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): Practica 6

Integrante(s): Vargas Hernandez Edgar Vicente

No. de lista o brigada: 50

Semestre: Primer semestre

Fecha de entrega:

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.

Actividades:
● Crear un archivo de texto (utilizando algún editor) y escribir un programa en lenguaje C que contenga variables de diferentes tipos, asignación de valores (por lectura desde la entrada estándar o asignación directa) y escritura del valor de las variables en la salida estándar
● Compilar un código fuente y ejecutarlo. ● Modificar y actualizar un programa usando un editor.
● Elaborar expresiones relacionales/lógicas en un programa en C y mostrar el resultado de su evaluación.

Editores Un programa en C debe ser escrito en un editor de texto para después generar un programa ejecutable en la computadora por medio de un compilador.

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.

La práctica De fundamentos del lenguaje tipo c la maestra nos empezó a explicar qué significaba el concepto y cómo deberíamos de utilizar el lenguaje tipo c puede realizar las actividades de las prácticas primero nos pidió hacer un archivo utilizando el formato que aparecía en la práctica que era este

Para iniciar *vi*, debe teclearse desde la línea de comandos: *vi* nombre_archivo[.ext]

Modo commando

- ↑ 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.

El modo comandos es la forma en la que nosotros nos podemos desplazar en el programa para poder modificar o realizar ciertas acciones donde el programador dese

Modo de última línea

El modo d última línea es el comando qué vamos a escribir en la parte inferior que sería la última línea que realizará las ciertas acciones que están puestas en los siguientes puntos

- **/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*.

GNU nano

nano es un editor clon de otro editor llamado pico.

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

nano nombre_archivo[.ext]

Donde "*nombre_archivo*" es el nombre del archivo a editar o el nombre de un archivo nuevo.

gcc (GNU Compiler Collection)

Este es la forma en la que 1 puede identificar los programas que se hayan creado mediante ciertas condiciones de nombre como los ejemplos que se dan

Suponiendo que se tiene un programa escrito en C y se le llamó *calculadora.c,* la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:

gcc calculadora.c

Esto creará un archivo *a.out* (en Windows *a.exe*) que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro *-o* de *gcc*. Por ejemplo, para que se llame *calculadora.out* (en Windows *calculadora.exe*), se escribe en la línea de comandos:

gcc calculadora.c *-o* calculadora.out

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se están usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro *-l* seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

gcc calculadora.c *-o* calculadora -lnombre_biblioteca

Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es *calculadora.out*, para ejecutar debe teclearse en línea de comandos:

```
./calculadora.out
```

Si el programa realizado necesita tener una entrada de información por medio de argumentos, éstos se colocan así:

```
./calculadora.out argumento1 argumento2
```

Lenguaje de programación C C es un lenguaje de propósito general basado en el paradigma estructurado. El teorema del programa estructurado, demostrado por Böhm-Jacopini, dicta que todo programa puede desarrollarse utilizando únicamente 3 instrucciones de control: Secuencia, Selección e Iteración.

Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 * Authors: Julio A. de León, Jorge A. Solano and Hugo Zuñiga
*/
```

Código con comentarios (se emplean los símbolos para indicar comentarios en C)

El siguiente programa muestra las sintaxis de comentarios en C:

Programa1.c

