```cpp
#include "VigenereForwardIterator.h"
#include <cctype>

VigenereForwardIterator::VigenereForwardIterator(const std::string& aKeyword, const
std::string& aSource, EVigenereMode aMode) noexcept
    : fMode(aMode), fKeys(aKeyword, aSource), fSource(aSource), fIndex(-1), fCurrentChar('\0')
{
        initializeTable();
}

void VigenereForwardIterator::encodeCurrentChar() noexcept {
    if (fIndex < fSource.length()) {
        char keyChar = *fKeys;
        char sourceChar = fSource[fIndex];

        if (std::isalpha(sourceChar)) {
            if (std::isupper(sourceChar)) {
                fCurrentChar = fMappingTable[keyChar - 'A'][sourceChar - 'A'];
            } else {
                fCurrentChar = fMappingTable[keyChar - 'A'][std::toupper(sourceChar) - 'A'] +
32;
            }
            // Increment fKeys only for alphabetic characters
            if (std::isalpha(sourceChar)) {
                ++fKeys;
                if (*fKeys == '\0') {
                    fKeys = fKeys.begin(); // Wrap around to the beginning of the keyword
                }
            }
        } else {
            fCurrentChar = sourceChar; // Preserve non-alphabetic characters
        }
    }
}

void VigenereForwardIterator::decodeCurrentChar() noexcept {
    if (fIndex < fSource.length()) {
        char keyChar = *fKeys;
        char sourceChar = fSource[fIndex];

        if (std::isalpha(sourceChar)) {
            if (std::isupper(sourceChar)) {
                for (int i = 0; i < CHARACTERS; ++i) {
                    if (fMappingTable[keyChar - 'A'][i] == sourceChar) {
                        fCurrentChar = 'A' + i;
                        break;
                    }
                }
            } else {
                for (int i = 0; i < CHARACTERS; ++i) {
                    if (fMappingTable[keyChar - 'A'][i] == std::toupper(sourceChar)) {
                        fCurrentChar = 'A' + i + 32;
                        break;
                    }
                }
            }
            // Increment fKeys only for alphabetic characters
            ++fKeys;
            if (*fKeys == '\0') {
                fKeys = fKeys.begin(); // Wrap around to the beginning of the keyword
            }
        } else {
            fCurrentChar = sourceChar; // Preserve non-alphabetic characters
        }
    }
}
```

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

VigenereForwardIterator& VigenereForwardIterator::operator++() noexcept {
    ++fIndex;
    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& aOther) const noexcept
{
    return fKeys == aOther.fKeys && fIndex == aOther.fIndex;
}

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

VigenereForwardIterator VigenereForwardIterator::begin() const noexcept {
    VigenereForwardIterator temp = *this;
    temp.fIndex = 0;
    if (fMode == EVigenereMode::Encode) {
        temp.encodeCurrentChar();
    } else {
        temp.decodeCurrentChar();
    }
    return temp;
}

VigenereForwardIterator VigenereForwardIterator::end() const noexcept {
    VigenereForwardIterator temp = *this;
    temp.fIndex = fSource.length(); // Set index to end of string
    temp.fCurrentChar = '\0'; // End of string
    return temp;
}
```