	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 22

No. de práctica(s): Práctica No.6

Integrante(s): Freddy Beckham Cedillo Arias

No. de lista o brigada: No.11

Semestre: 1er Semestre

Fecha de entrega: 27 de septiembre del 2024

Observaciones:

CALIFICACIÓN: _____

“ Practica 6 ”

Entorno y fundamentos del lenguaje C

Objetivo:

Se elaborará programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

Desarrollo:

- Para esta práctica se comenzó dando una breve introducción hacia el entorno y los fundamentos del lenguaje C. Antes de comenzar con una explicación mas detallada se explicaron los conceptos principales:
- Editores de texto: Es cualquier tipo de programa informático que permite a los usuarios crear, cambiar, editar, abrir y ver archivos de texto sin formato . Vienen ya instalados en la mayoría de los sistemas operativos, pero su aplicación principal ha evolucionado desde la toma de notas y la creación de documentos hasta la elaboración de código complejo.
- Lenguaje C: El lenguaje de programación C es uno de los más importantes y ampliamente utilizados en la historia de la informática. Desde su creación, C ha tenido un impacto profundo en la informática. Se convirtió rápidamente en el lenguaje de referencia para el desarrollo de sistemas operativos y software de sistemas. UNIX, escrito casi completamente en C, demostró la viabilidad y las ventajas de este lenguaje, lo que llevó a su adopción generalizada.
- Ciclo de vida software: Es el proceso de planificación, creación, prueba y despliegue de sistemas de información en hardware y software. Es decir la vida útil completa de una aplicación de software, desde el concepto hasta el final de su vida útil.
- Compilador: Se encarga de traducir el código fuente de un lenguaje de programación a un formato ejecutable, realizando diversas etapas de análisis, optimización y generación de código durante el proceso.
- Interfaz de usuario y computadora: Es el punto de interacción y comunicación humano-computadora en un dispositivo. Esto puede incluir pantallas de visualización, teclados, un mouse y la apariencia de un escritorio. También es la forma en que un usuario interactúa con una aplicación o un sitio web.

- Etapas de codificación:

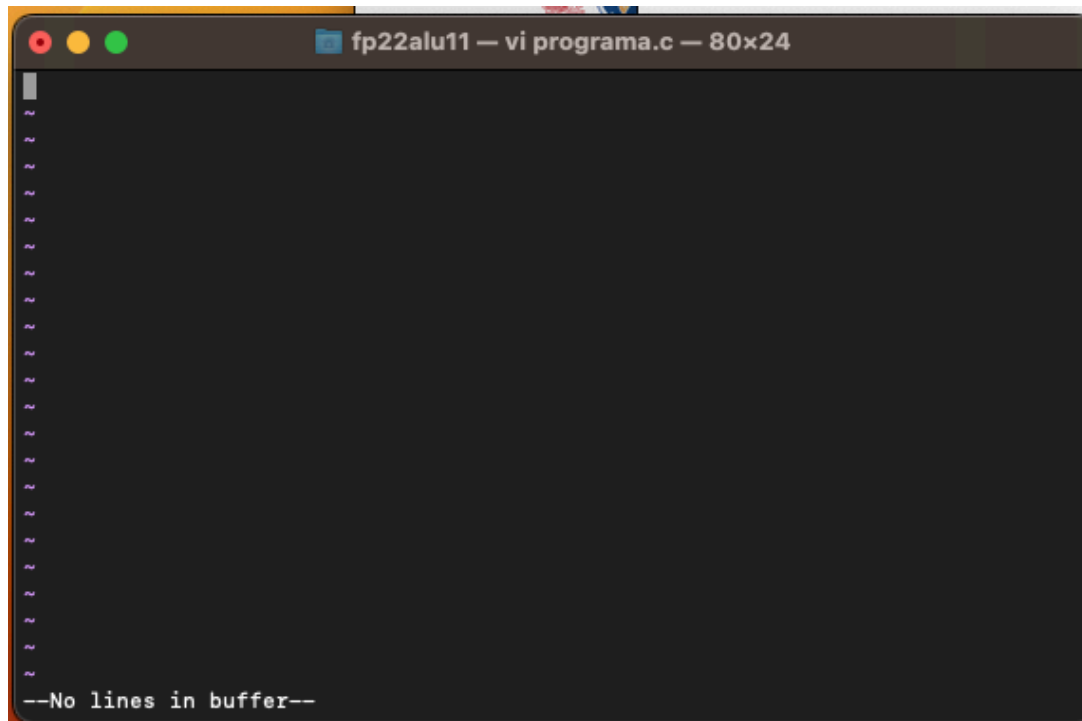
- Edición
- Compilación
- Ejecución

Crea/Abre archivos	vi programa.c
Compilador	gcc programa.c -o programa.out
Ejecutor	./programa.out

