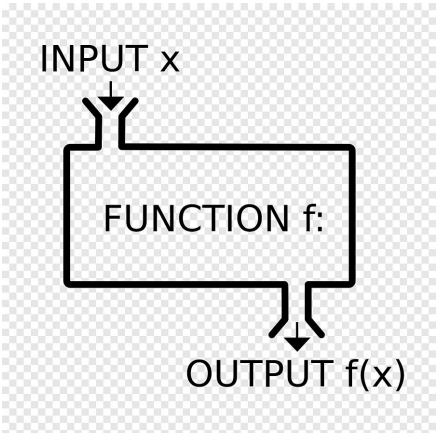
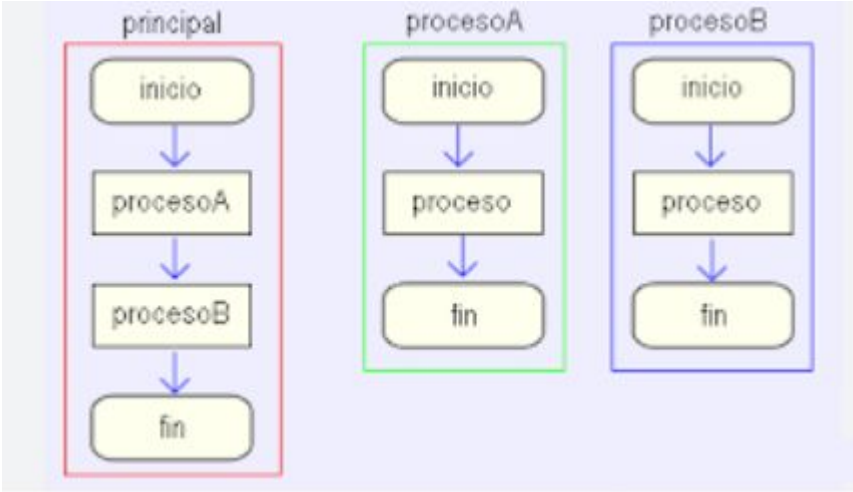
Funciones



Funciones

Básicamente una función puede realizar las mismas acciones que un programa:

- Aceptar datos.
- Realizar cálculos determinados y, finalmente,
- Devolver resultados



Funciones

Las funciones son invocadas desde otras funciones, con una excepción: la función global main(), que tienen todos los programas en C++. Permite al compilador conocer donde está el punto inicial de un programa. Por razones obvias, main() no puede ser invocada desde otras funciones.

Podemos distinguir 3 características de las funciones en C++:

- La definición
- La declaración
- La llamada

Definición de una función

Es el código que realiza las tareas para las que la función ha sido prevista. La primera línea de la definición recibe el nombre de encabezamiento y el resto, un bloque encerrado entre llaves, es el cuerpo de la función. La definición de una función se debe realizar en alguno de los ficheros que forman parte del programa.

La sintaxis habitual de una definición de función es la siguiente

```
tipo nombre_funcion(tipo_1 arg_1, ..., tipo_n arg_n)
{
    sentencias;
    return expresion;  // optativo
}
```

