



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* ALEJANDRO ESTEBAN PIMENTEL ALARCON

*Asignatura:* FUNDAMENTOS DE PROGRAMACION

*Grupo:* 03

*No de Práctica(s):* 7

*Integrante(s):* HINOJOSA RUIZ DIANA LAURA

*No. de Equipo de  
cómputo empleado:* 14

*No. de Lista o Brigada:* 6740

*Semestre:* PRIMER SEMESTRE

*Fecha de entrega:* 03 OCTUBRE 2019

*Observaciones:* Tarde entrega.  
Y recuerda que no solo es copiar el código, hay que entenderlo. En una parte dejaste corriendo el programa sin darle entrada

**CALIFICACIÓN:** 8

## Objetivo:

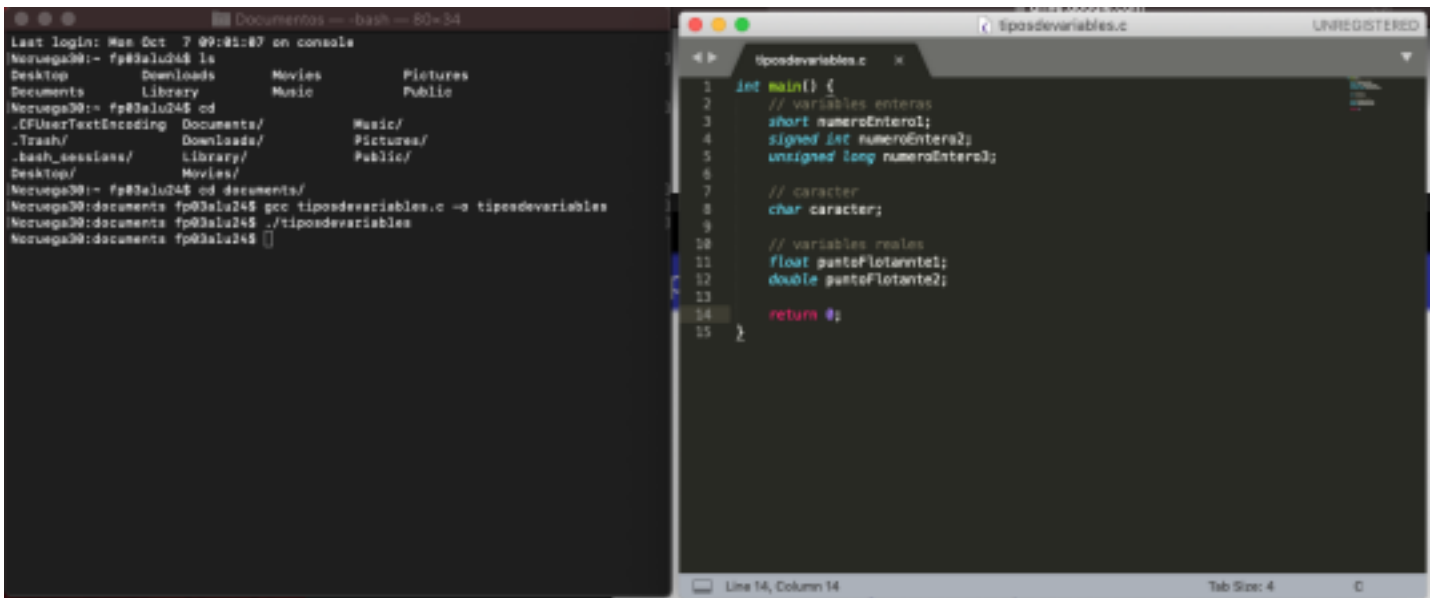
Elaborar programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

## Introducción:

Esta práctica tuvo muchas cosas nuevas, con las que comenzamos a trabajar en C, venían algunos ejemplos los cuales teníamos que hacer en notepad++ o sublime text y luego compilar y correr en la terminal, todas las hice en el laboratorio.

