' es menor que 0.

## 6 Condicionales

### 6.1 Condición única


Los condicionales son estructura de control de flujo en C, es decir, controlan el flujo del código. Permiten ejecutar una sección de código si una condición es verdadera. También permiten ejecutar otra sección de código, si la condición es falsa.

Ejemplo:

## 6 CONDICIONALES

---

Este código comprueba si una variable booleana (autorizado) es verdadera. Si es así, imprime un mensaje en la consola indicando que el usuario puede ingresar. Si no es así, imprime un mensaje indicando que el usuario no puede ingresar.

A screenshot of a code editor window with a dark background and light-colored text. The code is in C and uses the printf function to output messages based on the value of a boolean variable 'autorizado'. The code is as follows:

```
#include <stdio.h>

int main() {
    int autorizado = 1;

    if (autorizado) {
        printf("Puede ingresar\n");
    } else {
        printf("No puede ingresar\n");
    }

    return 0;
}
```

Reto:

Escribe código condicional que revise si un número 'a' es par o impar. La variable 'a' tiene el valor de 17 y debe imprimir a la consola 'Es par' si es par o 'Es impar' si es impar.

### 6.2 Múltiples condiciones

Adicionalmente, los condicionales permiten ejecutar varias secciones de código de acuerdo a varias condiciones.

Ejemplo:


En este caso podemos ver que el código que se ejecuta depende de varios valores.



## 6 CONDICIONALES

---

Este código comprueba si una variable llamada entero es igual a 99 o 100. Si es igual a 99, imprime “Es 99” en la consola. Si es igual a 100, imprime “Es 100” en la consola. Si no es ni 99 ni 100, imprime “No 99 ni 100” en la consola.



```
#include <stdio.h>

int main() {
    int entero = 100;

    if (entero == 99) {
        printf("Es 99");
    } else if (entero == 100) {
        printf("Es 100"); // ✓
    } else {
        printf("No 99 ni 100");
    }

    return 0;
}
```

Este condicional es útil cuando hay una gran cantidad de posibles resultados para una expresión. Este código compara una variable (color) con diferentes valores y ejecuta una acción dependiendo del resultado de la comparación. En este caso, si la variable color es igual a ‘amarillo’, se imprimirá ‘Advertencia’ en la consola.

```
#include <stdio.h>

int main() {
    char color = 'A';

    // switch acepta int y char
    switch (color) {
        case 'V':
            printf("Éxito");
            break;
        case 'A':
            printf("Advertencia"); // ✓
            break;
        default:
            printf("Error");
            break;
    }

    return 0;
}
```

Reto:

Crea un programa que evalúe una variable 'numero' y dependiendo del resultado, imprima un mensaje diferente. Si el número es mayor a 10, imprime "El número es mayor a 10" . Si el número es menor a 10, imprime "El número es menor a 10" . Si el número es igual a 10, imprime "El número es igual a 10" .


# 7 Bucles

## 7.1 Bucle for

Los bucles son otra estructura de control de flujo que se utilizan para iterar sobre una secuencia de elementos. También se los llama loops o ciclos.

Ejemplo:

Este código itera sobre una lista de animales e imprime cada elemento de la lista en la consola.

A screenshot of a code editor window with a dark background and a green border. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in C and uses syntax highlighting: keywords like #include, int, for, return, and printf are in orange; string literals are in green; and comments are in light gray. The code defines an array of animal names and iterates over it using a for loop.

```
#include <stdio.h>

int main() {
    char *animales[] = {"perro", "gato", "tigre"};

    // iniciamos i con 0
    // mientras i sea menor que 3
    // aumentamos i + 1 en cada iteración con i++
    for (int i = 0; i < 3; i++) {
        printf("%s\n", animales[i]);
    }

    return 0;
}
```

Reto:


Escribe un bucle que imprima los números del 1 al 10 en orden ascendente.

### 7.2 Bucle while

while es un tipo de bucle en C que se usa para ejecutar una serie de instrucciones mientras se cumpla una condición. Esta condición se evalúa antes de cada iteración del bucle.

Ejemplo:

Este código establece un bucle while que imprime los números enteros desde 100 hasta 911 en la consola. El bucle while se ejecutará mientras la variable entero sea menor o igual a la variable emergencia. Cada vez que el bucle se ejecuta, la variable entero se incrementa en 1.



```
#include <stdio.h>

int main() {
    int entero = 100;

    int emergencia = 911;

    while (entero <= emergencia) {
        printf("%d\n", entero);
        entero++;
    }

    return 0;
}
```

Ten cuidado de no incluir una condición para parar el bucle. Esto podría seguir corriendo tu código indefinidamente hasta congelar tu computador.