```c
#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;
}
```

La profesora en esta parte nos estuvo remarcando que al momento de nosotros colocar dos guiones (///) significaba que lo que estuviese escrito después era un comentario hacia el programador significa que la persona que está ejecutando el programa no podrá ver lo que hay escrito dicho programador por lo que la profesora lo estaba poniendo en la práctica para que pudiéramos seguir los pasos

Palabras reservadas

Las palabras reservadas (Tabla 1) son palabras que tienen un significado predefinido estándar y sólo se pueden utilizar para su propósito ya establecido; no se pueden utilizar como identificadores definidos por el programador. En lenguaje C son las siguientes:

Tabla 1: Palabras reservadas

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

Las palabras reservadas son aquellas las cuales ya tienen un valor asignado en el programa por lo tanto no se puede utilizar estas palabras para referirse a alguna variable porque el programa lo detectará con el significado o valor que le corresponda

Tipos de datos

El lenguaje C ofrece distintos tipos de datos (Tabla 2), cada uno de los cuales se puede encontrar representado de forma diferente en la memoria de la computadora.

Tabla 2: Tipos de datos.

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

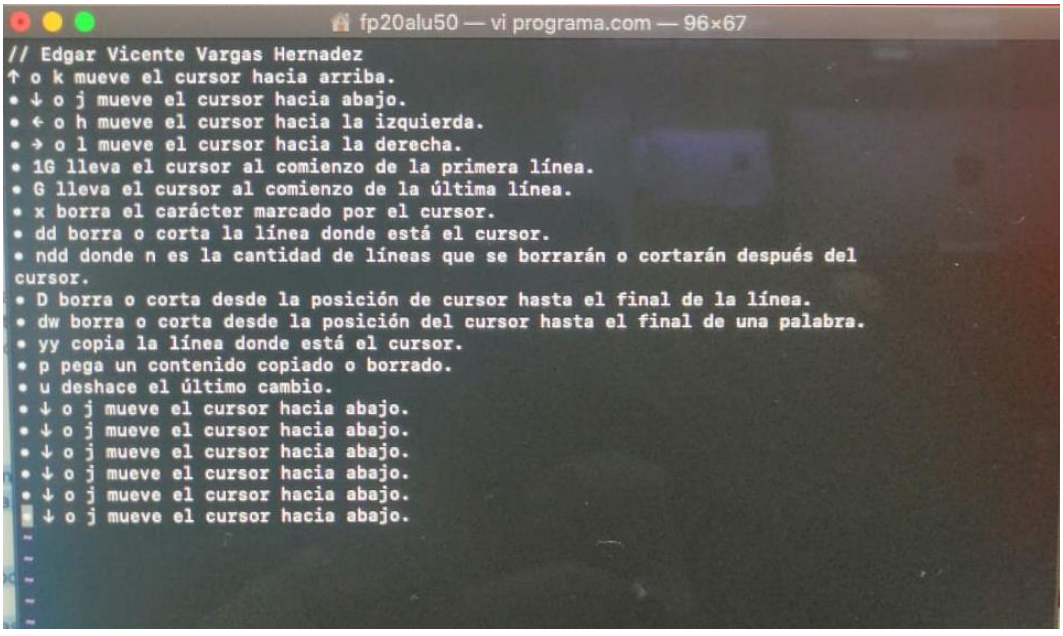
Código que utiliza las funciones *printf* y *scanf*.

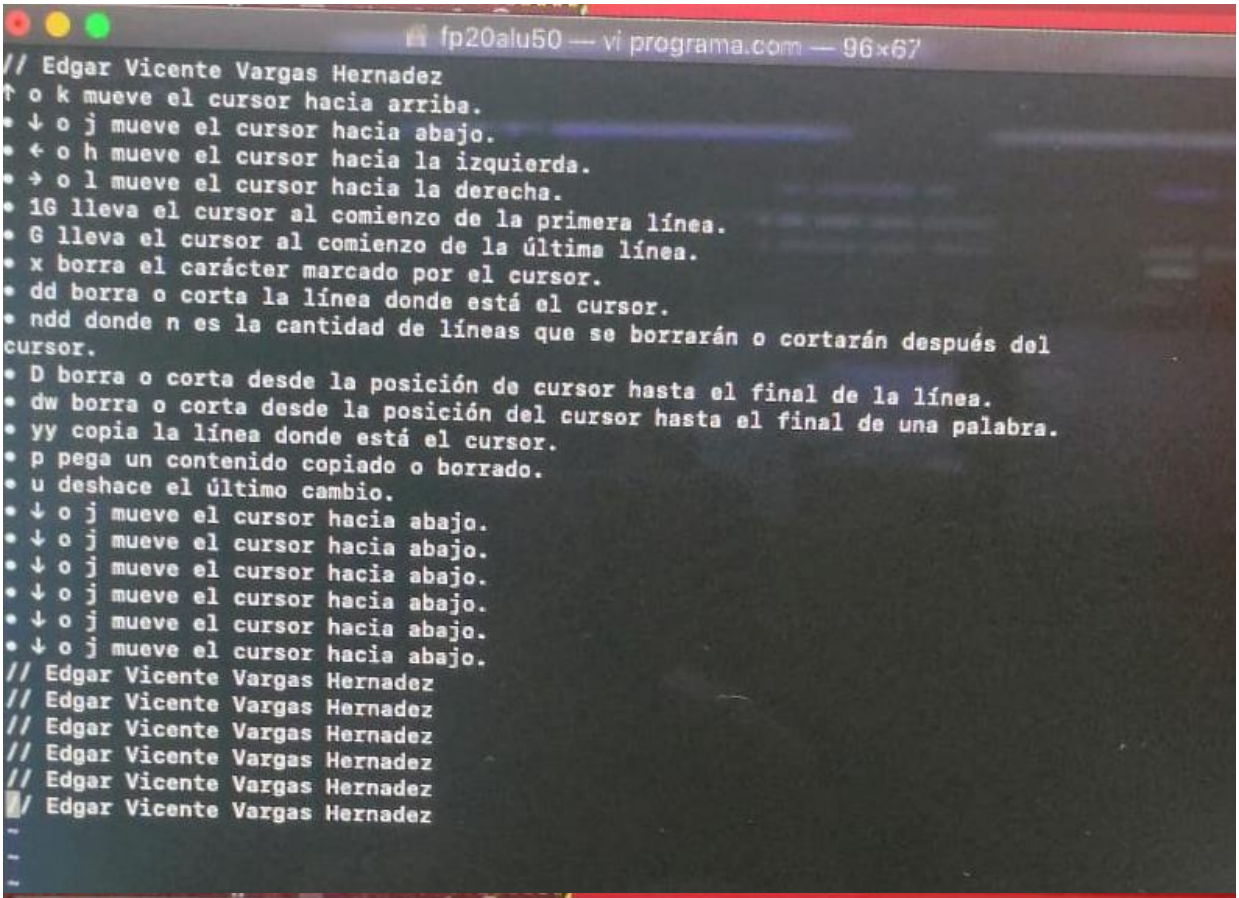
El siguiente código muestra cómo almacenar e imprimir variables. Compila y ejecuta bien.

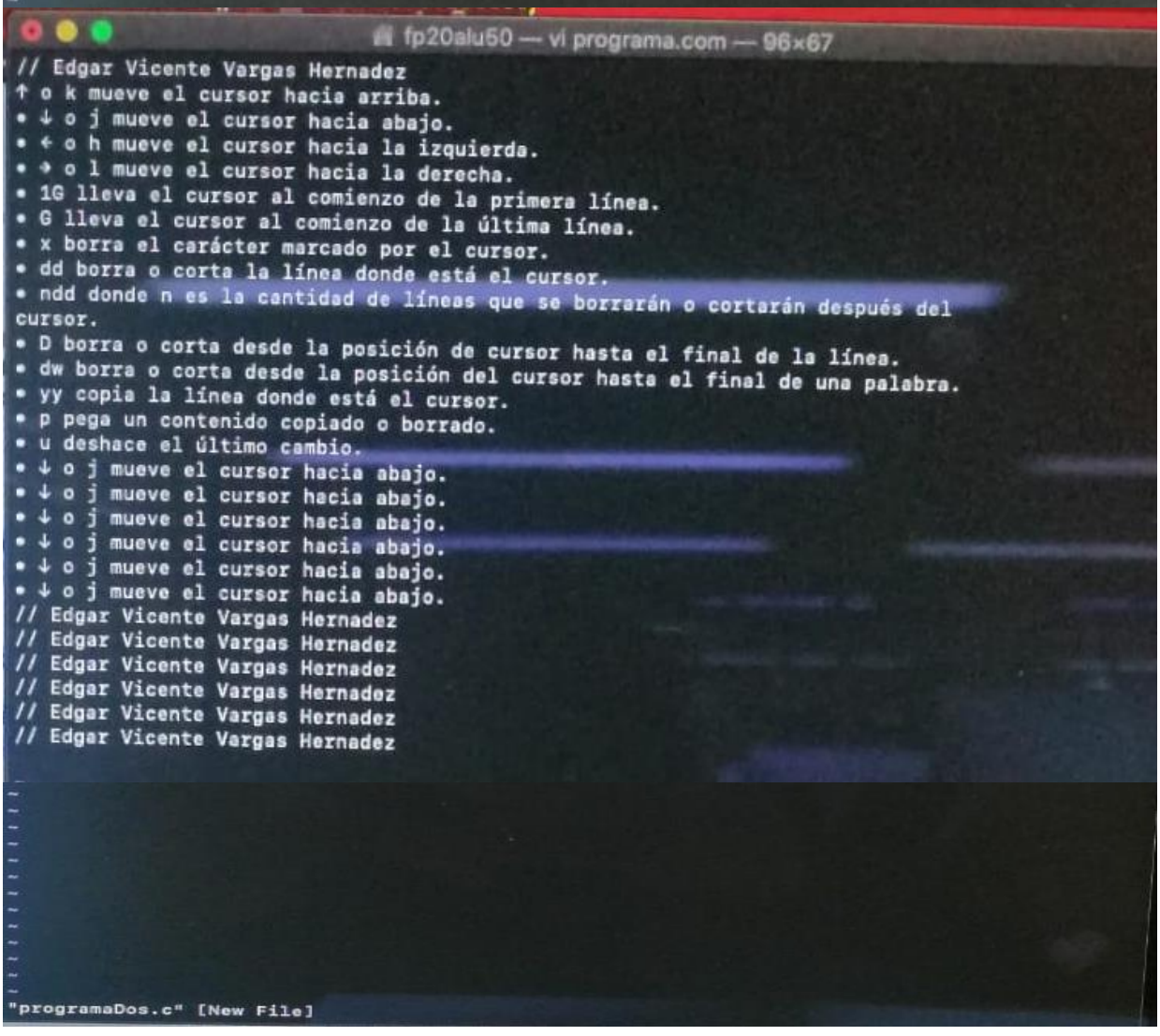
Programa5.c

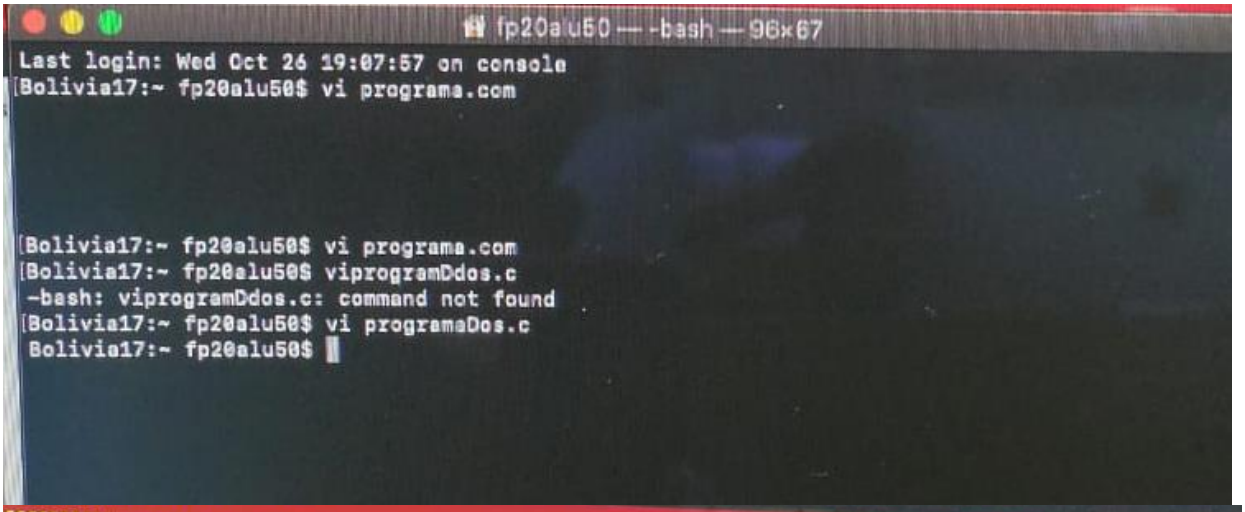
```c
#include <stdio.h>