## Desarrollo:

Tipos de variables: aquí vimos principalmente el tipo de variable puede haber, ya que hay distintos en los enteros, reales, etc., cuantos bits tiene y de que cifra a que cifra abarca.



The image shows two side-by-side windows. The left window is a terminal with the following commands and output:

```
Last login: Wed Oct 7 09:01:07 on console
Neruega39:~$ ls
Desktop  Downloads  Movies  Pictures
Documents  Library  Music  Public
Neruega39:~$ cd
.NCUserTextEncoding Documents/ Music/
.Trash/ Downloads/ Pictures/
.bash_sessions/ Library/ Public/
Desktop/ Movies/
Neruega39:~$ cd documents/
Neruega39:documents$ gcc tiposdevariables.c -o tiposdevariables
Neruega39:documents$ ./tiposdevariables
Neruega39:documents$
```

The right window is a code editor showing the source code for `tiposdevariables.c`:

```
1 int main() {
2     // variables enteras
3     short numeroEntero1;
4     signed int numeroEntero2;
5     unsigned long numeroEntero3;
6
7     // caracter
8     char caracter;
9
10    // variables reales
11    float puntoFlotante1;
12    double puntoFlotante2;
13
14    return 0;
15 }
```

Mostrar y leer: en esta parte el profesor nos explicó que igual hay distintas formas de expresar un dato, y sobre todo la que más se usa de cada una para mostrar y leer, para entero es `%i`, para flotante `%lf` y carácter `%c`.

```
Documents - mostraryleer - 80x34
3 warnings and 2 errors generated.
Noruega3@documents fp93alu245 gcc mostraryleer.c -s mostraryleer
mostraryleer.c:16:46: error: member reference base type 'char [36]' is not a
      structure or union
      printf("luego podemos poner un entero: %i\n", numeroEntero);
      ~~~~~^
mostraryleer.c:22:15: warning: format specifies type 'double*' but the argument
      has type 'double' [-Wformat]
      scanf("%d", realEntrada);
      ~~~~~^
mostraryleer.c:26:23: warning: more 'N' conversions than data arguments
      [-Wformat]
      printf("tu real: %.31f\n", realEntrada);
      ~~~~~^
mostraryleer.c:26:41: error: extraneous '}' before ';'
      printf("tu real: %.31f\n", realEntrada);
      ~~~~~^
mostraryleer.c:26:30: warning: expression result unused [-Wunused-value]
      printf("tu real: %.31f\n", realEntrada);
      ~~~~~^
3 warnings and 2 errors generated.
Noruega3@documents fp93alu245 gcc mostraryleer.c -s mostraryleer
mostraryleer.c:22:15: warning: format specifies type 'double*' but the argument
      has type 'double' [-Wformat]
      scanf("%d", realEntrada);
      ~~~~~^
1 warning generated.
Noruega3@documents fp93alu245 ./mostraryleer
primeretextosola
luego podemos poner un entero: 32768
tambien podemos poner un caracter: B
y un numero real: 89.89
```

```
mostraryleer.c
1 #include <stdio.h>
2
3 int main() {
4
5     // declaramos variables a leer
6     int numeroEntero;
7     double realEntrada;
8
9     // asignamos variables
10    int numeroEntero = 32768;
11    char caracter = 'B';
12    float numeroReal = 89.8;
13
14    // mostramos texto y valores
15    printf("primeretextosola\n");
16    printf("luego podemos poner un entero: %i\n", numeroEntero);
17    printf("tambien podemos poner un caracter: %c\n", caracter);
18    printf("y un numero real: %.2f\n", numeroReal);
19
20    // leemos valores
21    scanf("%i", &numeroEntero);
22    scanf("%lf", realEntrada);
23
24    // y ahora podemos mostrarlos tambien
25    printf("tu entero: %i\n", numeroEntero);
26    printf("tu real: %.31f\n", realEntrada);
27
28    return 0;
29 }
```