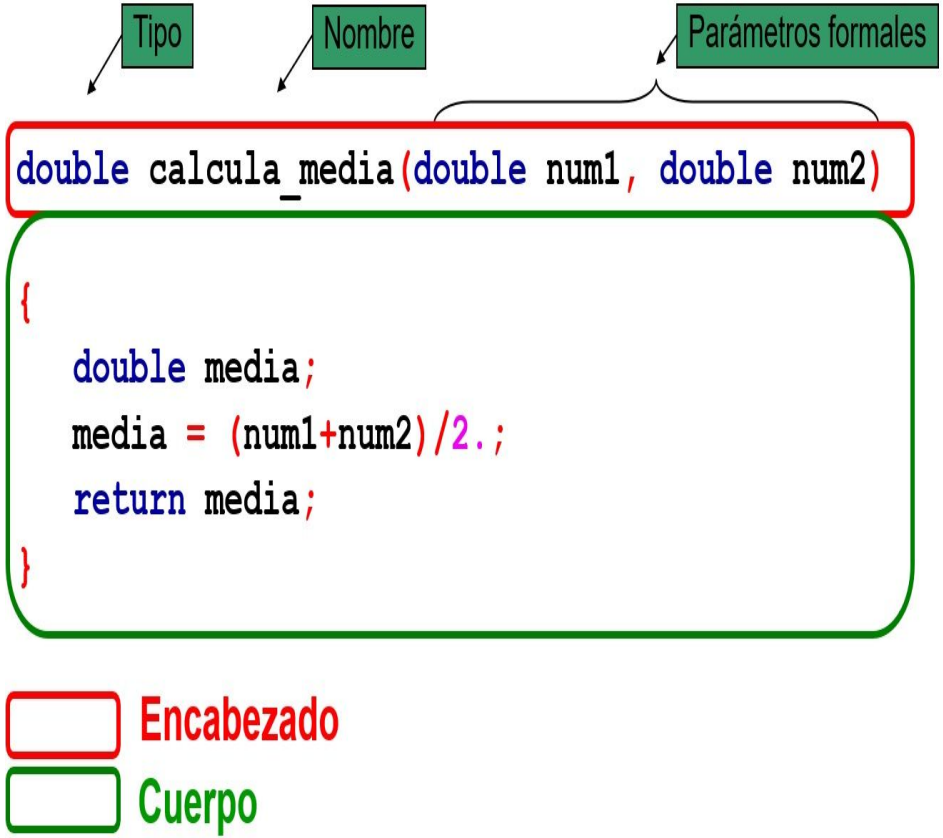
Funciones

Tipo: indica el tipo de valor (int, float, etc.) devuelto por la función.

Nombre_funcion: es el identificador usado para la función.

La lista de argumentos es una secuencia de declaración de parámetros separados por comas y encerrados entre paréntesis. Son los llamados parámetros formales de la función.

Return expresión es un salto incondicional que permite evaluar una expresión y devolver su valor a la función llamante.



LLamada a la función

La llamada a una función se hace especificando su nombre y, entre paréntesis, las expresiones cuyos valores se van a enviar como argumentos de la función.

Estos parámetros pueden ser identificadores o cualquier otra expresión válida. La llamada a una función se escribe de forma general como sigue:

```
salida = nombre_funcion(arg_1, arg_2, ..., arg_n);
```

Variable Local

Una variable local es aquella cuyo ámbito se restringe a la función que la ha declarado se dice entonces que la variable es local a esa función. Esto implica que esa variable sólo va a poder ser manipulada en dicha sección, y no se podrá hacer referencia fuera de dicha sección. Cualquier variable que se defina dentro de las llaves del cuerpo de una función se interpreta como una variable local a esa función.

Variables Globales

Una variable global es aquella que se define fuera del cuerpo de cualquier función, normalmente al principio del programa, después de la definición de los archivos de biblioteca (#include), de la definición de constantes simbólicas y antes de cualquier función. El ámbito de una variable global son todas las funciones que componen el programa, cualquier función puede acceder a dichas variables para leer y escribir en ellas. Es decir, se puede hacer referencia a su dirección de memoria en cualquier parte del programa.

Variables Locales(aclaración)

Las variables declaradas dentro de una función son automáticas por defecto, es decir, sólo existen mientras se ejecuta la función. Cuando se invoca la función se crean estas variables en la pila y se destruyen cuando la función termina. La única excepción la constituyen las variables locales declaradas como estáticas (static). En este caso, la variable mantiene su valor entre cada dos llamadas a la función aún cuando su visibilidad sigue siendo local a la función.

Variables Locales(aclaración)

También puede suceder que en un mismo ámbito aparezcan variables locales y globales con el mismo nombre. Cuando sucede esta situación, siempre son las variables locales y argumentos formales los que tienen prioridad sobre las globales.

Variables estáticas en una función

Cuando una variable se declara como estática, se le asigna espacio durante la vida útil del programa . Incluso si la función se llama varias veces, el espacio para la variable estática se asigna solo una vez y el valor de la variable en la llamada anterior se transfiere a la siguiente llamada de función. Esto es útil para implementar rutinas en C/C++ o cualquier otra aplicación en la que se deba almacenar el estado anterior de la función.

https://replit.com/join/wntdeknjam-maximo-arielari

Variables por & Referencia

No se pasa una copia de los valores originales, sino que se crea (cómo su nombre lo indica) una referencia que tiene la misma dirección en memoria del elemento original, esto quiere decir dos cosas, en primer lugar la variable con la cual se trabajará al interior de la función que recibe la referencia será por definición un sinónimo de la variable original, esto quiere decir que será idéntica y cuanto a la dirección de memoria, su contenido, etc; Por otro lado esto quiere decir que cualquier modificación que hagamos en la función al contenido de la referencia afectará de forma inmediata a las variables originales.

https://replit.com/join/tcerbwmjqm-maximo-arielari

Compilación Separada

Se emplea para generar tiempos de compilación más pequeños.

La labor es que las variables y funciones generadas en otros archivos deben ser asociados haciendo uso de :

- Extern
- Static
- Prototipos.

https://replit.com/join/nqoepxipxj-maximo-arielari

Bibliografía

https://www2.eii.uva.es/fund_inf/cpp/temas/8_funciones/funciones.html

https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap6/cap62.htm

https://www.geeksforgeeks.org/static-keyword-cpp/