```cpp
#include "FibonacciSequenceIterator.h"

FibonacciSequenceIterator::FibonacciSequenceIterator(const FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept : fSequenceObject(aSequenceObject), fIndex(aStart) {
    //Initialize the FibonacciSequenceIterator, make sure that the sequence object must be reset before iteration
    this->fSequenceObject.reset();
}

const long long& FibonacciSequenceIterator::operator*() const noexcept {
    //Getter, using the operator*() of FibonacciSequenceGenerator
    return this->fSequenceObject.operator*();
}

FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept {
    //++(FibonacciSequenceIterator)
    // Raise the fIndex to one value
    ++this->fIndex;
    //Check if could go to next value
    if (this->fSequenceObject.hasNext()) {
        //If can then jump to next value
        this->fSequenceObject.next();
    }
    //Return the iterator
    return *this;
}

FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept {
    //(FibonacciSequenceIterator)++
    //Copy iterator as temporary
    FibonacciSequenceIterator temporary = *this;
    //Raise the iterator to one value (both index and FibonacciSequenceGenerator object)
    ++temporary;
    //Return the copy
    return temporary;
}

bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator& aOther) const noexcept {
    //Compare if the two index of both objects are true and same id as well
    return this->fIndex == aOther.fIndex && this->fSequenceObject.id() == aOther.fSequenceObject.id();
}

bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator&
    aOther) const noexcept {
    //Negative of operator==
    return !(*this == aOther);
}

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept {
    //Copy the object as temporary
    FibonacciSequenceIterator temporary = *this;
    //Set the copy's index as 0
    temporary.fIndex = 0;
    //Reset the copy's FibonacciSequenceGenerator (Previous value will be 0 and Current value will be 1)
    temporary.fSequenceObject.reset();
    //Return the copy
    return temporary;
}

FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
    //Copy the object as temporary
    FibonacciSequenceIterator temporary = *this;
    //Do while loop while object is has next value
    while (temporary.fSequenceObject.hasNext()) {
        //Move fSequenceObject to one value
        temporary.fSequenceObject.next();
        //Raise the index to one value
        ++temporary.fIndex;
    }

    //Return the copy
    return temporary;
}
```