```cpp

//Sartaj Khan
//Date: 26/04/23

#include "LZWCompressor.h"


bool LZWCompressor::readK() noexcept {
    if (fInput[fIndex + 1]) {
        fK = fInput[++fIndex];
        return true;
    }
    fK = -1;
    return false;
}


void LZWCompressor::start() {
    fTable = LZWTable();
    fTable.initialize();
    fK = fInput[0];
    fW = fTable.lookupStart(fK);
    fCurrentCode = nextCode();
}


uint16_t LZWCompressor::nextCode() {
    if (fK == -1) {
        return -1;
    }
    while (readK()) {
        PrefixString next = fW + fK;
        if (fTable.contains(next)) {
            fW = next;
        }
        else {
            fTable.add(next);
            uint16_t code = fW.getCode();
            fW = fTable.lookupStart(fK);
            return code;
        }
    }
    return fW.getCode();
}


LZWCompressor::LZWCompressor(const std::string& aInput) :
    fInput(aInput),
    fIndex(0)
```

```
{
    start();
}

const uint16_t& LZWCompressor::operator*() const noexcept {
    return fCurrentCode;
}

LZWCompressor& LZWCompressor::operator++() noexcept {
    fCurrentCode = nextCode();
    return *this;
}

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

bool LZWCompressor::operator==(const LZWCompressor& aOther) const noexcept {
    return (fInput == aOther.fInput) && (fIndex == aOther.fIndex && fK ==
      aOther.fK && fCurrentCode == aOther.fCurrentCode);
}

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

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

LZWCompressor LZWCompressor::end() const noexcept {
    LZWCompressor end = *this;
    end.fIndex = end.fInput.size() - 1;
    end.fCurrentCode = -1;
    end.fK = -1;
    return end;
}

```