C++ For C Programmers Part B Ira Pohl

April 2016

Array, my\_container, and move semantics

C++11 introduces some new STL container classes. Among them is std::forward\_list(a singly linked list), std::unordered\_map( a hash based lookup), and an std::array( fixed length sequential container). This can be used to replace a C-style array without some of the performance hits that the more general std::vector has. It is an STL container; so it knows its size and can use iterators.

The introductory example in the videos and found in the downloadable code files show how we can code template class< class T, int n> my\_container our simpler version of std::array. The most novel idea in this example is that it uses move semantics. Move semantics are introduced with a new style of declaration: type && identifier (note in C++ “&&” is used contextually to be logical and)

int &&rvalue\_ref = 9; //literal 9 is an r value (as opposed to an lvalue

int l\_value; //simple declaration identifier is declared – it is an lvalue.

int &&rvalue\_ref = l\_value; //illegal i

int &&rvalue\_ref = move(l\_value); //okay – now an alias ; l\_value should no longer be used

So why all this esoterica? Because move semantics lead to efficiency when copying or assigning large aggregates. The copy is a shallow or referential copy.

Now take a look and play with various code examples to understand this further.