

Las funciones en programación son un conjunto de instrucciones o sentencias que están agrupadas bajo un nombre en la memoria principal. Realizan una tarea específica y por lo general retornan un valor (aunque pueden no retornar ningún valor – a esto se lo conoce como procedimientos -Procedures-).

Las características principales de la implementación de funciones en la programación son:

- **Atomicidad:** Las funciones deben cumplir una **única tarea específica**. Cuanto más específica es la tarea, mayor probabilidad de reutilización tiene la función.
- **Modularidad o Modularización:** Dividir el programa principal en sub-programas o programas más pequeños. A esto también se lo conoce como programación modular, que consiste dividir en programa principal en módulos. Cada sub-programa genera un segmento en la memoria del programa, en el cual tiene su ámbito de trabajo.
- **Reutilización del código o de instrucciones:** Reutilizar las funciones para evitar la redundancia de instrucciones y para su posterior reusabilidad en otros programas. Lo que genera una reducción del programa principal en líneas de código y tamaño.

Estructura de una función: Toda función está compuesta por un nombre (identificador), parámetros o argumentos (valores) y retorno. Las funciones pueden tener o no tener parámetros y, en el caso del lenguaje C, pueden o no retornar un valor. Las funciones pueden: No recibir parámetros y no retornar valor o no recibir parámetros y retornar valor o recibir parámetros y no retornar valor o recibir parámetros y retornar valor.

Por ejemplo: Llamar o invocar a funciones propias del lenguaje C

getch(); //Es una función sin parámetros. El nombre es getch, no retorna valor. Ídem a **getch(void);**


pow(base,numero);//Función con dos parámetros. El nombre es pow, retorna valor de tipo float, en este caso el llamado a la función se convierte en un valor de tipo float, es decir que se puede almacenarlo en una variable, utilizarlo como parte de una expresión o una condición o informarlo en pantalla. Ejemplo:

float potencia;

int base, exponente;

....

```
potencia = pow (base, exponente);  
printf("La potencia es %.2f",potencia);
```



Los ejemplos anteriores son ejemplos de llamadas o de invocar a una función. La forma general para llamar o invocar a una función es (una operación en diagramación):

Nombre de la función (parámetros)

A cada valor que se le envía la función se lo llama **parámetro real o actual o argumento**, cada uno de ellos puede ser una expresión (Constante numérica, constante caracter, variable, operación, llamado a otra función – Que retorne valor -, cualquier combinación entre ambas) o void.

El tipo de dato void en el parámetro actual, significa que a la función no se le envían parámetros.

“Cuando se invoca o llama a una función, se entrega el control de la ejecución del programa a la función”.

“Una función puede llamar a otras funciones o ella misma (recursividad). Cualquier función puede llamar o invocar a cualquier otra función sin ningún tipo de restricciones”.

Construcción de una función: Toda función contiene un encabezado, un cuerpo y un retorno del control.

Encabezado

Está compuesto por el nombre de la función (en la sección superior), los parámetros formales y el tipo de dato de retorno de la función (puede ser int, float, char o void) - en la sección inferior -. Los nombres de las funciones tienen las mismas restricciones que el nombre de una variable. **Los parámetros formales (variables paramétricas) son los valores de entrada o ingreso a la función, siempre es la declaración de una variable por cada parámetro actual o void. Cada parámetro formal se corresponde con un parámetro actual o real (relación 1 a 1), es decir que ambos deben pertenecer al mismo tipo de dato, el pasaje de parámetros es por copia o copiar-valor (Se copian los valores de los parámetros actuales en los parámetros formales) o por referencia. Si el tipo de dato de retorno de la función es void, implica que la misma no retorna valor.**

