|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Funciones** | | | **No.** | **1** |
| **Asignatura:** | **Métodos Numéricos** | **Carrera:** | **INGENIERÍA EN SISTEMAS COMPUTACIONALES** | **Duración de la práctica (Hrs)** | **2** |

**I. Competencia(s) específica(s):**

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

* otro

**III. Material empleado:**

* Computadora
* App Dev-Cpp

**IV. Desarrollo de la práctica:**

Antes de empezar cabe mencionar que una función es una operación que toma uno o más valores llamados argumentos y produce un valor llamado resultado.

Una función es un bloque de código reconocido por un identificador que realiza un trabajo específico.

Su propósito es dividir los programas en módulos manejables separados (divide y vencerás).

Ventajas

1. Facilita el diseño descendente.

2. Los procedimientos dentro de ellas se pueden ejecutar varias veces.

3. Facilita la división de tareas.

4. Se pueden probar individualmente

5. Con funciones apropiadamente diseñadas, es posible ignorar como se realiza una tarea, sabiendo qué es lo que hacen.

Modo de uso

1. Funciones diseñadas para realizar operaciones a partir de sus argumentos devolver un valor basado en sus cálculos.

2. Funciones que no reciben argumentos, realizan un proceso y devuelven un valor.

3. Funciones que no tienen argumentos ni valor de retorno explícito, operan sobre el entorno de variables globales o atributos del sistema operativo.

**Declaración de funciones**

El formato para la declaración de funciones es:

Tipo: especifica el tipo de valor que devuelve la función. Si no se especifica tipo, el compilador asume que es entero (int).

Tipo nombre\_funcion (lista de parámetros) {cuerpo de la función}

**La segunda para declarar una función**

**Declaración de funciones (2)**

**tipo nombre\_funcion(lista de parámetros){ cuerpo de la función }**

Lista de parámetros: es la lista de nombres de variables separados por comas con sus tipos asociados que reciben los valores de los argumentos actuales de la llamada a la función.

**Declaración de funciones (3)**

Forza la salida inmediata de la función en que se encuentra.

Una función puede retornar valor sólo cuando el tipo de retorno no es void.

Devuelve un valor a la función que realizó la llamada.

return (expresion);

**Declaración de funciones (4)**

Tradicionalmente en C se declaran como prototipos al inicio del programa. Después se declara la función main, y después se hace la declaración formal de las funciones.

También pueden declararse las funciones al inicio del programa y después declarar la función main sin declarar prototipo.

**Ejemplo de funciones**

**int suma (int a, int b) {int c; return int c = a + b;}**

**float promedio (int a, int b, int c) {int sum = a + b + c; float prom = sum/3; float prom;}**

**Llamadas a funciones**

Para llamar a una función se especifica su nombre y la lista de argumentos sin poner el tipo de dato.



En una llamada habrá un argumento real por cada argumento formal, respetando el orden de declaración.

Argumento formal: Los que aparecen en la definición de la función.

Argumento real: Los que se pasan en la llamada a la función.

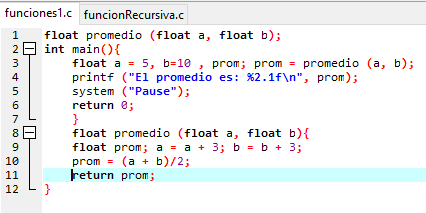
**Paso de parámetros por valor**

Se hace una copia del valor del argumento en el parámetro formal.

La función opera internamente con estos últimos.

Los parámetros formales se crean al entrar a la función y se destruyen al salir de ella, cualquier cambio realizado por la función en los parámetros formales no tienen ningún efecto sobre los argumentos.

Se crea la función



Se crean las variables

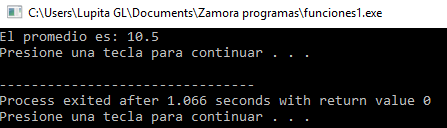
Se imprime en pantalla

Se crean las variables

Se manda a llamar la función

Ahora promedio es igual a (a mas b entre 2)

Retorna a promedio



**Variables locales y globales**

Variables Locales: Se declaran dentro de la función y sólo están disponibles durante su ejecución.

Se crean cuando se entra en ejecución una función y se destruyen cuando se termina.

Variables globales: Se declaran fuera de las funciones. Pueden ser utilizadas por todas las funciones.

Existen durante toda la vida del programa.

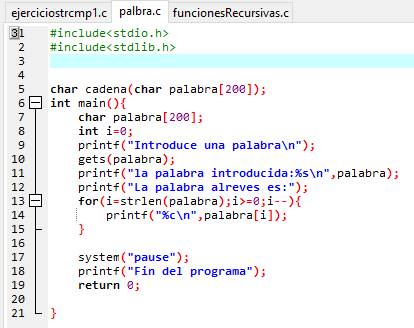
**Ejercicio**

Escribir una función que se llame máximo que reciba dos número por parámetros y que regrese el mayor de ellos.

