

Carátula para entrega de prácticas	
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>	3
<i>No de Práctica(s):</i>	12
<i>Integrante(s):</i>	Uno Karin Natalia
<i>No. de Equipo de cómputo empleado:</i>	26
<i>No. de Lista o Brigada:</i>	8723 #50
<i>Semestre:</i>	1
<i>Fecha de entrega:</i>	4 de noviembre del 2019
<i>Observaciones:</i>	Tarde entrega

CALIFICACIÓN: 8

# PRÁCTICA 12:

## FUNCIONES

### 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.

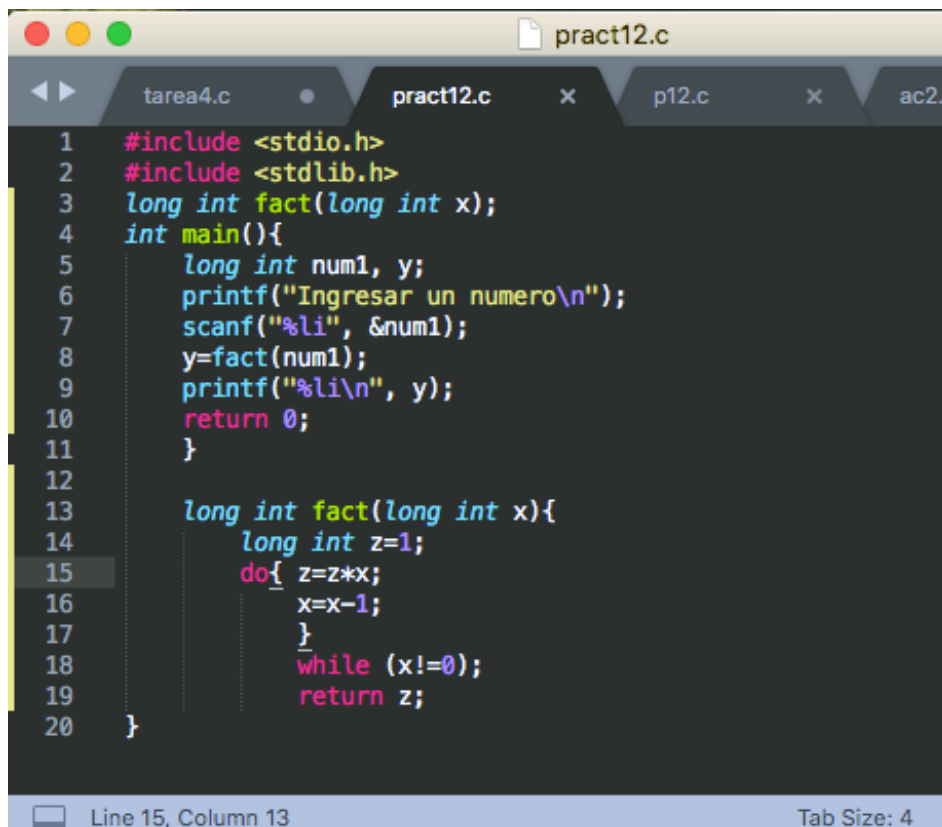
### INTRODUCCIÓN

Las funciones permiten a un programador modularizar un programa. Todas las variables declaradas en las definiciones de función son variables locales (son conocidas sólo en la función en la cual están definidas). La mayor parte de las funciones tienen una lista de parámetros. Los parámetros proporcionan la forma de comunicar información entre funciones. Los parámetros de función son también variables locales.

Existen varios intereses que dan motivo a la "funcionalización" de un programa. El enfoque de divide y vencerás hace que el desarrollo del programa sea más manipulable. Otra motivación es la reutilización del software, el uso de funciones existentes como bloques constructivos para crear nuevos programas. Cada función deberá limitarse a ejecutar una tarea sencilla y bien definida y el nombre de la función deberá expresar claramente dicha tarea. 1 Si no se puede elegir un nombre conciso es probable que la función esté intentando ejecutar demasiadas tareas diversas.