Reto:

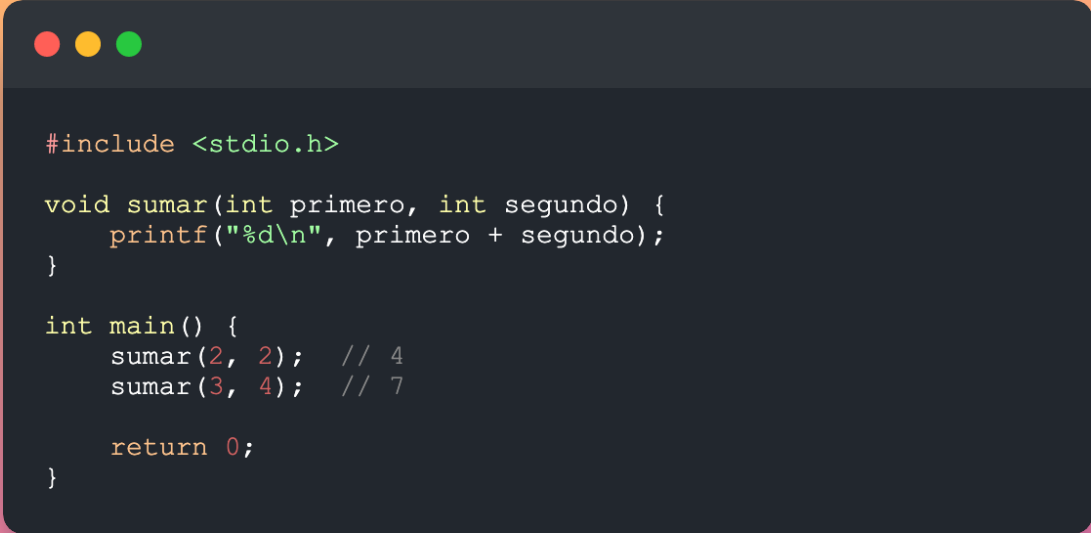
Crea un bucle while que imprima los números del 1 al 10 en la consola.

# 8 Funciones

En C, una función es un bloque de código diseñado para realizar una tarea específica y se puede reutilizar en varias partes el código. Las funciones se pueden definir para realizar cualquier tarea, desde realizar cálculos hasta mostrar mensajes en la pantalla.

Ejemplo:

Esta función toma dos argumentos (primero y segundo) de forma dinámica y los suma. Luego, imprime el resultado en la consola. Esta función se llama dos veces con diferentes argumentos para mostrar los resultados de la suma.



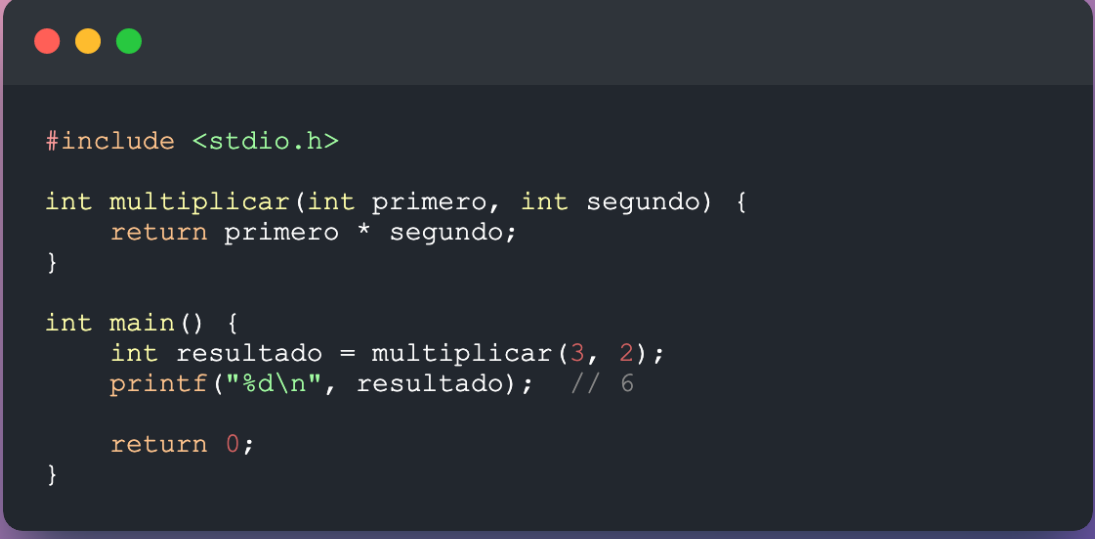
```
#include <stdio.h>

void sumar(int primero, int segundo) {
    printf("%d\n", primero + segundo);
}

int main() {
    sumar(2, 2); // 4
    sumar(3, 4); // 7

    return 0;
}
```

