**Type deduction: auto and decltype.**

When a new variable is initialized, the compiler can figure out what the type of the variable is automatically by the initializer. For this, it suffices to use auto as the type specifier for the variable:

|  |  |  |
| --- | --- | --- |
| 1 2 | int foo = 0;  auto bar = foo; // the same as: int bar = foo; |  |

Here, bar is declared as having an auto type; therefore, the type of bar is the type of the value used to initialize it: in this case it uses the type of foo, which is *int*.

Variables that are not initialized can also make use of type deduction with the decltype specifier:

|  |  |  |
| --- | --- | --- |
| 1 2 | int foo = 0;  decltype(foo) bar; // the same as: int bar; |  |

Here, bar is declared as having the same type as foo.

auto and decltype are powerful features recently added to the language. But the type deduction features they introduce are meant to be used either when the type cannot be obtained by other means or when using it improves code readability. The two examples above were likely neither of these use cases. In fact, they probably decreased readability, since, when reading the code, one has to search for the type of foo to actually know the type of bar.