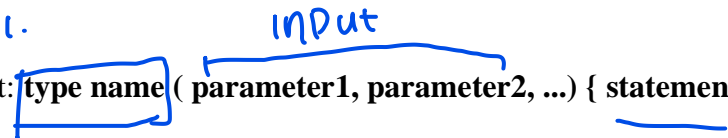


C++ FUNCTION

A function is a group of statements that is executed when it is called from some point of the program.

1. 
The following is its format: **type name** (parameter1, parameter2, ...) { statements }

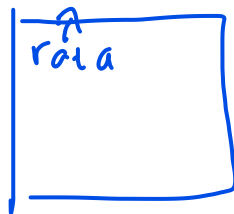
- type is the data type specifier of the data returned by the function.
- name is the identifier by which it will be possible to call the function.
- parameters (as many as needed): Each parameter consists of a data type specifier followed by an identifier, like any regular variable declaration (for example: int x) and which acts within the function as a regular local variable. They allow to pass arguments to the function when it is called. The different parameters are separated by commas.
- statements is the function's body. It is a block of statements surrounded by braces { }.

Example :

```
// function example
#include <iostream>
using namespace std;
int addition (int a, int b)
```

```
{
    int r;
    r=a+b;
    return (r);
}
```

```
int main ()
{
    int z;
    z = addition (5,3);
    cout << "The result is " << z;
    return 0;
}
```



int main

{ rata(a)
stdev(a).
median

```
int addition (int a, int b)
```

```
z = addition ( 5 , 3 );
```

Functions With No Type. The Use Of Void.

type name (argument1, argument2 ...) statement

you will see that the declaration begins with a type, that is the type of the function itself (i.e., the type of the datum that will be returned by the function with the return statement). But what if we want to return no value?

function \Rightarrow return.

Imagine that we want to make a function just to show a message on the screen. We do not need it to return any value. In this case we should use the void type specifier for the function. This is a special specifier that indicates absence of type.

// void function example

#include <iostream>

using namespace std;

void printmessage ()

{

cout << "I'm a function!";

}

int main ()

{

printmessage ();

return 0;

}

name (a).

Void \rightarrow cout.

function \rightarrow return.

int subtraction

void subtraction ()

void name (a).

cout << "hello " << a ;

Overloaded Functions.

In C++ two different functions can have the same name if their parameter types or number are different. That means that you can give the same name to more than one function if they have either a different number of parameters or different types in their parameters. For example:

```
// overloaded function
#include <iostream>
using namespace std;
① int operate (int a, int b) →
{
    return (a*b);
}
② float operate (float a, float b) → float.
{
    return (a/b);
}
int main ()
{
    int x=5,y=2;
    float n=5.0,m=2.0; ✓
    cout << operate (x,y);
    cout << "\n"; → int ✓
    cout << operate (n,m);
    cout << "\n"; → float ✓
    return 0;
}
```

berulang

Recursive Function

- ✓ Recursivity is the property that functions have to be called by themselves. It is useful for many tasks, like sorting or calculate the factorial of numbers. For example, to obtain the factorial of a number (n!) the mathematical formula would be:

$$n! = n * (n-1) * (n-2) * (n-3) \dots * 1$$

```
// factorial calculator
#include <iostream>
using namespace std;
long factorial (long a)
{
    if (a > 1)
        return (a * factorial (a-1));
    else
        return (1);
}

int main ()
{
    long number;
    cout << "Please type a number: ";
    cin >> number;
    cout << number << "! = " << factorial (number);
    return 0;
}
```

factorial(4)
4 x factorial(3)
3 x factorial(2)
2 x factorial(1) → 1

long → float

b = 3

b = 2

b = 1

$$n! = n \times (n-1)! \times (n-2)! \dots$$

$$5! = 5 \times 4.$$

$$5^3 = 5 \times 5 \times 5.$$

$$a = 5 \quad b = 3$$

long pangkat (int a, int b) {

if (b > 1)

return (a * pangkat(a, b));

else

return (1);

}