También puedes crear funciones que retornen un valor. Esto significa que una vez que se ejecuta una función, el valor devuelto se puede usar en otra parte del código. Esta función toma dos argumentos (primero y segundo) y los multiplica para devolver el resultado. En este caso, el resultado es 6.




```
#include <stdio.h>

int multiplicar(int primero, int segundo) {
    return primero * segundo;
}

int main() {
    int resultado = multiplicar(3, 2);
    printf("%d\n", resultado); // 6

    return 0;
}
```

Esta función imprime el primer elemento de una lista. En este caso, la lista es una lista de animales, y la función imprime el primer elemento de la lista, que es 'perro'. Funciones como esta son útiles para evitar la repetición de código y para ahorrar tiempo.



```
#include <stdio.h>

void imprimirPrimerElemento(char *lista[]) {
    printf("%s\n", lista[0]);
}

int main() {
    char *animales[] = {"perro", "gato", "tigre"};
    imprimirPrimerElemento(animales); // "perro"

    return 0;
}
```

Finalmente, puedes ver otro ejemplo de una función bastante compleja. Esta función implementa el algoritmo de ordenamiento QuickSort para ordenar una lista de elementos. El algoritmo comienza tomando un elemento de la lista como pivote y luego coloca los elementos menores que el pivote a su izquierda y los elementos mayores a su derecha. Luego, se aplica el mismo algoritmo a las sublistas izquierda y derecha hasta que la lista esté ordenada. No tienes que entender todo lo que hace internamente pero te dará una idea de la complejidad que pueden tener programas como apps con miles de funciones.

```
#include <stdio.h>

void quicksort(int lista[], int n) {
    int pivote = lista[0];
    int izquierda[n];
    int derecha[n];
    int i, j, k;
    i = 0;
    j = 0;
    k = 0;

    if (n <= 1) return;

    for (i = 1; i < n; i++) {
        if (lista[i] < pivote)
            izquierda[j++] = lista[i];
        else
            derecha[k++] = lista[i];
    }

    quicksort(izquierda, j);
    quicksort(derecha, k);

    for (i = 0; i < j; i++) lista[i] = izquierda[i];

    lista[i] = pivote;

    for (i = i + 1; i < n; i++) lista[i] = derecha[i - j -
1];
}

int main() {
    int numeros[] = {23, 45, 16, 37, 3, 99, 22};
    int n = sizeof(numeros) / sizeof(numeros[0]);

    quicksort(numeros, n);

    int i;
    for (i = 0; i < n; i++) printf("%d ", numeros[i]);

    return 0;
}
```



Reto:

Escribe una función que tome dos números como argumentos y devuelva el mayor de los dos.

## 9 Estructuras

Las estructuras en C son colecciones de variables relacionadas bajo un nombre.

Ejemplo:

En este ejemplo creamos una estructura llamada 'Lenguaje' que inicializamos con las propiedades 'nombre' y 'año'. Adicionalmente creamos la función 'descripción' que imprime un texto usando las propiedades previas. Con esa estructura podemos crear los lenguajes 'HTML' y 'CSS'. Ahora podemos crear, en teoría, un número infinito de lenguajes sin reutilizando la misma estructura.

```
#include <stdio.h>

typedef struct Lenguaje {
    char* nombre;
    int año;
} Lenguaje;

void descripcion(Lenguaje lenguaje) {
    printf("%s fue creado en %d\n", lenguaje.nombre,
lenguaje.año);
}

int main() {
    Lenguaje html;
    html.nombre = "HTML";
    html.año = 1993;
    Lenguaje css;
    css.nombre = "CSS";
    css.año = 1996;
    descripcion(html); // 'HTML fue creado en 1993'
    descripcion(css);  // 'CSS fue creado en 1996'

    return 0;
}
```

