	Práctica	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación salas A y B

<i>Profesor:</i>	Alejandro Pimentel
<i>Asignatura:</i>	Fundamentos de Programación
<i>Grupo:</i>	Grupo 3 Bloque 135
<i>No de Práctica(s):</i>	Practica # 12
<i>Integrante(s):</i>	Muñoz Reyes Laura Vanessa – 2823 Gutierrez Ramos Alitzel Anaid-9370
<i>No. de Equipo de cómputo empleado:</i>	Luxemburgo 10
<i>No. de Lista o Brigada:</i>	No. de Cuenta: 317752 2823 31704 9370
<i>Semestre:</i>	2020-1
<i>Fecha de entrega:</i>	04 de Noviembre del 2019
<i>Observaciones:</i>	

CALIFICACIÓN: _____

Practica #12: Función

Introducción:

La modularización, es una técnica usada por los programadores para hacer sus códigos más cortos, ya que consiste en reducir un gran problema complejo, en pequeños problemitas más sencillos, concentrándose en la solución por separado, de cada uno de ellos.

La programación en lenguaje C consiste en combinar una o más funciones. C permite tener dentro de un archivo fuente varias funciones, esto con el fin de dividir las tareas y que sea más fácil la depuración, la mejora y el entendimiento del código.

Objetivo:

Elaborar programas en C donde la solución del problema se divida en funciones. Distinguir lo que es el prototipo o firma de una función y la implementación de ella, así como manipular parámetros tanto en la función principal como en otras.

Marco Referencial:

¿Que son las funciones?

Las funciones son los bloques de construcción básicos de C. Dentro de ellas se da toda la actividad del programa.

La sintaxis básica para definir una función es la siguiente:

```
valorRetorno nombre (parámetros) {  
    // bloque de código de la función  
}
```

Una función puede recibir parámetros de entrada, los cuales son datos de entrada con los que trabajará la función, dichos parámetros se deben definir dentro de los paréntesis de la función, separados por comas e indicando su tipo de dato, de la siguiente forma:

(tipoDato nom1, tipoDato nom2, tipoDato nom3...)

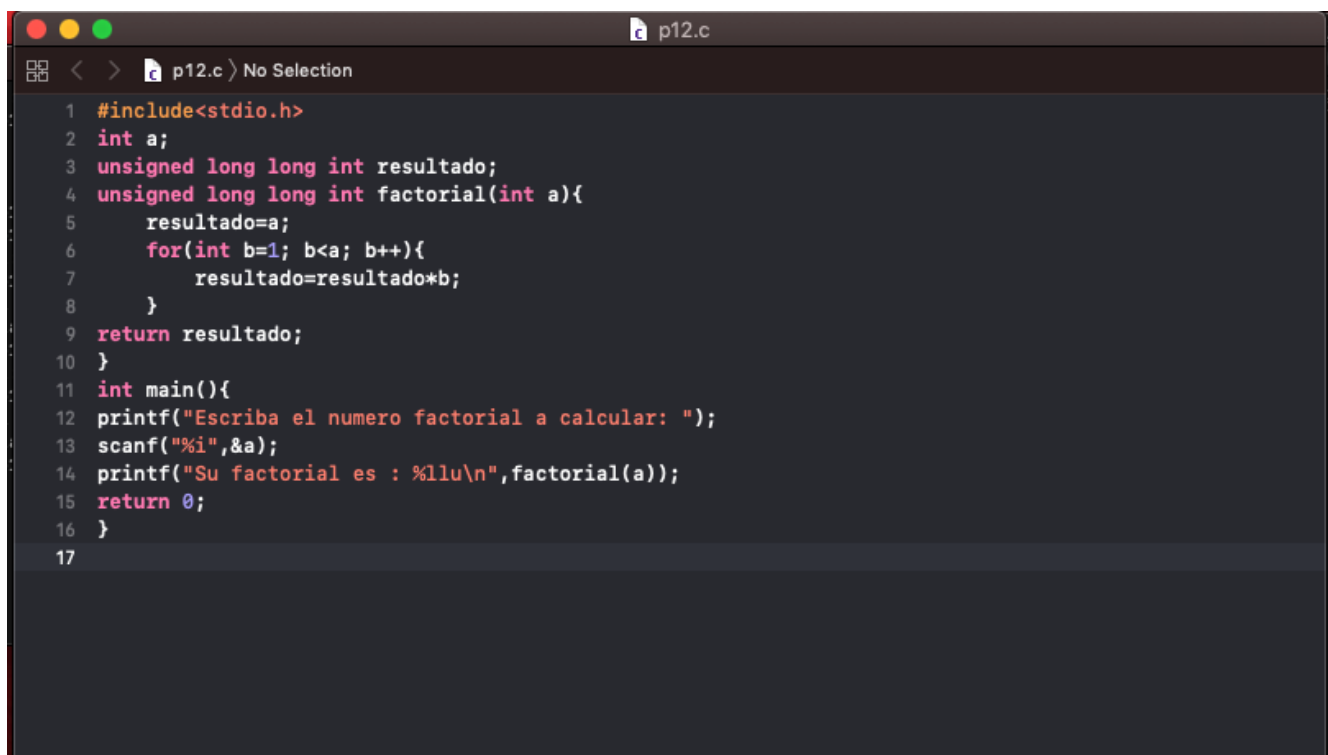
El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma. El valor de retorno puede ser cualquiera de los tipos de datos vistos hasta el momento (entero, real, carácter o arreglo), aunque también se puede regresar el elemento vacío (void).

La firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función; finaliza con punto y coma (;). Los nombres de los parámetros no necesariamente deben ser iguales a los que se encuentran en la definición de la función. Las funciones definidas en el programa no necesariamente deberán ser declaradas; esto dependerá de su ubicación en el código.

En lenguaje C la función principal se llama *main*. Cuando se ordena la ejecución del programa, se inicia con la ejecución de las instrucciones que se encuentran dentro de la función *main*, y ésta puede llamar a ejecutar otras funciones, que a su vez éstas pueden llamar a ejecutar a otras funciones, y así sucesivamente.

Procedimiento/Resultados:

1. Crear un programa que tenga una función que regrese el factorial de numero de entrada



```
1 #include<stdio.h>
2 int a;
3 unsigned long long int resultado;
4 unsigned long long int factorial(int a){
5     resultado=a;
6     for(int b=1; b<a; b++){
7         resultado=resultado*b;
8     }
9     return resultado;
10 }
11 int main(){
12     printf("Escriba el numero factorial a calcular: ");
13     scanf("%i",&a);
14     printf("Su factorial es : %llu\n",factorial(a));
15     return 0;
16 }
17
```

Nota: se utilizó unsigned long long int para dar mayor rango al problema

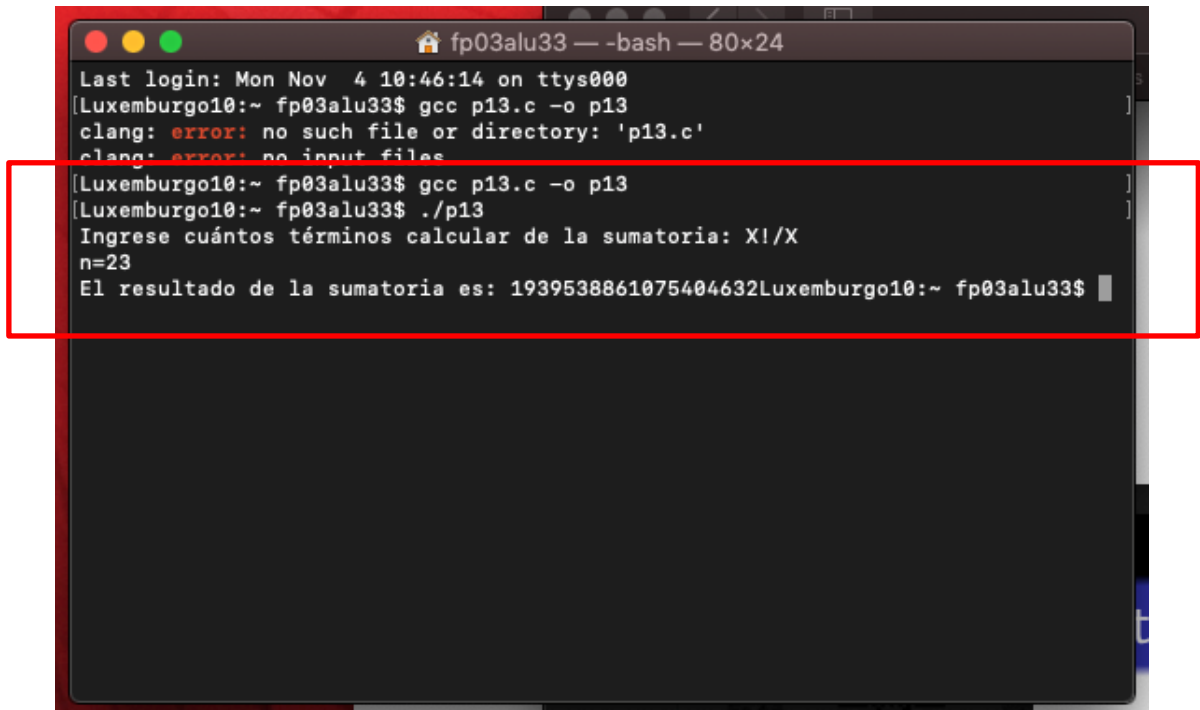
Compilado

```
fp03alu33 — -bash — 80x24
Last login: Mon Nov  4 10:39:45 on ttys000
[Luxemburgo10:~ fp03alu33$ ls
Desktop      Library      Pictures     p12.c
Documents    Movies       Public       tarea6
Downloads    Music        p12          tarea6.c
[Luxemburgo10:~ fp03alu33$ gcc p12.c -o p12
[Luxemburgo10:~ fp03alu33$ ./p12
Escriba el numero factorial a calcular: 4
Su factorial es : 24
[Luxemburgo10:~ fp03alu33$ ./p12
Escriba el numero factorial a calcular: 42
Su factorial es : 7538058755741581312
[Luxemburgo10:~ fp03alu33$ ./p12
```

