



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: 20

No. de práctica(s): 06

Integrante(s): Suzán Herrera Álvaro

No. de lista o brigada: 49

Semestre: 1

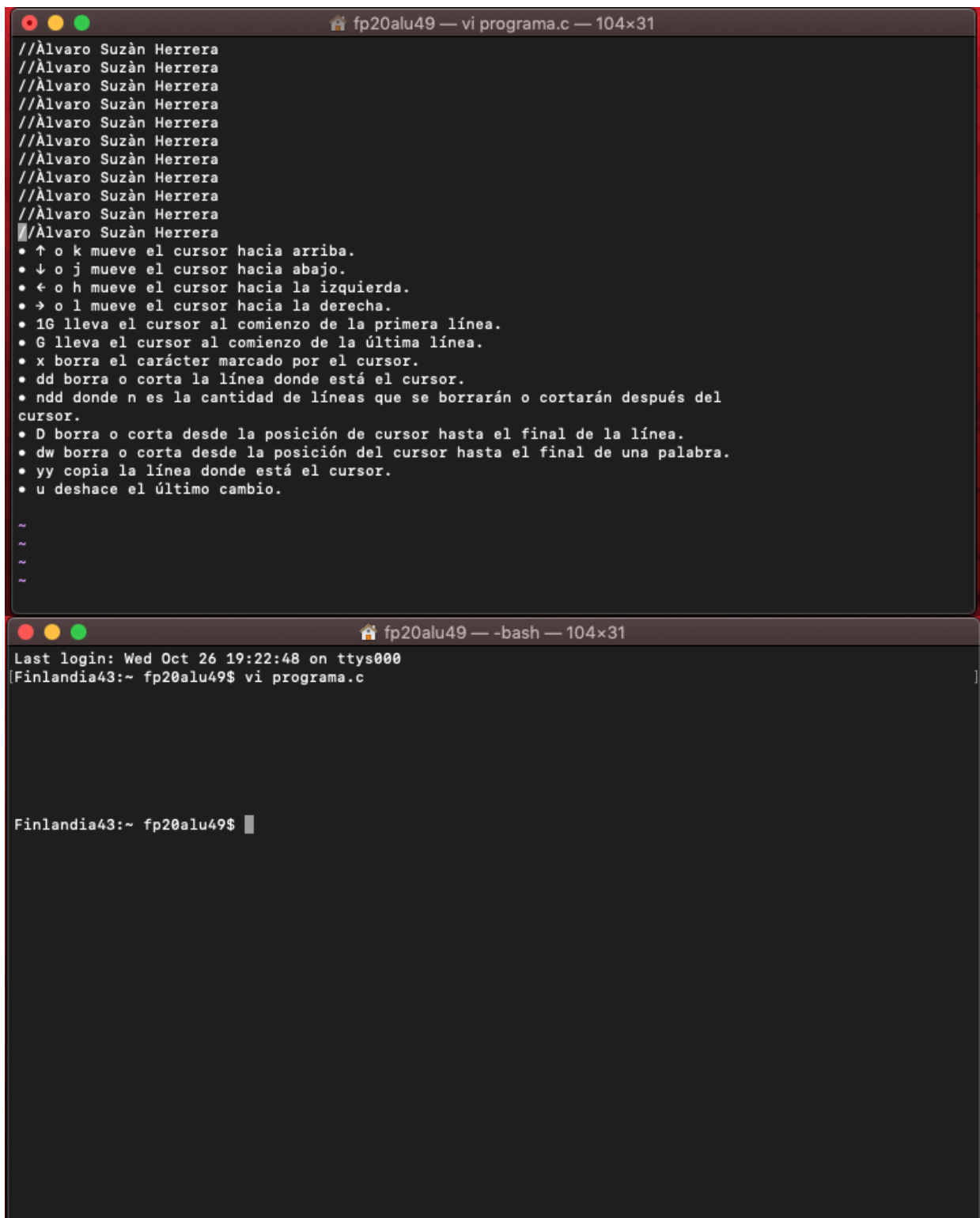
Fecha de entrega: 25 octubre 2022

Observaciones:

CALIFICACIÓN: _____

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.

