```cpp
#include "ifstream12.h"

#include <iostream>
#include <cstddef>

ifstream12::ifstream12(const char* aFileName, size_t aBufferSize)
    : fBuffer(new std::byte[aBufferSize]), fBufferSize(aBufferSize)
{
    if (aFileName)
    {
        open(aFileName);
    }

    reset();
}

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

void ifstream12::open(const char* aFileName)
{
    if (aFileName)
    {
        fIStream.open(aFileName, std::ios::binary);
    }
}

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

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

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

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


void ifstream12::reset()
{
    for (size_t i = 0; i < fBufferSize; i++)
    {
        fBuffer[i] = static_cast<std::byte>(0);
```

```
57      }
58
59      fByteCount = 0;
60      fByteIndex = 0;
61      fBitIndex = 7; // Bits are read from MSB to LSB
62  }
63
64  void ifstream12::fetch_data()
65  {
66      // Only attempt to fetch if we are not known to be at the end of the file.
67      if (!fIStream.eof())
68      {
69          fIStream.read(reinterpret_cast<char*>(fBuffer), fBufferSize);
70          fByteCount = fIStream.gcount();
71          fByteIndex = 0;
72          fBitIndex = 7;
73
74          if (fByteCount == 0)
75          {
76              // If no data is fetched, manually check EOF again.
77              fIStream.peek();
78          }
79      }
80  }
81
82  std::optional<size_t> ifstream12::readBit()
83  {
84      // Check if we need to fetch more data.
85      if (fByteIndex >= fByteCount)
86      {
87          fetch_data();
88
89          // After fetching, if still no data, then attempt to read beyond
90          if (fByteCount == 0)
91          {
92              // Check for EOF directly after fetch attempt
93              if (fIStream.eof())
94              {
95                  return std::nullopt;
96              }
97              else
98              {
99                  // Explicitly try to peek to trigger EOF if no more data is
                      available
100                 if (fIStream.peek() == EOF)
101                 {
102                     return std::nullopt;
103                 }
104             }
105         }
106     }
107
108     std::byte lByte = fBuffer[fByteIndex] & (std::byte{1} << fBitIndex);
109     size_t bitValue = std::to_integer<size_t>(lByte) ? 1 : 0;
110
111     // Move to the next bit
```

```
112     fBitIndex--;
113
114     // Check if we have exhausted this byte
115     if (fBitIndex < 0)
116     {
117         fBitIndex = 7;
118         fByteIndex++;
119
120         // Reduce byte count and check if we need to fetch again
121         if (fByteIndex >= fByteCount)
122         {
123             fetch_data();
124             if (fByteCount == 0 && fIStream.eof())
125             {
126                 return std::nullopt;
127             }
128         }
129     }
130
131     return bitValue;
132 }
133
134
135 ifstream12& ifstream12::operator>>(size_t& aValue)
136 {
137     aValue = 0;
138     for (int i = 0; i < 12; i++)
139     {
140         auto bit = readBit();
141
142         if (!bit.has_value()) break;
143
144         else if (bit == 1)
145         {
146             aValue += (bit.value() << i);
147         }
148     }
149
150     return *this;
151 }
```