



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor(a):* Ing. Karina García Morales

*Asignatura:* Fundamentos de Programación (L)

*Grupo:* 22

*No. de práctica(s):* Práctica 6: Entorno y fundamentos del lenguaje C

*Integrante(s):* González Márquez José Luis

*No. de lista o brigada:* 20

*Semestre:* 2026-1

*Fecha de entrega:* Jueves 16 de octubre de 2025

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

**Objetivo:** El alumnado 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:** Programar en lenguaje C es el proceso de escribir instrucciones en un archivo de texto plano usando un editor (como vi, nano o Visual Studio Code). Este código, que se organiza alrededor de una función principal `main()`, utiliza variables de distintos tipos (como `int`, `char`, `float`), palabras reservadas y comentarios para ser legible.

Una vez escrito, se usa un compilador (como gcc) que traduce el código fuente a un programa ejecutable que la computadora puede entender, avisando si hay errores de sintaxis en el proceso. Finalmente, este programa se ejecuta desde una línea de comandos para que realice su tarea, etapa en la que se deben hacer pruebas para encontrar errores lógicos.

Para simplificar todo este ciclo de editar, compilar y ejecutar, se pueden usar los Entornos de Desarrollo Integrado (IDE) como Code Blocks o Xcode, que combinan todas estas herramientas en una sola aplicación.

Tabla 1: Palabras reservadas

<i>auto</i>	<i>double</i>	<i>int</i>	<i>struct</i>
<i>break</i>	<i>else</i>	<i>long</i>	<i>switch</i>
<i>case</i>	<i>enum</i>	<i>register</i>	<i>typedef</i>
<i>char</i>	<i>extern</i>	<i>return</i>	<i>union</i>
<i>const</i>	<i>float</i>	<i>short</i>	<i>unsigned</i>
<i>continue</i>	<i>for</i>	<i>signed</i>	<i>void</i>
<i>default</i>	<i>goto</i>	<i>sizeof</i>	<i>volatile</i>
<i>do</i>	<i>if</i>	<i>static</i>	<i>while</i>



Figura 1: Ciclo de vida del software.

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)	-32767 a 32766
<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>

## **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..**

En el lenguaje de programación C, los tipos de datos enteros como int, short, long y long long son signed (con signo) por defecto. Esto significa que pueden representar tanto números positivos como negativos.

Por ejemplo, si declaras una variable como int, el compilador la tratará como signed int. La única excepción a esta regla es el tipo char. El estándar de C no especifica si char es signed o unsigned por defecto, y esto puede variar dependiendo del compilador y la arquitectura del sistema.

### **Diferencia entre signed y unsigned**

signed: Utiliza uno de sus bits para representar el signo del número (positivo o negativo). Por ejemplo, un signed int de 32 bits puede almacenar valores en un rango aproximado de -2 mil millones a +2 mil millones.

unsigned: Utiliza todos sus bits para representar el valor del número, por lo que solo puede almacenar valores no negativos (cero y positivos). Un unsigned int de 32 bits puede almacenar valores en un rango de 0 a aproximadamente 4 mil millones.

### **Características para Crear una Variable en C**

Para crear una variable en C, debes seguir un conjunto de reglas para nombrar el identificador (el nombre de la variable). Aquí están las reglas y convenciones más importantes:

#### **Reglas Obligatorias**

Caracteres permitidos: El nombre de una variable solo puede contener letras (mayúsculas y minúsculas), números y el carácter de guion bajo (\_).