Escribir una función que reciba caracteres del teclado hasta recibir un espacio o un salto de línea (enter) y a continuación mostrar todos los caracteres en orden inverso**.**

**Ejemplo:  Entrada: Hola  Salida: aloH**

Bibliotecas



Introducimos un ciclo donde i es igual a strlen almacena a palabra donde i es mayor a 0 y donde i --

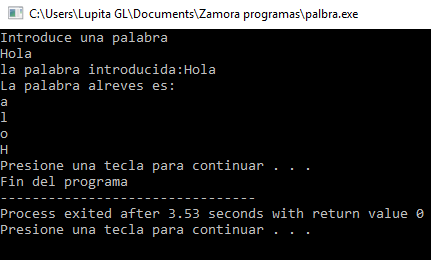
Variable de tipo entero inicializada en 0

Imprime un mensaje

Se declara la función

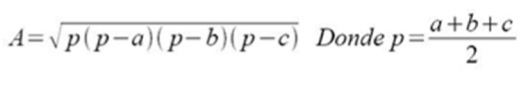
Almacena la palabra

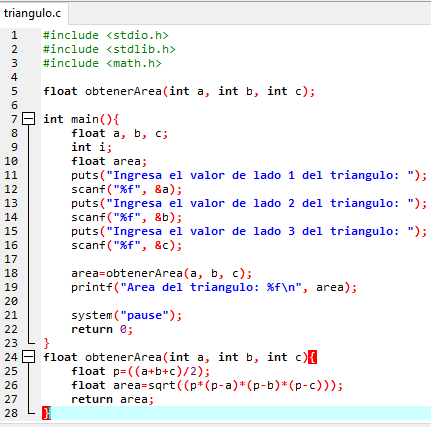
Se crea el método principal

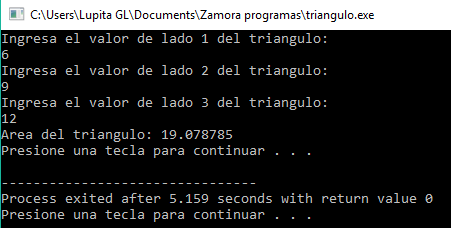


**Ejercicio**

Escribir una función que tome como parámetros las longitudes de los tres lados de un triángulo (a, b, c) y devuelva el área del triángulo.







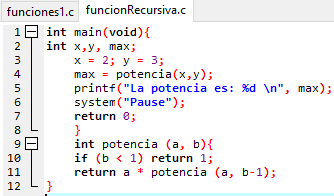
**FUNCIONES RECURSIVAS**

Se llaman funciones recursivas a aquellas que se llaman a sus mismas de forma repetida hasta que se cumpla alguna condición.

Cada llamada implica el almacenamiento de variables de estado y otros parámetros.

**Ejemplo**: Calcular la potencia de forma recursiva

Variables de tipo entero



Dentro del parámetro se crea una variable llamada potencia y almacena a y b.

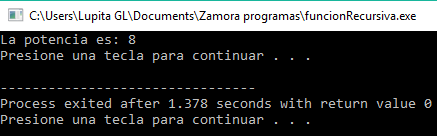
Posteriormente la variable max será igual a la potencia y esta almacena las variables de x y (y).

Se imprime en pantalla

Las variables x y (y) tienen un valor

Después retorna a por potencia dentro de un valor almacenado (a,b-1).

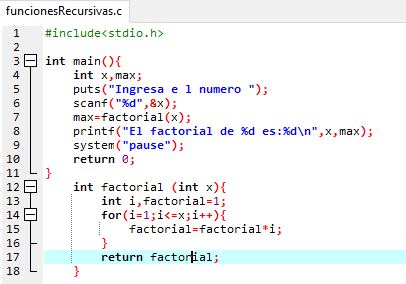
Ahora con un if muestra que si b es menor que 1 y retorna 1.



**Ejercicios**

Haz un programa con funciones recursivas que calcule el factorial de un número n ingresado desde teclado

Se crean las variables de tipo entero en este caso x y max.



Se crean las variables

Dentro de la variable max será igual a factorial que almacena a x.

Para mostrar en pantalla los datos utlizaremos un Scanf.

Con ayuda del Puts pedimos al usuario que ingrese el número.

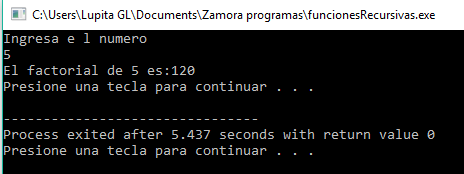
Se manda a llamar la función

Se crean las variables donde la variable factorial se inicializa en 1.

Se crea el ciclo for donde i es igual a 1 y i es menor o igual que x y i ++.

Ahora la factorial es igual a factorial por i

Retorna la factorial.



**V. Conclusiones:** al realizar las funciones nos permiten realizar los programas rápido y más fácil en lo particular se hizo difícil realizar los programas ya que en algunas partes me confundía pero con la práctica se me fue haciendo más fácil.

Las funciones nos permiten interactuar con parámetros donde una función es una operación que toma uno o más valores llamados argumentos y produce un valor llamado resultado.