int main()
{
	int enteroNumero;
	char caracterA = 65;		// Convierte el entero a carácter ASCII.
	double puntoFlotanteNumero;


	// Asignar valor de teclado a una variable.
	printf("Escriba un valor entero: ");
	scanf("%i", &enteroNumero);
	printf("Escriba un valor real: ");
	scanf("%lf", &puntoFlotanteNumero);

	// Imprimir valores con formato.
	printf("\nImprimiendo las variables enteras \a\n");
	printf("\t Valor de enteroNumero = %i \a\n", enteroNumero);
	printf("\t Valor de caracterA = %c \a\n", caracterA);
	printf("\t Valor de puntoFlotanteNumero = %lf \a\n",
		puntoFlotanteNumero);

	printf("\t Valor de enteroNumero en base 16 = %x \a\n", enteroNumero);
	printf("\t Valor de caracterA en código hexadecimal = %x\n", caracterA);
	printf("\t Valor de puntoFlotanteNumero\n");
	printf("en notación científica = %e\n", puntoFlotanteNumero);


	return 0;
}
```

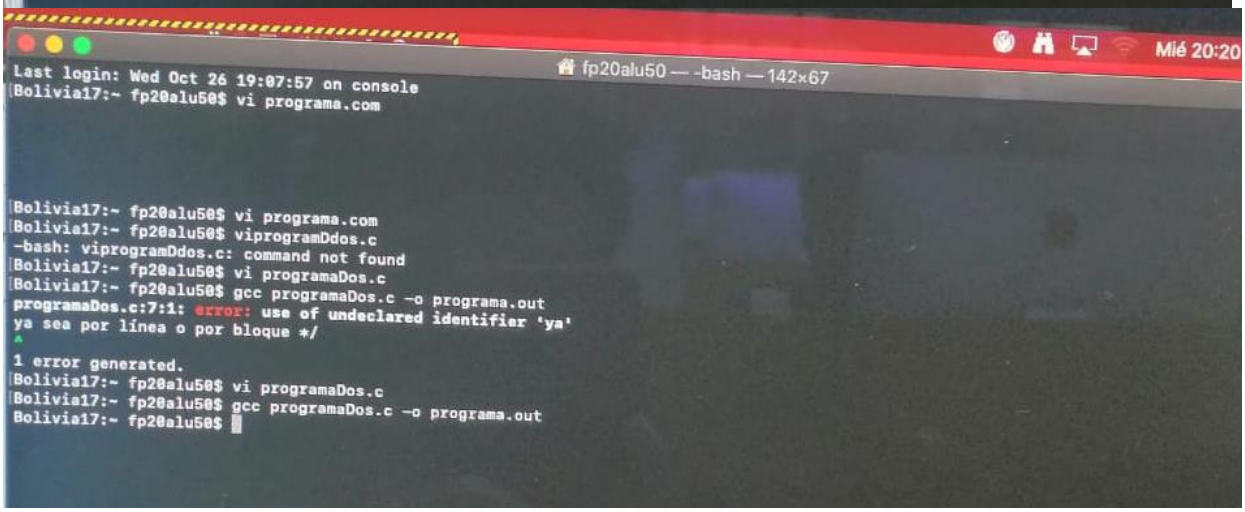
En esta parte la profesora nos fue dando los pasos de la programación que íbamos a realizar en la práctica el cual iba a tener esta estructura dónde nos enseñó a copiar y pegar en el programa y el ver los errores de este mismo

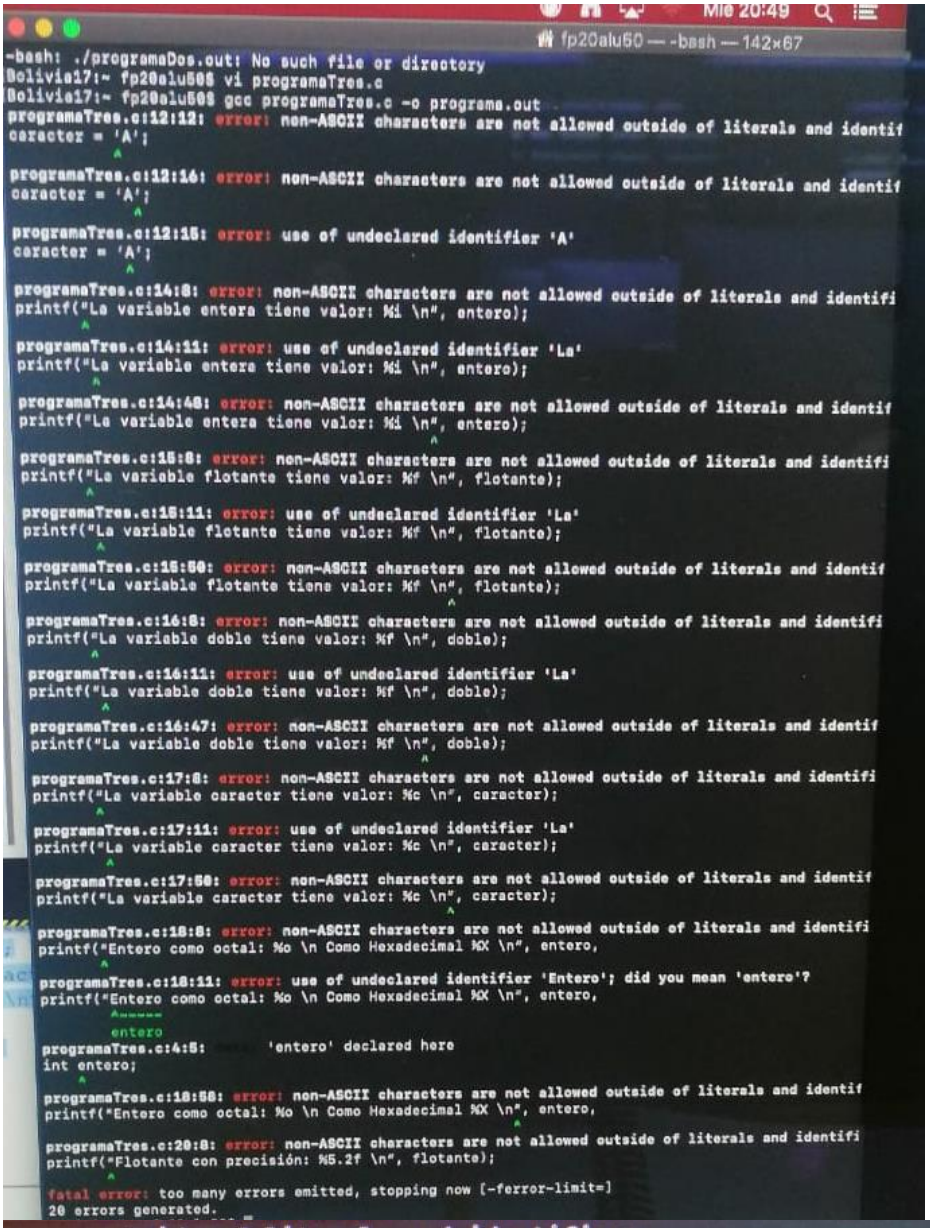



fp20alu50 — vi programa.com — 96×67
// Edgar Vicente Vargas Hernadez
↑ o k mueve el cursor hacia arriba.
• ↓ o j mueve el cursor hacia abajo.
• ← o h mueve el cursor hacia la izquierda.