### DESARROLLO/RESULTADO

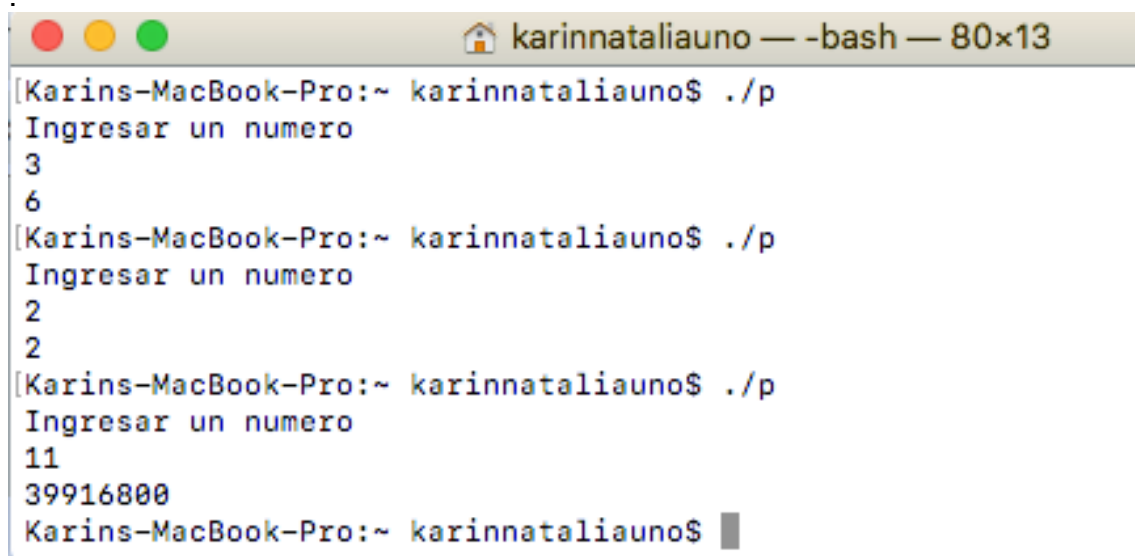
Actividad 1. Crear un programa que tenga una función que regrese el factorial de un número de entrada.



```
1  #include <stdio.h>
2  #include <stdlib.h>
3  long int fact(long int x);
4  int main(){
5      long int num1, y;
6      printf("Ingresar un numero\n");
7      scanf("%li", &num1);
8      y=fact(num1);
9      printf("%li\n", y);
10     return 0;
11 }
12
13 long int fact(long int x){
14     long int z=1;
15     do{ z=z*x;
16         x=x-1;
17     }
18     while (x!=0);
19     return z;
20 }
```

Line 15, Column 13      Tab Size: 4

El programa funciona de esta manera:

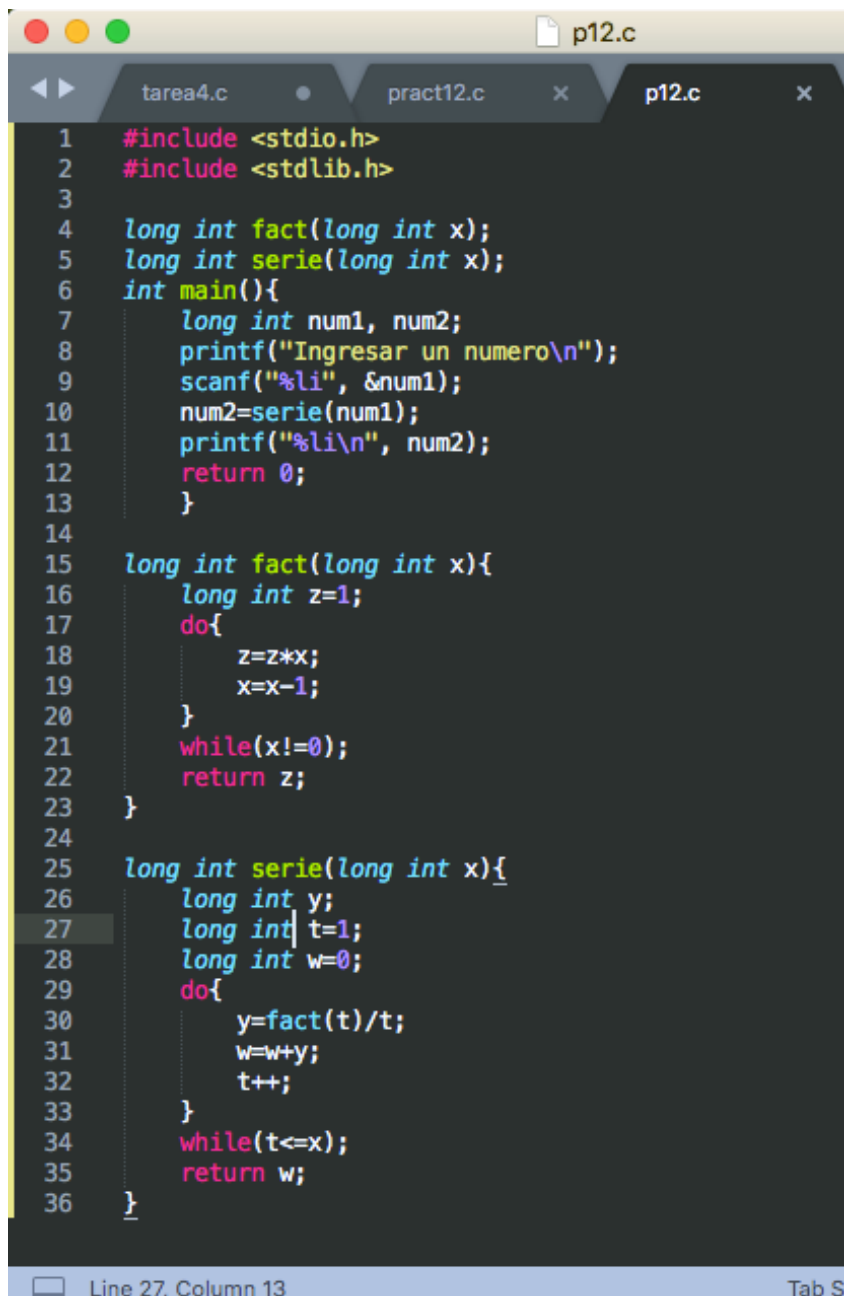


```
karinnataliauno — -bash — 80x13
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
3
6
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
2
2
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
11
39916800
Karins-MacBook-Pro:~ karinnataliauno$
```