[illegible]



The image shows two terminal windows. The top window is titled 'fp20alu49 — vi programa.c — 104x31' and displays the contents of a file named 'programa.c'. The file contains ten lines of comments, each starting with '//' and followed by the text 'Álvaro Suzàn Herrera'. Below the comments is a list of vi editor commands and their functions, preceded by a bullet point. The bottom window is titled 'fp20alu49 — -bash — 104x31' and shows a bash shell prompt. It displays the last login time as 'Wed Oct 26 19:22:48 on ttys000' and the current directory as '~'. The prompt is 'Finlandia43:~ fp20alu49\$'.

```
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
//Álvaro Suzàn Herrera
• ↑ 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.
• n dd 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.
• u deshace el último cambio.

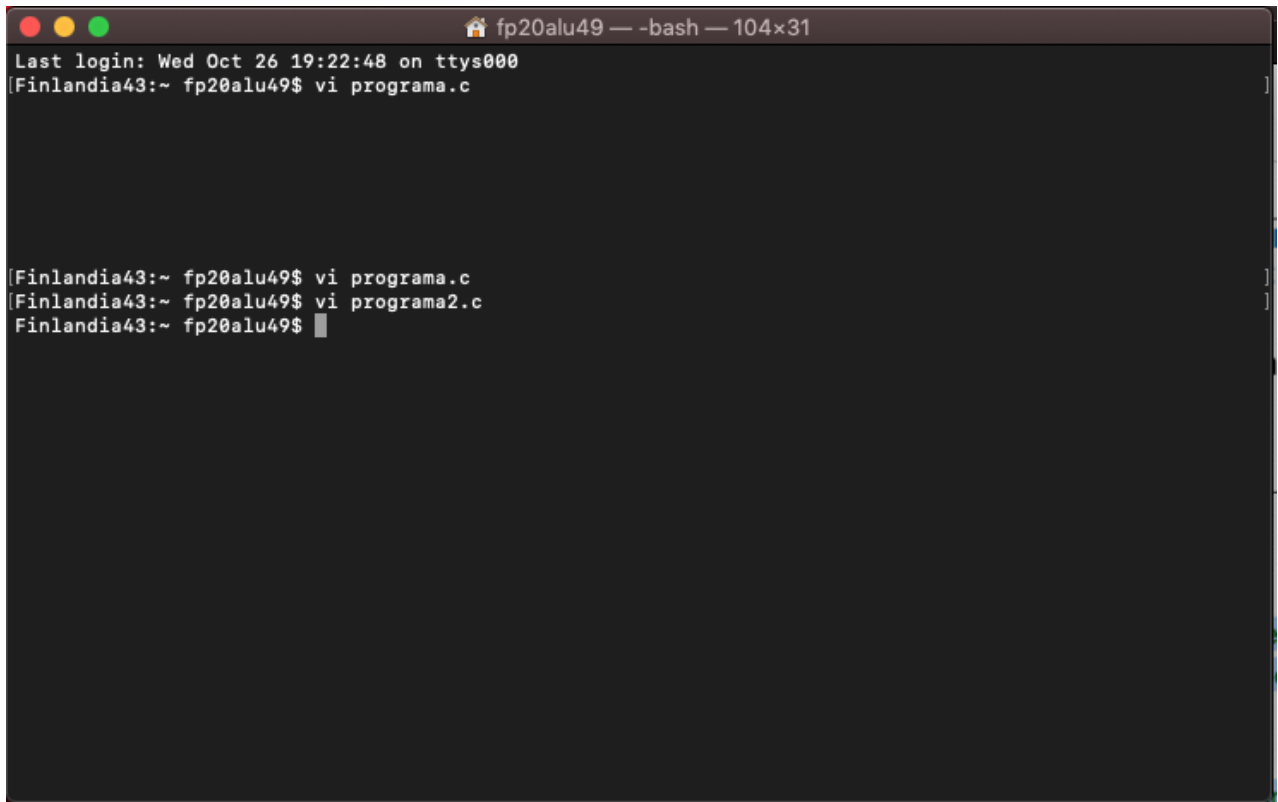
~
~
~
~
```

```
Last login: Wed Oct 26 19:22:48 on ttys000
[Finlandia43:~ fp20alu49$ vi programa.c

Finlandia43:~ fp20alu49$
```

