Global Variables and Scope in C

In C programming language, **scope** is the region of the program where a given variable can be accessed using its name. Each variable has a particular scope attached to it.

On the basis of scope, there are two types of variables:

- Local Variables
- Global Variables

Local Variables

These are the variables declared inside a function or block and accessible only within the function or block where they are defined. These variables are specific to the function or block.

When a local variable is declared and not initialized, it may hold a random value (garbage value).

Example:

C

```c
#include <stdio.h>
int main()
{
    int x = 10; // Local variable
    printf("%d", x);
    return 0;
}
```

Output

```
10
```

Global Variables

Global Variables are declared outside all functions, usually at the top of the program and accessible across the entire program including all the functions.

If a global variable is not initialized, it is automatically assigned a default value of 0. This holds true for integers, characters, and floating-point types.

Example

C

```c
#include <stdio.h>
int x = 10;   // Global variable
int main()
{
    printf("%d", x);
    return 0;
}
```

Output

```
10
```

Extern Keyword and Global Variables

Global variables can be used before their definition by using the **extern** keyword. This tells the compiler that the variable will be defined elsewhere.

C

```c
#include <stdio.h>
extern int x;
int main()
{
    printf("%d", x);
    return 0;
}
int x = 10;
```

Output

```
10
```

Variable Shadowing

A variable declared in an inner block or scope can shadow a variable with the same name from an outer scope. When a variable is accessed, the innermost scope is checked first. If the variable is not found, the compiler checks outer scopes until it finds a match.

Two such cases are:

1. When local and global variable have same name.
2. When we have same variable name in multiple scopes.

Example 1: Local and global variable have same name.

C

```c
#include <stdio.h>

int x = 10;
int main()
{
    int x = 5;
    printf("%d", x);
    return 0;
}
```

Output

5

Example 2: Same variable name in multiple scopes

C

```c
#include <stdio.h>

int x = 20;
int main()
{
    int x = 10;
    {
      int x = 30;
      printf("%d", x);
    }
    return 0;
}
```

Output

30