Actividad 2.

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

$$\sum_{x=1}^n \frac{x!}{x}$$

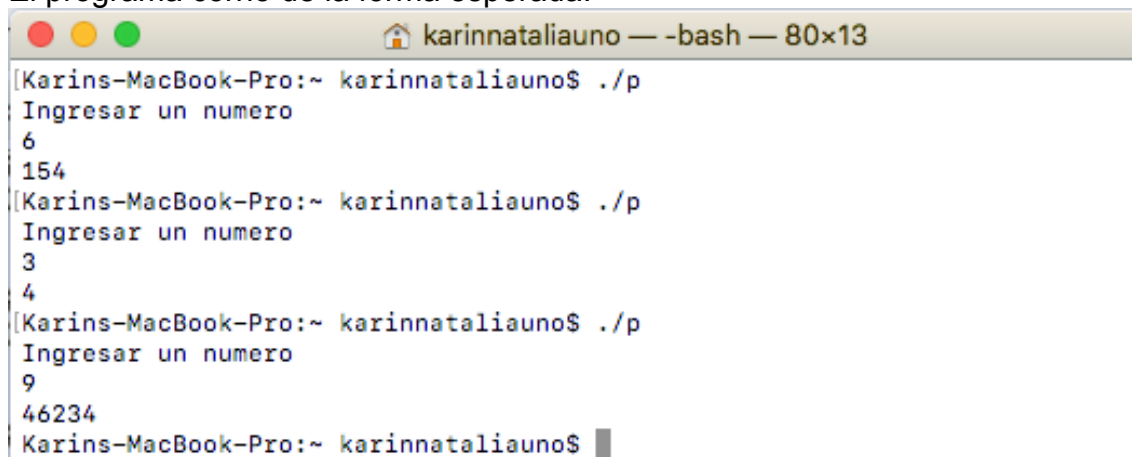


```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  long int fact(long int x);
5  long int serie(long int x);
6  int main(){
7      long int num1, num2;
8      printf("Ingresar un numero\n");
9      scanf("%li", &num1);
10     num2=serie(num1);
11     printf("%li\n", num2);
12     return 0;
13 }
14
15 long int fact(long int x){
16     long int z=1;
17     do{
18         z=z*x;
19         x=x-1;
20     }
21     while(x!=0);
22     return z;
23 }
24
25 long int serie(long int x){
26     long int y;
27     long int t=1;
28     long int w=0;
29     do{
30         y=fact(t)/t;
31         w=w+y;
32         t++;
33     }
34     while(t<=x);
35     return w;
36 }
```

Line 27, Column 13

Tab S

El programa corrió de la forma esperada.



```
karinnataliauno — -bash — 80x13
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
6
154
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
3
4
[Karins-MacBook-Pro:~ karinnataliauno$ ./p
Ingresar un numero
9
46234
Karins-MacBook-Pro:~ karinnataliauno$
```

## CONCLUSIÓN

Podemos concluir que el uso de funciones vuelve la forma de programar un tanto más ordenada y más clara. Además, al utilizar o crear varias funciones se vuelve más práctico el “llamarlas” por medio de la función principal, sin tener que volver a escribir todo el bloque de códigos que contiene cierta función, a la que se le quiere ejecutar más procesos si es que lo requiere.