Editor Visual Interface de GNU/Linux (vi) El editor vi (visual interface) es el editor más común en cualquier distribución de sistemas operativos con núcleo basado en UNIX. Está disponible en línea de comandos y si el sistema operativo tiene entorno gráfico se puede acceder a él desde la terminal. vi es un editor que puede resultar difícil de usar en un inicio. Aunque existen editores más intuitivos en su uso; en muchas ocasiones vi es el único

disponible. Para iniciar vi, debe teclearse desde la línea de comandos: `vi nombre_archivo[.ext]` Donde `nombre_archivo` es el nombre del archivo a editar o el nombre de un archivo nuevo que se creará con vi, y `[.ext]` se refiere a la extensión que indica que el texto es una programa escrito en algún lenguaje o es texto plano, por ejemplo. Es válido incluir la ruta donde se localiza o localizará el archivo. Existen más métodos de apertura para usuarios más avanzados.



```
fp20alu49 — -bash — 104x31
Last login: Wed Oct 26 19:22:48 on ttys000
[Finlandia43:~ fp20alu49$ vi programa.c

[Finlandia43:~ fp20alu49$ vi programa.c
[Finlandia43:~ fp20alu49$ vi programa2.c
[Finlandia43:~ fp20alu49$
```

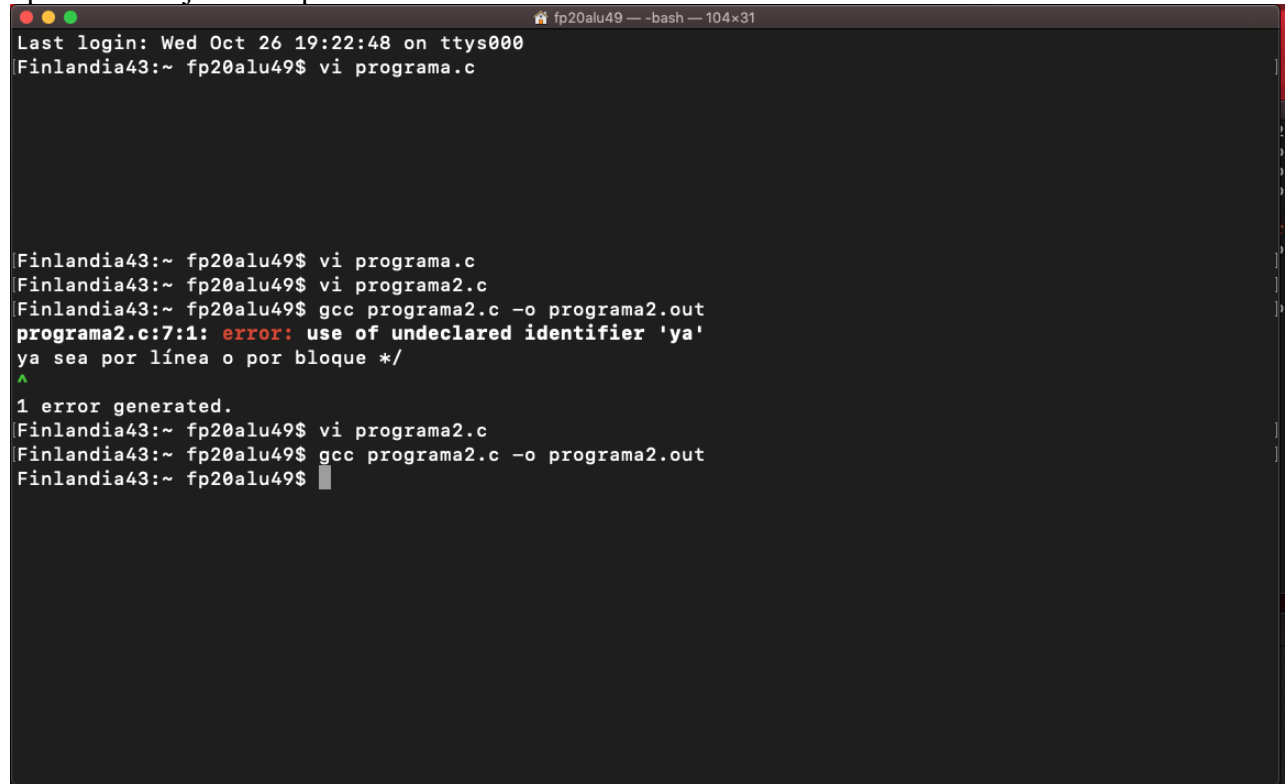
Es el modo por defecto de vi cuando se abre. Las teclas presionadas ejecutan diversas acciones predeterminadas y no se puede editar el texto libremente. Los comandos son sensitivos a las mayúsculas y a las minúsculas.

