```cpp
#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;
}

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

std::optional<size_t> ifstream12::readBit() {
    if (fByteCount == 0) {
        if (fIStream.eof()) {
            return std::nullopt;  // Return no value if EOF is reached
        }
        fetch_data();
    }

    std::byte lByte = fBuffer[fByteIndex] & (std::byte{ 1 } << fBitIndex);
    size_t lBitValue = std::to_integer<size_t>(lByte);

    fBitIndex--;
    if (fBitIndex < 0) {
        fBitIndex = 7;
        fByteIndex++;
        fByteCount--;
    }

    if (lBitValue == 0) {
        return 0;
    }
    else {
        return 1;
    }
}

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

    if (aFileName) {
        open(aFileName);
    }
}
```

```
53
54  ifstream12::~ifstream12() {
55      close();
56      delete[] fBuffer;
57  }
58
59  void ifstream12::open(const char* aFileName) {
60      assert(!isOpen());
61
62      if (aFileName) {
63          fIStream.open(aFileName, std::ifstream::binary);
64      }
65  }
66
67  void ifstream12::close() {
68      assert(isOpen());
69      fIStream.close();
70  }
71
72  bool ifstream12::isOpen() const {
73      return fIStream.is_open();
74  }
75
76  bool ifstream12::good() const {
77      return fIStream.good() && (fByteCount > 0 || !fIStream.eof());
78  }
79
80  bool ifstream12::eof() const {
81      return fByteCount == 0;
82  }
83
84  ifstream12& ifstream12::operator>>(size_t& aValue) {
85      aValue = 0;
86      for (size_t i = 0; i < 12; i++) {
87          auto bitOpt = readBit();
88          if (!bitOpt) {
89              break;
90          }
91          if (bitOpt == 1) {
92              aValue += (1 << i);
93          }
94      }
95
96      return *this;
97  }
```