```cpp
#include <iostream>
#include "ifstream12.h"
#include <cassert>

void ifstream12::reset() {
    for (size_t i = 0; i < fBufferSize; i++)
        fBuffer[i] &= std::byte{0};
    fByteCount = 0;
    fByteIndex = 0;
    fBitIndex = 7;
}

std::optional<size_t> ifstream12::readBit() {
    // Check if there are no bytes left in the buffer
    if (fByteCount == 0) {
        fetch_data();
        if (fIStream.eof()) {
            return std::nullopt; // EOF reached
        }
    }
    std::byte lByte = fBuffer[fByteIndex] & (std::byte{1} << fBitIndex);
    // Convert the resulting value to size_t to interpret it as an integer
    size_t bitValue = std::to_integer<size_t>(lByte);
    // If the resulting value is greater than zero, it denotes bit 1, else it denotes bit 0
    bitValue = (bitValue > 0) ? 1 : 0;

    fBitIndex--;
    if (fBitIndex < 0) {
        fByteIndex++; // Move to next Byte
        fBitIndex = 7; // Reset bit index
        fByteCount--; // Decrement the count of remaining bytes
    }
    return bitValue;
}

void ifstream12::fetch_data() {
    fIStream.read(reinterpret_cast<char*>(fBuffer), fBufferSize);
    fByteCount = fIStream.gcount();
    fByteIndex = 0;
    fBitIndex = 7;
}

ifstream12::ifstream12(const char* aFileName, size_t aBufferSize) :
    fBuffer(new std::byte[aBufferSize]),
    fBufferSize(aBufferSize),
    fByteCount(0),
    fByteIndex(0),
    fBitIndex(7)
    {
        reset();
        open(aFileName);
    }

ifstream12::~ifstream12() {
    close();
    delete[] fBuffer;
}

void ifstream12::open(const char* aFileName) {
    assert(!isOpen());
    if (aFileName) {
        fIStream.open(aFileName, std::ifstream::binary);
    }
}

void ifstream12::close() {
    fIStream.close();
```

```
    reset();
}

bool ifstream12::isOpen() const {
    return fIStream.is_open();
}

bool ifstream12::good() const {
    return fIStream.good();
}

bool ifstream12::eof() const {
    return (fByteCount == 0 || fIStream.eof());
}

ifstream12& ifstream12::operator>>(size_t& aValue) {
    aValue = 0;
    size_t i = 0;
    for (; i < 12; i++) {
        auto bit = readBit();
        if (!bit.has_value() || fIStream.eof()) {
            break; // readBit has no value or EOF reached
        }
        aValue = (aValue >> 1) | (static_cast<size_t>(*bit) << 11);
    }
    return *this;
}
```