



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Ing. Karina García Morales

*Asignatura:* Fundamentos de la programación

*Grupo:* 20

*No. de práctica(s):* Practica 06

*Integrante(s):* Francisco Javier Gómez Mendoza

*No. de lista o brigada:* 22

*Semestre:* 2023-1

*Fecha de entrega:* 08 de noviembre del 2023

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

# Entorno y fundamentos del lenguaje C

## Objetivo:

El alumno 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:

El software tiene un ciclo en el hasta el momento sólo habíamos llegado a la parte del pseudocódigo, ahora pasaremos a la etapa de codificación y las pruebas.



## Modo comando

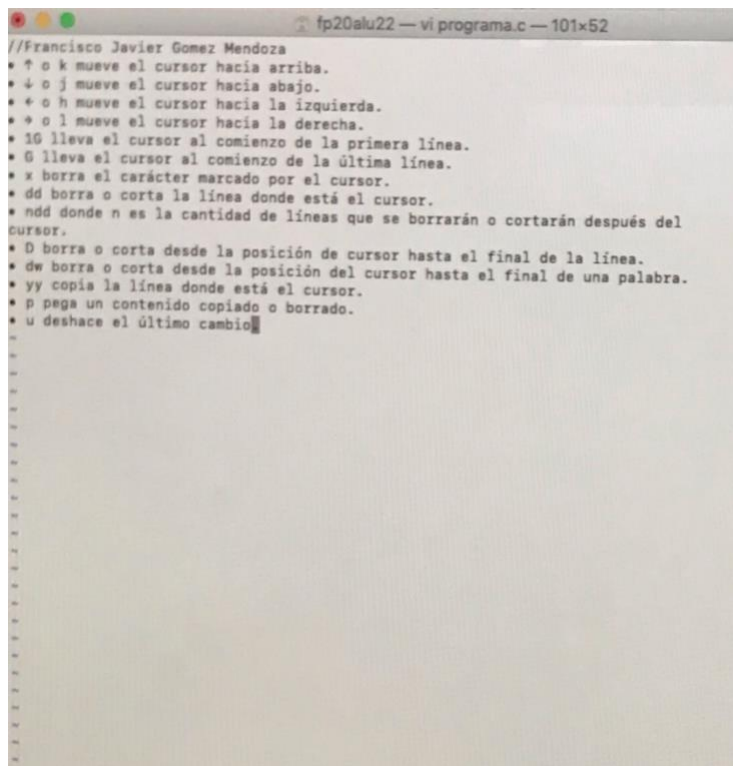
Como parte del trabajo en el laboratorio, aprendimos a usar diversas teclas que al ser presionadas pueden ejecutar acciones determinadas. Los comandos son sensitivos a las mayúsculas y a las minúsculas.

- ↑ 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.

Figura 2: vi en modo comando.

Área/Departamento: Laboratorio de computación salas A y B

Con ayuda de la practica solo lo copiamos y pegamos en nuestro demostrado y probamos las funciones de cada comando.



```
//Francisco Javier Gomez Mendoza
* ↑ 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
```

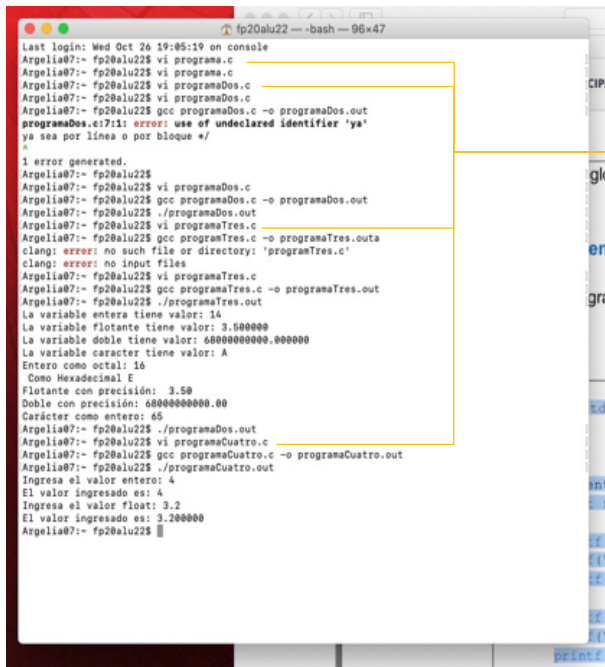
## Editor Visual Interface de GNU/Linux (vi)

