UNIVERSIDAD NACIONAL EXPERIMENTAL DEL TÁCHIRA
VICERRECTORADO ACADÉMICO
DECANATO DE DOCENCIA
DEPARTAMENTO DE ING. EN INFORMÁTICA

UNIDAD CURRICULAR: # COMPUTACIÓN I

(Código: 0415102T)

Clase No: 9

Unidad V: Funciones en Lenguaje C

Profesor: Armando Carrero

UNIDAD No. V
FUNCIONES EN LENGUAJE C

TIPOS :
- .- Funciones estándar o **predefinidas**
- .- Funciones **creadas** por el programador.

Funciones Estándar o Predefinidas

Este tipo de funciones vienen incorporadas en el propio compilador del lenguaje, para dar soporte a las operaciones que se usan con mayor frecuencia, algunas ya se han utilizado en las unidades estudiadas anteriormente, por ejemplo las funciones: printf( ), scanf( ).

Estas funciones se encuentran almacenadas en bibliotecas y para usarlas se requiere incorporar en la cabecera del programa junto a la directiva #***include*** el nombre de la respectiva biblioteca, para posteriormente hacer uso directamente de la función mediante su nombre, su sintaxis y la llamada respectiva. Antes de utilizar una función, se deben conocer las características de dicha función, es decir, el número y tipo de datos de sus argumentos y el tipo de valor que devuelve.

Los grupos de funciones estándar más comunes, se encuentran agrupadas en un mismo archivo de cabecera, algunas de ellas se identifican a continuación:

APLICACIÓN	DIRECTIVA(archivo)
Operaciones de entrada/salida estándar	#***include<stdio.h>***
Operaciones matemáticas	#***include<math.h>***
Manipulación de caracteres	#***include<ctype.h>***
Operaciones de manipulación de memoria	#***include<alloc.h>***
Manejo de directorios	#***include<dir.h>***
Fecha y hora	#***include<time.h>***
Búsqueda y ordenación	#***include<stdlib.h>***
Manejo de cadenas	#***include<string.h>***

A continuación se estudian algunas de las funciones de mayor utilidad como: las funciones de entrada, de salida, numéricas, de manejo de cadenas de caracteres y de otras utilidades.

Funciones de Entrada - Salida.

Incluidas en la biblioteca **stdio.h** :

Descripción	Función	Ejemplo
Lee un carácter desde el dispositivo de entrada estándar y lo guarda en la variable indicada. El carácter se visualiza en pantalla y se requiere pulsar ENTER.	**getchar** ( )	**a=getchar( );**
Lee un carácter desde el dispositivo de entrada estándar y lo guarda en la variable indicada. El carácter se visualiza en pantalla y no se requiere pulsar ENTER.	**getche** ( )	**b=getche();**
Lee un carácter desde el dispositivo de entrada estándar y lo guarda en la variable indicada. El carácter no se visualiza en pantalla y no se requiere ENTER.	**getch** ( )	**c=getch ();**

Descripción	Función	Ejemplo
Escribe un carácter en el dispositivo de salida estándar.	**putchar**()	**h='R' ;** **putchar (h) ;** se muestra el caracter 'R' en pantalla **putchar ('S') ;** se muestra el caracter 'S' en pantalla.
Lee una cadena de caracteres desde el dispositivo de entrada estándar.	**gets**()	**gets(nombre) ;** Guarda la cadena ingresada desde el teclado, incluyendo los espacios en blanco, en la variable 'nombre' ,que debe ser de tipo cadena.
Muestra la cadena de caracteres en el dispositivo de salida estándar.	**puts**( )	**puts(nombre) ;** Muestra la cadena almacenada en la variable 'nombre'.
Permite leer datos desde el dispositivo de entrada estándar.	**scanf**("cc", &h) cc: representa la cadena de control que incluye el formato del tipo de dato a leer, que puede ser: %c : un carácter %d : entero decimal %x : hexadecimal %s : cadena %f : número real h: Variable donde se almacena el valor leído	**scanf("%c%d",&sex,&edad) ;** Guarda en las variables 'sex' y 'edad' los valores ingresados desde el teclado, separados por un espacio en blanco y luego pulsando ENTER, ó pulsando ENTER luego de cada valor ingresado desde el teclado. Ejemplo: F 25 (pulsar ENTER) ó F (pulsar ENTER) 25 (pulsar ENTER)

Descripción	Función	Ejemplo
Permite mostrar datos e información en el dispositivo de salida estándar.	**printf**("cc", h) cc: representa la cadena de control que puede incluir: texto o mensajes, secuencias de escape y/o el formato del tipo de dato a escribir, que puede ser: %c : un carácter %d : entero decimal %x : hexadecimal %s : cadena %f : número real h: puede ser una constante, un valor, una variable o una expresión.	**printf("Sexo = %c \n", sex) ;** **printf("Edad = %d", edad) ;** Muestra en pantalla: **Sexo = F** **Edad =25** Observación: **\n** es una secuencia de escape que indica mover el cursor a la línea siguiente después de mostrar el valor de la variable sex.

Funciones para el manejo de caracteres

Incluidas en la biblioteca **ctype.h:**

Descripción	Función	Ejemplo
Retorna un valor diferente de 0 (cero) si c es un carácter **alfanumérico**. Retorna 0 (cero) en otro caso.	**isalnum(c)**	**m = '%' ;** **h=isalnum (m);** se asigna el valor 0 a la variable h **m = 'k' ;** **h=isalnum (m);** se asigna un valor ≠0 a la variable h **m = '5' ;** **h=isalnum (m);** se asigna un valor ≠0 a la variable h
Retorna un valor diferente de 0 (cero) si c es un carácter **alfabético**. Retorna 0 (cero) en otro caso.	**isalpha(c)**	**w = 'P' ;** **h=isalpha (w);** se asigna un valor ≠ 0 a la variable h **w = '8' ;** **h=isalpha (w);** se asigna el valor 0 a la variable h
Retorna un valor diferente de 0 (cero) si c es un **digito decimal**. Retorna 0 (cero) en otro caso.	**isdigit(c)**	**w = '9' ;** **h=isdigit (w);** se asigna un valor ≠ 0 a la variable h **w = 'A' ;** **h=isdigit (w);** se asigna el valor 0 a la variable h
Retorna un valor diferente de 0 (cero) si c es un carácter del alfabeto en **minúscula**. Retorna 0 (cero) en otro caso.	**islower(c)**	**w = 'd' ;** **h=islower (w);** se asigna un valor ≠ 0 a la variable h **w = '(' ;** **h=islower (w);** se asigna el valor 0 a la variable h **w = 'L' ;** **h=islower (w);** se asigna el valor 0 a la variable h

