A assignment cover sheet with text and numbers

Description automatically generated

Problem 1 – FibonacciSequenceGenerator.cpp

//COS30008 - 104416957 - Avery Flannery. Semester 1 2024

//Fibonacci Sequence Generator

#include "FibonacciSequenceGenerator.h"

#include <cassert>

// constructor

FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string& aID) noexcept :

fID(aID), fPrevious(0), fCurrent(1) {}

// get sequence id

const std::string& FibonacciSequenceGenerator::id() const noexcept {

return fID;

}

//deference operator to get current Fibonacci number

const long long& FibonacciSequenceGenerator::operator\*() const noexcept {

return fCurrent; // returning the current number

}

// converting operator to bool

FibonacciSequenceGenerator::operator bool() const noexcept {

return hasNext(); // check if there is a next Fibonacci number

}

// used to reset the sequence generator

void FibonacciSequenceGenerator::reset() noexcept {

fPrevious = 0; // return previous

fCurrent = 1; // return current

}

// used to check if there is a next Fibonacci number

bool FibonacciSequenceGenerator::hasNext() const noexcept {

long long next = fPrevious + fCurrent; // calculates the next number

return next >= 0; // checks if the number is negative

}

// used to go on to the next Fibonacci number in the sequence

void FibonacciSequenceGenerator::next() noexcept {

long long next = fPrevious + fCurrent; // calculates the number

fPrevious = fCurrent; // updates the previous Fibonacci number

fCurrent = next; // updates the current Fibonacci number

}

Problem 2 – FibonacciSequenceIterator.cpp

//COS30008 - 104416957 - Avery Flannery. Semester 1 2024

//Fibonacci Sequence Iterator

#include "FibonacciSequenceIterator.h"

// constructor

FibonacciSequenceIterator::FibonacciSequenceIterator(const FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept

: fSequenceObject(aSequenceObject), fIndex(aStart) {}

// deference operator

const long long& FibonacciSequenceIterator::operator\*() const noexcept {

return \*fSequenceObject; // deferring the object

}

// prefix increment operator

FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept {

++fIndex; // increments iterator position

fSequenceObject.next(); //move to the next Fibonacci number

return \*this;

}

// postfix increment operator

FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept {

FibonacciSequenceIterator temp = \*this; // create a copy of the current iterator

++(\*this); // increment the iterator position

return temp; // return the copy

}

// == equals comparison operator

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

// comparing the sequence objects' ID and iterator positions

return (fSequenceObject.id() == aOther.fSequenceObject.id()) && (fIndex == aOther.fIndex);

}

// != does not equal comparison operator

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

return !(\*this == aOther);

}

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept {

FibonacciSequenceIterator beginIterator = \*this;

beginIterator.fIndex = 1; // sets the iterator to 1 to start

beginIterator.fSequenceObject.reset();

return beginIterator; // returns the iterator

}

FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {

FibonacciSequenceIterator endIterator = \*this;

endIterator.fIndex = 93; // sets the iterator position to 93 at the end

return endIterator; // returns the iterator

}

Output

A screen shot of a building

Description automatically generated

A black screen with a black background

Description automatically generated