Bonus operations for C++/WinRT iterators: The IIterator

devblogs.microsoft.com/oldnewthing/20200930-00

September 30, 2020

Raymond Chen

C++/WinRT provides iterators for a number of basic Windows Runtime collections. Let's start with the lowest-level Windows Runtime iterator-ish thing: The `IIterator<T>`, which represents a cursor in a collection.

C++/WinRT adds two additional operators to the Windows Runtime-defined methods on the C++/WinRT `IIterator<T>` interface:

- The dereferencing `*` operator fetches the current item.
- The preincrement `++` operator moves the iterator to the next item. If the iterator has moved past the last item of the collection, then the iterator is set to `nullptr`.

This allows you to write code that consumes the contents of an `IIterator` in a somewhat more idiomatic way.

```
for (IIterator<int> it = start.HasCurrent() ? start : nullptr; it; ++it) {
  int value = *it;
  /* do something with the value */
}
```

This is still rather awkward because you have to check whether the starting point is already beyond the end of the collection and convert it to a `nullptr`. A little helper function can help with that.

```
template<typename T>
IIterator<T> as_cpp_iterator(IIterator<T> const& it)
{
  return it.HasCurrent() ? it : nullptr;
}
```

This simplifies the `for` loop slightly:

```
for (IIterator<int> it = as_cpp_iterator(start); it; ++it) {
  int value = *it;
  /* do something with the value */
}
```

These operators are added primarily for use in C++ standard algorithms that accept input iterators.

```
std::vector<int> to_vector(IIterator<int> const& it)
{
  std::vector<int> v;
  std::copy(as_cpp_iterator(it), {}, std::back_inserter(v));
  return v;
}

int sum(IIterator<int> const& it)
{
    return std::reduce(as_cpp_iterator(it), {}, 0);
}
```

Iterators are a rather clumsy way of walking through a collection. Next time, we'll look at something better.

Raymond Chen

Follow