- Terminando de aclarar los conceptos principales de la práctica y dar la introducción adecuada continuamos a la práctica:

(Programa 1)

Iniciamos colocando vi programa.C, copiamos el texto de la práctica y para guardar apretamos esc y utilizamos los comandos:




```
fp22alu11 — -bash — 144x35
/bin/bash: wq: command not found
shell returned 127
Press ENTER or type command to continue
[No write since last change]
/bin/bash: wq: command not found
shell returned 127
Press ENTER or type command to continue
[No write since last change]
/bin/bash: wq: command not found
shell returned 127
Press ENTER or type command to continue
Congo41:~ fp22alu11$ gcc programa1.c -o programa.out
programa1.c: In function 'main':
programa1.c:10:11: error: unknown type name 'ya'
   10 | ya sea por línea o por bloque */
      | ^~
programa1.c:10:8: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'por'
   10 | ya sea por línea o por bloque */
      | ^~~
programa1.c:10:14: error: stray '\314' in program
   10 | ya sea por li?nea o por bloque */
      | ^
programa1.c:10:15: error: stray '\201' in program
   10 | ya sea por li?nea o por bloque */
      | ^
Congo41:~ fp22alu11$ vi programa1.c
Congo41:~ fp22alu11$ gcc programa1.c -o programa.out
Congo41:~ fp22alu11$ ./programa.out
Cedillo Arias Freddy BEckhamCongo41:~ fp22alu11$
```

```
fp22alu11 — vi programa.c — 80x24
#include <stdio.h>
int main()
// Comentario por línea
/* Comentario por bloque
que puede ocupar varios renglones */
// Este código compila y ejecuta /* pero no muestra salida alguna debido a que u
n comentario
ya sea por línea o por bloque */
// no es tomado en cuenta al momento
// de compilar el programa,
/* sólo sirve como documentación en el */ /*
código fuente */
return 0;
~
~
~
~
~
~
```

Modo de última línea Se puede acceder a él utilizando el modo comando, pero los comandos no tendrán efecto hasta que se presiona la tecla Enter además de que se visualizará el comando en la última línea del editor. Es posible cancelar el comando con la tecla Esc. Los comandos de última línea se caracterizan porque inician con /, ? o :. Algunos ejemplos son:

- /texto donde la cadena texto será buscada hacia delante de donde se encuentra el cursor.

- ?texto donde la cadena texto será buscada hacia atrás de donde se encuentra el cursor.
 - :q para salir de vi sin haber editado el texto desde la última vez que se guardó.
 - :q! para salir de vi sin guardar los cambios.
 - :w para guardar los cambios sin salir de vi.
 - :w archivo para realizar la orden “guardar como”, siendo archivo el nombre donde se guardará el documento.
 - :wq guarda los cambios y sale de vi.
- Para este ejercicio usamos esc :wq guardar y salir

Modo comando (Figura 2) Es el modo por defecto de vi cuando se abre. Las teclas presionadas ejecutan diversas acciones predeterminadas y no se puede editar el texto libremente. Los comandos son sensitivos a las mayúsculas y a las minúsculas. Algunos ejemplos son:

- ↑ o k mueve el cursor hacia arriba.
- ↓ o j mueve el cursor hacia abajo.
- ← o h mueve el cursor hacia la izquierda.
- → o l mueve el cursor hacia la derecha.
- 1G lleva el cursor al comienzo de la primera línea.
- G lleva el cursor al comienzo de la última línea.
- x borra el carácter marcado por el cursor.
- dd borra o corta la línea donde está el cursor.
- ndd donde n es la cantidad de líneas que se borrarán o cortarán después del cursor.
- D borra o corta desde la posición de cursor hasta el final de la línea.
- dw borra o corta desde la posición del cursor hasta el final de una palabra.
- yy copia la línea donde está el cursor.
- p pega un contenido copiado o borrado.
- u deshace el último cambio.