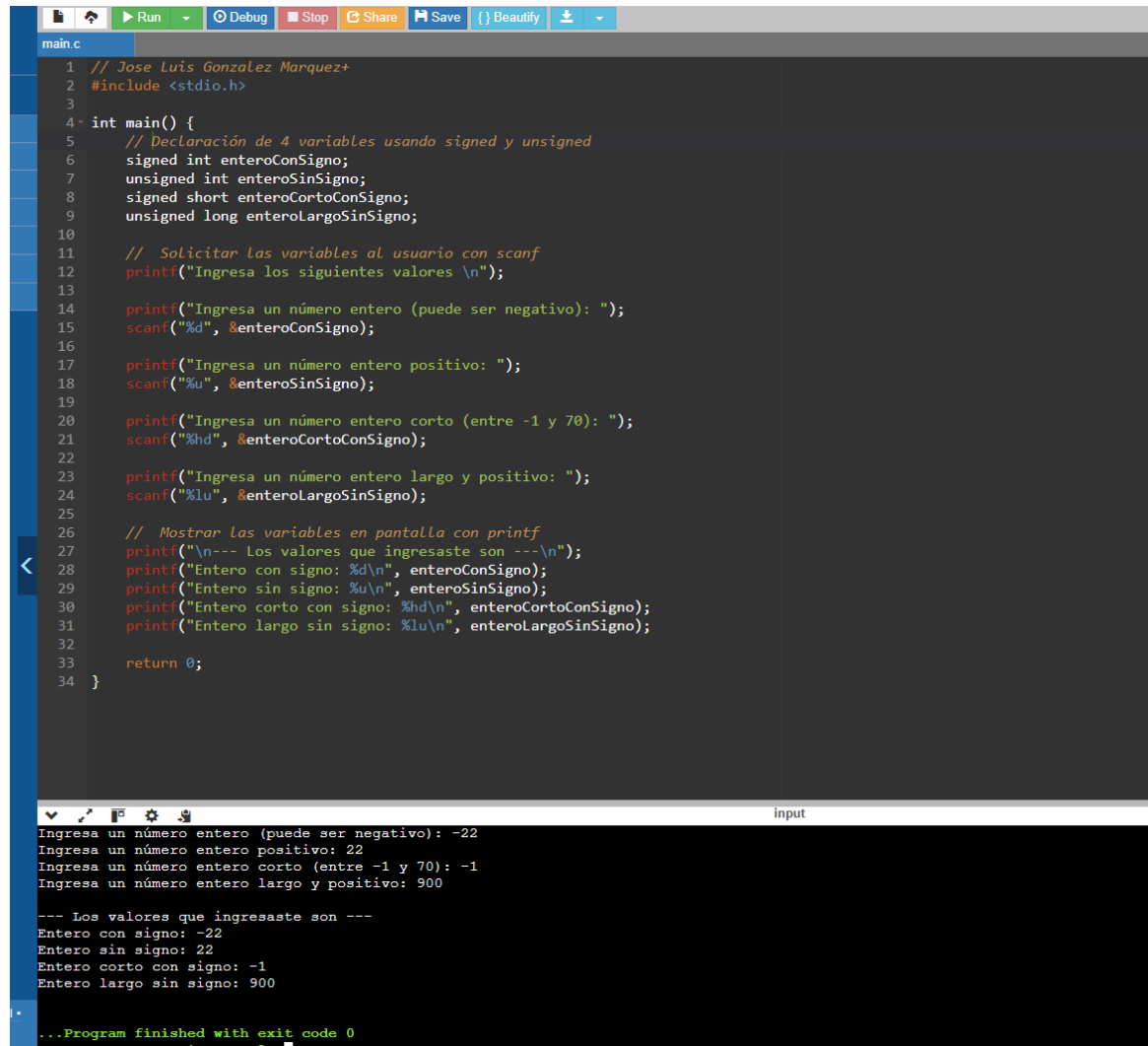
Primer carácter: El nombre de la variable debe comenzar con una letra o un guion bajo. No puede comenzar con un número.

Sin palabras clave: No puedes usar ninguna de las palabras reservadas del lenguaje C como nombre de variable (por ejemplo, int, while, for, return, etc.).

Sin espacios: El nombre de la variable no puede contener espacios.

Sensibilidad a mayúsculas y minúsculas: C distingue entre mayúsculas y minúsculas. Por lo tanto, miVariable, mivariable y MIVARIABLE serían tres variables diferentes.

