**Storage classes.**

The storage for variables with *global* or *namespace scope* is allocated for the entire duration of the program. This is known as *static storage*, and it contrasts with the storage for *local variables* (those declared within a block). These use what is known as automatic storage. The storage for local variables is only available during the block in which they are declared; after that, that same storage may be used for a local variable of some other function, or used otherwise.

But there is another substantial difference between variables with *static storage* and variables with *automatic storage*:

- Variables with *static storage* (such as *global variables*) that are not explicitly initialized are automatically initialized to zeroes.  
- Variables with *automatic storage* (such as *local variables*) that are not explicitly initialized are left uninitialized, and thus have an undetermined value.

For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | //static vs automatic storage  #include <iostream>  using namespace std;  int x;  int main ()  {  int y;  cout << x << '\n';  cout << y << '\n';  return 0;  } | 0  4285838 | [Edit & Run](https://cplusplus.com/doc/tutorial/namespaces/) |

The actual output may vary, but only the value of *x* is guaranteed to be zero. *y* can actually contain just about any value (including zero).