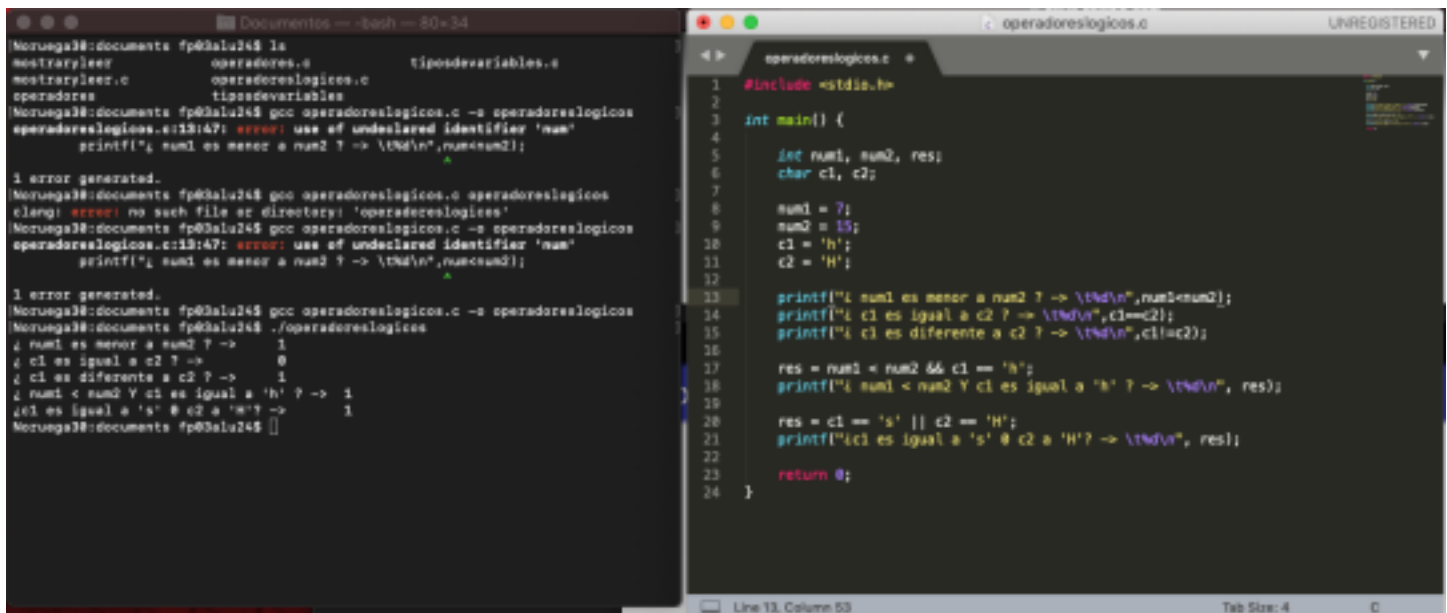
Seguimos con los operadores, los cuales ya habíamos visto la mayoría antes, a excepción de que ahora la diagonal la utilizamos para dividir en lugar del signo de porcentaje, como lo veníamos haciendo cuando escribíamos a mano.

```
Documents - operadores - 80x34
Noruega3@documents fp93alu245 gcc operadores.c -s operadores
operadores.c:9:12: error: use of undeclared identifier 'cuatro'; did you mean
      'cuatro'?
      cuatro = 4;
      ~~~~~^
operadores.c:4:17: note: 'cuatro' declared here
      int dos, tres, cuatro, cinco;
      ~~~~~^
operadores.c:13:21: warning: implicit declaration of function 'printf' is invalid
      in C99 [-Wimplicit-function-declaration]
      printf("5 / 2 = %.11f\n", resultado);
      ~~~~~^
1 warning and 1 error generated.
Noruega3@documents fp93alu245 gcc operadores.c -s operadores
operadores.c:13:21: warning: implicit declaration of function 'printf' is invalid
      in C99 [-Wimplicit-function-declaration]
      printf("5 / 2 = %.11f\n", resultado);
      ~~~~~^
1 warning generated.
Undefined symbols for architecture x86_64:
  "_printf", referenced from:
      _main in operadores-ee21c5.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -x to see invocation)
Noruega3@documents fp93alu245 gcc operadores.c -o operadores
-bash: operadores.c: command not found
Noruega3@documents fp93alu245 gcc operadores.c operadores
clang: error: no such file or directory: 'operadores'
Noruega3@documents fp93alu245 gcc operadores.c -s operadores
Noruega3@documents fp93alu245 ./operadores
5 / 2 = 2.5
5 / 2 = 2.5
Noruega3@documents fp93alu245
```

```
operadores.c
1 #include <stdio.h>
2
3 int main() {
4     int dos, tres, cuatro, cinco;
5     double resultado;
6
7     dos = 2;
8     tres = 3;
9     cuatro = 4;
10    cinco = 5;
11
12    resultado = cinco/dos;
13    printf("5 / 2 = %.11f\n", resultado);
14
15    resultado = (double)cinco/dos;
16    printf("5 / 2 = %.11f\n", resultado);
17
18    return 0;
19 }
```

