**FibonacciSequence.cpp**

#include "FibonacciSequence.h"

#include "FibonacciSequenceIterator.h"

#include <stdexcept>

FibonacciSequence::FibonacciSequence(unsigned long aLimit)

{

fLimit = aLimit;

fPrevious = 0;

fCurrent = 1;

fPosition = 1;

}

const unsigned long& FibonacciSequence::current() const

{

return fCurrent;

}

void FibonacciSequence::advance()

{

unsigned long next = fCurrent + fPrevious;

fPrevious = fCurrent;

fCurrent = next;

if (fLimit < fPosition)

{

throw std::out\_of\_range("Limit exceeded, please try again");

}

fPosition++;

}

const unsigned long& FibonacciSequence::getLimit() const

{

return fLimit;

}

FibonacciSequenceIterator FibonacciSequence::begin()

{

\*this = FibonacciSequence(getLimit());

return FibonacciSequenceIterator(\*this, 1);

}

FibonacciSequenceIterator FibonacciSequence::end()

{

return FibonacciSequenceIterator(\*this, getLimit()+1);

**FibonacciSequenceIterator.cpp**

#include "FibonacciSequenceIterator.h"

#include "FibonacciSequence.h"

FibonacciSequenceIterator::FibonacciSequenceIterator(FibonacciSequence& aSequenceObject, unsigned long aStart) : fSequenceObject(aSequenceObject), fIndex(aStart)

{

}

const unsigned long& FibonacciSequenceIterator::operator\*() const

{

return fSequenceObject.current();

}

FibonacciSequenceIterator& FibonacciSequenceIterator::operator++()

{

fSequenceObject.advance();

fIndex++;

return \*this;

}

FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int)

{

FibonacciSequenceIterator temp = \*this;

fSequenceObject.advance();

fIndex++;

return temp;

}

bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator& aOther) const

{

return (fIndex == aOther.fIndex)

&& fSequenceObject.current() == aOther.fSequenceObject.current()

&& fSequenceObject.getLimit() == aOther.fSequenceObject.getLimit();

}

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

{

return !(\*this == aOther);

}

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const

{

return fSequenceObject.begin();

}

FibonacciSequenceIterator FibonacciSequenceIterator::end() const

{

return fSequenceObject.end();

}

Problem 3:

The last output was off by 1 Fibonacci number because in the while loop it implements \*lIteratorC++. That skips the first IteratorC.

A picture containing knife

Description automatically generated

A picture containing bird

Description automatically generatedA close up of a logo

Description automatically generated