Con este comando (vi), podremos crear y nombrar un archivo

Para iniciar vi, debe teclearse desde la línea de comandos:

```
vi nombre_archivo[.ext]
```

En la práctica lo aplicamos para nombrar nuestros programas



```
Last login: Wed Oct 26 19:05:19 on console
Argelia@7:~$ fp20alu225 vi programa.c
Argelia@7:~$ fp20alu225 vi programa.c
Argelia@7:~$ fp20alu225 vi programaDos.c
Argelia@7:~$ fp20alu225 vi programaDos.c
Argelia@7:~$ fp20alu225 gcc programaDos.c -o programaDos.out
Argelia@7:~$ fp20alu225 gcc programaDos.c -o programaDos.out
programaDos.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Argelia@7:~$ fp20alu225
Argelia@7:~$ fp20alu225 vi programaDos.c
Argelia@7:~$ fp20alu225 gcc programaDos.c -o programaDos.out
Argelia@7:~$ fp20alu225 ./programaDos.out
Argelia@7:~$ fp20alu225 vi programaTres.c
Argelia@7:~$ fp20alu225 gcc programaTres.c -o programaTres.out
clang: error: no such file or directory: 'programTres.c'
clang: error: no input files
Argelia@7:~$ fp20alu225 vi programaTres.c
Argelia@7:~$ fp20alu225 gcc programaTres.c -o programaTres.out
Argelia@7:~$ fp20alu225 ./programaTres.out
La variable entera tiene valor: 14
La variable flotante tiene valor: 3.500000
La variable doble tiene valor: 6000000000.000000
La variable caracter tiene valor: A
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 6000000000.00
Carácter como entero: 65
Argelia@7:~$ fp20alu225 ./programaDos.out
Argelia@7:~$ fp20alu225 vi programaCuatro.c
Argelia@7:~$ fp20alu225 gcc programaCuatro.c -o programaCuatro.out
Argelia@7:~$ fp20alu225 ./programaCuatro.out
Ingresa el valor entero: 4
El valor ingresado es: 4
Ingresa el valor float: 3.2
El valor ingresado es: 3.200000
Argelia@7:~$ fp20alu225
```

En la práctica usamos el comando (vi) para nombrar nuestro archivo, ej. Programa.c, ProgramaDos.c, ProgramaTres.c, PprogramaCuatro.c.

### Modo de última línea

- /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.

Los elementos que se encuentran marcados de color verde son comandos que usamos en la práctica, los cuales también especifican su función.

```
#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 un 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;
}
```

Aquí hay un ejemplo del uso de :wq el cual se uso para guardar lo que esta dentro de mi programa dos y salir sin riesgo alguno

:wq||

División de Ingeniería Eléctrica

## Compiladores

Una vez codificado un programa en C en algún editor de texto, éste debe ser leído por un programa que produzca un archivo ejecutable. A este programa se le conoce como compilador.

### Compiladores gcc

En la practica para compilar nuestros programas usamos los compiladores gcc.

