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

std::string KeyProvider::preprocessString(const std::string& aString) noexcept {
    std::string processedString;
    for (char c : aString) {
        if (std::isalpha(c)) {
            processedString.push_back(std::toupper(c));
        }
    }
    return processedString;
}

KeyProvider::KeyProvider(const std::string& aKeyword, const std::string& aSource) noexcept
    : fKeys(preprocessString(aKeyword)), fIndex(0) {
    std::string processedSource = preprocessString(aSource);

    while (fKeys.length() < processedSource.length()) {
        fKeys += preprocessString(aKeyword); // Append the keyword again
    }

    // Truncate if necessary
    fKeys = fKeys.substr(0, processedSource.length());
}

char KeyProvider::operator*() const noexcept {
    if (fIndex >= 0 && fIndex < fKeys.length()) {
        return fKeys[fIndex];
    } else {
        return '\0'; // Return null character or any other suitable default value
    }
}

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 {
    // Create a copy of this iterator
    KeyProvider temp = *this;
    // Reset the iterator index to the beginning
    temp.fIndex = 0;
    return temp;
}


KeyProvider KeyProvider::end() const noexcept {
    KeyProvider temp = *this;
    // Set the iterator index to one position after the last keyword character
    temp.fIndex = fKeys.length();
    return temp;
```

}