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



```
main.c
1 // Jose Luis Gonzalez Marquez+
2 #include <stdio.h>
3
4 int main() {
5     // Declaración de 4 variables usando signed y unsigned
6     signed int enteroConSigno;
7     unsigned int enteroSinSigno;
8     signed short enteroCortoConSigno;
9     unsigned long enteroLargoSinSigno;
10
11     // Solicitar las variables al usuario con scanf
12     printf("Ingresa los siguientes valores \n");
13
14     printf("Ingresa un número entero (puede ser negativo): ");
15     scanf("%d", &enteroConSigno);
16
17     printf("Ingresa un número entero positivo: ");
18     scanf("%u", &enteroSinSigno);
19
20     printf("Ingresa un número entero corto (entre -1 y 70): ");
21     scanf("%hd", &enteroCortoConSigno);
22
23     printf("Ingresa un número entero largo y positivo: ");
24     scanf("%lu", &enteroLargoSinSigno);
25
26     // Mostrar las variables en pantalla con printf
27     printf("\n--- Los valores que ingresaste son ---\n");
28     printf("Entero con signo: %d\n", enteroConSigno);
29     printf("Entero sin signo: %u\n", enteroSinSigno);
30     printf("Entero corto con signo: %hd\n", enteroCortoConSigno);
31     printf("Entero largo sin signo: %lu\n", enteroLargoSinSigno);
32
33     return 0;
34 }
```