Con la sintaxis siguiente

**gcc calculadora.c -o calculadora.out**

En forma de que se escribe **gcc** nombre del archivo con un guion ( - ) un punto (.) y un **aut**

```
Last login: Wed Oct 26 19:05:19 on console
Argelia07:~$ fp20alu22$ vi programa.c
Argelia07:~$ fp20alu22$ vi programa.c
Argelia07:~$ fp20alu22$ vi programaDos.c
Argelia07:~$ fp20alu22$ vi programaDos.c
Argelia07:~$ fp20alu22$ gcc programaDos.c -o programaDos.out
programaDos.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Argelia07:~$ fp20alu22$ vi programaDos.c
Argelia07:~$ fp20alu22$ gcc programaDos.c -o programaDos.out
Argelia07:~$ fp20alu22$ ./programaDos.out
Argelia07:~$ fp20alu22$ vi programaTres.c
Argelia07:~$ fp20alu22$ gcc programTres.c -o programaTres.outa
clang: error: no such file or directory: 'programTres.c'
clang: error: no input files
Argelia07:~$ fp20alu22$ vi programaTres.c
Argelia07:~$ fp20alu22$ gcc programaTres.c -o programaTres.out
Argelia07:~$ fp20alu22$ ./programaTres.out
La variable entera tiene valor: 14
La variable flotante tiene valor: 3.500000
La variable doble tiene valor: 68000000000.000000
La variable caracter tiene valor: A
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 68000000000.00
Carácter como entero: 65
Argelia07:~$ fp20alu22$ ./programaDos.out
Argelia07:~$ fp20alu22$ vi programaCuatro.c
Argelia07:~$ fp20alu22$ gcc programaCuatro.c -o programaCuatro.out
Argelia07:~$ fp20alu22$ ./programaCuatro.out
Ingresar el valor entero: 4
El valor ingresado es: 4
Ingresar el valor float: 3.2
El valor ingresado es: 3.200000
Argelia07:~$ fp20alu22$
```

En el caso de la práctica usamos los compiladores gcc para justamente complicar nuestros tres programas

## Ejecución

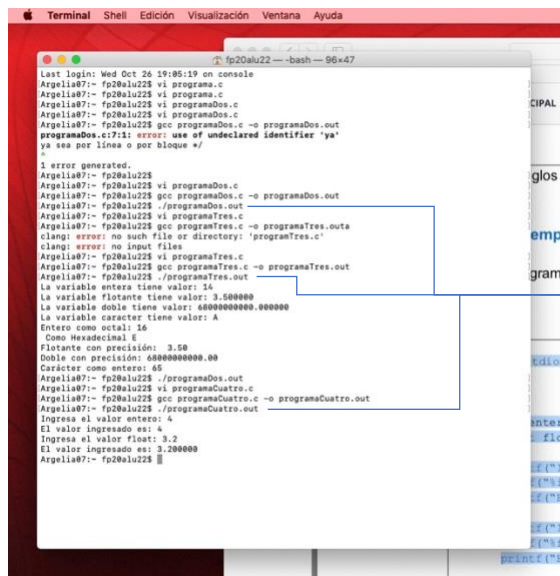
La ejecución es la etapa que sigue después de haber compilado el programa. Y este nos ayuda a dar una vista previa de lo que hemos codificado.

Hay una forma para ejecutar el programa de la siguiente forma y con este comando.

```
./calculadora.out
```

La cual va con la siguiente nomenclatura punto (.) barra inclinada o slash (/), seguido por el nombre del archivo y por ultimo un **.out**

Un ejemplo dentro de la actividad en clase de la practica son los siguientes.



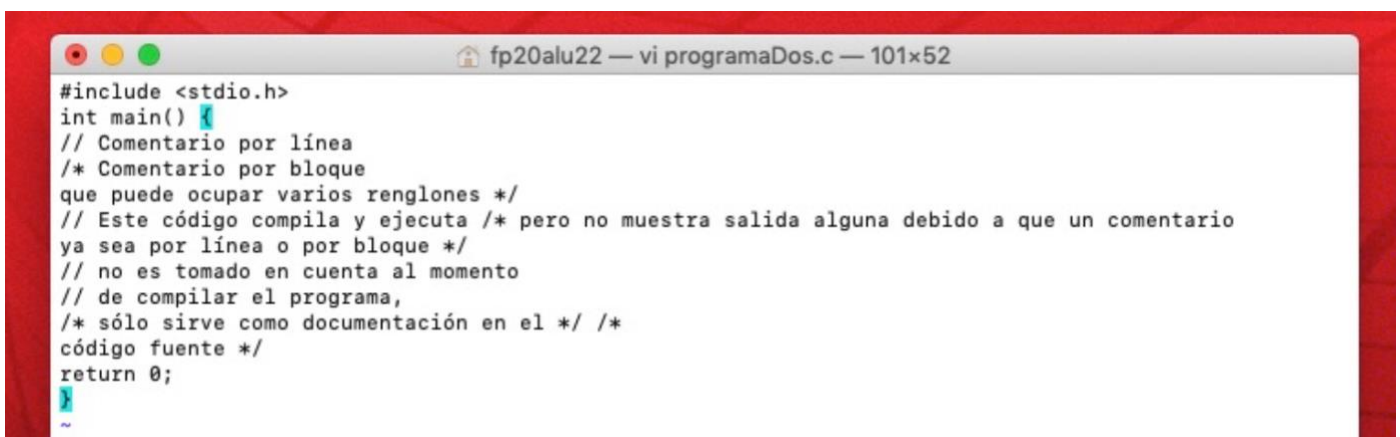
En este caso usamos los comandos .  
/nombre del archivo. Out, y asi ejecutar  
nuestros programas

