2.21 Auto (since C++11)

Versions of C++ since C++11 support auto specifiers. In a variable declaration, using ***auto*** as the type specifier causes the compiler to automatically deduce the type from the initializer. Ex: `auto i = 5;` causes i to be of type int, and `auto j = 5.0;` causes j to be of type double.

PARTICIPATION ACTIVITY 2.21.1: Auto in variable declarations. ✓

Indicate the type that a compiler (C++11 or later) would deduce for the variable declaration.

1) auto x = 9;
- [✓ int]
- double
- (error)

Correct

9 is an integer, so the compiler makes x of type int. ✓

2) auto x = -5;
- [✓ int]
- (error)

Correct

A negative integer is still an integer, so the compiler makes x an int. ✓

3) auto x = 0.01;
- int
- [✓ double]
- (error)

Correct

0.01 is a floating-point type, so the compiler makes x a double. ✓

4) const auto x = 5;
- [✓ int]
- (error)

Correct

const may be used along with auto. x will be a const int. ✓

5) auto x;
- int
- double
- [✓ (error)]

Correct

For auto, the compiler uses the initializer to determine the type. This declaration has no initializer, so the compiler will generate an error message. ✓

6) auto x = '9';
- int
- [✓ char]

Correct

The single quotes indicate a character. A character may be a letter like 'c' or 'X', or may be a digit like '9'. Thus, the ✓

- ○ (error)

compiler will deduce a char type.

7) auto x = "Hello";

- ○ char
- ○ string
- [✓ (something else)]

Correct

"Hello" is a string literal. The compiler deduces the type to be: const char *. That type is simpler than the more advanced string type. As such, programmers may wish to declare such variables as: `string x = "Hello";`

Feedback?

Exploring further:

- CppReference.com (auto)
- MSDN C++ reference (auto)