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.

gcc (GNU Compiler Collection)

Es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX. Entre sus compiladores existe el que sirve para programas escritos en C. Se encuentra por defecto en diversas distribuciones de GNU/Linux. El compilador trabaja en línea de comandos. Existe también una versión modificada que puede ejecutar y crear programas para plataformas Windows en un paquete llamado MinGW (Minimalist GNU for Windows). Al compilar un programa en C el compilador genera diversos archivos intermedios que corresponden a las distintas fases que realiza. Éstas no son de interés por el momento y son eliminadas una vez obtenido el archivo ejecutable. gcc tiene diferentes opciones de ejecución para usuarios más avanzados.

A screenshot of a terminal window with a dark background. The window title bar shows 'fp20alu49' and '-bash' with a window size of '104x31'. The terminal text shows a user logging in on 'Wed Oct 26 19:22:48 on ttys000'. The user runs 'vi programa.c'. Then, they run 'gcc programa2.c -o programa2.out', which results in a compilation error: 'programa2.c:7:1: error: use of undeclared identifier 'ya''. The error message is followed by a green arrow cursor pointing to the first line of the file and the text '1 error generated.'. The user then runs 'vi programa2.c' and 'gcc programa2.c -o programa2.out' again, but the terminal text is cut off before the second compilation command completes.

```
fp20alu49 — -bash — 104x31
Last login: Wed Oct 26 19:22:48 on ttys000
Finlandia43:~ fp20alu49$ vi programa.c

Finlandia43:~ fp20alu49$ vi programa.c
Finlandia43:~ fp20alu49$ gcc programa2.c -o programa2.out
programa2.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Finlandia43:~ fp20alu49$ vi programa2.c
Finlandia43:~ fp20alu49$ gcc programa2.c -o programa2.out
Finlandia43:~ fp20alu49$
```



```
programa3.c:21:42: warning: missing terminating '"' character [-Winvalid-pp-token]
printf("Doble con precisión: %5.2f \n", doble);

programa3.c:22:8: error: non-ASCII characters are not allowed outside of literals and identifiers
printf("Carácter como entero: %d \n", caracter);

programa3.c:22:40: warning: missing terminating '"' character [-Winvalid-pp-token]
printf("Carácter como entero: %d \n", caracter);

8 warnings and 12 errors generated.
Finlandia43:~ fp20alu49$ vi programa3.c
Finlandia43:~ fp20alu49$ gcc programa3.c -o programa3.out
programa3.c:12:12: error: non-ASCII characters are not allowed outside of literals and identifiers
caracter = 'A';

programa3.c:12:16: error: non-ASCII characters are not allowed outside of literals and identifiers
caracter = 'A';

programa3.c:12:15: error: use of undeclared identifier 'A'
caracter = 'A';

3 errors generated.
Finlandia43:~ fp20alu49$ vi programa3.c
Finlandia43:~ fp20alu49$ gcc programa3.c -o programa3.out
programa3.c:12:10: warning: incompatible pointer to integer conversion assigning to 'char' from
      'char [2]' [-Wint-conversion]
caracter = "A";

1 warning generated.
Finlandia43:~ fp20alu49$
```