Descripción	**Función**	**Ejemplo**
Retorna un valor diferente de 0 (cero) si c es un carácter del alfabeto en **mayúscula**. Retorna 0 (cero) en otro caso.	**isupper(c)**	**w = 'R' ;** **h=isupper (w);** se asigna un valor ≠ 0 a la variable h **w = 'a' ;** **h=isupper (w);** se asigna el valor 0 a la variable h **w = ':' ;** **h=isupper (w);** se asigna el valor 0 a la variable h
Convierte la letra almacenada en c a minúscula.	**tolower(c)**	**j = 'E' ;** **j =tolower (j);** se asigna el carácter 'e' a la variable j **t = 'a' ;** **t =tolower (t);** La variable t no se modifica, es decir, sigue almacenando el carácter 'a'
Convierte la letra almacenada en c a mayúscula.	**toupper(c)**	**p = 't' ;** **p =tolower (p);** se asigna el carácter 'T' a la variable p **x = 'B' ;** **x =tolower (x);** La variable x no se modifica, es decir, sigue almacenando el carácter 'B'

Funciones para el manejo de cadena de caracteres.

Incluidas en la biblioteca **string.h** :

Descripción	Función	Ejemplo
Retorna el número de caracteres de una cadena.	**strlen** (h)	Suponiendo que la variable 'nombre' contenga la cadena: " Luis Ramírez" **a=strlen(**nombre **);** **printf**("La cadena tiene % d caracteres", a **) ;** Se imprime : La cadena tiene 12 caracteres
Copia la cadena o el valor de la cadena S1, en la variable tipo cadena S2	**strcpy** (S2 , S1)	Suponiendo que la variable 'nombre' contenga la cadena: " Luis Ramírez" **strcpy** (ident , nombre) ; Se copia la cadena " Luis Ramírez" en la variable ident. **strcpy** (caso ,"Razón") ; Se copia la cadena "Razón" en la variable caso.
Compara lexicográficamente las cadenas p1 y p2	**strcmp** (p1 , p2) En caso de: a) p1 > p2, la función devuelve un valor entero >0. b) p1 < p2, la función devuelve un valor entero <0. c) p1 = p2, la función devuelve el valor entero cero (0).	Suponiendo que la variable 'valor' contenga la cadena:"material" y la variable 'codigo' la cadena "casa" : **p = strcmp** (valor, codigo) ; Se asigna a la variable p, un entero mayor a cero. **p = strcmp** ("marca" , valor) ; Se asigna a la variable p, un entero menor a cero. **p = strcmp** ("casa" , codigo) ; Se asigna a la variable p, el entero cero.
Compara lexicográficamente las cadenas p1 y p2, sin diferenciar mayúsculas de minúsculas.	**strcmpi** (p1 , p2)	**p = strcmpi** ("CASA" , codigo) ; Se asigna a la variable p el entero cero.

Funciones Numéricas

En la formación básica en el área de la ingeniería se requiere el uso de funciones matemáticas para el cálculo y la resolución de problemas, por ello se dará importancia al estudio de aquellas funciones más usuales como las funciones matemáticas de uso general, las trigonométricas, las logarítmicas, las exponenciales y las aleatorias. La mayoría de estas funciones se encuentran agrupadas en el archivo de cabecera: #***include<math.h>***

Constantes matemáticas de uso general, incluidas en la biblioteca **math.h :**

Descripción	Constante	Valor que representa
Valor de la constante e	**M_E**	2.71828…
Valor de la constante π	**M_PI**	3.14159….
Valor de la constante $\frac{\pi}{2}$	**M_PI_2**	1.57079….
Valor de la constante $\frac{\pi}{4}$	**M_PI_4**	0.78539….
Valor de la constante $\frac{1}{\pi}$	**M_1_PI**	0.31831…
Valor de la constante $\frac{2}{\pi}$	**M_2_PI**	0.63662…
Valor de la constante $\sqrt{2}$	**M_SQRT2**	1.41421…
Valor de la constante $\sqrt{\frac{1}{2}}$ ó $\frac{1}{\sqrt{2}}$ ó $\frac{\sqrt{2}}{2}$	**M_SQRT1_2**	0.70710…
Valor de la constante $Log_{10}(e)$	**M_LOG10E**	0.43429…
Valor de la constante $Ln_e(2)$	**M_LN2**	0.69314…
Valor de la constante $Ln_e(10)$	**M_LN10**	2.30258…
Valor de la constante $\frac{2}{\sqrt{\pi}}$	**M_2_SQRTPI**	1.12837…

Funciones matemáticas de uso general, incluidas en la biblioteca **math.h**:

Descripción	Función	Ejemplo
Valor absoluto del entero h	**abs** (h)	**a=abs(-8);** se asigna el valor 8 a la variable a
Valor absoluto del real h	**fabs** (h)	**b=fabs(-3.5);** se asigna el valor 3.5 a la variable b
Valor de e^x, es decir $(2.7172)^x$	**exp** (x)	**c=exp(4);** se asigna el valor $(2.7172)^4$= 54.598… a la variable c
Valor de la potencia a^b	**pow** (a , b)	**d** = **pow(2 , 3);** se asigna el valor 8 (2^3) a la variable d
Raíz cuadrada de un valor z	**sqrt** (z)	**m=sqrt(25);** se asigna el valor 5 a la variable m **n=sqrt(25.0);** se asigna el valor 5.0 a la variable n
Logaritmo decimal delvalor x	**log10** (x)	**p=log10(1000);** se asigna el valor 3.0 a la variable p
Logaritmo neperiano de un valor y	**log**(y)	**w=log(M_E);** se asigna el valor 1.0[$\ln_e(e)$]a la variablew
Seno del ángulo α (α debe expresarse en radianes)	**sin**(α)	**q=sin(M_PI_2);** se asigna el valor 1.0 a la variable q (90° ~$\frac{\pi}{2}$: M_PI_2)
Coseno del ángulo α (α debe expresarse en radianes)	**cos**(α)	**b=cos(M_PI / 3);** se asigna el valor: ½=0.5 a la variable b(60° ~ $\frac{\pi}{3}$: $\frac{\text{M\_PI}}{3}$)
Tangente del ángulo α (α debe expresarse en radianes)	**tan**(α)	**w=tan(M_PI_4);** se asigna el valor 1.0 a la variable w (45° ~ $\frac{\pi}{4}$: M_PI_4)
Retorna el ángulo β, cuyo seno es el valor q. El ángulo retornado viene expresado en radianes.	**asin**(q)	**z=asin(0.965925);** se asigna el valor 1.308994 a la variable z (β = 1.308994x$\frac{180°}{\pi}$ = 75°)