## Comentarios

Los comentarios son únicamente para el programador, puesto que el solo los puede ver, dentro de la codificación, siempre y cuando se use la nomenclatura correcta, para agregar estos comentarios

El comentario por línea inicia cuando se insertan los símbolos // y termina con el salto de línea (hasta donde termine el renglón).

Dentro de la practica creamos un archivo donde se hace uso de los comentarios y lo nombramos **vi Programa.c**



**Código declaración de variables (se declaran variables y se asignan datos)**

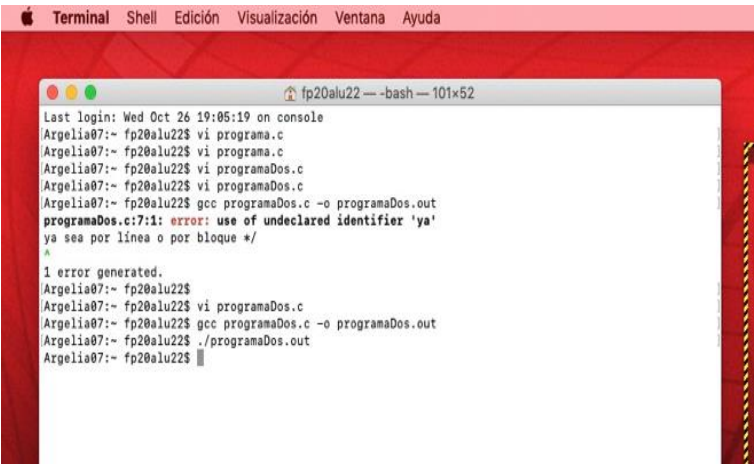
El siguiente programa muestra la manera en la que se declaran y asignan variables de diferentes tipos: numéricas (enteros y reales) y tipo carácter.

Con la siguiente tabla podemos ver como declarar nuestras variables en lenguaje c.

**Tabla 2: Tipos de datos.**

Tipo	Descripción	Espacio en Memoria	Rango
<i>int</i>	Cantidad entera	2 bytes o una palabra* (varía de un compilador a otro)	-32 767 a 32 766
<i>char</i>	Carácter	1 byte	-128 a 127
<i>float</i>	Número en punto flotante (un número que incluye punto decimal y/o exponente)	1 palabra* (4 bytes)	-3.4E <sup>38</sup> a 3.4 E <sup>38</sup>
<i>double</i>	Número en punto flotante de doble precisión (más cifras significativas y mayor valor posible del exponente)	2 palabras* (8 bytes)	-1.7E <sup>308</sup> a 1.7E <sup>308</sup>