Comentarios Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En C existen dos tipos de comentarios: el comentario por línea y el comentario por bloque. El comentario por línea inicia cuando se insertan los símbolos // y termina con el salto de línea (hasta donde termine el renglón). El comentario por bloque inicia cuando se insertan los símbolos /* y termina cuando se encuentran los símbolos */. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.

```
#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;
}

~
~
~
~
~
~
"programa3.c" 24L, 745C
```

La primera línea del Programa1.c #include es una directiva al preprocesador de C que agrega la biblioteca (que contiene funciones pre-definidas en el lenguaje) que requiere el programa para su ejecución. En este caso particular, la biblioteca stdio.h (standard input(i) output(o)) provee funciones externas necesarias para lectura y escritura de datos que se verán más adelante. Además, revisaremos la ubicación de estas directivas dentro de la estructura de un programa en C.

```
Finlandia43:~ fp20alu49$ vi programa3.c
Finlandia43:~ fp20alu49$ gcc programa3.c -o programa3.out
programa3.c:12:12: error: non-ASCII characters are not allowed outside of literals and identifiers
character = 'A';
               ^
programa3.c:12:16: error: non-ASCII characters are not allowed outside of literals and identifiers
character = 'A';
               ^
programa3.c:12:15: error: use of undeclared identifier 'A'
character = 'A';
               ^
3 errors generated.
Finlandia43:~ fp20alu49$ vi programa3.c
Finlandia43:~ fp20alu49$ gcc programa3.c -o programa3.out
programa3.c:12:10: warning: incompatible pointer to integer conversion assigning to 'char' from
'char [2]' [-Wint-conversion]
character = "A";
               ^~~~~~
1 warning generated.
Finlandia43:~ fp20alu49$ vi programa3.c
Finlandia43:~ fp20alu49$ ./programa3.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: ?
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 68000000000.00
Carácter como entero: -116
Finlandia43:~ fp20alu49$
```

- 1-Cuál es el dato que se encuentra por default en en lenguaje C Signed.
- 2- Indicar que sucede cuando en una variable tipo carácter se emplea el formato %d, %i, %o, %x
 - %d: Imprime una variable int en formato decimal.
 - %i: Especifica a un entero.
 - %o: Imprime un entero en octal.

%x: Imprime un entero en hexadecimal.

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

Variable: Es un nombre para identificar una o varias posiciones de memoria donde el programa guarda los distintos valores de una misma entidad.

Así, una variable es un lugar donde se puede almacenar temporalmente un dato. En C las variables tienen un nombre que las identifica, y sirve para hacer referencia a ellas. También tienen un tipo, que es el tipo de datos que puede almacenar. El valor de las variables es, como su propio nombre indica, variable. Podemos alterar su valor en cualquier punto del programa.

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

Estáticas: se crean al principio del programa y duran mientras el programa se ejecute.

