```cpp
//
//  VigenereForwardIterator.cpp
//  midSem
//
//  Created by H M Asfaq Ahmed Shihab on 25/4/2024.
//
#include "VigenereForwardIterator.hpp"
//Constractor
VigenereForwardIterator::VigenereForwardIterator(const std::string&
 keyword, const std::string& source, EVigenereMode mode) noexcept
    : fKeys(keyword, source), fSource(source), fMode(mode), fIndex(0)
     {
    initializeTable();
    if (!fSource.empty()) {
        if (fMode == EVigenereMode::Encode) {
            encodeCurrentChar();
        } else {
            decodeCurrentChar();
        }
    }
}
// Encode the current character based on the Vigenere cipher
void VigenereForwardIterator::encodeCurrentChar() noexcept {
    if (std::isalpha(fSource[fIndex])) {
        int keyCharIndex = std::toupper(static_cast<unsigned
         char>(fKeys.operator*())) - 'A';
        int sourceCharIndex = std::toupper(static_cast<unsigned
         char>(fSource[fIndex])) - 'A';
        fCurrentChar = fMappingTable[keyCharIndex][sourceCharIndex];
        fCurrentChar = std::isupper(fSource[fIndex]) ? fCurrentChar :
         std::tolower(fCurrentChar);
        ++fKeys;
    } else {
        fCurrentChar = fSource[fIndex];
    }
}

void VigenereForwardIterator::decodeCurrentChar() noexcept {
    if (std::isalpha(fSource[fIndex])) {
        char keyChar = std::toupper(static_cast<unsigned
         char>(fKeys.operator*()));
        char encodedChar = std::toupper(static_cast<unsigned
         char>(fSource[fIndex]));
        int keyCharIndex = keyChar - 'A';

        int decodedCharIndex = 0;
        for (decodedCharIndex = 0; decodedCharIndex < CHARACTERS;
         ++decodedCharIndex) {
            if (fMappingTable[keyCharIndex][decodedCharIndex] ==
             encodedChar) {
                break;
```

```
            }
        }

        for (decodedCharIndex = 0; decodedCharIndex < CHARACTERS;
         ++decodedCharIndex) {
            if (fMappingTable[keyCharIndex][decodedCharIndex] ==
             encodedChar) {
                break;
            }
        }
        char decodedChar = 'A' + decodedCharIndex;
        if (!std::isupper(fSource[fIndex])) {
            decodedChar = std::tolower(decodedChar);
        }
        fCurrentChar = decodedChar;
        ++fKeys;

    } else {
        fCurrentChar = fSource[fIndex];
    }
}

char VigenereForwardIterator::operator*() const noexcept {
    return fCurrentChar;
}

VigenereForwardIterator& VigenereForwardIterator::operator++()
 noexcept {
    ++fIndex;
    if (fIndex < fSource.length()) {
        if (fMode == EVigenereMode::Encode) {
            encodeCurrentChar();
        } else {
            decodeCurrentChar();
        }
    }
    return *this;
}

VigenereForwardIterator VigenereForwardIterator::operator++(int)
 noexcept {
    VigenereForwardIterator temp = *this;
    ++(*this);
    return temp;
}

bool VigenereForwardIterator::operator==(const
 VigenereForwardIterator& other) const noexcept {
    return fIndex == other.fIndex && fSource == other.fSource;
}
```

```
bool VigenereForwardIterator::operator!=(const
 VigenereForwardIterator& other) const noexcept {
    return !(*this == other);
}

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

// Get the iterator pointing to the end of the source string
VigenereForwardIterator VigenereForwardIterator::end() const noexcept
 {
    VigenereForwardIterator it(*this);
    it.fIndex = fSource.length();
    return it;
}
```