**Investigación de las funciones**

# Gutiérrez Padilla Luis Pablo

# 220277726

## ¿Qué es una función?

Es un conjunto de líneas de código que hacen una tarea en concreto, esta función puede retornar un valor o no. Las funciones tienen la posibilidad de tener parámetros o argumentos, que son variables que modifican su comportamiento.

## ¿Cuál es la sintaxis de una función?

Se debe empezar por asignar la función principal con un **main()**, despues declarar las variables que se van a usar, que pueden ser de tipo **int, char, float** y **double,** y después empezar a darle instrucciones al programa, ya sea con operaciones matemáticas directas, con condicionales o con muchos otros tipos de opciones.

## ¿Qué es la programación modular?

Es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable. Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que ésta puede resolver.

## ¿Qué son las variables globales y locales?

Una variable global es una [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) accesible en todos los [ámbitos](https://es.wikipedia.org/wiki/%C3%81mbito_(programaci%C3%B3n)) de un [programa informático](https://es.wikipedia.org/wiki/Programa_(inform%C3%A1tica)). Y una variable local es una [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) a la que se le otorga un [ámbito](https://es.wikipedia.org/wiki/%C3%81mbito_(programaci%C3%B3n)) local. Tales variables sólo pueden accederse desde la [función](https://es.wikipedia.org/wiki/Subrutina) o [bloque de instrucciones](https://es.wikipedia.org/w/index.php?title=Bloque_de_instrucciones&action=edit&redlink=1) en donde se [declaran](https://es.wikipedia.org/w/index.php?title=Declaraci%C3%B3n_(inform%C3%A1tica)&action=edit&redlink=1). Las variables locales se contraponen a las [variables globales](https://es.wikipedia.org/wiki/Variable_global).

## Ventajas y desventajas de las variables globales y locales.

Las variables locales hacen que los programas de computadora sean más fáciles de depurar y mantener. Los programadores pueden determinar el punto exacto en el que un programa modifica el valor de una variable local, mientras que las variables globales pueden ser modificadas en cualquier lugar dentro del código fuente. Las variables locales también producen menos interacciones inesperadas con llamadas a funciones, o el mundo exterior, conocidas como efectos secundarios, que las variables globales. Las variables globales son accesibles en funciones en las que no son utilizadas, en contradicción con el concepto de programación modular, en el que el código del programa se compone de bloques o módulos pequeños aislados, por lo que los programadores que favorecen la estructura modular rara vez las utilizan.

## ¿Qué es un parámetro?

es una [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) utilizada para recibir valores de entrada en una [rutina](https://es.wikipedia.org/wiki/Rutina_(inform%C3%A1tica)), [subrutina](https://es.wikipedia.org/wiki/Subrutina) o [método](https://es.wikipedia.org/wiki/M%C3%A9todo). Dichos valores, que serán enviados desde la rutina invocante, son llamados argumentos. La [subrutina](https://es.wikipedia.org/wiki/Subrutina) usa los valores asignados a sus parámetros para alterar su comportamiento en tiempo de ejecución. La mayor parte de los [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguajes_de_programaci%C3%B3n) pueden definir subrutinas que aceptan cero o más argumentos.

## Funciones con y sin prototipo.

Una declaración de función precede a la definición de función y especifica el nombre, el tipo devuelto, la clase de almacenamiento y otros atributos de una función. Para ser un prototipo, la declaración de función también debe establecer tipos e identificadores para los argumentos de la función. El prototipo tiene la misma forma que la definición de función, excepto en que termina con un punto y coma inmediatamente después del paréntesis de cierre y, por consiguiente, no tiene ningún cuerpo. En cualquier caso, el tipo de valor devuelto debe coincidir con el tipo de valor devuelto especificado en la definición de función.

Los prototipos de función tienen los siguientes usos importantes:

* Establecen el tipo de valor devuelto para funciones que devuelven tipos diferentes de int. Aunque las funciones que devuelven valores int no requieren prototipos, estos se recomiendan.
* Sin prototipos completos se realizan conversiones estándar, pero no se hace ningún intento de comprobar el tipo o el número de argumentos con el número de parámetros.
* Los prototipos se utilizan para inicializar punteros a funciones antes de que se definan esas funciones.
* La lista de parámetros se utiliza para comprobar la correspondencia de los argumentos de la llamada de función con los parámetros de la definición de función.

