```cpp
constexpr long long MAX_FIBONACCI = 92; //to have a fixed end limit

#include "FibonacciSequenceIterator.h"
#include <iostream>

// iterator constructor
FibonacciSequenceIterator::FibonacciSequenceIterator(const FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept
    : fSequenceObject(aSequenceObject), fIndex(aStart)
{
    if (aStart == 1)
    { //index 1 = fCurrent = 1
        fSequenceObject.reset();
    }
}

// return current Fibonacci number
const long long& FibonacciSequenceIterator::operator*() const noexcept
{
    return *fSequenceObject;
}

// prefix, next Fibonacci number
FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept
{
    //if has next number -> move on
    if (fSequenceObject.hasNext())
    {
        fSequenceObject.next();
        ++fIndex;
    }
    else
    {
        // Set fIndex out-of-scope to signal the end -> break the loop
        fIndex = MAX_FIBONACCI + 1;
    }
    return *this;
}

// postfix (extra unused argument)
FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept
{
    FibonacciSequenceIterator temp = *this;
    ++(*this);
    return temp;
}

bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator& aOther) const noexcept
{
    return this->fIndex == aOther.fIndex && this->fSequenceObject.id() == aOther.fSequenceObject.id();
}

bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator& aOther) const noexcept
```

```
53  {
54      return !(*this == aOther);
55  }
56
57  // return new iterator positioned at start
58  FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept {
59      return FibonacciSequenceIterator(fSequenceObject, 1);
60  }
61
62  // return new iterator positioned at limit
63  FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
64      return FibonacciSequenceIterator(fSequenceObject, MAX_FIBONACCI + 1);
65      //Indicate the end based on limits
66  }
67
```