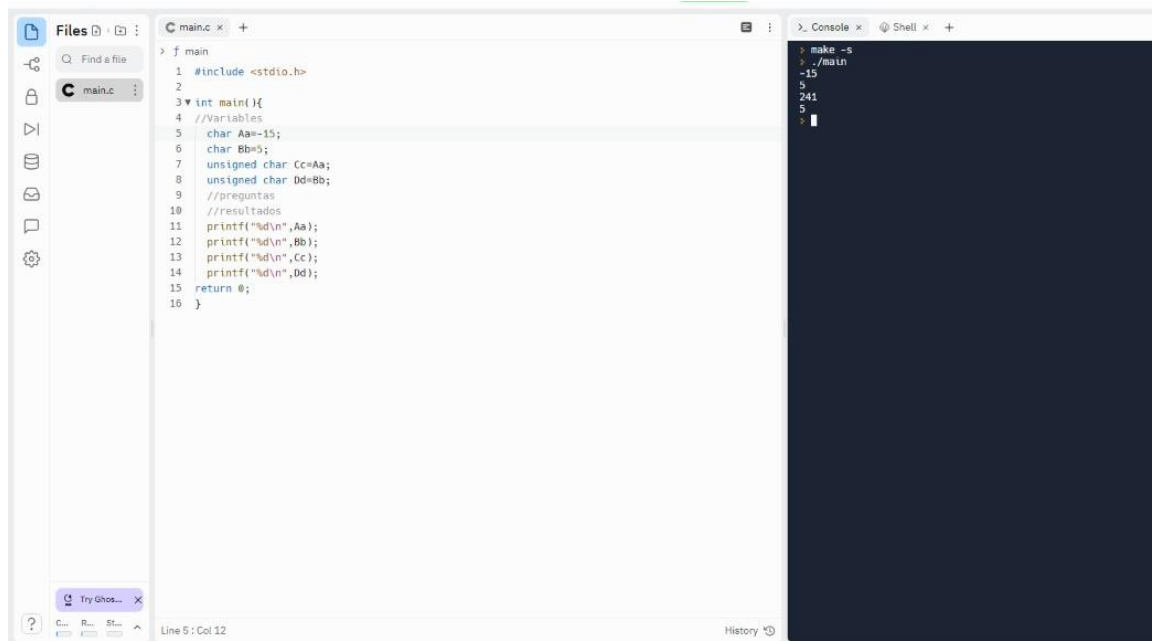
Constantes: Las variables son constantes si son creadas dentro de una función. Su existencia está ligada a la existencia de la función.

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

Prioridad de los operadores aritméticos, de índice de un array, de llamada a una función, relacionales, lógicos, de asignación y de conversión de tipo (de mayor a menor) en C:

() []	Llamada a una función e índice de un array
+ - ++ -- ! (<tipo>)	Signo más, signo menos, incremento, decremento, negación y conversión de tipo
* / %	Multiplicación, división, módulo
+ -	Suma y resta
< <= > >=	Menor que, menor o igual que, mayor que, mayor o igual que
== !=	Igual que y distinto que
&&	Conjunción
	Disyunción
= += -= *= /= %=	Operadores de asignación

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)



