FUNCIONES Y PASO DE PARAMETROS (Resumen)

Las palabras parámetro y argumento, aunque de significado similar, tiene distintas connotaciones semánticas: Se denominan parámetros los tipos declarados en el prototipo de la función y se le llama argumentos a los "valores" que se pasan de una función a otra. Aunque por lo general siempre les llamamos igual ☺

Una vez que entendimos que es un parámetro o argumento (le puedes decir como quieras) vamos a enfocarnos a ver como se manejan las funciones y el paso de valores (parámetros) en el lenguaje ANSI C.

Funciones

Una función es una pequeña parte de código que se encarga de realizar una sola tarea, de tal forma que nuestro programa puede estar formado por cualquier número de funciones, donde cada una hace algo distinto, y todas en conjunto se encargan de resolver un problema.

Por ejemplo, el cuerpo humano tiene distintas partes, cada una se encarga de una actividad diferente (ojos, corazón, boca, oídos, etc.) pero si juntamos cada parte, logramos que una persona pueda vivir. Así pasa con un programa: ☺ cada función hace algo y todas logran un resultado.

Declaración de una Función

Para declarar una función debemos seguir la siguiente sintaxis:

```
tipo_devuelto nombre_función (parámetros_cada_uno_con_su_tipo_de_dato){
  //aquí vamos a escribir las instrucciones que debe realizar la función
  return expresión;
}
```

tipo_devuelto

Si el *tipo_devuelto* es *void*, se considera que la función no devuelve ningún valor, por lo tanto, no debe escribirse la palabra **return.**

Si el *tipo_devuelto* es cualquiera de los siguientes:

char	short	int	long	unsigned	double	float
struct	union	enum	auto	extern	static	register

Entonces se debe escribir la palabra return junto con el valor que se desea retornar.
! Recuerda: si vas a "devolver" un valor, éste debe ser del mismo tipo de dato que la función va a devolver (*tipo_devuelto*).

nombre_función

El nombre de la función debe ser de acuerdo a lo que esa parte de código va a realizar, así que es importante que no inventes nombres y que escribas el nombre lo más parecido posible a lo que esa función hace. El nombre siempre debe comenzar con una letra minúscula y si está formado por dos o más palabras, la segunda palabra y debe comenzar con una letra mayúscula. Por ejemplo:

```
captura              //nombre sencillo y comienza con minúscula
mostrarDatos         //nombre compuesto y comienza con minúscula y luego mayúscula
calcularSueldoNeto //nombre compuesto y comienza con minúscula y luego mayúscula
```

(parámetros_cada_uno_con_su_tipo_de_dato)
Los parámetros son los nombres de las variables que la función va a recibir para trabajar con ellos. Más adelante veremos que los parámetros se pueden recibir por valor o referencia. **! Recuerda:** cada parámetro debe ir junto con su tipo de dato.

Llamada de una Función

Ahora veamos cómo hacer que las funciones se ejecuten. Para esto deben ser "llamadas". La llamada a una función se puede hacer desde cualquier parte del programa, por lo general la función main es quien llama a las demás funciones, aunque por lo general se acostumbra hacer una función con un menú, y dependiendo de la opción seleccionada se hace la llamada a la función que corresponda. Existen distintas formas de llamar a una función y estas dependen del tipo de función:

TIPOS DE FUNCIONES Y FORMA DE LLAMARLAS	
NO recibe parámetros y NO retorna valores	***NO recibe parámetros y SI retorna valores***

`void nombre_funcion( ){`
`  // código de la función`
`}`
	`tipo_devuelto nombre_funcion( ){`
	`  // código de la función`
	`  return valor;`
	`}`

Ejemplo:
 	
`void captura( ){`
`  int id;`
`  printf("Dame tu id");`
`  scanf(%d",&id);`
`}`
	Ejemplo:
	
	`int captura( ){`
	`  int id;`
	`  printf("Dame tu id");`
	`  scanf(%d",&id);`
	`  return id;`
	`  // id es del mismo tipo que devuelve la función`
	`}`

Para llamar a este tipo de función:
 	
`void main( ){`
`  captura( );`
`}`
	Para llamar a este tipo de función:
	
	`void main( ){`
	`  int res;`
	`  res = captura( );`
	`}`

<table>
<tr><th colspan="2">TIPOS DE FUNCIONES</th></tr>
<tr><th>SI recibe parámetros y NO retorna valores</th><th>SI recibe parámetros y SI retorna valores</th></tr>
<tr><td>

```
void nombre_funcion(tipo nomvar, tipo nomvar){
    // código de la función
}
```

</td><td>

```
tipo_devuelto nombre_funcion(tipo nomvar, tipo nomvar){
    // código de la función
    return valor;
}
```

</td></tr>
<tr><td>Ejemplo:

```
void mostrarDatos(int id, float calif){
   printf("El alumno %d tiene la calif %f", id, calif);
}
```

</td><td>Ejemplo:

```
float calcularPromedio(float calif1, float calif2){
  float prom;
  prom =  (calif1 + calif2) / 2;
  return prom;
  //prom es del mismo tipo que devuelve la función
}
```

</td></tr>
<tr><td>Para llamar a este tipo de función:

```
void main( ){
    int id =15;
   float calif = 8.5;
    mostrarDatos(id, calif);
}
```

</td><td>Para llamar a este tipo de función:

```
void main( ){
    float calif1 = 8.5, calif2 = 9.4, res;
    res = calcularPromedio(calif1, calif2);
}
```

</td></tr>
</table>

Paso de parámetros

Los parámetros se pasan normalmente por *valor*, pero también se pueden pasar por *referencia*. El paso de parámetros por referencia admite dos tipos de sintaxis: escribiendo el operador & antes del nombre del parámetro y otro caso es escribiendo el operador * (asterisco) antes del nombre del parámetro (esto lo analizaremos cuando veamos apuntadores).

Paso de parámetros es por valor:

```
int funcion1 (int x, int y)
```

Esto quiere decir que la función1 recibirá únicamente el valor de los dos parámetros, x e y. Podrá utilizar esos valores dentro de su código, e incluso podrá cambiarlos. Pero cualquier cambio en x e y no afectará a los parámetros actuales, es decir, a los parámetros del programa que llamó a función1.

Llamada a la función:

```
resultado = funcion1 ( x, y);
```

Paso de parámetros por referencia con el operador &

```
int funcion2 (int x, int *y)
```

En esta función, el parámetro "x" se pasa por valor y el parámetro "y" se pasa por referencia. Por lo que esta función podrá hacer cambios en la variable "x" sin afectar al valor original, mientras que los cambios que haga a la variable "y" se verán reflejados en todo el programa.

Llamada a la función:

```
resultado = funcion2 ( x, &y);
```

Veamos un resumen de cómo pasar datos por valor y por referencia, además de cómo deben ser recibidos por la función:

PASO DE PARAMETRO	***FORMA DE RECIBIRLO***
Por valor: mostrar(id, prom, gpo);	void mostrar(int id, float prom, char gpo);
Por referencia: mostrar(&id, &prom, &gpo);	void mostrar(int *id, float *prom, char *gpo);