**Scopes.**

Named entities, such as variables, functions, and compound types need to be declared before being used in C++. The point in the program where this declaration happens influences its visibility:

An entity declared outside any block has *global scope*, meaning that its name is valid anywhere in the code. While an entity declared within a block, such as a function or a selective statement, has *block scope*, and is only visible within the specific block in which it is declared, but not outside it.

Variables with block scope are known as *local variables*.

For example, a variable declared in the body of a function is a *local variable* that extends until the end of the function (i.e., until the brace } that closes the function definition), but not outside it:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | int foo; // global variable  int some\_function ()  {  int bar; // local variable  bar = 0;  }  int other\_function ()  {  foo = 1; // ok: foo is a global variable  bar = 2; // wrong: bar is not visible from this function, as identifier "bar" is undefined  } |  |

In each scope, a name can only represent one entity. For example, there cannot be two variables with the same name in the same scope:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 | int some\_function ()  {  int x;  x = 0;  double x; // wrong: name already used in this scope  x = 0.0;  } |  |

The visibility of an entity with *block scope* extends until the end of the block, including inner blocks. Nevertheless, an inner block, because it is a different block, can re-utilize a name existing in an outer scope to refer to a different entity; in this case, the name will refer to a different entity only within the inner block, hiding the entity it names outside. While outside it, it will still refer to the original entity.

For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  21  22 | // inner block scopes  #include <iostream>  using namespace std;  int k = 8; // the most global variable  int main () {  int x = 10;  int y = 20;  {  int x; // ok, inner scope.  x = 50; // sets value to inner x  y = 50; // sets value to (outer) y  cout << "inner block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  }  cout << "outer block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  return 0;  } | inner block:  x: 50  y: 50  k: 8  outer block:  x: 10  y: 50  k: 8 | [Edit & Run](https://cplusplus.com/doc/tutorial/namespaces/) |

Note that *y* is not hidden in the inner block, and thus accessing *y* still accesses the outer variable.

Variables declared in declarations that introduce a block, such as function parameters and variables declared in loops and conditions (such as those declared on a for or an if) are local to the block they introduce.

So, for better understanding, if change the value of *k* in the above code in the inner block, then the *k* in the outer block will be changed as well:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  21  22  23 | // inner block scopes  #include <iostream>  using namespace std;  int k = 8; // the most global variable  int main () {  int x = 10;  int y = 20;  {  int x; // ok, inner scope.  x = 50; // sets value to inner x  y = 50; // sets value to (outer) y  k = 5; // sets value for global variable k  cout << "inner block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  }  cout << "outer block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  return 0;  } | inner block:  x: 50  y: 50  k: 5  outer block:  x: 10  y: 50  k: 5 | [Edit & Run](https://cplusplus.com/doc/tutorial/namespaces/) |

But if we declare and initialize variable *k* anew in the inner block, the results of *k* in the outer and inner blocks will be different as they are now different variables in different scopes:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  21  22  23  24 | // inner block scopes  #include <iostream>  using namespace std;  int k = 8; // the most global variable  int main () {  int x = 10;  int y = 20;  {  int x; // ok, inner scope.  x = 50; // sets value to inner x  y = 50; // sets value to (outer) y  int k; // inner scope.  k = 5; // sets value for k in the inner scope  cout << "inner block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  }  cout << "outer block:\n";  cout << "x: " << x << '\n';  cout << "y: " << y << '\n';  cout << "k: " << k << '\n';  return 0;  } | inner block:  x: 50  y: 50  k: 5  outer block:  x: 10  y: 50  k: 8 | [Edit & Run](https://cplusplus.com/doc/tutorial/namespaces/) |