Para regresar al archivo volvemos a poner vi programa.c

Para que imprima mi nombre en pantalla colocamos el comando

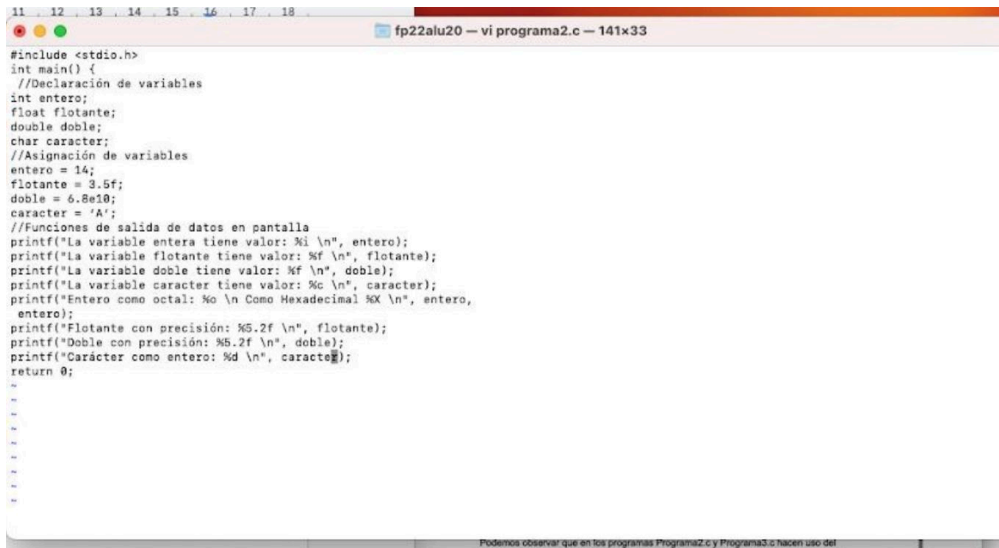
```
printf("Carlos Eduardo Gonzalez Villa"); //Imprime mi nombre en pantalla
```

Ahora compilamos el programa con gcc programa.c -o programa.out

(Programa 2)

Las funciones printf y scanf son herramientas fundamentales en la programación en C para establecer una comunicación efectiva entre el programa y el usuario. La función scanf permite al programa leer datos ingresados por el usuario desde el teclado y almacenarlos en variables, lo que otorga dinamismo y flexibilidad al código. Por otro lado, printf se encarga de mostrar información en pantalla de forma organizada y formateada, permitiendo al programa presentar los resultados de los cálculos o cualquier otro tipo de datos de manera clara y comprensible. En conjunto, estas funciones son esenciales para crear programas interactivos y útiles que puedan realizar tareas diversas según las necesidades del usuario.

El siguiente programa imprime en pantalla diversos tipos de datos utilizando la función printf. Compila y ejecuta bien. creamos un nuevo programa con vi programa2.c y pegamos los comandos de la práctica. Para este programa volveremos a ocupar :wq para guardar y salir y compilamos y ejecutamos el programa.



```
11 12 13 14 15 16 17 18
fp22alu20 - vi programa2.c - 141x33

#include <stdio.h>
int main() {
    //Declaración de variables
    int entero;
    float flotante;
    double doble;
    char caracter;
    //Asignación de variables
    entero = 14;
    flotante = 3.5f;
    doble = 6.8e10;
    caracter = 'A';
    //Funciones de salida de datos en pantalla
    printf("La variable entera tiene valor: %i \n", entero);
    printf("La variable flotante tiene valor: %f \n", flotante);
    printf("La variable doble tiene valor: %f \n", doble);
    printf("La variable caracter tiene valor: %c \n", caracter);
    printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero,
    entero);
    printf("Flotante con precisión: %5.2f \n", flotante);
    printf("Doble con precisión: %8.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);
    return 0;
}
```

Podemos observar que en los programas Programa2.c y Programa3.c hacen uso del

(Programa 3)

En el lenguaje de programación C, los operadores aritméticos son herramientas fundamentales para realizar cálculos matemáticos. Estos operadores permiten realizar operaciones como suma, resta, multiplicación, división y módulo (resto de una división entera). Cada operador tiene un símbolo específico y se utiliza para combinar valores numéricos. Por ejemplo, el operador + se utiliza para sumar dos números, mientras que el operador % se emplea para obtener el resto de una división. La tabla proporcionada muestra los operadores aritméticos más comunes en C junto con ejemplos de su uso.