Descripción	Función	Ejemplo
Retorna el ángulo β, cuyocoseno es el valor q. El ángulo retornado viene expresado en radianes.	**acos**(q)	**z=acos(0.5);** se asigna el valor1.0471 a la variable z (β = 1.0471 x$\frac{180°}{\pi}$ = 60°)
Retorna el ángulo β, cuya tangente es el valor q. El ángulo retornado viene expresado en radianes.	**atan**(q)	**z=atan(1);** se asigna el valor 0.7853… a la variable z (β = 0.7853 x$\frac{180°}{\pi}$ = 45°)
Retorna el ángulo β, cuya tangente es el valor $\frac{a}{b}$. El ángulo retornado viene expresado en radianes.	**atan2**(a , b)	**z=atan2(**$\sqrt{3}$ **, 3);** se asigna el valor 0.5235… a la variable z (β = 0.5235 x$\frac{180°}{\pi}$ = 30°)
Redondea el valor de q, al entero menor más cercano a él.	**floor**(q)	**p=floor (** 6.82**);** se asigna 6 a la variable p **p=floor (** 6.03**);** se asigna 6 a la variable p
Redondea el valor de q, al entero mayor más cercano a él.	**ceil**(q)	**h=ceil (** 6.82**);** se asigna 7 a la variable h **h=ceil (** 6.03**);** se asigna 7 a la variable h

Funciones para generar números aleatorios.

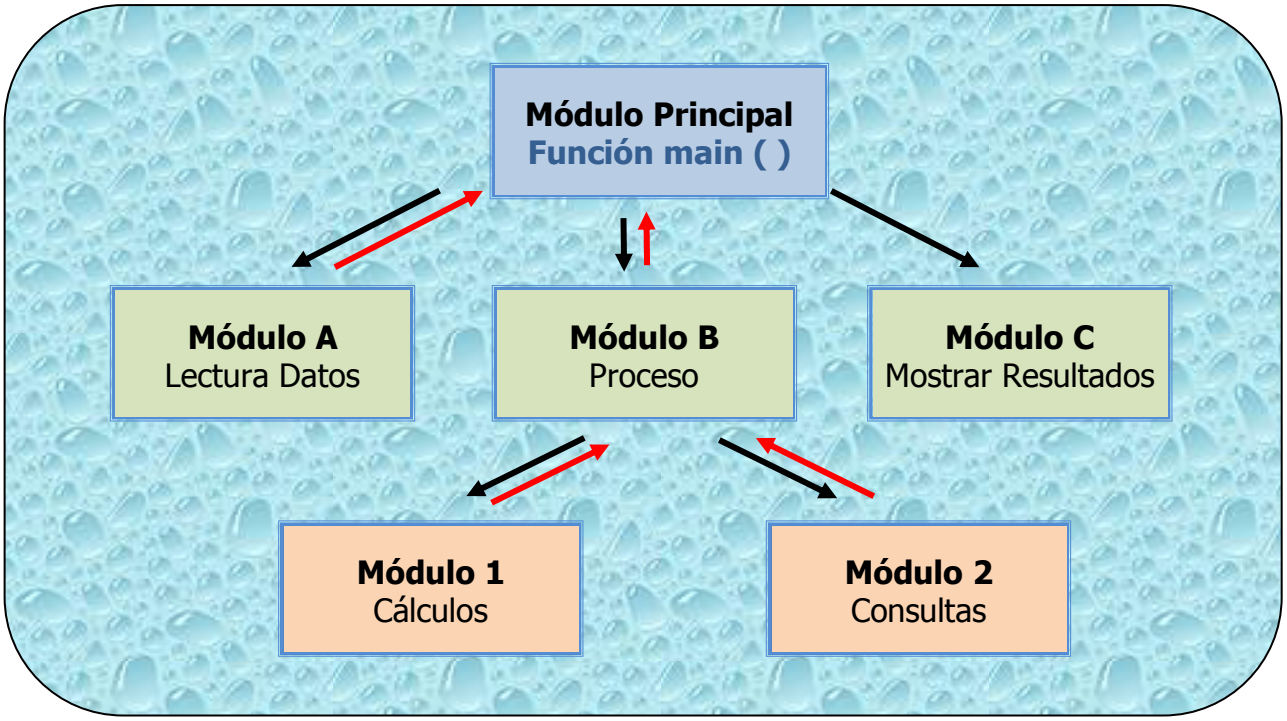
Incluidas en la biblioteca **stdlib.h** :

Descripción	Función	Ejemplo
Constante de valor 32767	**RAND_MAX**	
Genera un numero entero aleatorio, en el rango 0 a RAND_MAX.	**rand**( )	**z=rand ();** se asigna a la variable z, un valor entero aleatorio
Inicializa el generador de números aleatorios. Fija la semilla para que la función rand genere una secuencia diferente de números aleatorios.	**srand**( )	**srand (time(NULL));** **z = rand ( ) ;** time(): función que retorna el tiempo en segundos desde el 01/01/1970

Funciones creadas por el programador.

Un programa bien elaborado debe consistir de un módulo principal (programa principal), que sea el más alto nivel de mando, y (sub)módulos, que sean los niveles más bajos, donde se realicen las tareas ordenadas desde el nivel superior, a su vez los (sub) módulos pudieran asignar actividades a otros que se encuentren en otro nivel más inferior. A éste método de programación se le conoce como programación modular, que es una técnica que atiende al proverbio "divide y vencerás" y que se fundamenta en fraccionar un programa que puede resultar muy extenso o complejo, en varias partes, de modo que la tarea se pueda llevar acabo mediante la ejecución de programas más pequeños, que resulten más sencillos de diseñar.

El diagrama siguiente muestra a manera de ejemplo, una representación gráfica de una posible subdivisión de un programa en varias tareas.

