C++ Versions Report

**C++ 11**

* **auto** – the auto keyword is used as a replacement typecast for variable initializations. The compiler infers the type of a variable initialized with auto based on the value it is initialized to. Since C++ is not able to deduce type without context, the variable must be initialized when it is created, rather than just declared. The auto keyword is useful for when types get very complex, and it would be easier to just put auto and let the compiler figure it out on its own.1
* **nullptr** – the nullptr keyword is used to denote a pointer literal of type std::nullptr\_t. The value of null pointers was changed from 0 due to the issue that would arise from the auto type assuming the 0 represented an integer rather than a null pointer. The use of nullptr is important when there are multiple definitions for the same function name since its value is recognized as an actual pointer, whereas the value of NULL is recognized as an int.2
* **for(var:range)** – the range-based for loop was introduced in C++ 11 to make for loops simpler and quicker to implement when the contents of the loop are not concerned with indices. For example, when you want to execute a command a certain number of times, but the command is not influenced by what iteration of the loop it is, then a ranged-based for loop would be useful.3
* **enums (strongly typed)** – one of the main pitfalls of enums are that they cannot have a user-specified type – they are always converted to an integral type. There are also scoping problems with regular enums, since they export their enumerators to the surrounding scope. Strongly-typed enums fix these issues, as they can have a user-specified type and are no longer exported to the surrounding scope.4
* **lambda functions** – also known as anonymous functions, lambda functions are a concept taken from functional programming. They are useful for inclusion in functions like std::for\_each which requires a functor as a parameter. Instead of defining an entire class just to make one function to put into the for\_each, the function can be declared inline as a lambda.2
* **std::array** – this new array container was added to address the functionality problems of fixed and dynamic arrays that were already built into previous versions of the language. Dynamic arrays are difficult to resize or deallocate, and fixed arrays are converted into a pointer when passed into a function, losing the array.length functionality. std::array comes with “fixed array functionality that won’t decay when passed into a function.”5

**C++ 14**

* **auto return types** – similar to the auto keyword implemented in C++ 11, auto can now be used to infer return types of functions. Like the variables, the function must be defined and have a return type within it, otherwise the compiler will not be able to deduce the return type.6
* **variable templates** – used for manipulating constant values as different types, without having to create data members and template definitions. C++ 14 makes this process quicker and cleaner with the variable templates.7
* **The [[deprecated]] attribute** – this attribute produces a warning at compile time when a deprecated entity is used elsewhere in the program. This could be useful for when there are multiple people working on some piece of code and the developer would like to draw the attention to that specific area to tell the other developers to be careful. It can also include a custom message.8
* **tuple accessing by type** – tuples can now be accessed by asking for a type from the tuple rather than an index. However, this will result in an error if there are more than one elements of that type within the tuple, since it cannot differentiate between those elements.8

**C++ 17**

* **Initializers in if and switch statements** – similar to how you can initialize a variable in the parameters of a for loop, you will now be able to do the same in both if and switch statements. The variable will be inside the scope of the statement, so it is essentially the same as initializing a variable before the statement, then wrapping it all in brackets.9
* **Structured bindings** – multiple variables will be declarable from the result of a function that returns a tuple. Essentially, the tuple will be able to be split up into its own separate variables all in one line – something that was not possible before without a lengthy and complex workaround.9

References

1. *http://www.learncpp.com/cpp-tutorial/4-8-the-auto-keyword/*
2. *http://www.codeproject.com/Articles/570638/Ten-Cplusplus11-Features-Every-Cplusplus-Developer*
3. *http://en.cppreference.com/w/cpp/language/range-for*
4. *http://www.codeguru.com/cpp/cpp/article.php/c19083/C-2011-Stronglytyped-Enums.htm*
5. *http://www.learncpp.com/cpp-tutorial/6-15-an-introduction-to-stdarray/*
6. *http://www.drdobbs.com/cpp/the-c14-standard-what-you-need-to-know/240169034*
7. *https://www.ibm.com/developerworks/community/blogs/5894415f-be62-4bc0-81c5-3956e82276f3/entry/Introduction\_to\_Variable\_Templates\_of\_C\_14?lang=en*
8. *https://en.wikipedia.org/wiki/C%2B%2B14*
9. *https://meetingcpp.com/index.php/br/items/final-features-of-c17.html*