```cpp
//
//  CharacterFrequencyIterator.cpp
//  Assignment2
//
//  Created by Ekrar Efaz on 12/4/23.
//

#include "CharacterFrequencyIterator.h"
#include <algorithm>

void CharacterFrequencyIterator::mapIndices() noexcept {
    for (int i = 0; i < 256; ++i){
        fMappedIndices[i] = static_cast<unsigned char>(i);
    }

    // stable insertion sort algorithm
    /*

     */
    for (int i = 1; i < 256; ++i){
        unsigned char currentCharacter = fMappedIndices[i]; // start with a character at the current index
        int previousCharacterIndex = i - 1; // set the previous character index
        while(previousCharacterIndex>=0 && fCollection->operator[](fMappedIndices[previousCharacterIndex]).frequency() < fCollection->operator[](currentCharacter).frequency()){
            // continue the loop until you go through all the previous elements until the first onw to keep swapping
            std::swap(fMappedIndices[previousCharacterIndex + 1], fMappedIndices[previousCharacterIndex]);
            -- previousCharacterIndex;
        }
    }
}

CharacterFrequencyIterator::CharacterFrequencyIterator( const CharacterCounter* aCollection ) noexcept:
    fCollection(aCollection),
    fIndex(0)
{
    mapIndices();
}

const CharacterMap& CharacterFrequencyIterator::operator*() const noexcept
{
    return fCollection->operator[](fMappedIndices[fIndex]);
}

CharacterFrequencyIterator& CharacterFrequencyIterator::operator++() noexcept
{
    fIndex ++;
    return *this;
}

CharacterFrequencyIterator CharacterFrequencyIterator::operator++( int ) noexcept{
    CharacterFrequencyIterator old = *this;
    ++(*this);
    return old;
}

bool CharacterFrequencyIterator::operator==( const CharacterFrequencyIterator& aOther ) const noexcept{
    return fCollection == aOther.fCollection && fIndex == aOther.fIndex;
}

bool CharacterFrequencyIterator::operator!=( const CharacterFrequencyIterator& aOther ) const noexcept{
    return !(*this == aOther);
}

CharacterFrequencyIterator CharacterFrequencyIterator::begin() const noexcept{
    CharacterFrequencyIterator copy = *this;
    copy.fIndex = 0;
    return copy;
}

/*
I started off with implemented fIndex=256 which resulted in showing a lot more characters which had zero frequency. My solution suggests
that I find the last non-zero character index and then set end to one past that. I use a while loop for this implementation
*/

CharacterFrequencyIterator CharacterFrequencyIterator::end() const noexcept {
    CharacterFrequencyIterator copy = *this;
    int i = 255; // start from the last character index
    while (i >= 0 && copy.fCollection->operator[](copy.fMappedIndices[i]).frequency() == 0) {
        --i; // search for last non-zero frequency character
    }
    copy.fIndex = i + 1; // set index to one past the last non-zero frequency character
    return copy;
}
```