• → o l mueve el cursor hacia la derecha.
• 1G lleva el cursor al comienzo de la primera línea.
• G lleva el cursor 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.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez


fp20alu50 — vi programa.com — 96×67
// Edgar Vicente Vargas Hernadez
↑ 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.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
• ↓ o j mueve el cursor hacia abajo.
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
// Edgar Vicente Vargas Hernadez
"programaDos.c" [New File]


fp20alu50 — -bash — 96×67
Last login: Wed Oct 26 19:07:57 on console
Bolivia17:~ fp20alu50$ vi programa.com
Bolivia17:~ fp20alu50$ vi programa.com
Bolivia17:~ fp20alu50$ viprogramDdos.c
-bash: viprogramDdos.c: command not found
Bolivia17:~ fp20alu50$ vi programaDos.c
Bolivia17:~ fp20alu50$


fp20alu50 — -bash — 142×67
Mié 20:20
Last login: Wed Oct 26 19:07:57 on console
Bolivia17:~ fp20alu50$ vi programa.com
Bolivia17:~ fp20alu50$ vi programa.com
Bolivia17:~ fp20alu50$ viprogramDdos.c
-bash: viprogramDdos.c: command not found
Bolivia17:~ fp20alu50$ vi programaDos.c
Bolivia17:~ fp20alu50$ gcc programaDos.c -o programa.out
programaDos.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Bolivia17:~ fp20alu50$ vi programaDos.c
Bolivia17:~ fp20alu50$ gcc programaDos.c -o programa.out
Bolivia17:~ fp20alu50$