input

```
Ingresa un número entero (puede ser negativo): -22
Ingresa un número entero positivo: 22
Ingresa un número entero corto (entre -1 y 70): -1
Ingresa un número entero largo y positivo: 900

--- Los valores que ingresaste son ---
Entero con signo: -22
Entero sin signo: 22
Entero corto con signo: -1
Entero largo sin signo: 900

...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)

	A	B	C
1	Características	Editor de Texto	Procesador de Texto
2	Propósito Principal	Escribir y editar texto sin formato . Ideal para programación y	Crear, dar formato y diseñar documentos visualmente atractivos.
3	Contenido Multimedia	No permite insertar imágenes, gráficos, tablas o videos.	Permite y facilita la inserción de imágenes, gráficos, tablas y otros elementos multimedia.
4	Formato de Texto	No maneja formato. El texto es plano, sin estilos, fuentes o co	Amplias capacidades de formato: negritas, cursivas, diferentes fuentes, tamaños, colores, etc.
5	Uso Principal	Programación y escritura de código fuente. Creación de archivos de configuración. Tomar notas rápidas.	Redacción de informes, cartas, currículums. Creación de documentos académicos y profesionales. Elaboración de folletos y otros materiales de diseño simple
6	Funcionalidades Clave	Resaltado de sintaxis para lenguajes de programación. Autocompletado de código. Búsqueda y reemplazo avanzado . Ligero y rápido.	Corrector ortográfico y gramatical. Diseño de página (márgenes, encabezados, pies de página). Plantillas predefinidas. Generación de índices y tablas de contenido.
7	Tipos de Archivo	.txt, .html, .css, .js, .py, .java, etc. (archivos de texto plano).	.docx, .odt, .rtf, .pdf (archivos con formato enriquecido).
8	Rendimiento	Muy rápidos y consumen pocos recursos del sistema.	Más lentos y consumen más recursos debido a sus complejas funcionalidades.
9	Ejemplos populares	Visual Studio Code Sublime Text Notepad++ Bloc de Notas (Windows) Atom	Microsoft Word Google Docs LibreOffice Writer Pages (Apple)
10			

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

En Linux y sistemas similares, usas la terminal. El proceso tiene dos pasos: Compilar: Traduces tu código a un programa ejecutable con el comando gcc.

Por ejemplo: gcc mi\_codigo.c -o mi\_programa

Ejecutar: Corres el programa recién creado con el comando: ./mi\_programa

En iOS (iPhone/iPad), es totalmente diferente. No se usan comandos directamente en el dispositivo. Todo el proceso se hace en una computadora Mac usando la aplicación Xcode. Dentro de Xcode, simplemente haces clic en el botón de Play . Este botón se encarga de compilar, empaquetar e instalar la aplicación en un iPhone o simulador para ejecutarla automáticamente.

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

De la forma más sencilla: cuando usas un formato de número (%d, %i, %o, %x) para mostrar una variable de tipo char, le pides a C que te muestre el número interno que representa a ese carácter, en lugar del carácter en sí.

Para una computadora, todo son números. No entiende la letra 'A', pero sí entiende el número 65. Existe una tabla estándar llamada código ASCII que funciona como un "diccionario" que asigna un número a cada letra, símbolo y número que puedes escribir.

Cuando usas %c (el formato para *character*), printf busca el número en la tabla ASCII y muestra el símbolo correspondiente.

Cuando usas %d, %i, %o o %x, printf simplemente toma ese número y lo muestra en el formato numérico que le pediste.

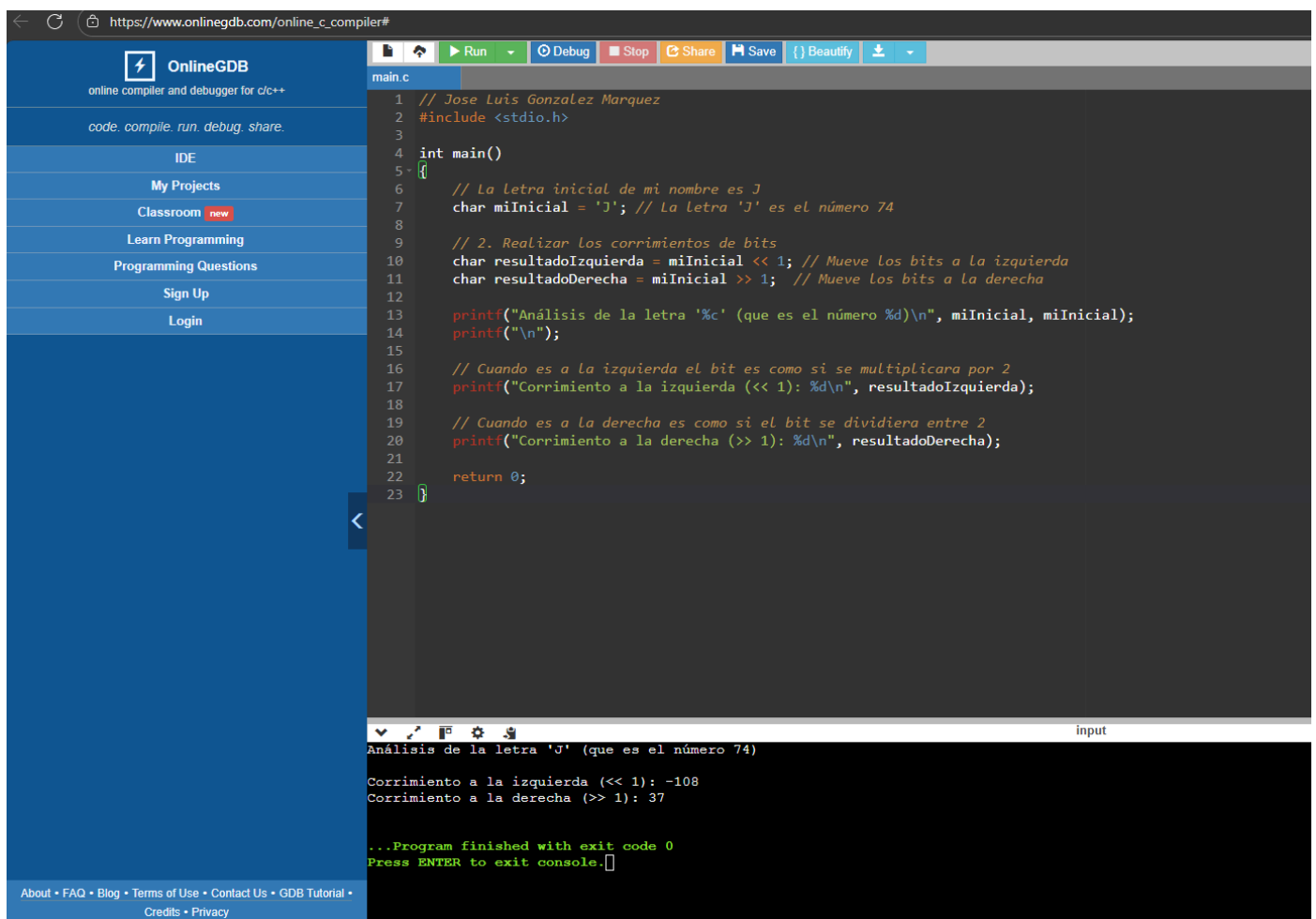
%c: muestra el carácter

%d y %i Muestra el entero en decimal.

%o Muestra el entero en octal.

%x Muestra el entero en hexadecimal

## 6. 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.