En la practica copiamos y solo nos encargamos de de compilar y ejecutar



```
Terminal Shell Edición Visualización Ventana Ayuda
fp20alu22 — bash — 101x52
Last login: Wed Oct 26 19:05:19 on console
Argelia07:~ fp20alu22$ vi programa.c
Argelia07:~ fp20alu22$ vi programa.c
Argelia07:~ fp20alu22$ vi programaDos.c
Argelia07:~ fp20alu22$ vi programaDos.c
Argelia07:~ fp20alu22$ gcc programaDos.c -o programaDos.out
programaDos.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Argelia07:~ fp20alu22$
Argelia07:~ fp20alu22$ vi programaDos.c
Argelia07:~ fp20alu22$ gcc programaDos.c -o programaDos.out
Argelia07:~ fp20alu22$ ./programaDos.out
Argelia07:~ fp20alu22$
```

```
#include <stdio.h>

int main() {
    short enteroNumero1 = 115;
    signed int enteroNumero2 = 55;
    unsigned long enteroNumero3 = 789;
    char caracterA = 65;
    char caracterB = 'B';

    float puntoFlotanteNumero1 = 89.8;

    unsigned double puntoFlotanteNumero2 = 238.2236;
    return 0;
}
```

Tambien en el momento de compilar nuestro ProgramaDos Detectamos un error de comillas, los cuales corregimos y asi logramos ejecutar el programa



## Código que ejemplifica el uso de la función printf.

En este programa usamos algunos comandos del segundo programa, pero en esta ocasión el programa 3 usamos el comando **printf** el cual es el escribir de nuestro pseudocódigo y imprécamos del diagrama de flujo, como bien lo mencionamos antes este solo va imprimir en pantalla lo que le escribimos

De igual manera con ayuda de la práctica sólo tuvimos que compilar y ejectutar.

```
Argelia07:~ fp20alu22$ vi programaTres.c
Argelia07:~ fp20alu22$ gcc programaTres.c -o programaTres.out
Argelia07:~ fp20alu22$ ./programaTres.out
```

```
#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: %5.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);

    return 0;
}
```

## Código que ejemplifica el uso de la función scanf

El siguiente programa muestra el almacenamiento de datos en variables. Compila y ejecuta bien.

En esta ocasión usamos lo que aplicamos en los tres programas anteriores, a exepcion de uno que es scanf el cual nos ayuda a compilar los datos que el usuarios nos da para continuar con el proceso de nuestro programa.

En esta ocasión no solo compilamos y ejecutamos, pues tambien le dimos valores a scanf al momento de compilar

```
Argelia07:~ fp20alu22$ vi programaCuatro.c
Argelia07:~ fp20alu22$ gcc programaCuatro.c -o programaCuatro.out
Argelia07:~ fp20alu22$ ./programaCuatro.out
Ingresa el valor entero: 4
El valor ingresado es: 4
Ingresa el valor float: 3.2
El valor ingresado es: 3.200000
Argelia07:~ fp20alu22$ █
```

```
#include <stdio.h>

int main()
{
    int entero;
    float flotante;

    printf("Ingresa el valor entero: ");
    scanf("%i", &entero);
    printf("El valor ingresado es: %d\n", entero);

    printf("Ingresa el valor float: ");
    scanf("%f", &flotante);
    printf("El valor ingresado es: %f\n", flotante);

    return 0;
}
```



## Tarea

1.- Investigar cual es el dato que se encuentra por default en en lenguaje C( signed o unsigned)

### signed

Le indica a la variable que va a llevar signo. Es el utilizado por defecto.

2.- Indicar que sucede cuando en una variable tipo carácter se emplea el formato %d, %i, %o, %x

Se emplean cuando el número es ENTERO.

3.- Mencionar las características con las que debe crearse una variable

La variable se debe declarar segun sea el caso, como Real, Entera o de carácter

4.- ¿Cuál es la diferencia entre variable estática y constante?

Variable costante	Variable estatica
constantes no se pueden cambiar.	variables estáticas tienen más que ver con cómo se asignan y dónde son accesibles.