Además de los operadores aritméticos, C también cuenta con operadores lógicos que permiten realizar comparaciones entre valores. Estos operadores son esenciales para tomar decisiones dentro de un programa, ya que permiten evaluar si una condición es verdadera o falsa. Los operadores lógicos más utilizados son: igual que (==), diferente de (!=), menor que (<), mayor que (>), menor o igual que (<=) y mayor o igual que (>=). Estos operadores se aplican a valores numéricos, caracteres o expresiones más complejas. Por ejemplo, la expresión `5 > 3` es verdadera, mientras que `'a' == 'A'` es falsa, ya que los caracteres 'a' y 'A' son diferentes. Es importante destacar la diferencia entre el operador de igualdad (==) y el operador de asignación (=). El operador de igualdad se utiliza para comparar dos valores y determinar si son iguales, mientras que el operador de asignación se utiliza para almacenar un valor en una variable. Por ejemplo, la expresión `x = 5` asigna el valor 5 a la variable x, mientras que la expresión `x == 5` compara el valor de x con 5 y devuelve verdadero si son iguales. Confundir estos dos operadores es un error común en la programación en C y puede llevar a resultados inesperados. Además de los operadores relacionales que comparan valores, C cuenta con operadores lógicos que permiten combinar expresiones booleanas (verdaderas o falsas). Estos operadores son fundamentales para construir condiciones más complejas y tomar decisiones en nuestros programas.

- ! (NOT): Niega una expresión. Si una expresión es verdadera, ! la convierte en falsa, y viceversa.
- && (AND): Devuelve verdadero solo si ambas expresiones a ambos lados del

operador son verdaderas.

- `||` (OR): Devuelve verdadero si al menos una de las expresiones a ambos lados del operador es verdadera.

Ejemplos:

- `!(a > 10)`: Verdadero si a no es mayor que 10.
- `(a > 5) && (a < 10)`: Verdadero si a está entre 5 y 10.
- `(x == 0) || (y == 0)`: Verdadero si x o y es igual a 0.

También hay operadores a Nivel de Bits, estos operadores permiten manipular los bits individuales de un número. Son más avanzados y requieren una comprensión más profunda de la representación binaria de los números.

- `>>` (Shift Right): Desplaza los bits de un número hacia la derecha.
- `<<` (Shift Left): Desplaza los bits de un número hacia la izquierda.
- `&` (AND): Realiza una operación AND bit a bit entre dos números.
- `|` (OR): Realiza una operación OR bit a bit entre dos números.
- `~` (NOT): Invierte todos los bits de un número.

Ejemplos:

- `8 >> 2`: Desplaza los bits de 8 dos posiciones a la derecha, resultando en 2.
- `5 & 4`: Realiza un AND bit a bit entre 5 y 4, resultando en 4.

El siguiente programa muestra cómo manipular números a nivel de bits:

- Corrimiento de bits a la izquierda y a la derecha
- Operador AND a nivel de bits
- Operador OR a nivel de bits

Utilizando el código proporcionado por la Práctica

```
8
9 #include <stdio.h>
10
11 int main()
12 {
13     short ocho, cinco, cuatro, tres, dos, uno;
14
15     // 8 en binario: 0000 0000 0000 1000
16     ocho = 8;
17     // 5 en binario: 0000 0000 0000 0101
18     cinco = 5;
19     // 4 en binario: 0000 0000 0000 0100
20     cuatro = 4;
21     // 3 en binario: 0000 0000 0000 0011
22     tres = 3;
23     // 2 en binario: 0000 0000 0000 0010
24     dos = 2;
25     // 1 en binario: 0000 0000 0000 0001
26     uno = 1;
27
28     printf("Operadores aritméticos\n");
29     printf("5 modulo 2 = %d\n", cinco%dos);
30     printf("Operadores lógicos\n");
31     printf("8 >> 2 = %d\n", ocho>>dos);
32     printf("8 << 1 = %d\n", ocho<<1);
33     printf("5 & 4 = %d\n", cinco&cuatro);
34     printf("3 | 2 = %d\n", tres|dos);
35
36     printf("\n");
37     return 0;
38 }
39
```

input

```
Operadores aritméticos
5 modulo 2 = 1
Operadores lógicos
8 >> 2 = 2
8 << 1 = 16
5 & 4 = 4
3 | 2 = 3

...Program finished with exit code 0
Press ENTER to exit console.
```