```
1 // Jose Luis Gonzalez Marquez
2 #include <stdio.h>
3
4 int main()
5 {
6     // La Letra inicial de mi nombre es J
7     char miInicial = 'J'; // La Letra 'J' es el número 74
8
9     // 2. Realizar Los corrimientos de bits
10    char resultadoIzquierda = miInicial << 1; // Mueve los bits a la izquierda
11    char resultadoDerecha = miInicial >> 1; // Mueve los bits a la derecha
12
13    printf("Análisis de la letra '%c' (que es el número %d)\n", miInicial, miInicial);
14    printf("\n");
15
16    // Cuando es a la izquierda el bit es como si se multiplicara por 2
17    printf("Corrimiento a la izquierda (<< 1): %d\n", resultadoIzquierda);
18
19    // Cuando es a la derecha es como si el bit se dividiera entre 2
20    printf("Corrimiento a la derecha (>> 1): %d\n", resultadoDerecha);
21
22    return 0;
23 }
```

input

```
Análisis de la letra 'J' (que es el número 74)
Corrimiento a la izquierda (<< 1): -108
Corrimiento a la derecha (>> 1): 37
...Program finished with exit code 0
Press ENTER to exit console.
```

## Evidencias laboratorio:

Last login: Wed Oct 8 19:11:19 on console

The default interactive shell is now zsh.

To update your account to use zsh, please run `chsh -s /bin/zsh`.

For more details, please visit <https://support.apple.com/kb/HT208050>.

Bielorrusia22:~ fp22alu20\$ ls

Desktop Documents Downloads Library Movies Music Pictures Public

Bielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ gcc programa1.c -o programa1.out

gcc-11: error: unrecognized command-line option '-0'

Bielorrusia22:~ fp22alu20\$ gcc programa1.c -o programa1.out

Bielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ gcc programa1.c -o programa1.out

Bielorrusia22:~ fp22alu20\$ ./programa1.out

Gonzalez Marquez Jose LuisGonzalez Marquez Jose LuisGonzalez Marquez Jose LuisBielorrusia22:~ fp22alu20\$ gcc programa1.c -vi programa1.c

Bielorrusia22:~ fp22alu20\$ ./programa1.out

Gonzalez Marquez Jose LuisGonzalez Marquez Jose LuisGonzalez Marquez Jose LuisBielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ ./programa1.out

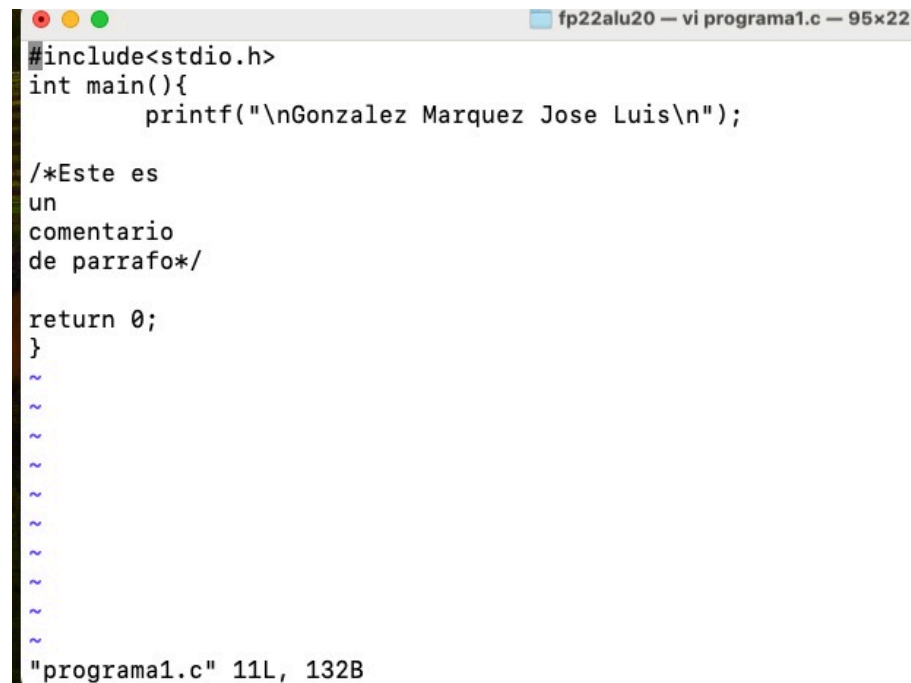
Gonzalez Marquez Jose LuisGonzalez Marquez Jose LuisGonzalez Marquez Jose LuisBielorrusia22:~ fp22alu20\$ vi programa1.c

Bielorrusia22:~ fp22alu20\$ gcc programa1.c -o programa1.out

Bielorrusia22:~ fp22alu20\$ ./programa1.out

Gonzalez Marquez Jose Luis

Bielorrusia22:~ fp22alu20\$ █