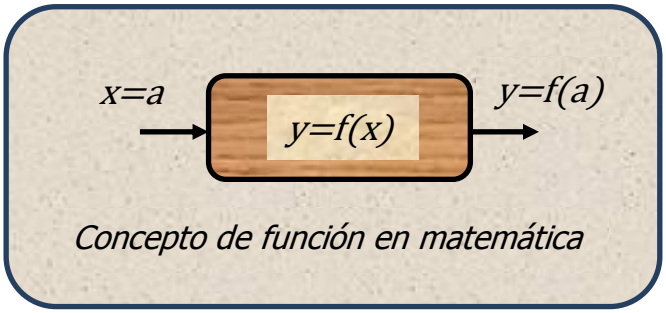


El modulo principal, representa la función principal "main", es por ello que todo programa debe contener obligatoriamente, al menos, una función, y esa es la función "main". La función main es ejecutada directamente desde el sistema operativo y jamás desde otra función. La importancia de la función main radica en que la ejecución del programa inicia con las instrucciones contenidas en su cuerpo. Una vez iniciada la ejecución del programa, desde la función main se pueden llamar a otras funciones, conocidas como funciones secundarias y, posiblemente, desde estas funciones a otras.

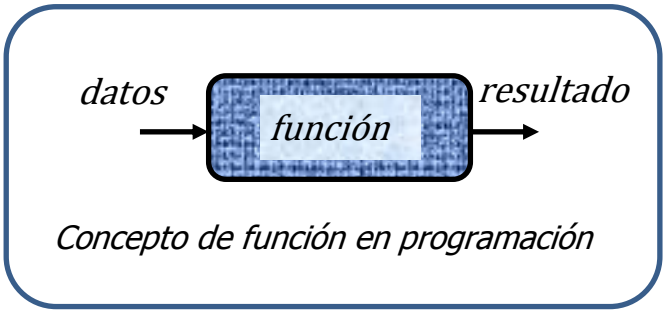
A los módulos(A, B, C, 1 y 2) se les conoce como subprogramas, en algunos lenguajes de programación también se les identifica como subrutinas o procedimientos; y en lenguaje C se les llama: ***funciones***, fundamentalmente funciones creadas o diseñadas por el programador, a las que se dedica el siguiente estudio.

Concepto de: Función creada por el programador.

Aunque no hay una similitud exacta con el concepto de función en matemáticas, existe cierta analogía que permite introducir el tema. En matemáticas la función $y = f(x)$, requiere de un valor de la variable independiente "x", para generar un valor de la variable dependiente "y"; es decir para un valor de entrada se genera un valor de salida, éste se obtiene al evaluar la función en un valor posible de "x". De igual forma se puede ilustrar el propósito de una función en programación, en donde, luego de realizar las actividades indicadas, con los datos suministrados, ésta devuelve un resultado. Ambos procedimientos se pueden graficar como a continuación:



Concepto de función en matemática



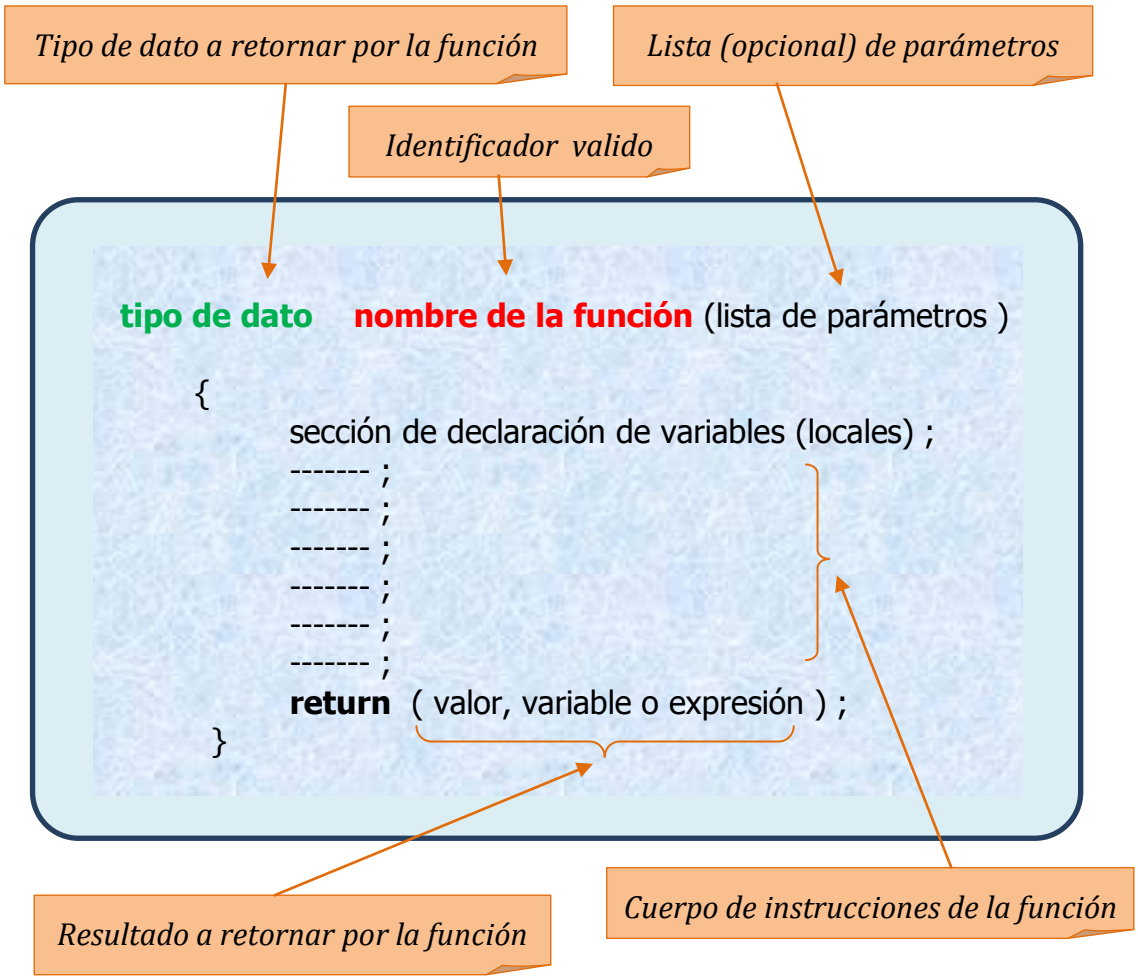
Concepto de función en programación

Concepto:

Una función es un conjunto de instrucciones, que conllevan a resolver una tarea específica, permitiendo dividir un problema en módulos de solución, llamados subprogramas, facilitando el trabajo y haciendo más simple el mantenimiento de un programa.