2. Crear un programa que tenga una función que regrese el resultado de la serie

```
p13.c
1 #include <stdio.h>
2 #include <math.h>
3 int a;
4 unsigned long long int resultado;
5 unsigned long long int factorial(int a){
6     resultado=a;
7     for(int b=1; b<a; b++){
8         resultado=resultado*b;
9     }
10    return resultado;
11 }
12 int main()
13 {
14     unsigned long long int suma;
15     int n;
16     printf("Ingrese cuántos términos calcular de la sumatoria: X!/X\n");
17     printf("n=");
18     scanf("%i",&n);
19     suma=factorial(1)/1;
20     for (a=2;a<=n;a++){
21         suma=suma+ factorial(a)/a;
22     }
23     printf("El resultado de la sumatoria es: %llu",suma);
24     return 0;
25 }
26
```

Compilado

A terminal window titled 'fp03alu33 — -bash — 80x24' showing the compilation and execution of a C program. The user 'Luxemburgo10' is at the prompt. The terminal output is as follows:

```
Last login: Mon Nov  4 10:46:14 on ttys000
Luxemburgo10:~ fp03alu33$ gcc p13.c -o p13
clang: error: no such file or directory: 'p13.c'
clang: error: no input files
Luxemburgo10:~ fp03alu33$ gcc p13.c -o p13
Luxemburgo10:~ fp03alu33$ ./p13
Ingrese cuántos términos calcular de la sumatoria: X!/X
n=23
El resultado de la sumatoria es: 1939538861075404632Luxemburgo10:~ fp03alu33$
```

Conclusiones:

Con esta práctica vimos una manera más fácil de escribir un programa, “desmenuzándolo” en pequeños problemas que nos llevaran a la solución final. También aprendimos otras herramientas del lenguaje C para hacer nuestros programas de una manera más completa. En conclusión los subprogramas son una herramienta bastante útil para desarrollar programas más complejos y formales.

Bibliografía:

<https://drive.google.com/drive/folders/1ifhWPHI9jT9Nma-l6pT32JOzahsVmUG9>

<http://lcp02.fi-b.unam.mx/>

<http://programandoenc.over-blog.es/article-32481588.html>