```
#include<stdio.h>
int main(){
    printf("\nGonzalez Marquez Jose Luis\n");

    /*Este es
    un
    comentario
    de parrafo*/

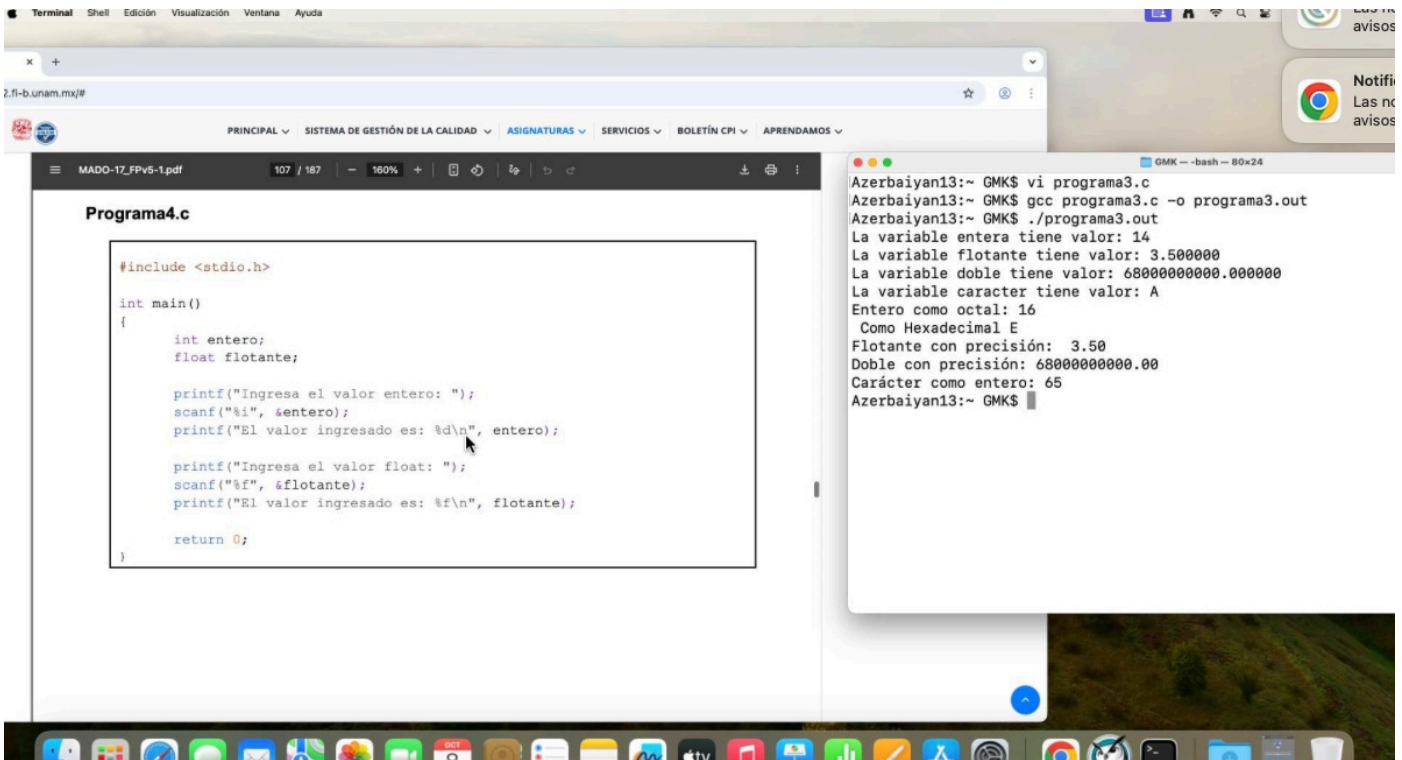
    return 0;
}
~
~
~
~
~
~
~
~
~
"programa1.c" 11L, 132B
```

```

fp22alu20 — vi programa3.c — 95x22

//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;
}

```





fp22alu20 — -bash — 113x29