5.- Menciona en que momento empleas los dos tipos de diferentes ( < > != )

Para designar que es menor que <

Para designar que debe ser mayor que >

Usamos este símbolo para decir que es diferente que !=

6.- 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

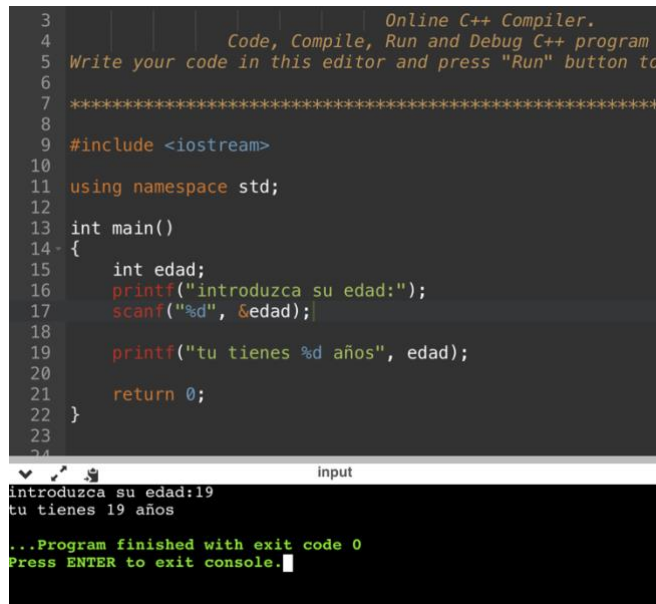
pantalla (emplear printf)

```
10
11 using namespace std;
12
13 int main()
14 {
15     signed int numero1, numero2;
16     unsigned int numero3, numero4;
17
18     printf("introduca el 1er numero");
19     scanf("%d", &numero1);
20
21     printf("introduca el 2do numero");
22     scanf("%d", &numero2);
23
24     printf("introduzca el 3er numero");
25     scanf("%d", &numero3);
26
27     printf("introduzca el 4to numero");
28     scanf("%d", &numero4);
29
30     printf("su 1er numero es: %d", numero1);
31     printf("su 2do numero es: %d", numero2);
32     printf("su 3er numero es: %d", numero3);
33     printf("su 4to numero es: %d", numero4);
34
35     return 0;
36 }
37
```

input

```
introduca el 2do numero
-70
introduzca el 3er numero
18
introduzca el 4to numero
-666
su 1er numero es: 25su 2do numero es: -70su 3er numero es: 18su 4to numero es: -666
...Program finished with exit code 0
Press ENTER to exit console.[]
```

7.- Crea un programa que le solicite su edad al usuario, leer el datos( emplear scanf) y mostrarlo en pantalla.



```
3 | Online C++ Compiler.
4 | Code, Compile, Run and Debug C++ program
5 | Write your code in this editor and press "Run" button to
6 |
7 | *****
8 |
9 | #include <iostream>
10 |
11 | using namespace std;
12 |
13 | int main()
14 | {
15 |     int edad;
16 |     printf("introduzca su edad:");
17 |     scanf("%d", &edad);
18 |
19 |     printf("tu tienes %d años", edad);
20 |
21 |     return 0;
22 | }
23 |
24 |
```

input

```
introduzca su edad:19
tu tienes 19 años

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

8.- Revisar y colocar cuando se emplea MOD y cuando se emplea % (pseudocódigo y codificación) agrega un ejemplo de su uso.

MOD: función MOD() esta función para probar si dos números pueden dividirse de manera exacta o aislar el resto de un cálculo de división: Ej.  $20\%7$  da como resultado 6 que es el resto de la división entre 20 y 7

%. Se emplea cuando necesitamos decirle a a nuestro programa que después del paciente se guardara lo que el usuario no proporcione

Ej.

```
printf("introduzca su edad:");
scanf("%d", &edad);
```

Pueden usarse para distintos datos como enteros, reales o de tipo carácter, etc.

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

	EDITOR DE TEXTOS	PROCESADOR DE TEXTOS
¿Qué es?	Un procesador de textos es una aplicación informática que permite crear documentos de texto en una computadora.	Un editor de textos es un programa que permite escribir y modificar archivos digitales compuestos por textos sin formato.
Diferencias	Un editor de texto es un programa que permite crear y modificar archivos digitales compuestos por texto sin formato	permiten poner formato al texto como regla general , trabajar con distintos tamaños y tipos de letra entre otras, además de brindar la posibilidad de intercalar.
Ejemplos	Word Pad y NotePad para Windows y SimpleText y TextEdit para Mac.28 feb 2022	OpenOffice Writer, WordPad, Pages, Microsoft Word, AbiWord, Google Docs, Overleaf.

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

Para compilar

gcc nombredelprograma.c -o nombredelprograma.out

Para ejecutar

./nombredelprograma.out

## 12.- Genera un programa que solicite dos variables enteras al usuario y realice las 4 operaciones básicas, compila y ejecuta el programa utilizando terminal y los comandos indicados para cada

