

# KeyProvider.cpp

```
//
// KeyProvider.cpp
// midsem
//
// Created by Vu Duc Tran on 24/4/2024.
//

#include "KeyProvider.h"
#include <cctype>
#include <cassert>

std::string KeyProvider::preprocessString(const std::string& aString) noexcept {
    std::string result;
    for (char c : aString) {
        if (std::isalpha(c)) {
            result += std::toupper(c);
        }
    }
    return result;
}

KeyProvider::KeyProvider(const std::string& aKeyword, const std::string& aSource) noexcept :
    fKeys(preprocessString(aKeyword)), fIndex(0) {
    size_t originalLength = fKeys.length();
    while (originalLength < aSource.length()) {
        fKeys += fKeys;
        originalLength += fKeys.length();
    }
    fKeys = fKeys.substr(0, preprocessString(aSource).length()); //substring
    assert(fKeys.length() <= preprocessString(aSource).length() && "The size of fKeys should
match the size of the preprocessed input string.");
}

char KeyProvider::operator*() const noexcept {
    return fKeys[fIndex];
}

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

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

```
bool KeyProvider::operator==(const KeyProvider& aOther) const noexcept {  
    return fKeys == aOther.fKeys && fIndex == aOther.fIndex;  
}
```

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

```
KeyProvider KeyProvider::begin() const noexcept {  
    return *this;  
}
```

```
KeyProvider KeyProvider::end() const noexcept {  
    KeyProvider temp = *this;  
    temp.fIndex = fKeys.length(); // Position after the last keyword character  
    return temp;  
}
```

# VigenereForwardIterator.cpp

```
//  
// VigenereForwardIterator.cpp  
// midsem  
//  
// Created by Vu Duc Tran on 24/4/2024.  
//  
#include "VigenereForwardIterator.h"  
#include <cctype>  
  
void VigenereForwardIterator::encodeCurrentChar() noexcept {  
    char sourceChar = fSource[fIndex];  
    char keyChar = *fKeys;  
  
    if (std::isupper(sourceChar)) {  
        fCurrentChar = fMappingTable[keyChar - 'A'][sourceChar - 'A'];  
        ++fKeys;  
    } else if (std::islower(sourceChar)) {  
        fCurrentChar = std::tolower(fMappingTable[keyChar - 'A'][std::toupper(sourceChar) - 'A']);  
        ++fKeys;  
    } else {  
        fCurrentChar = sourceChar;  
    }  
    ++fIndex;  
}  
  
void VigenereForwardIterator::decodeCurrentChar() noexcept {  
    char sourceChar = fSource[fIndex];  
    char keyChar = *fKeys;  
  
    if (std::isupper(sourceChar)) {  
        for (int i = 0; i < CHARACTERS; ++i) {  
            if (fMappingTable[keyChar - 'A'][i] == sourceChar) {  
                fCurrentChar = 'A' + i;  
                break;  
            }  
        }  
        ++fKeys;  
    } else if (std::islower(sourceChar)) {  
        for (int i = 0; i < CHARACTERS; ++i) {  
            if (fMappingTable[keyChar - 'A'][i] == std::toupper(sourceChar)) {  
                fCurrentChar = std::tolower('A' + i);  
                break;  
            }  
        }  
        ++fKeys;  
    }  
}
```

```

    }
}
++fKeys;
} else {
    fCurrentChar = sourceChar;
}
++fIndex;
}

```

```

VigenereForwardIterator::VigenereForwardIterator(
    const std::string& aKeyword,
    const std::string& aSource,
    EVigenereMode aMode) noexcept :
    fMode(aMode), fKeys(aKeyword, aSource), fSource(aSource), fIndex(0) {
    initializeTable();
    if (fMode == EVigenereMode::Encode) {
        encodeCurrentChar();
    } else {
        decodeCurrentChar();
    }
}

```

```

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

```

```

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

    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 iter = *this;
    iter.fKeys = fKeys.begin();
    return iter;
}
```

```
VigenereForwardIterator VigenereForwardIterator::end() const noexcept {
    VigenereForwardIterator iter = *this;
    iter.fKeys = fKeys.end();
    iter.fIndex = fSource.length() + 1;
    return iter;
}
```