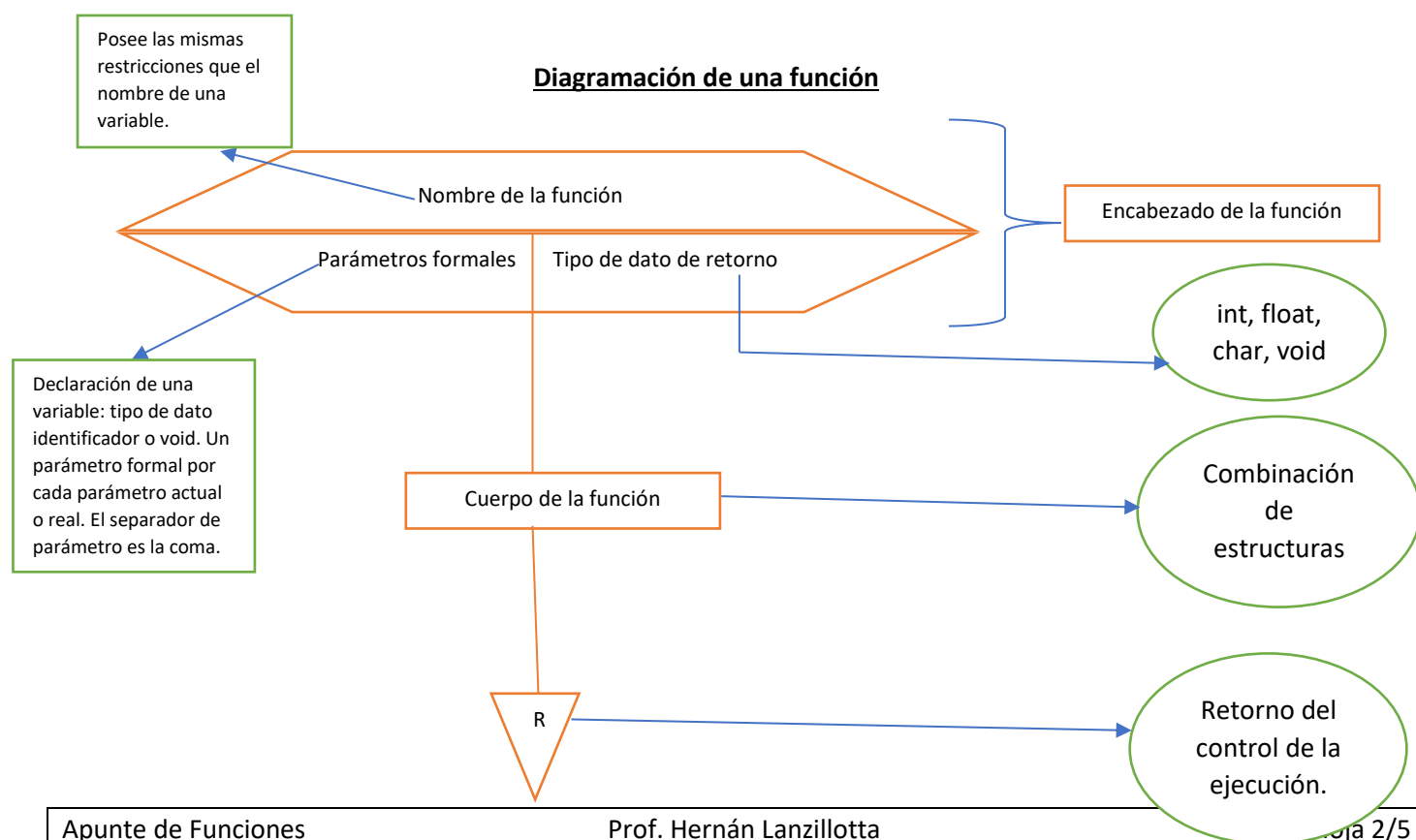
Cuerpo

Es el conjunto de instrucciones que desarrollan la tarea específica de la función, dentro de este se colocan todas las combinaciones de estructuras estudiadas (Secuencial, Selección, Iteración).

Retorno de control

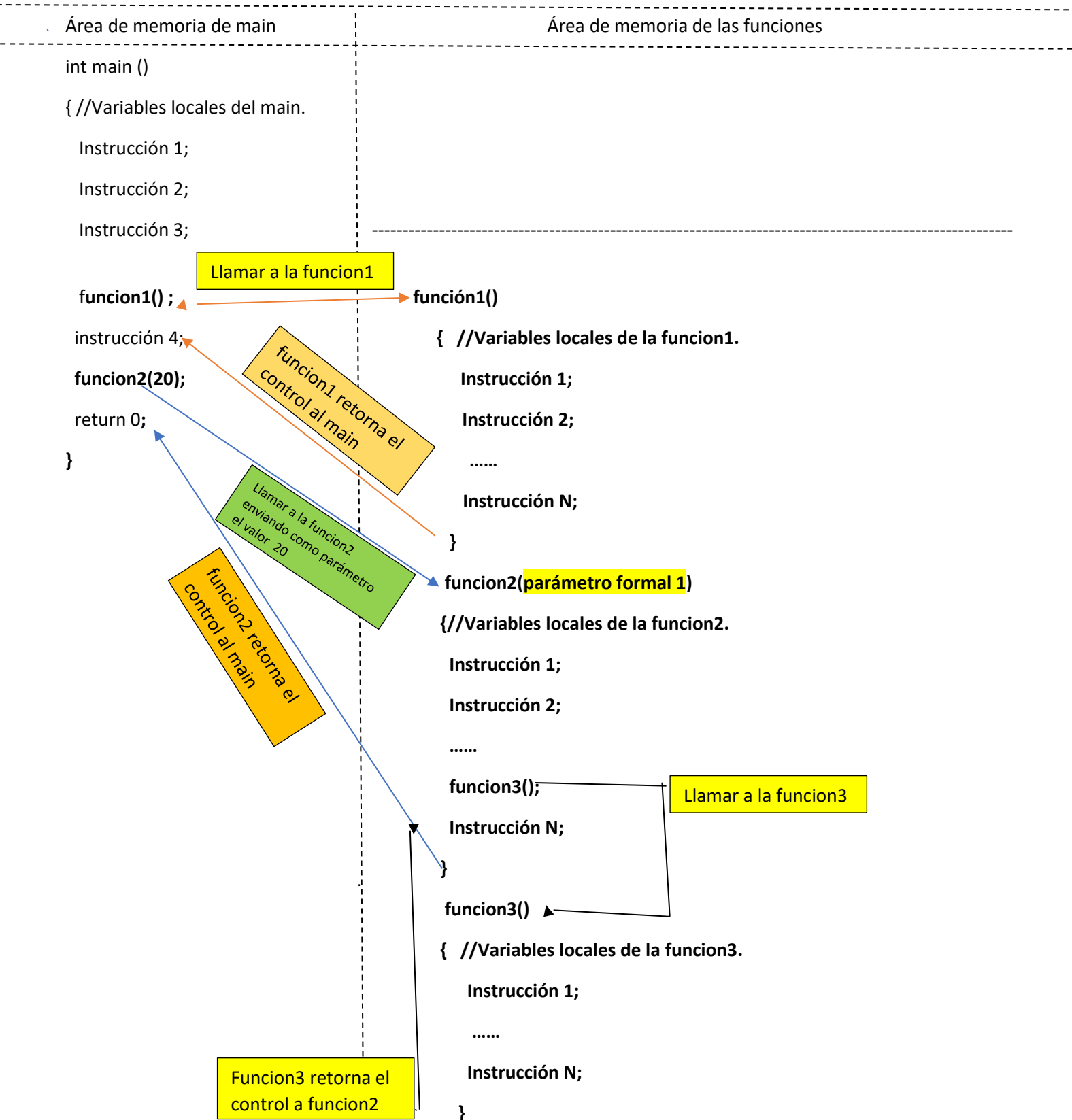
Cuando la función finaliza, le entrega el control de la ejecución a quién la llamó o invocó retornando el valor (en el caso de que el tipo de dato de retorno sea int, float o char) o void. Luego, la misma, elimina todos los recursos utilizados (variables locales y paramétricas) de la memoria principal.

