	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 programación

Grupo: 3

No de Práctica(s): 12

Integrante(s): Rivera Sosa Arlethe

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: 317083033

Semestre: 2020-1

Fecha de entrega: 04/noviembre/19

Observaciones: Muy bien

CALIFICACIÓN: **10**

Prototipos y parámetros

Un prototipo de función le da información importante al compilador. En el prototipo se indican el tipo de dato que retorna la función, el número, tipo y orden de parámetros que recibe la misma. El compilador utiliza los prototipos para verificar las llamadas a funciones. Antes el prototipado no existía y el compilador no podía detectar ciertos errores.

Un ejemplo de un prototipo de función:

- int máximo (int, int, int): este prototipo indica que la función de nombre máximo retorna como resultado un valor de tipo entero. Además, informa que la función debe ser llamada con tres parámetros del tipo entero también.

El prototipo también sirve para la coerción de argumentos, o sea, si la función es llamada con valores de otro tipo diferentes a los definidos en el prototipo de esa función, el compilador tratará de convertir esos valores a los tipos de datos correspondientes declarados en el prototipo. Si la función espera un entero y recibe un decimal, intentará convertir ese decimal al entero truncando la parte decimal, lo cual puede generar errores si la exactitud de ese dato es importante. Por esta razón se debe tener mucho cuidado en la coerción de tipos automática. El prototipado de funciones puede omitirse cuando se programa; el compilador formará el prototipo dependiendo de la primera aparición de la función. También, por omisión, el compilador asume que toda función regresa un valor del tipo entero de forma predeterminada.

Normalmente, las funciones operan sobre ciertos valores pasados a las mismas ya sea como constantes literales o como variables, aunque se pueden definir funciones que reciban parámetros. Existen dos formas en C++ de pasar parámetros a una función; por referencia o por valor. El hecho es que, si en una declaración de función se declaran parámetros por referencia, a los mismos no se les podrá pasar valores literales ya que las referencias apuntan a objetos residentes en la memoria; por otro lado, si un parámetro es declarado para ser pasado por valor, el mismo puede pasarse como una constante literal o como una variable. Los parámetros pasados por referencia pueden ser alterados por la función que los reciba, mientras que los parámetros pasados por valor o copia no pueden ser alterados por la función que los recibe, es decir, la función puede manipular a su antojo al parámetro, pero ningún cambio hecho sobre este se reflejará en el parámetro original. Los parámetros son variables locales a los que se les asigna un valor antes de comenzar la ejecución del cuerpo de una función. Su ámbito de validez, por tanto, es el propio cuerpo de la función. El mecanismo de paso de parámetros a las funciones es fundamental para comprender el comportamiento de los programas en C.

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.

Actividad: las actividades deben tener los prototipos de sus funciones, y sus funciones implementadas después del main.

- Crear un programa que tenga una función que regrese la factorial de un número de entrada.

p12.c

```c
#include <stdio.h>
#include <stdlib.h>
long int factorial(long int a);
int main(){
    long int num1, b;
    printf("ingresar numero\n");
    scanf("%li", &num1);
    b=factorial(num1);
    printf("%li\n", b);
    return 0;
}
long int factorial(long int a){
    long int c=1;
    do{
        c=c*a;
        a=a-1;
    }
    while(a!=0);
    return c;
}
```

```
Familia@DESKTOP-NE4QAFJ ~
$ gcc p12.c -o act

Familia@DESKTOP-NE4QAFJ ~
$ ./act
ingresar numero
5
120

Familia@DESKTOP-NE4QAFJ ~
$ ./act
ingresar numero
7
5040
```

Hacemos un programa usando un prototipo de función, para que nos de la factorial de un número. Haciendo que se multiplique por los números que están antes que él.

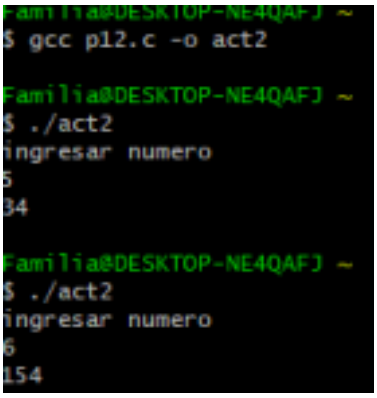
- Crear un programa que tenga una función que regrese el resultado de la serie:

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

Para un número n de entrada. Utilizar la función de factorial de la primera actividad.

p12.c

```c
#include <stdio.h>
#include <stdlib.h>
long int factorial(long int a);
long int form(long int a);
int main(){
    long int num1, cat;
    printf("ingresar numero\n");
    scanf("%li", &num1);
    cat=form(num1);
    printf("%li\n", cat);
    return 0;
}
long int factorial(long int a){
    long int c=1;
    do{
        c=c*a;
        a=a-1;
    }
    while(a!=0);
    return c;
}
long int form(long int a){
    long int b;
    long int e=1;
    long int r=0;
    do{
        b=factorial (e)/e;
        r=r+b;
        e++;
    }
    while(e<=a);
    return r;
}
```



Usando nuestro programa anterior, hay que agregarle la división, pues en la primera parte solo sacaba la factorial y en la segunda buscaos que encuentre en factorial y se divida en el número.

Conclusión: usar parámetros dentro de un programa para que nos haga una función en específico, nos permite escribir nuevamente la misma y que se vuelva a nombrar. Y los prototipos nos ayudan a tener un programa más estético, para poderlo desarrollar bien, aparte de que nos dice que dato retorna la función.