Definición de una Función

Formato general:



En donde debe entenderse lo siguiente:

tipo de dato : Se refiere al tipo de dato a retornar por la función, indicándose con la palabra reservada respectiva: int, float, char. Es posible que la función no retorne valor alguno, en este caso se usará la palabra reservada void. Si antes del nombre se deja en blanco éste lugar, se supone que la función retornará un valor de tipo int.

nombre de la función: Debe ser un identificador valido, es decir, que respete las reglas ya conocidas. Se sugiere, sea representativo de la tarea que resuelve la función.

lista de parámetros: Se trata de variables separadas por comas y encerradas entre paréntesis. Aunque los parámetros son opcionales, sirven para recibir los datos o valores de entrada con los que trabajara la función para realizar la tarea. En ocasiones pudieran servir para retornar algún

resultado. Cada parámetro debe estar precedido del respectivo tipo de dato, así: (tipo1 parámetro1, tipo2 parámetro2, ...).

El tipo de dato, el nombre de la función y la lista de parámetros determinan lo que se conoce como la cabecera dela función.

cuerpo de la función: Se compone de las instrucciones, encerradas entre llaves, que resuelven la tarea específica y que puede contener: la declaración de variables locales, funciones de entrada o salida, asignaciones, tomas de decisión, estructuras de repetición y la instrucción return, si se requiere.

variables locales Se trata de posibles variables a usar dentro del cuerpo de la función. Su uso se restringe a ella y queda anulada cuando el control del programa retorna hacia donde fue llamada la función.

return ( ) : Esta es la última instrucción a ser ejecutada por la función. Se usa para indicar, dentro de los paréntesis, el resultado a retornar, que puede ser el valor almacenado en una un variable o una expresión que al resolverse arroja el resultado a retornar. El resultado a retornar debe estar acorde al tipo de dato declarado en la cabecera de la función. En caso de ser una función tipo void, la instrucción return puede omitirse o sólo insertar: *return*. Una función puede contener varios return, pero como norma de programación se considera que la función finaliza al ejecutar la primera que se encuentre. En caso de estar ausente, la función finaliza al encontrar la llave de cierre "}".

Características de una función:

1.- Una función se define y se declara.
2.- A una función se le asigna un único nombre.
3.- Una función puede estar formada por una o varias sentencias.
4.- Una función se asocia a un tipo de dato.
5.- Una función se invoca (se llama, se ejecuta o se activa)
6.- Una función puede ser ejecutada más de una vez.
7.- Se pueden usar como factores de una expresión
8.- Una función se ejecuta cuando su nombre se coloca en el cuerpo de la función principal, o en el cuerpo de otra función.
9.- Una función puede recibir datos, desde donde sea llamada, y devolver uno o varios resultados. Con el return solo puede retornar un único valor.
10.- Pueden ejecutar tareas sin retornar ningún valor.
11.-Pueden llamarse desde distintas partes en un mismo programa o desde otra función

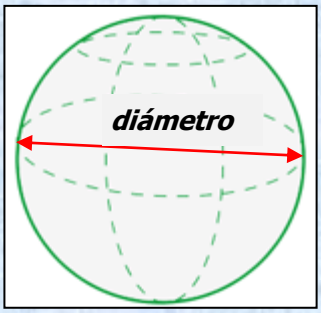
Declaración de una Función

Antes de hacer uso de una función se debe hacer su respectiva declaración, con el propósito de que sea reconocida cuando se ejecute, bien desde el programa principal (función main) o desde otra función secundaria. La declaración también se conoce con el nombre de prototipo de la función y consiste en ubicar en la cabecera del programa, la cabecera de la función, que consiste en la primera línea de la definición de una función, finalizando con (;), así el formato de una declaración sería:

tipo de dato nombre de la función (lista de parámetros) ;

Ejemplo: Definir una función que calcule el volumen de una esfera, en base al diámetro medido en centímetros (valor entero).

Solución: Una esfera es un cuerpo geométrico, cuya forma se muestra a continuación y su volumen se calcula con la formula dada:



volumen de una esfera

$$V = \frac{4}{3}\pi r^3$$

r : radio de la esfera

La función a definir debe recibir a través de un parámetro el diámetro de la esfera, luego en el cuerpo de la función se calcula el radio en base al diámetro (radio = diámetro ÷ 2), seguidamente se calcula y se almacena en una variable local el volumen, usando la formula suministrada, para luego retornar el valor calculado que de acuerdo a la formula, debe ser un valor real. A continuación la función denominada "volumen de esfera".

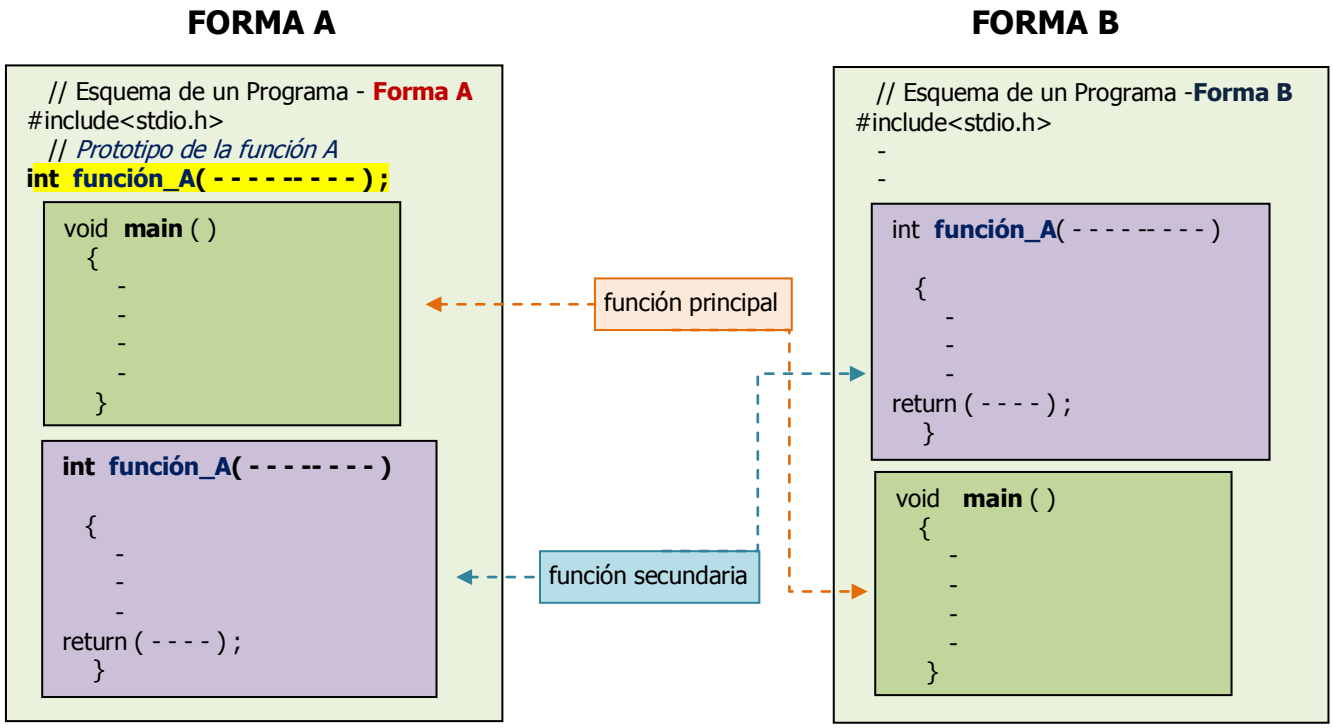
```
float   volumen_de_esfera  ( int   diametro )
    {
        //declaración de variables locales
        float    volumen, radio ;

        //cuerpo de la función
        radio = diametro / 2.0 ;
        volumen = 4/3.0 * M_PI * pow ( radio , 3 ) ;
        return  ( volumen ) ;
    }
```