Reto:

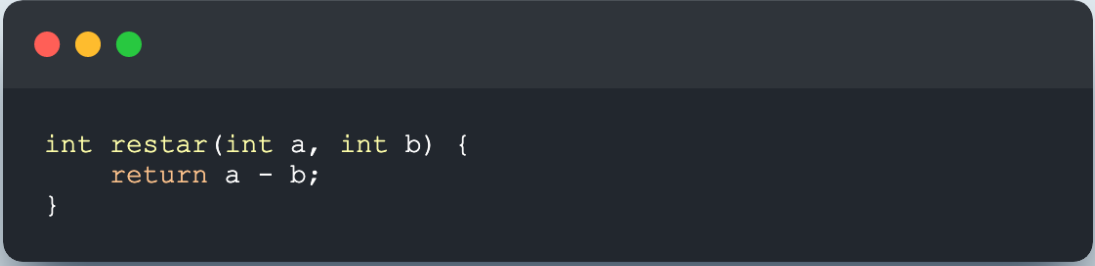
Crea una estructura llamada Auto que tenga propiedades como marca, modelo y año. Luego, inicializa un par de estas estructuras.

# 10 Módulos

Los módulos son una forma de incluir código externo de otro archivo. Esto permite a los desarrolladores reutilizar código y ahorrar tiempo al escribir código.


Ejemplo:

Este archivo define una función que toma dos argumentos (a y b) y luego imprime el resultado de la resta de a y b en la consola. El código exporta la función para que pueda ser usada en otros archivos. Exportar es una forma de compartir código entre archivos. Esto significa que un archivo puede exportar variables, funciones y objetos para que otros archivos los puedan usar.



```
int restar(int a, int b) {  
    return a - b;  
}
```

Y en otro archivo podemos importar y utilizar esta función. Este código importa una función desde el archivo externo y luego la usa para restar 10 menos 2, resultando en 8.



```
#include <stdio.h>
#include "restar.c"

int main() {
    int resultado = restar(10, 2);
    printf("%d\n", resultado); // 8

    return 0;
}
```

Reto:

Crea un archivo llamado que contenga una función para calcular el área de un rectángulo. Luego, importa la función en otro archivo y usala para calcular el área de un rectángulo con lados de 3 y 4.

## 11 Siguientes pasos

### 11.1 Herramientas

- **GCC:** Es un compilador de lenguaje de programación C que se utiliza para compilar programas escritos en lenguaje C. Está disponible para la mayoría de los sistemas operativos, incluidos Windows, Linux y Mac OS X.
- **GDB:** Es un depurador de lenguaje de programación C que se utiliza para depurar programas escritos en lenguaje C. Está disponible para la mayoría de los sistemas operativos, incluidos Windows, Linux y Mac OS X.
- **Make:** Es una herramienta de línea de comandos que se utiliza para compilar

y administrar proyectos de programación C. Está disponible para la mayoría de los sistemas operativos, incluidos Windows, Linux y Mac OS X.

- **Valgrind:** Es una herramienta de depuración de memoria para lenguaje de programación C que se utiliza para detectar errores de memoria en programas escritos en lenguaje C. Está disponible para la mayoría de los sistemas operativos, incluidos Windows, Linux y Mac OS X.
- **GProf:** Es una herramienta de análisis de rendimiento para lenguaje de programación C que se utiliza para analizar el rendimiento de programas escritos en lenguaje C. Está disponible para la mayoría de los sistemas operativos, incluidos Windows, Linux y Mac OS X.

### 11.2 Recursos

- **Stack Overflow:** Stack Overflow es un sitio web de preguntas y respuestas para programadores. Está diseñado para ayudar a los programadores a encontrar y compartir soluciones a problemas comunes de programación. También ofrece una comunidad de programadores donde pueden discutir problemas y soluciones. Enlace: <https://stackoverflow.com/>
- **GitHub:** GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Ofrece una variedad de herramientas para colaborar en proyectos de código abierto y privado, así como herramientas para administrar y compartir proyectos. Enlace: <https://github.com/>
- **TutorialesPoint:** TutorialesPoint es un sitio web de recursos para programadores. Ofrece tutoriales y ejemplos de código para una variedad de lenguajes de programación, incluyendo C. También ofrece una comunidad de programadores donde pueden discutir problemas y soluciones. Enlace: <https://www.tutorialspoint.com/cprogramming/index.htm>