Tarea:

1.- Investigar cual es el dato que se encuentra por default en en lenguaje C(signed o unsigned) y mencionar las características con las que debe crearse una variable.

R=

- Signed : define que el valor de una variable numérica puede ser positivo o negativo. Este modificador puede ser aplicado a los tipos básicos int, char, long, short y __int64. Según se indica en el ejemplo, si no se indica el tipo básico, el modificador por si solo supone signed int.
- Unsigned :Devuelve datos numéricos convertidos al tipo de datos sin signo.

- Una variable en programación se caracteriza por:
 - El nombre único que la identifica dentro del programa
 - El valor guardado que puede cambiar durante la ejecución del programa
 - El tipo de datos que se almacena, como números, textos, estados booleanos, etc.
 - El alcance, que puede ser global, si se puede acceder desde cualquier parte del programa, o local, si solo se puede acceder dentro de un bloque de código.

2.- Crea un programa en el que declares 4 variables haciendo uso de las reglas signed/unsigned, las cuatro variables deben ser solicitadas al usuario(se emplea scanf) y deben mostrarse en pantalla (emplear printf)

```

#include <stdio.h>

int main() {
    signed int a;
    unsigned int b;
    signed char c;
    unsigned char d;

    printf("Introduce un número entero con signo: ");
    scanf("%d", &a);

    printf("Introduce un número entero sin signo: ");
    scanf("%u", &b);

    printf("Introduce un carácter con signo (ASCII): ");
    scanf(" %c", &c);

    printf("Introduce un carácter sin signo (ASCII): ");
    scanf(" %c", &d);

    printf("\nValores introducidos:\n");
    printf("Número entero con signo: %d\n", a);
    printf("Número entero sin signo: %u\n", b);
    printf("Carácter con signo: %c\n", c);
    printf("Carácter sin signo: %c\n", d);

    return 0;
}

```

```

Introduce un número entero con signo: -4
Introduce un número entero sin signo: 6
Introduce un carácter con signo (ASCII): -5
Introduce un carácter sin signo (ASCII):
Valores introducidos:
Número entero con signo: -4
Número entero sin signo: 6
Carácter con signo: -
Carácter sin signo: 5

```

```

...Program finished with exit code 0
Press ENTER to exit console.

```

3.- Comparación entre Editor de Texto y Procesador de Texto(Realizar una tabla comparativa)

	Editor de Texto	Procesador de Texto
Pros	<ul style="list-style-type: none"> • Puedes proteger el documento contra edición. • Diferentes herramientas para revisión • Permite insertar distintos tipos de archivo • Diferentes formas de visualización • Facilidad de copiar y pegar contenido • Personalización del documento • Uso de archivos de la red • Personalización del documento • Uso de archivos de la red • Autoguardado de documentos • Corrector ortográfico • Autoguardado de documentos • Plantillas o diseños preestablecidos • Versatilidad 	Incluyen permitir la edición flexible de textos
Contras	<ul style="list-style-type: none"> • Difícil de utilizar para algunos usuarios • Carece de texto predictivo • Costo de 	Los procesadores de texto se usan comúnmente en escuelas u oficinas, y los más conocidos son el Bloc de Notas, WordPad, Microsoft Word,

	suscripción elevado <ul style="list-style-type: none"> • Fuentes y diccionario limitados • Fallas en la diagramación • Incompatibilidad con otros editores 	OpenOffice y StarOffice
--	---	-------------------------

4.- Indica los comandos utilizados para compilar y para ejecutar un programa en iOS o Linux