En el cuadro anterior solo se presenta la definición de la función, que no actúa por si sola, es necesario ejecutarla (también *se usa el término: llamarla o invocarla*), de allí que es necesario ubicarla dentro del programa.

Ubicación de las funciones secundarias

Las funciones secundarias pueden ubicarse antes o después de la función principal (main). En caso de ubicarse después a la función principal, es necesario incluir en la cabecera del programa, su declaración o el correspondiente prototipo. A continuación se muestra una representación de las dos formas, considerando sólo un esquema general de un programa.



Aunque ambas formas son válidas, la forma A es la más utilizada debido a que todo programa se empieza a diseñar desde la función principal, y de lo requerido allí, se van anexando las funciones secundarias necesarias.

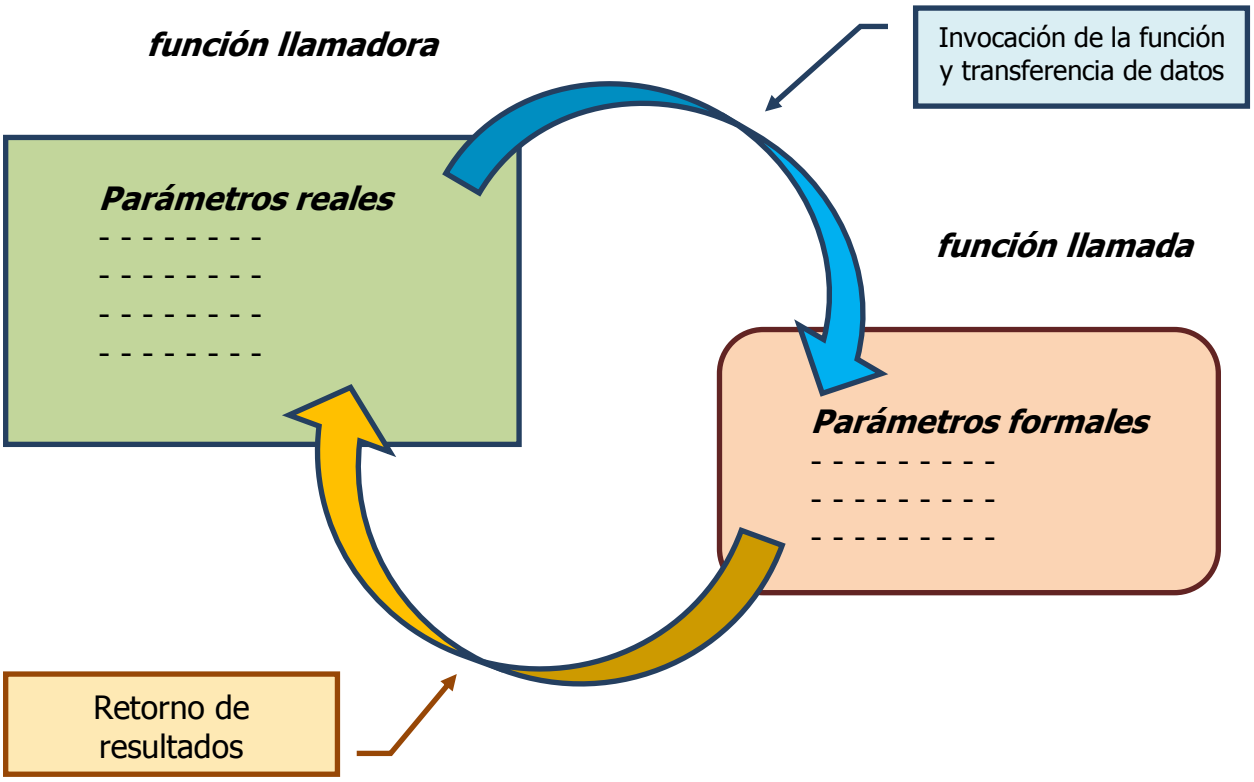
Dado que se usara la forma A, respecto a la ubicación de las funciones secundarias, se debe insistir en la necesidad de colocar antes de la función principal los prototipos de las funciones. El prototipo de una función, no es más que la cabecera de la función, para el ejemplo anterior el prototipo es:

float volumen_de_esfera (int diametro) ;

Modos o formas de ejecución de una función

Existen varias maneras de ordenar la ejecución (también denominada: *llamada* o *invocación*) de una función. Casi siempre la llamada a una función se hace desde la función principal (main), sin embargo es perfectamente valido que se haga desde otra función secundaria.