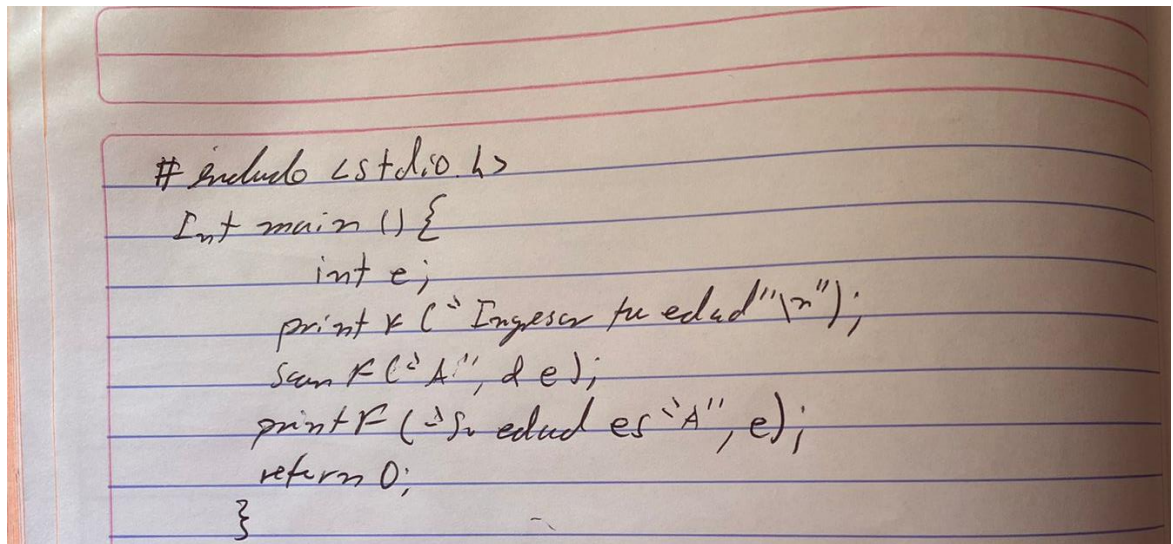
The screenshot shows a code editor with a file named `main.c`. The code is as follows:

```
f main
1 #include <stdio.h>
2
3 int main() {
4     //Variables
5     char Aa=15;
6     char Bb=5;
7     unsigned char Cc=Aa;
8     unsigned char Dd=Bb;
9     //preguntas
10    //resultados
11    printf("%d\n", Aa);
12    printf("%d\n", Bb);
13    printf("%d\n", Cc);
14    printf("%d\n", Dd);
15    return 0;
16 }
```

On the right, a terminal window shows the output of the program after running `make -s` and `./main`:

```
> make -s
> ./main
-15
5
241
5
>
```

las cuatro variables deben ser solicitadas al usuario(se emplea scanf) y deben mostrarse en pantalla (emplear printf)



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

```
#include [stdio.h]
```

```
main () {
```

```
    int 1
```

```
    int 0
```

```

Char Edad

Float 18

Printf ("Ingrese su edad")

Scanf ("%d & EDAD)

IF (EDAD >= 18)

Print ("Eres mayor de edad")

ELSE ("No eres mayor de edad")

Return 0;

}

```

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

	Editor de texto	Procesador de texto
Ventajas	<ul style="list-style-type: none"> • Acceso a información en red • visualización • Personalización 	<ul style="list-style-type: none"> • Personalización • Permite elaborar textos •

Desventajas	<ul style="list-style-type: none"> • Compatibilidad • Diccionario limitado • Costos 	<ul style="list-style-type: none"> • Complejidad • No hay texto predictivo
-------------	--	--

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

Crear archivos, llamar archivos: vi programa.c

- Compilador: gcc programa.c -o programa
- Ejecución: ./programa.out

10-Compilación y prueba del programa antes mostrado en DEV C++ u otro IDE en sistema operativo Windows.

Windows y Linux no compilan en el mismo código, por ende no se puede.

11- 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 instrucción.

```
#include "stdio.h"
int main() {
```

```
    float a, b, Ans;
    printf("Ingresar un número: ");
    scanf("%f", &a);
    printf("Ingresar otro número: ");
    scanf("%f", &b);
```

Operaciones

```
    Ans = a + b;
    printf("Sumadas: %f\n", Ans);
    Ans = a - b;
    printf("Restadas: %f\n", Ans);
    Ans = a * b;
    printf("Multiplicadas: %f\n", Ans);
    if (b != 0) {
        Ans = a / b;
    }
    printf("Divididas: %f\n", Ans);
```

Norma

```
    close {
    }
    printf("Divididos: Error");
    return 0;
}
```

Conclusiones

La práctica me ayudó a comprender el funcionamiento de códigos y pseudocódigos más aplicados, así como la creación de programas con la terminal, es un mundo un poco complejo el de la programación, pero interesante.

Aprendí diferentes tipos de códigos que usan los programas para crear comandos y funciones.

<https://github.com/3201050964/Pr-ctica-6>