



# Scope of Variable

Let's classify variables according to their accessibility in the program.

We'll cover the following



- Introduction
- Local variable
  - Illustration
  - Example program
- Explanation
- Global variable
  - Illustration
  - Example program
  - Explanation

## Introduction#

The scope of a variable defines which part of the program that particular variable is accessible in. In C++, the variable can be either of these two:

- Local variable
- Global variable

## Local variable#

Suppose you are staying at a hotel. The hotel manager gives you a key to room No. 5. You can only access room No. 5 in the hotel.

ROOM NO. 5. You can only access ROOM NO. 5 in the hotel.



The local variable is just like a hotel room-specific key. It is only accessible within the block in which it is declared.



Block is a section of code enclosed inside the curly braces.

The **local variable** can only be accessed within the block in which it is declared.

A block can be a function, loop, or conditional statement. These variables are created when the compiler executes that particular block and destroyed when the compiler exits that block.

## Illustration#

See the illustration given below!



// Function func1

```
void func1 () {  
    int local1;
```

```
    for loop {  
        int local3;  
    }  
}
```

local3 is only accessible  
inside the for loop

local1 is only accessible  
inside the func1

// Function func2

```
int func2 () {  
    int local2;  
}
```

local2 is only accessible  
inside the func2

In the figure above, `local1`, `local2`, and `local3` are the local variables. They are only accessible within the block in which they are declared. We cannot access them outside the block in which they are declared. For example, `local3` is not accessible in `func2()`.

## Example program#

**Run** the program and see what happens!

The program will not compile.

```
1 #include <iostream>  
2 using namespace std;  
-
```



```
3
4 void function () {
5     int function_local = 10;
6     cout << main_local;
7 }
8
9 int main() {
10     int main_local = 20;
11     cout << function_local;
12     return 0;
13 }
```



Output

0.96s

```
main.cpp: In function 'void function()':
main.cpp:6:11: error: 'main_local' was not declared in this scope
    cout << main_local;
           ^
main.cpp: In function 'int main()':
main.cpp:11:11: error: 'function_local' was not declared in this scope
    cout << function_local;
           ^
```

## Explanation#

- In the code above, variable `function_local` is only accessible within the body of the `function()`. We cannot access it in the `main()`.
- Similarly, `main_local` is only accessible within the body of the `main()`. We cannot access it in the `function()`.
- The program is generating an error because we are trying to access the variable `main_local` in the body of the `function()` and `function local` in the body of the `main()`.



# Global variable#

Again, consider the example of a **hotel**. The hotel manager has the master key. Unlike us, the hotel manager can access each and every room in the hotel.

Similar to the master key, global variables are accessible in the whole program.

***Global variables** can be accessed from the point they are declared to the end of the program. They are declared at the very start of the program before defining any function.*

## Illustration#

See the illustration given below!

```
int global;
```

```
// Function func1
```

```
void func1 () {  
  int local1;  
}
```

local1 is only accessible  
inside the func1

global is accessible  
throughout the program

```
// Function func2
```

```
int func2 () {  
  int local2;  
}
```

local2 is only accessible  
inside the func2



In the above illustration, `global` is declared in the start of the program. Therefore, we can access it anywhere in the program. We can access the `global` in `func1()` and `func2()`.

## Example program#

Press the **RUN** button and see the output!

```
1 #include <iostream>
2 using namespace std;
3 int global = 3;
4
5 void function () {
6     int function_local = 10;
7     cout << "global = " << global << endl;
8 }
9 int main() {
10     int main_local = 20;
11     cout << "global = " << global << endl;
12     function();
13     return 0;
14 }
```



Output

0.91s

```
global = 3
global = 3
```

## Exnlanation#



In the above program, the value of global is accessible in both `main()` and `function()`.

If two variables with the same name are declared twice within the same scope, the compiler will generate an error.

## Quiz



What is the output of the following code?

```
void function1 () {
    cout << "global = " << global << endl;
}

int global = 3;

void function2 () {
    cout << "global = " << global << endl;
}

int main() {
    cout << "global = " << global << endl;
    function1();
    function2();
}
```



A) 3



☐ B) 3

3

☐ C) 3

Your Answer



D) Error

Explanation

Global variable is accessible from the point it is declared to the end of the program. Since the `global` is declared after the `function1()`. Therefore, it is not accessible inside the `function1()`

Submit Answer

Reset Quiz ↻

In the upcoming lesson, we will discuss global variables in more detail.

← Back

Next →