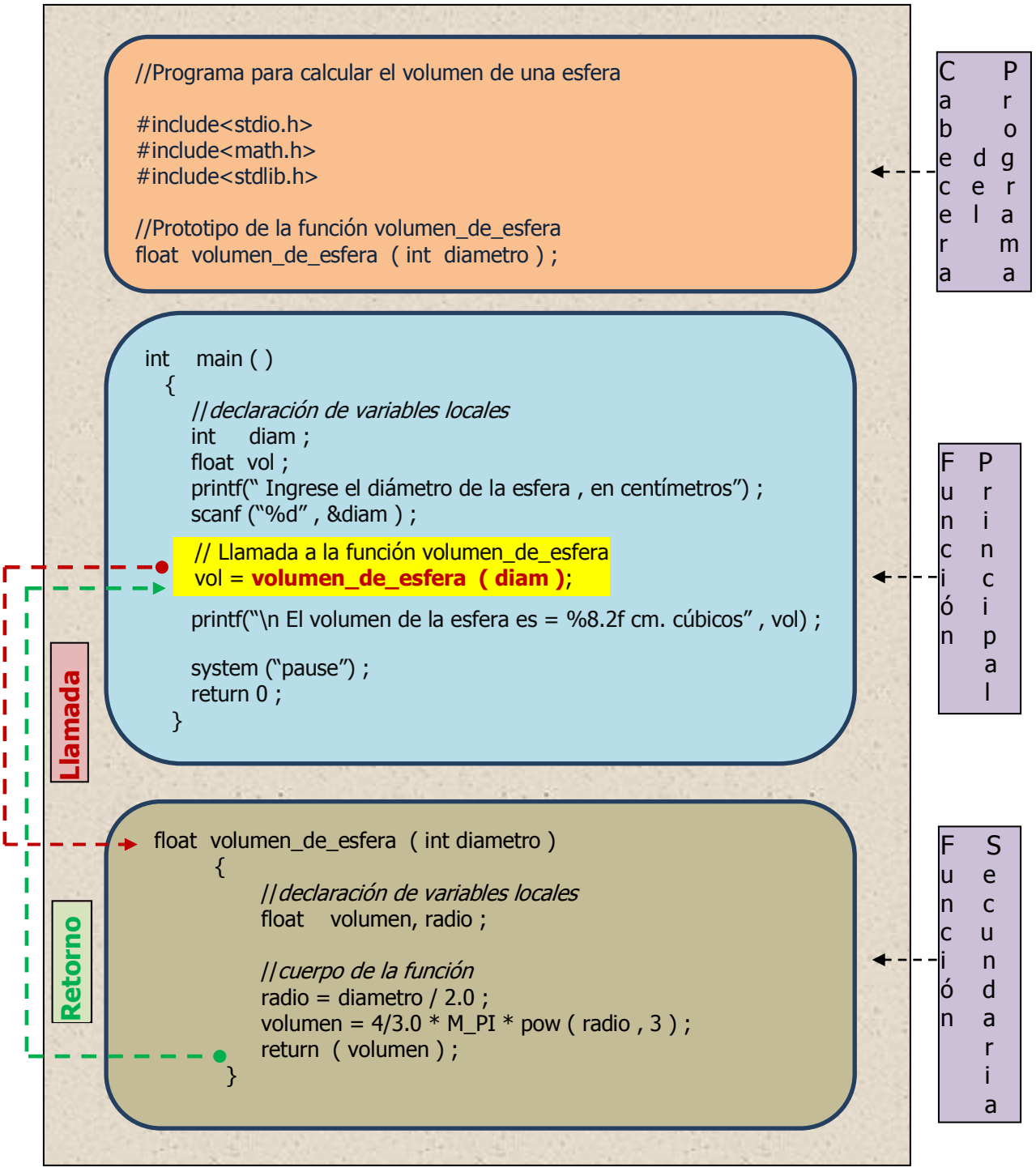
La función desde donde se efectúa la llamada recibe el nombre de "***función llamadora***" y la función invocada se denomina "***función llamada***". Al ejecutarse la ejecución, el control del programa lo toma la función llamada, y simultáneamente se efectúa el traspaso de los datos almacenados en la lista de parámetros, si la hay, que se identifican como "***parámetros reales***", hacia los parámetros ubicados en la cabecera de la función, identificados como "***parámetros formales"***. Una vez ejecutado el cuerpo de instrucciones de la función llamada y se ejecute la función return o se llegue a la llave final "}", el control del programa retorna a la instrucción desde donde fue invocada en la función llamadora, retornando el valor resultado o los resultados, si es el caso. Este proceso se ilustra en el siguiente gráfico:



A continuación se explican las distintas formas o maneras de llamar a una función, usando como ejemplo la función "*volumen_de_esfera*", definida anteriormente.

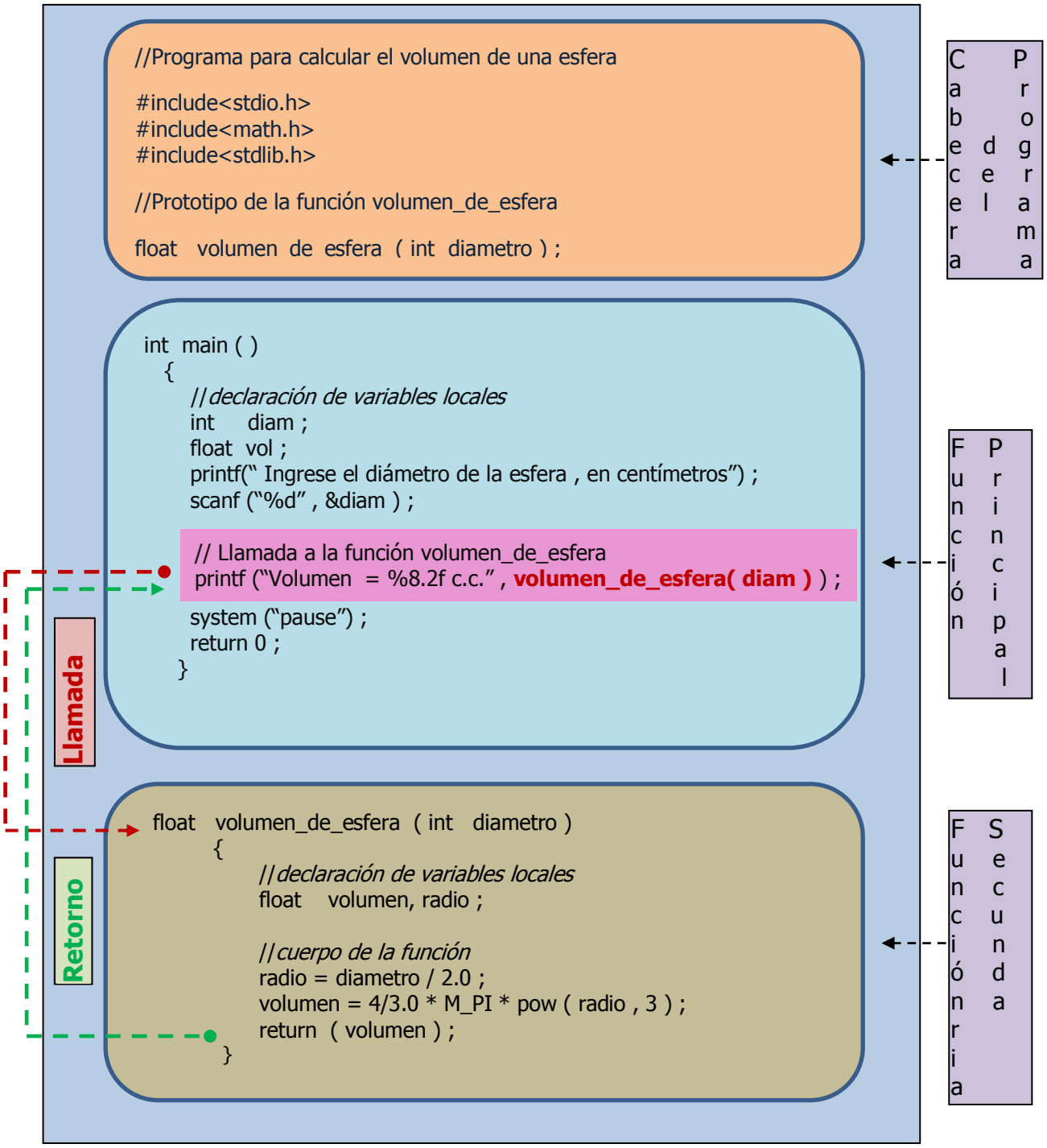
1.- Llamada desde una instrucción de asignación: Se trata de llamar a la función desde una asignación con el propósito de guardar en una variable, el valor retornado por la función, siguiendo el siguiente formato:

variable= nombre de la función (lista parámetros) ;



2.- Llamada desde una instrucción de salida: En este caso la llamada se ejecuta desde una instrucción que muestra directamente el valor retornado por la función; esa instrucción en lenguaje C, requiere del uso de la función *printf*, en donde se llama a la función y a su vez se muestra el valor retornado. A diferencia de la forma anterior, el valor retornado se muestra pero no queda almacenado en una variable. La invocación, se indica en el siguiente formato:

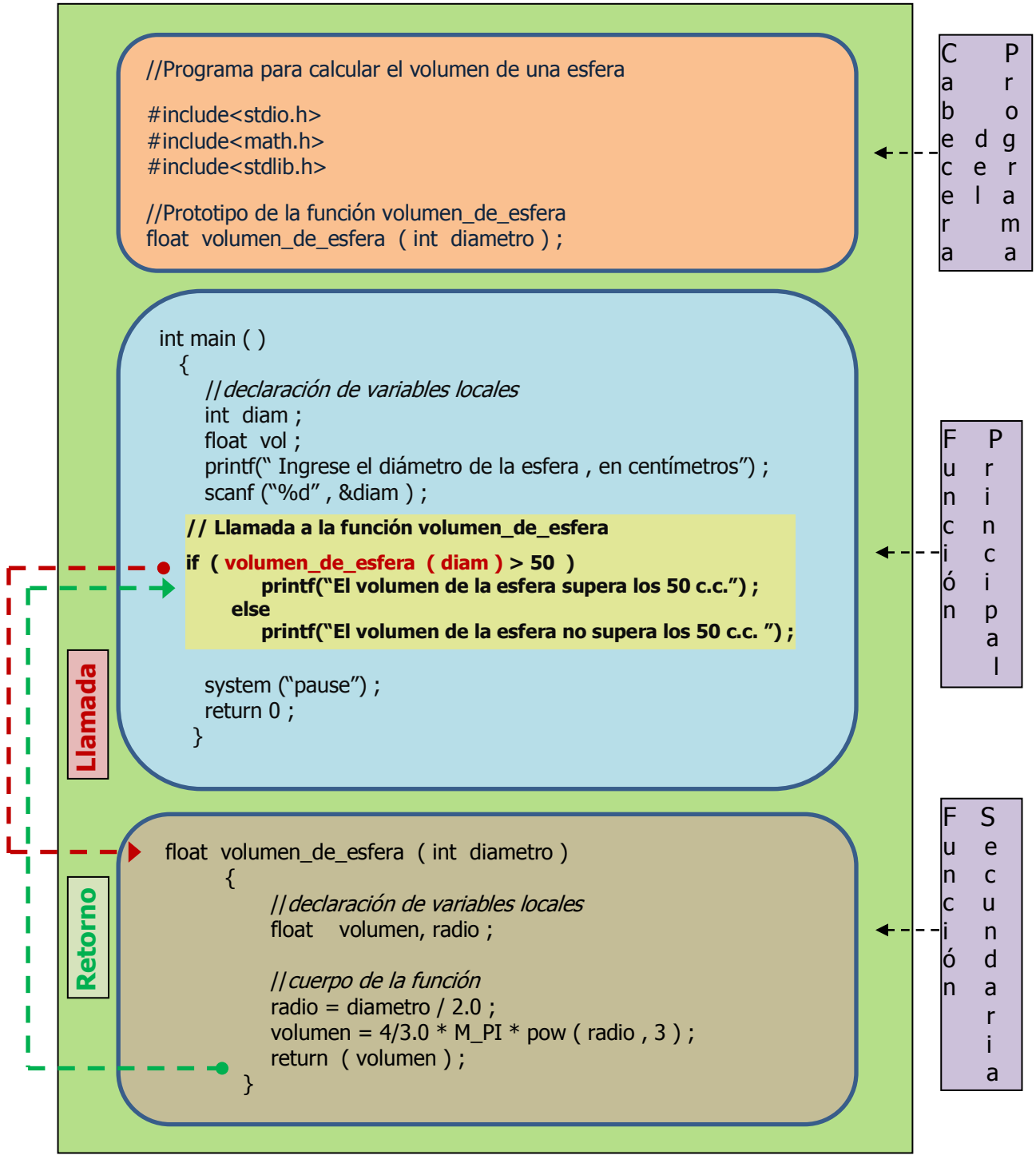
printf (" cadena de control ", nombre de la función (lista parámetros)) ;



3.- Llamada desde una instrucción de proceso: Es posible ejecutar una función desde una parte en donde el valor retornado sea utilizado para ejecutar otros procesos o tomardecisiones que conduzcan a otros procesos. Para ejemplificar, a continuación se indica cómo se puede invocar una función desde una toma de decisión:

```
if ( nombre de la función ( lista parámetros ) > valor )
        ---------------;
   else
        ---------------;
```

Como ejemplo, consideremos que en el programa anterior, se debe verificar si el volumen de la esfera supera o no, 50 centímetros cúbicos:



4.- Llamada desde un cálculo o término de una expresión: Una función puede ser invocada también ubicándola como un término de una expresión, bien aritmética, lógica o relacional, dependiendo del valor que ésta retorne. Un formato que puede servir de ejemplo se da a continuación:

variable = valor1 + nombre de la función (lista parámetros) – valor2 ;

Como ejemplo, modifiquemos el programa anterior para calcular el 75% del volumen de la esfera, siendo el volumen el resultado que retorne la función *volumen_de_esfera*.