```
Bielorrusia22:~ fp22alu20$ vi programa1.c
Bielorrusia22:~ fp22alu20$ gcc programa1.c -o programa1.out
Bielorrusia22:~ fp22alu20$ ./programa1.out
```

Gonzalez Marquez Jose Luis

```
Bielorrusia22:~ fp22alu20$ clear
```

```
Bielorrusia22:~ fp22alu20$ vi programa3.c
Bielorrusia22:~ fp22alu20$ gcc programa3.c -o programa3.out
Bielorrusia22:~ fp22alu20$ ./programa3.out
La variable entera tiene valor: 14
La variable flotante tiene valor: 3.500000
La variable doble tiene valor: 6800000000.000000
La variable caracter tiene valor: A
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 6800000000.00
Carácter como entero: 65
Bielorrusia22:~ fp22alu20$
```

fp22alu20 — -bash — 113x29

```
Bielorrusia22:~ fp22alu20$ vi programa5.c
Bielorrusia22:~ fp22alu20$ gcc programa5.c -o programa5.out
Bielorrusia22:~ fp22alu20$ ./programa5.out
Escriba un valor entero: 69
Escriba un valor real: 697.6
```

Imprimiendo las variables

```
Valor de enteroNumero = 69
Valor de caracterA = A
Valor de puntoFlotanteNumero = 697.600000
Valor de enteroNumero en base 16 = 45
Valor de caracterA en código hexadecimal = 41
Valor de puntoFlotanteNumero
en notación científica = 6.976000e+02
Bielorrusia22:~ fp22alu20$
```

```
fp22alu20 — -bash — 113x29
Bielorrusia22:~ fp22alu20$ vi programa6.c
Bielorrusia22:~ fp22alu20$ gcc programa6.c -o programa6.out
Bielorrusia22:~ fp22alu20$ ./programa6.out
Operadores aritméticos
5 modulo 2 = 1
Operadores lógicos
8 >> 2 = 2
8 << 1 = 16
5 & 4 = 4
3 | 2 = 3
Bielorrusia22:~ fp22alu20$
```

## Conclusión:

La verdad esta conclusión va a estar bastante completa porque me gusto mucho este tema tanto en la clase como en el laboratorio. Es muy padre poder entender poco a poco como funciona cada parte de nuestro programa y sobre todo la satisfacción de que cada vez sientas que estás más cerca de poder comprender lo necesario. Sin duda alguna mi práctica favorita fue la práctica de laboratorio porque pude poner a prueba cómo funcionan diferentes tipos de reglas y sobre todo para experimentar todas las curiosidades que tenía o que podían surgir al momento de emplearlo. En lo personal me gusto como nos lo enseñó maestra, espero podamos seguir con esto y volvernos mejores con cada práctica.

## Fuentes:

Facultad de Ingeniería. (2025). Manual de prácticas del laboratorio de Fundamentos de Programación. Laboratorio de computación. Salas A y B. Universidad Nacional Autónoma de México. pp. 85-112. Recuperado el 15 de octubre de 2025 de <http://lcp02.fi-b.unam.mx/>

The GNU Project. (s.f.). *GCC, the GNU Compiler Collection*. Recuperado el 16 de octubre de 2025, de <https://gcc.gnu.org/>

Apple Inc. (s.f.). *Xcode*. Recuperado el 16 de octubre de 2025, de <https://developer.apple.com/xcode/>

The GNU Project. (s.f.). *GCC, the GNU Compiler Collection*. Recuperado el 16 de octubre de 2025, de <https://gcc.gnu.org/>

Roskilde University. (s.f.). *How to Compile and Run a C Program on Ubuntu Linux*. Recuperado el 16 de octubre de 2025, de [http://akira.ruc.dk/~keld/teaching/CAN\\_e14/Readings/How%20to%20Compile%20and%20Run%20a%20C%20Program%20on%20Ubuntu%20Linux.pdf](http://akira.ruc.dk/~keld/teaching/CAN_e14/Readings/How%20to%20Compile%20and%20Run%20a%20C%20Program%20on%20Ubuntu%20Linux.pdf)

Apple Inc. (s.f.). *Xcode*. Recuperado el 16 de octubre de 2025, de <https://developer.apple.com/xcode/>