Las variables locales: Son variables que se declaran dentro de la función y tienen su ámbito de trabajo en el área de memoria de esta, no son accesibles por otras funciones ni por el main. Como las variables declaradas en main no son accesibles por las funciones del programa. **Todo lo que se declara previo al main tiene ámbito o alcance global**, es decir que lo puede utilizar todo el programa, como por ejemplo las constantes declaradas con #define nombre valor antes del main es accesible por el main y las funciones.



Programa con funciones en pseudo código

“Las flechas expresan el control de la ejecución del programa”



Codificación de una función

- 1) Prototipo de una función: Es la codificación de casi todo o todo del encabezado de la función (Nombre de la función, parámetros y tipo de dato de retorno). Se coloca luego de las directivas al procesador, previo al main.

Tipo de dato de retorno Nombre de la función (tipos de datos de los parámetros formales separados por comas ó void);

Nota: Se puede colocar también el tipo de dato y el nombre de los parámetros formales, pero en la definición se deben colocar los mismos nombres que en el prototipo.

Por ejemplo:

-El prototipo de la función pow es:

`float pow (float, float);` //tiene dos parámetros formales de tipo float y retorna un tipo de dato float.

-El prototipo de la función getch() es

`void getch();` //No tiene parámetros formales, es opcional colocar el void. Pero sí se debe colocar en el tipo de dato del retorno.

- 2) Llamado o invocar a una función: Se coloca el nombre de la función y entre paréntesis los parámetros actuales o reales o argumentos, finalmente se coloca ;

Por ejemplo:

`float raiz = sqrt(5);` //A la variable raíz le asignamos lo que retorna la función sqrt.

`int valor=0;`

`mensaje(raiz,valor);` //Informar por pantalla el contenido de las variables raíz y valor.

`calculo (valor*2);` // Primero resuelve valor * 2 y el resultado es el valor que envía a la función.

`calculo(sumatoria(valor,5));` //Primero invoca a la función sumatoria y el resultado que retorna la misma, lo envía como parámetro a la función calculo.

`getch();` //generar una pausa.

- 3) Definición de una función: Es la codificación completa del desarrollo de la función (El encabezado completo + el cuerpo + retorno). Se coloca luego de la llave de cierre del main. **“Es el sub-programa, tiene la misma estructura que el main”**.

Encabezado

Regla práctica: Copiar el prototipo de la función sin él ; y pegarlo, luego asignar los nombres de los parámetros formales (variables paramétricas).

Cuerpo

Abrir llave.

Declaración de las variables locales (Si posee).

Codificación de todo el cuerpo de la función.

Si el tipo de dato de retorno es int, float o char colocar ->return valor o variable u operación;

Sino no colocar nada (Si la función retorna void se puede colocar return;)

Cerrar llave.

Ejemplo de definición de algunas funciones

float potencia (float base, float exponente) **//Encabezado de la función.**

{

float resultado; **// Variable local.**

resultado = pow(base) ; **//Invocar a una función que se encuentra en la biblioteca <math.h>.**

return resultado; **//Retorno del valor float (el tipo de dato de retorno de la función es float).**

}

void mensaje (float raiz, int valor) **//Encabezado de la función**

{

printf(“El valor de la raiz es =%.2f \nEl valor del numero entero es =%d”, raiz, valor);

// Tipo de dato de retorno void => No retorna valor.

}