```
8
9 #include <iostream>
10
11 using namespace std;
12
13 int main()
14 {
15     float number1, number2, suma, resta, multiplicacion, division;
16     printf("introduce el 1er numero");
17     scanf("%f", &number1);
18     printf("introduce el 2do numero menor a 1er numero");
19     scanf("%f", &number2);
20
21     suma = number1 + number2;
22     printf("su resultado de la suma es: %f", suma);
23
24     resta = number1 - number2;
25     printf("su resultado de la resta es: %f", resta);
26
27     multiplicacion = number1 * number2;
28     printf("su resultado de la multiplicacion es: %f", multiplicacion);
29
30     division = number1 / number2;
31     printf("su resultado de la division es: %f", division);
32     return 0;
33 }
34
```

input

introduce el 1er numero  
8  
introduce el 2do numero menor a 1er numero  
su resultado de la suma es: 10.000000su resultado de la resta es: 6.000000su resultado de la m  
multiplicacion es: 16.000000su resultado de la division es: 4.000000  
...Program finished with exit code 0  
Press ENTER to exit console.

## Conclusiones

- Esta práctica me gusto mucho, soy muy malo para recordar comandos pero en esta práctica no se uso mucho la memoria.
- Aprendí a usar comando que tienen una función muy especifica, puesto que nunca antes había hecho algo así se me hizo algo muy divertido.
- También había algo en contra pues muchas cosas en la practica no conocía, pero ahora que ya tuvimos clases logre entender mejor las funciones de unas cosas, como el printf y scanf que no sabia como pronunciarlos.
- Tengo que admitir que la codificación se me hace más fácil que los procesos anteriores como hacer el diagrama de flujo, pero sin el diagrama de flujo no podemos tener el código fuente.
- Había cosas que no había notado en clase que justo en la practica estan, siempre que realizo las practica hay lagunas las cuales muchas de ellas desaparecen por que en la practica me quedan mas claras esas dudas que me cree en la clase.
- Al programar se me hizo difcil por que lo hice atraves de un iPad, así que use un programa web, tambien me ayudo un compañero, por que no funciono mi primer programa, eso fue lo que se me dificultó un poco, tampoco hice uso del if, así que fue un poco más fácil.

## Referencias

- Hugo Delgado. (2022, 22 agosto). Calificadores de Datos en C: signed, unsigned, short y long. Diseño Web akus.net. <https://disenowebakus.net/calificadores-datos.php>
- ¿Qué es una CONSTANTE en programación? (2019, 6 mayo). Lenguajes de programación. <https://lenguajesdeprogramacion.net/diccionario/que-es-una-constante-en-programacion/>
- Online C++ Compiler - online editor. (s. f.). GDB online Debugger. [https://www.onlinegdb.com/online\\_c++\\_compiler](https://www.onlinegdb.com/online_c++_compiler)
- Procesador y editor de texto - dulce101CEL. (s. f.). <https://sites.google.com/site/dulce101cel/procesador-y-editor-de-texto>
- Guías BibUpo: Procesadores de texto: Tipos de procesadores de texto. (s. f.). [https://guiasbib.upo.es/procesadores\\_de\\_texto/tipos](https://guiasbib.upo.es/procesadores_de_texto/tipos)
- Programación en C/Compilar un programa/Linux - Wikilibros. (s. f.-b). [https://es.wikibooks.org/wiki/Programaci%C3%B3n\\_en\\_C/Compilar\\_un\\_programa/Linux](https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C/Compilar_un_programa/Linux)
- Laboratorios Salas A y B (unam.mx)

