

Introducción a C++

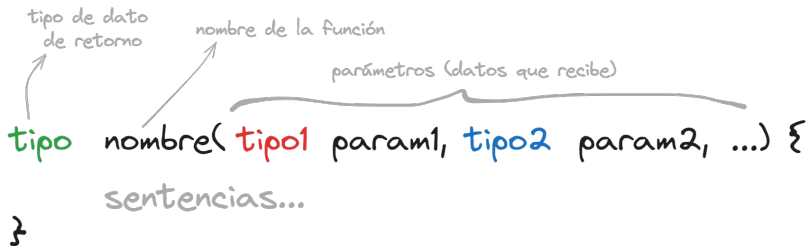
Funciones

Tomás Peiretti

Estructura de una función

Una función es un conjunto de sentencias que puede ser llamado/utilizado desde cualquier punto en nuestro programa.

Las funciones permiten modularizar y estructurar los programas en diferentes segmentos de código, favoreciendo la abstracción, legibilidad y reusabilidad.



The diagram illustrates the structure of a function signature with handwritten annotations in grey:

- tipo de dato de retorno**: An arrow points from this text to the word **tipo** (highlighted in green).
- nombre de la función**: An arrow points from this text to the word **nombre**.
- parámetros (datos que recibe)**: A bracket spans over the parameters **tipo1 param1, tipo2 param2, ...**.

The function signature is written as follows:

```
tipo nombre( tipo1 param1, tipo2 param2, ...) {  
    sentencias...  
}
```

Estructura de una función: ejemplos

```
1 int sumar(int a, int b, int c) {
2     return a + b + c;
3 }
4
5 int alCuadrado(int x) {
6     return x * x;
7 }
8
9 double distanciaEntrePuntos(double x1, double y1, double x2, double y2) {
10     double restaX = x2 - x1;
11     double restaY = y2 - y1;
12     return sqrt(restaX*restaX + restaY*restaY);
13 }
14
15 void imprimirDigitos(int num) {
16     while(num > 0) {
17         cout << num % 10 << endl;
18         num /= 10;
19     }
20 }
21
22 int main() {
23     int x = 1995;
24     cout << sumar(10, x, 5) << endl; // imprime 2010
25
26     double distancia = distanciaEntrePuntos(1, 1, 2.5, 2.5);
27     cout << distancia << endl; // imprime la distancia entre (1,1) y (2.5,2.5)
28
29     imprimirDigitos(x); // imprime 5 9 9 1
30 }
```

Pasaje de parámetros: por copia

Al momento de invocar una función, se crea una copia de los parámetros que se brindan. Esto significa que dentro de la función estaremos trabajando con nuevas variables.

```
1 void imprimirTriple(int x) {  
2     x = 3 * x;  
3     cout << "3*x = " << x << endl;  
4 }  
5  
6 int main() {  
7     int miVar = 10;  
8     cout << "valor de miVar: " << miVar << endl;  
9     imprimirTriple(miVar);  
10    cout << "valor de miVar: " << miVar << endl;  
11    // salida:  
12    // valor de miVar: 10  
13    // 3*x = 30  
14    // valor de miVar: 10  
15 }
```

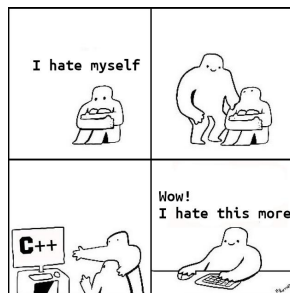
En el ejemplo **miVar** no se modifica, ya que al invocar la función se copia su contenido en la variable **x** de la función.

Pasaje de parámetros: por referencia

Si queremos que el contenido de **miVar** se modifique, debemos definir en la función que **el parámetro x pasa por referencia**.

Para indicar que un parámetro pasa por referencia, debemos agregar el símbolo **&** antes del nombre del parámetro

```
1 void imprimirTriple(int &x) {  
2     x = 3 * x;  
3     cout << "3*x = " << x << endl;  
4 }  
5  
6 int main() {  
7     int miVar = 10;  
8     cout << "valor de miVar: " << miVar << endl;  
9     imprimirTriple(miVar);  
10    cout << "valor de miVar: " << miVar << endl;  
11    // salida:  
12    // valor de miVar: 10  
13    // 3*x = 30  
14    // valor de miVar: 30  
15 }
```



Pasaje de parámetros: ¿Qué se imprime?

```
1 int funcion1(int x, int & y) {  
2     x = y - 10;  
3     return y;  
4 }  
5  
6 int funcion2(int & param) {  
7     int y = 10;  
8     y = funcion1(param, y);  
9     return y + param++;  
10 }  
11  
12 int main () {  
13     int x = 100;  
14     cout << funcion2(x) << endl;  
15     cout << x << endl;  
16 }
```

Funciones recursivas

Las funciones recursivas son aquellas que **se invocan a sí mismas**. Tienen:

- Un conjunto de **casos base**: aquellos que se utilizan para terminar con la recursión.
- Un conjunto de **casos recursivos**: aquellos en donde se invoca a la propia función.

```
1 void forRecursivo(int i, int fin) {  
2     if (i == fin)  
3         return;  
4  
5     cout << i << endl;  
6     forRecursivo(i+1, fin);  
7 }  
8  
9 int main() {  
10     // imprime todos los numeros  
11     // entre 0 y 10  
12     forRecursivo(0, 10);  
13 }
```



Ejercicios de funciones

Cualquier ejercicio, todos se pueden plantear con funciones