```
     outside of literals and identifiers
caracter = 'A';
           ^

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

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

3 errors generated.
Belgica13:~ GMK$ vi programaTres.c
Belgica13:~ GMK$ gcc programaTres.c -o programaTres.out
Belgica13:~ GMK$ ./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
Belgica13:~ GMK$
```

Tarea:

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

El dato que esta por default en lenguaje C es el Signed: El modificador define que el valor de una variable numérica puede ser positivo o negativo

Unsigned: Devuelve datos numéricos convertidos al tipo de datos sin signo.

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

Es la forma en la que se escribe cuando se imprime el valor del código ASCII del tipo carácter

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

Darle un nombre a la variable (que no lleve signos).

● Debes definir la variable como entera,carácter, real etc; para almacenarla en el programa.

● Al tener almacenada se determina hasta dónde se puede leer o cambiar el valor de una variable.

● Hay dos tipos de variable local y global, la local solo se pueden usar en la función o procedimiento donde

Se declaró y la global se pueden usar a lo largo de todo el programa. Es decir, su alcance es la aplicación

Completa.

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

La variable estática mantiene su valor en la memoria, La variable constante se escribe con mayúsculas y se inicializan al momento de declararse.

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

“<”se utiliza para mencionar una condición en el programa, el cual se significa “menor que”

“>”Se emplea para mencionar una condición en le programa, la cual significa “mayor que “

“!=”Se utiliza para mencionar una condición, este significa “ es distinto de”

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)

```
#Include <stdio.h> //libreria
Int a,b,c,d
Main() {
Printf(“ingresa 4 variables””);
Scanf( “%d, %d, %d, %d” &a, &b, &c, &d)
{printf (“a”); }
{printf (“b”); }
{printf (“c”); }
{printf (“d”); }

Return0.;
}
```

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

```
#Include <stdio.h> //libreria
Int edad
Main() {
Printf(“¿cuál es tu edad?”);
Scanf( “%d” & edad)
printf (“tu edad es:%d”, edad)
Return0.;
}
```

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

La palabra reservada mod y %, es un operador aritmético es la expresión que se utiliza entre 2 operadores para obtener el módulo del primer partido por el segundo

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

Procesador de texto	Editor de texto
Es la aplicación informática que permite crear los documentos de texto en una computadora	El editor de texto es el programa que se encarga de permitir le escritura y modificación de un archivo el cual está compuesto por texto o al igual que un formato

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

gcc nombre del programa.c -o programa.out ➞ sirve para compilar el programa.

./nombre del programa.out ➞ Ejecuta el programa

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

Dev-C++: Este emplea el compilador MinGW. Se trata de un software libre, sencillo, ligero y eficiente, para la
plataforma Windows.

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 instrucción.

```
#Include <stdio.h> //libreria
Main() {
Int op;
Float a,b,res;
Printf(“/n/t favor de seleccionar una operación”);
Printf(“/n/t opción de suma1/n”);
Printf(“/n/t opción de resta2/n”);
Printf(“/n/t opción de multiplicacion3/n”);
Printf(“/n/t opción de divicion4/n/n/t/t/t”);
Scanf( “%d, &op)
If (op>0 && op<5  {
Printf(“/n/t teclear el valor de a/t””);
Scanf( “%f”, &a)
Printf(“/n/t teclear el valor de b/t””);
Scanf( “%f”, &b) }
Switch(op);     {
Case1: res=a+b;
Printf(“/n/t la suma es:%f/n”,res);
Break;
Case2: res=a-b;
Printf(“/n/t la resta es:%f/n”,res);
Break;
Case3: res=a*b;
Printf(“/n/t la multiplicacion es:%f/n”,res);
Break;
Case4:
If( b!=0)    {
Res=a/b;
Printf(“/n/t la divicion es:%f/n”,res); }
Else
Printf(“/n/t error!!!);
Break;
Default: Printf(“/n/t opción invalida/n”);}
Return0.;
}
```

Conclusion

Mi conclusión sobre la práctica es que el programar con lenguaje tipo c no llega a ser muy diferente al pseudocódigo se llegan a utilizar diversas variables un modos de comando distintos al pseudocódigo por lo que llega a ser un poco más complicado el estar recordando cada una pero creo que en esencia es lo más puro al momento de realizar programación lo cual me gusta porque es el centro de toda la materia hasta el momento sin esto nuestro programa no funciona y no se puede ejecutar mientras que el seudocódigo es la forma de representarlo creo que esta práctica estuvo bastante difícil de comprender pero me gustó.

https://github.com/Chente117/practica1_fbp.git

Bibliografia

Función UNSIGNED( ). (2021). Highbond.com. https://help.highbond.com/helpdocs/analytics/15/es/Content/analytics/scripting/functions/r_unsigned.htm#:~:text=Devuelve%20datos%20num%C3%A9ricos%20convertidos%20al%20tipo%20de%20datos%20sin%20signo.

Modificadores de tipo. (2016). Zator.com. https://www.zator.com/Cpp/E2_2_3.htm#:~:text=El%20modificador%20de%20tipo%20signed,si%20solo%20supone%20signed%20int.