

Funciones definidas por el usuario en C++

¿Cómo enviar valores a una función? - Parámetros y argumentos

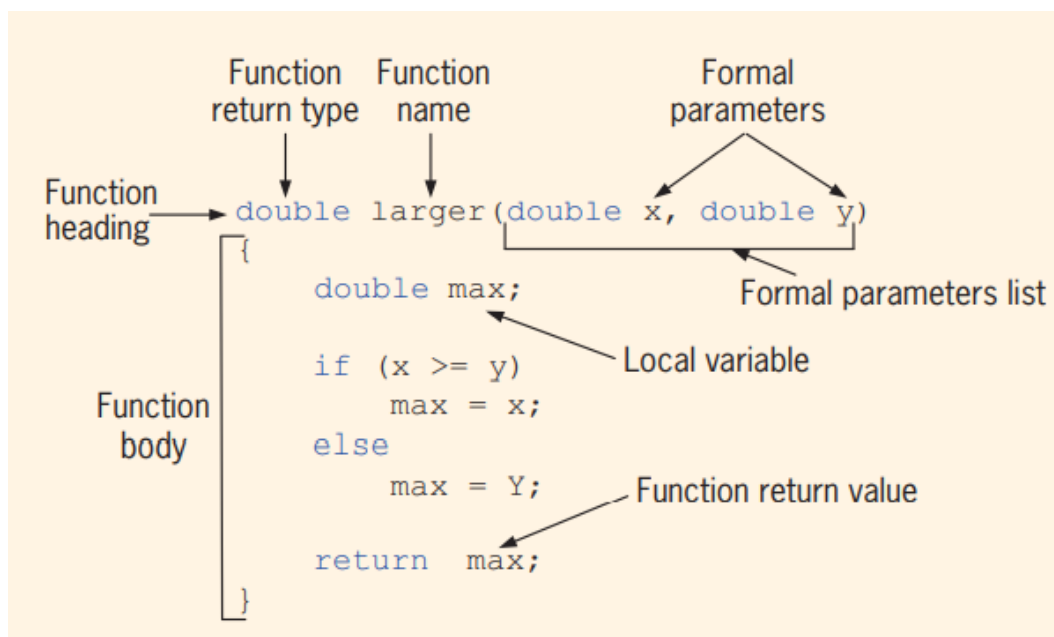
La información se puede pasar a las funciones como **parámetro**. Los parámetros actúan como variables dentro de la función.

Los parámetros se especifican después del nombre de la función, entre paréntesis. Puede agregar tantos parámetros como desee, solo sepárelos con una coma.

Cuando se pasa un parámetro a la función, se llama **argumento**.

Sintaxis básica del *formal parameter list*

```
dataType identifier, dataType identifier, ...
```



Valor de parámetro predeterminado

También puede utilizar un valor de parámetro predeterminado, utilizando el signo igual (=).

```
1 void myFunction(string country = "Norway") {
2     cout << country << "\n";
3 }
4
5 int main() {
6     myFunction("Sweden");
7     myFunction("India");
8     myFunction();
9     myFunction("USA");
10    return 0;
11 }
12
13 // Sweden
14 // India
15 // Norway
16 // USA
```

Parámetros múltiples y simples

Dentro de la función, puede agregar tantos parámetros como desee.

Tenga en cuenta que cuando trabaja con varios parámetros, la llamada a la función debe tener el mismo número de argumentos que parámetros y los argumentos deben pasarse en el mismo orden.

```
1 void myFunction(string fname, int age) {
2     cout << fname << " Refsnes. " << age << " years old. \n";
3 }
4
5 int main() {
6     myFunction("Liam", 3);
7     myFunction("Jenny", 14);
8     myFunction("Anja", 30);
9     return 0;
10 }
11
12 // Liam Refsnes. 3 years old.
13 // Jenny Refsnes. 14 years old.
14 // Anja Refsnes. 30 years old.
```

Pasar por referencia

En los ejemplos de la página anterior, usamos variables normales cuando pasamos parámetros a una función. También puede pasar una referencia a la función.

Los parámetros de referencia son útiles en tres situaciones:

- Cuando es necesario cambiar el valor del parámetro real.
- Cuando desee devolver más de un valor de una función (recuperar que la declaración de devolución puede devolver solo un valor).
- Al pasar la dirección, se ahorraría espacio de memoria y tiempo en relación con copiando una gran cantidad de datos

```
1 void swapNums(int &x, int &y) {
2     int z = x;
3     x = y;
4     y = z;
5 }
6
7 int main() {
8     int firstNum = 10;
9     int secondNum = 20;
10
11     cout << "Before swap: " << "\n";
12     cout << firstNum << secondNum << "\n";
13
14     // Call the function, which will change the values of firstNum and secondNum
15     swapNums(firstNum, secondNum);
16
17     cout << "After swap: " << "\n";
18     cout << firstNum << secondNum << "\n";
19
20     return 0;
21 }
```

Diferencia entre los parámetros por valor y por referencia

Las reglas que se han de seguir cuando se utilizan parámetros valor y referencia son las siguientes:

- Los parámetros valor (declarados sin &) reciben copias de los valores de los argumentos que se les pasan;
- La asignación a parámetros valor de una función nunca cambian el valor del argumento original pasado a los parámetros;
- Los parámetros referencia (declarados con &) reciben la dirección de los argumentos pasados;
- En una función, las asignaciones a parámetros referencia cambian los valores de los argumentos originales.

Sobrecarga de funciones (Function Overloading) - Polimorfismo

En un programa de C ++, varias funciones pueden tener el mismo nombre. Esto se llama función sobrecargada o sobrecargar el nombre de una función. Antes de establecer las reglas para sobrecargar una función, definamos lo siguiente:

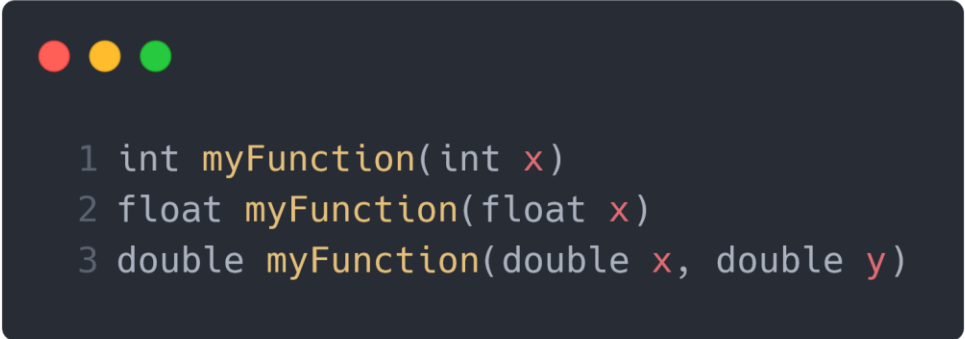
Se dice que dos funciones tienen listas de parámetros formales diferentes si ambas funciones tienen:

- Un número diferente de parámetros formales o
- Si el número de parámetros formales es el mismo, entonces el tipo de datos de los parámetros, en el orden en que los enumere, deben diferir en al menos una posición.

Sobrecarga de funciones: Creación de varias funciones con el mismo nombre.

Polimorfismo: "muchas formas" y ocurre cuando tenemos muchas clases que están relacionadas entre sí por herencia.

Nota: Varias funciones pueden tener el mismo nombre siempre que el número y / o tipo de parámetros sean diferentes.



```
1 int myFunction(int x)
2 float myFunction(float x)
3 double myFunction(double x, double y)
```