R= **gcc nombre_programa.c**

5. Genera un programa y ejecútalo en la interfaz que elijas con el número binario de tu letra inicial del nombre y realiza un corrimiento a la izquierda y uno a la derecha del bit más significativo.

```

#include <stdio.h>

int main() {
    unsigned char letra = 'A'; // Valor ASCII de 'A' es 65 (01000001)

    printf("Valor original: %d (Binario: 01000001)\n", letra);

    // Corrimiento a la izquierda
    unsigned char corrimiento_izq = letra << 1;
    printf("Corrimiento a la izquierda: %d\n", corrimiento_izq);

    // Corrimiento a la derecha
    unsigned char corrimiento_der = letra >> 1;
    printf("Corrimiento a la derecha: %d\n", corrimiento_der);

    return 0;
}

```

```

Valor original: 65 (Binario: 01000001)
Corrimiento a la izquierda: 130
Corrimiento a la derecha: 32

...Program finished with exit code 0
Press ENTER to exit console.

```

Conclusión:

En conclusión el lenguaje C es un lenguaje que nos brinda de forma organizada crear programas de una fácil, ordenada y eficaz; qué resuelvan desde los problemas más fáciles como saber el día actual, hasta realizar cálculos difíciles y tardados; reduciendolos a segundos, desde mi punto de vista este lenguaje al ser versátil y más fácil que otros, ayuda a introducir a gente sin conocimiento a la programación desde lo más básico, además de que es un lenguaje sumamente utilizado en la actividad.

Bibliografía:

Araceli. (2011, 5 diciembre). Ventajas y desventajas de los procesadores de textos [Diapositivas]. SlideShare. Recopilado de :
<https://es.slideshare.net/slideshow/ventajas-y-desventajas-de-los-procesadores-de-textos/10471645#:~:text=Las%20principales%20ventajas%20de%20los,aprender%20a%20utilizar%20estas%20herramientas>

VENTAJAS y DESVENTAJAS. (s. f.). Issuu. Recopilado de:
https://issuu.com/lessvia/docs/buenas_practicas_-_informatica_i_-_2022_definitivo/s/15885610

Modificadores de tipo. (s. f.). Recopilado de:
https://www.zator.com/Cpp/E2_2_3.htm#:~:text=%3E%20%3B-,Descripci%C3%B3n%3A,si%20solo%20supone%20signed%20int

Función UNSIGNED(). (s. f.). Recopilado de:
https://help.highbond.com/helpdocs/analytics/142/scripting-guide/es/Content/lang_ref/functions/r_unsigned.htm#:~:text=Devuelve%20datos%20num%C3%A9ricos%20convertidos%20al%20tipo%20de%20datos%20sin%20signo.

Qué es un compilador: para qué sirve y ejemplos. (2024, 23 abril). Blog Hubspot. Recopilado de:
<https://blog.hubspot.es/website/que-es-compilador#:~:text=compilador%20en%20programaci%C3%B3n-.Un%20compilador%20es%20una%20herramienta%20esencial%20en%20el%20desarrollo%20de,de%20c%C3%B3digo%20durante%20el%20proceso.>

Churchville, F. (2020, 8 septiembre). Interfaz de usuario (UI). ComputerWeekly.es. Recopilado de:
[https://www.computerweekly.com/es/definicion/Interfaz-de-usuario-UI#:~:text=La%20interfaz%20de%20usuario%20\(UI,aplicaci%C3%B3n%20o%20un%20sitio%20web.](https://www.computerweekly.com/es/definicion/Interfaz-de-usuario-UI#:~:text=La%20interfaz%20de%20usuario%20(UI,aplicaci%C3%B3n%20o%20un%20sitio%20web.)

Thales Group. (s. f.). Application Lifecycle Management. Recopilado de:
<https://cpl.thalesgroup.com/es/software-monetization/application-lifecycle-management#:~:text=El%20ciclo%20de%20vida%20de%20una%20aplicaci%C3%B3n%20es%20la%20%22vida,que%20se%20retira%20del%20uso.>

Brooks, R. (2023, 17 agosto). Tech basics: An introduction to text editors - University of York. University Of York. Recopilado de: <https://online.york.ac.uk/tech-basics-an-introduction-to-text-editors/>

Cuadrado, G. C. (2024, 7 agosto). Qué es C: Todo lo que debes saber. OpenWebinars.net. Recopilado de: <https://openwebinars.net/blog/que-es-c/>