Scoping Rules

The *scope* of a name (variable names, data structure names, procedure names) is the part of the program within which the name can be used. Darwin employs *dynamic scoping*, that is, Darwin does not require that you declare variables before you use them.

Thus,

> x := 5:

> x := x \* 99:

> y := z:

are perfectly digestible by the Darwin system.

**Scope** refers to the visibility of variables. In other words, which parts of your **program** can see or use it. Normally, every variable has a global **scope**. Once defined, every part of your **program** can access a variable. It is very useful to be able to limit a variable's **scope** to a single function.

*Lexical scoping* (sometimes known as **static scoping**) is a convention used with many programming languages that sets the scope (range of functionality) of a variable so that it may only be called (referenced) from within the block of code in which it is defined. The scope is determined when the code is compiled. A variable declared in this fashion is sometimes called a private variable.

The opposite approach is known as **dynamic scoping**. Dynamic scoping creates variables that can be called from outside the block of code in which they are defined. A variable declared in this fashion is sometimes called a **public variable**.

*Block Scope*

A **variable** can be either of global or local **scope**. A global **variable** is a variable declared in the main body of the source code, outside all functions, while a local variable is one declared within the body of a function or a **block**. Modern versions allow nested *lexical scoping*.

Scope

A scope is a region of the program and broadly speaking there are three places, where variables can be declared −

* Inside a function or a block which is called local variables,
* In the definition of function parameters which is called formal parameters.
* Outside of all functions which is called global variables.

We will learn what a function is and it's parameter in subsequent chapters. Here let us explain what local and global variables are.

Local Variables

Variables that are declared inside a function or block are local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. Following is the example using local variables −

[Live Demo](http://tpcg.io/QIjnPh)

#include <iostream>

using namespace std;

int main () {

// Local variable declaration:

int a, b;

int c;

// actual initialization

a = 10;

b = 20;

c = a + b;

cout << c;

return 0;

}

Global Variables

Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the life-time of your program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. Following is the example using global and local variables −

[Live Demo](http://tpcg.io/dRHHpD)

#include <iostream>

using namespace std;

// Global variable declaration:

int g;

int main () {

// Local variable declaration:

int a, b;

// actual initialization

a = 10;

b = 20;

g = a + b;

cout << g;

return 0;

}

A program can have same name for local and global variables but value of local variable inside a function will take preference. For example −

[Live Demo](http://tpcg.io/dt7MP9)

#include <iostream>

using namespace std;

// Global variable declaration:

int g = 20;

int main () {

// Local variable declaration:

int g = 10;

cout << g;

return 0;

}

When the above code is compiled and executed, it produces the following result −

10

Initializing Local and Global Variables

When a local variable is defined, it is not initialized by the system, you must initialize it yourself. Global variables are initialized automatically by the system when you define them as follows −

|  |  |
| --- | --- |
| **Data Type** | **Initializer** |
| int | 0 |
| char | '\0' |
| float | 0 |
| double | 0 |
| pointer | NULL |

It is a good programming practice to initialize variables properly, otherwise sometimes program would produce unexpected result.