Comparaciones y operadores lógicos: ambos ya los habíamos visto en clase, a excepción del NO y O, lo diferente a como los conocíamos antes es la forma de indicarlos, por ejemplo si tenemos un Y lo indicamos como &&.

Aquí el programa se quedó esperando por una entrada numérica que no le diste, revisa el código



```
Noruega38:documents fp@3alu24$ ls
mostraryleer      operadores.o      tiposdevariables.o
mostraryleer.c    operadoreslogicos.c
operadores        tiposdevariables
Noruega38:documents fp@3alu24$ gcc operadoreslogicos.c -o operadoreslogicos
operadoreslogicos.c:13:47: error: use of undeclared identifier 'num'
    printf("%i num1 es menor a num2 ? -> %td\n", num < num2);
                                          ^
1 error generated.
Noruega38:documents fp@3alu24$ gcc operadoreslogicos.c operadoreslogicos
clang: error: no such file or directory: 'operadoreslogicos'
Noruega38:documents fp@3alu24$ gcc operadoreslogicos.c -o operadoreslogicos
operadoreslogicos.c:13:47: error: use of undeclared identifier 'num'
    printf("%i num1 es menor a num2 ? -> %td\n", num < num2);
                                          ^
1 error generated.
Noruega38:documents fp@3alu24$ gcc operadoreslogicos.c -o operadoreslogicos
Noruega38:documents fp@3alu24$ ./operadoreslogicos
i num1 es menor a num2 ? -> 1
i c1 es igual a c2 ? -> 0
i c1 es diferente a c2 ? -> 1
i num1 < num2 y c1 es igual a 'h' ? -> 1
i c1 es igual a 'h' & c2 a 'H' ? -> 1
Noruega38:documents fp@3alu24$
```

```
operadoreslogicos.c
1 #include <stdio.h>
2
3 int main() {
4
5     int num1, num2, res;
6     char c1, c2;
7
8     num1 = 7;
9     num2 = 15;
10    c1 = 'h';
11    c2 = 'H';
12
13    printf("%i num1 es menor a num2 ? -> %td\n", num1 < num2);
14    printf("%i c1 es igual a c2 ? -> %td\n", c1 == c2);
15    printf("%i c1 es diferente a c2 ? -> %td\n", c1 != c2);
16
17    res = num1 < num2 && c1 == 'h';
18    printf("%i num1 < num2 y c1 es igual a 'h' ? -> %td\n", res);
19
20    res = c1 == 's' || c2 == 'H';
21    printf("%i c1 es igual a 's' & c2 a 'H' ? -> %td\n", res);
22
23    return 0;
24 }
```

## Conclusión:

La práctica fue algo sencilla en cuanto a actividades pero si es un poco complicado aprenderse los distintos nombres de los datos y de que cantidad a que cantidad abarca, son bastantes cosas nuevas que tenemos que memorizar, pero considero que con la práctica después se me va a facilitar un poco más.