### 11.3 ¿Que viene después?

1. **Practicar:** Una vez que hayas aprendido los conceptos básicos de C, es importante practicar lo que has aprendido. Esto te ayudará a reforzar tus conocimientos y a desarrollar habilidades de programación.
2. **Compilar y ejecutar programas:** Una vez que hayas escrito un programa en C, necesitarás compilarlo y ejecutarlo para ver los resultados. Esto te ayudará a entender mejor cómo funciona el lenguaje.
3. **Aprender conceptos avanzados:** Una vez que hayas dominado los conceptos básicos de C, puedes comenzar a aprender conceptos más avanzados como punteros, estructuras de datos, algoritmos y programación orientada a objetos. Esto te ayudará a convertirte en un programador más experto.
4. **Participar en proyectos:** Una vez que hayas aprendido los conceptos básicos y avanzados de C, puedes comenzar a participar en proyectos reales. Esto te ayudará a aplicar tus conocimientos y a desarrollar habilidades de programación.

### 11.4 Preguntas de entrevista

1. ¿Cuál es la diferencia entre C y C++?
  - La principal diferencia entre C y C++ es que C es un lenguaje de programación procedimental, mientras que C++ es un lenguaje de programación orientado a objetos.
2. ¿Qué es un puntero en C?
  - Un puntero en C es una variable que almacena una dirección de memoria. Esta dirección de memoria se refiere a una ubicación en la memoria en la que se almacena un valor.

## 11 SIGUIENTES PASOS

---

### 3. ¿Qué es una variable en C?

- Una variable en C es una ubicación de memoria que se utiliza para almacenar un valor.

### 4. ¿Qué es una estructura en C?

- Una estructura en C es una colección de variables agrupadas bajo un nombre común.

### 5. ¿Qué es una función en C?

- Una función en C es un bloque de código que realiza una tarea específica.

### 6. ¿Qué es una cadena en C?

- Una cadena en C es una secuencia de caracteres almacenada en una matriz de caracteres.

### 7. ¿Qué es una matriz en C?

- Una matriz en C es una colección de datos almacenados en una sola ubicación de memoria.

### 8. ¿Qué es un puntero a función en C?

- Un puntero a función en C es un puntero que apunta a una función.

### 9. ¿Qué es una enumeración en C?

- Una enumeración en C es una lista de constantes numéricas con nombres asignados.

### 10. ¿Qué es una sentencia switch en C?

- Una sentencia switch en C es una sentencia de control que permite a un programa realizar una acción diferente dependiendo del valor de una variable.

### 11. ¿Qué es una sentencia if en C?

## 11 SIGUIENTES PASOS

---

- Una sentencia if en C es una sentencia de control que permite a un programa realizar una acción diferente dependiendo de si una condición es verdadera o falsa.
12. ¿Qué es una sentencia for en C?
- Una sentencia for en C es una sentencia de control que permite a un programa realizar una acción repetidamente hasta que una condición especificada se cumpla.
13. ¿Qué es una sentencia while en C?
- Una sentencia while en C es una sentencia de control que permite a un programa realizar una acción repetidamente mientras una condición especificada se cumpla.
14. ¿Qué es una sentencia do-while en C?
- Una sentencia do-while en C es una sentencia de control que permite a un programa realizar una acción repetidamente al menos una vez, mientras una condición especificada se cumpla.
15. ¿Qué es una sentencia break en C?
- Una sentencia break en C es una sentencia de control que permite a un programa salir de un bucle.
16. ¿Qué es una sentencia continue en C?
- Una sentencia continue en C es una sentencia de control que permite a un programa saltar a la siguiente iteración de un bucle.
17. ¿Qué es una sentencia goto en C?
- Una sentencia goto en C es una sentencia de control que permite a un programa saltar a una etiqueta específica en el código.



## 11 SIGUIENTES PASOS

---

18. ¿Qué es una sentencia return en C?

- Una sentencia return en C es una sentencia de control que permite a un programa devolver un valor desde una función.

19. ¿Qué es una sentencia sizeof en C?

- Una sentencia sizeof en C es una sentencia de control que permite a un programa determinar el tamaño en bytes de una variable o tipo de datos.

20. ¿Qué es una sentencia sizeof en C?

- Una sentencia sizeof en C es una sentencia de control que permite a un programa determinar el tamaño en bytes de una variable o tipo de datos.