## Clasificación de las funciones.

## Funciones definidas por el compilador.

## Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código. Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.

## Funciones de paso de parámetros.

Paso de parámetros por valor: No se modifica el valor de ninguno de los parámetros que son pasados, es decir, cuando se produce el paso de parámetros, se hace una copia de cada parámetro y se modifica y se evalúan las copias dentro de la función, hasta que se ejecuta la sentencia return o se sale de la función es cuando se eliminan las copias existentes y se siguen utilizando las variables que se habían pasado (por valor) con los valores que tenían cuando se produjo la llamada a la función.

Paso de parámetros por referencia En este tipo, sí que se modifica el valor de los parámetros, lo que se pasa no es el valor de la(s) variable(s), sino la dirección de memoria que ésta(s) ocupa(n), con lo que al modificar su valor dentro de la función, se modifica también el valor fuera de la función. Cuando se ejecuta la sentencia return o se salga de la función las variables que se pasaron (por referencia) tienen el valor que tenían una vez se sale de la función.

## Funciones sin paso de parámetros.

La función sin parámetros significa que no lleva variables desde la función prototipo. Con resultado significa que la función lleva un return y retorna su valor a bien otra función o a la función principal para normalmente seguir con otras operaciones.

## Funciones con paso de parámetros de referencia.

El paso de parámetros por referencia consiste en proporcionar a la subrutina a la que se le quiere pasar el argumento la [dirección de memoria](https://es.wikipedia.org/wiki/Direcci%C3%B3n_de_memoria) del dato. En este caso se tiene un único valor referenciado (o apuntado) desde dos puntos diferentes, el programa principal y la subrutina a la que se le pasa el argumento, por lo que cualquier acción sobre el parámetro se realiza sobre la misma posición de memoria.

El paso de la [dirección de memoria](https://es.wikipedia.org/wiki/Direcci%C3%B3n_de_memoria) debe ser transparente al programador. Los lenguajes que permiten tratar con direcciones de memoria, como C, permiten simular el paso de argumentos por referencia, sin disponer realmente de esta característica.

## Recursividad.

La recursividad es la propiedad de una función de poder llamarse a sí mismo cuando así se requiera. Si una función (o Subfunción) requiere volver a ejecutar alguna parte de su código (sin hacer uso de ciclos de repetición), tan solo se llama a sí misma.

Las características que deben cumplir los problemas recursivos son:

* La recursividad debe terminar alguna vez: caso base
* Cada nueva formulación estamos mas cerca del caso final

Ventajas de la Recursión:

Soluciones simples, claras

Soluciones elegantes.

Soluciones a problemas complejos.

# Bibliografías:

* Microsoft. (2016, 4 noviembre). *Prototipos de función*. Microsoft Docs. https://docs.microsoft.com/es-es/cpp/c-language/function-prototypes?view=msvc-160
* Universidad de Wisconsin. (s. f.). *Local and Global Variables*. Computer Sciences User Pages. http://pages.cs.wisc.edu/%7Ecalvin/cs110/LOCAL\_GLOBAL.html
* García, M. E. (s. f.). *Funciones: Tipos de funciones y Recursividad*. Universidad Tecnológica de la Mixteca. http://www.utm.mx/~mgarcia/PE4(Funciones).pdf
* Fabian Cruz, R. (2013, 22 mayo). *FUNCIONES DEFINIDAS POR EL USUARIO Y POR EL PROPIO LENGUAJE*. raquelfabiancruz.com. https://raquelfabiancruz-com.webnode.mx/news/trabajo-de-investigacion-funciones-definidas-por-el-usuario-y-por-el-propio-lenguaje/
* colaboradores de Wikipedia. (2020, 17 abril). *Argumento (informática)*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Argumento\_(inform%C3%A1tica)#:%7E:text=afecta%20al%20otro.-,Paso%20por%20referencia,direcci%C3%B3n%20de%20memoria%20del%20dato.
* Vallejos, O. A. (s. f.). *Tema VII. Recursividad*. Universidad Nacional del Nodeste. http://exa.unne.edu.ar/informatica/programacion1/public\_html/archivos/recursividad\_1.pdf