A document with text and numbers

Description automatically generated

**Problem 1**

//COS30008 - 1044169576 - Avery Flannery

//Problem Set 3 - Design Pattern and 12-bit I/o

#include "ifstream12.h"

#include <cassert>

// Reset function to reset the input stream

void ifstream12::reset() {

// Reset buffer to all zeros

for (size\_t i = 0; i < fBufferSize; i++) {

fBuffer[i] = std::byte{ 0 };

}

// Reset byte count, byte index, and bit index

fByteCount = 0;

fByteIndex = 0;

fBitIndex = 7;

}

// Function to fetch data from the input stream

void ifstream12::fetch\_data() {

reset(); // Reset input stream

fIStream.read(reinterpret\_cast<char\*>(fBuffer), fBufferSize); // Read data from the input stream

fByteCount = fIStream.gcount(); // Get the number of bytes read

}

// Function to read the next bit from the input stream

std::optional<size\_t> ifstream12::readBit() {

if (fByteCount == 0) {

fetch\_data();

}

if (eof()) { // If end of file is reached

return std::nullopt;

}

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

fBitIndex--; // Move to the next bit

if (fBitIndex < 0) {

fByteCount--;

fByteIndex++;

fBitIndex = 7;

}

return std::to\_integer<size\_t>(lByte) != 0;

}

// Constructor

ifstream12::ifstream12(const char\* aFileName, size\_t aBufferSize) {

fBuffer = new std::byte[aBufferSize]; //

fBufferSize = aBufferSize;

reset();

open(aFileName);

}

// Destructor

ifstream12::~ifstream12() {

close();

delete[] fBuffer;

}

// Function to open a file

void ifstream12::open(const char\* aFileName) {

assert(!isOpen());

if (aFileName) {

fIStream.open(aFileName, std::ifstream::binary);

}

}

// Function to close the file

void ifstream12::close() {

fIStream.close();

}

// Function to check if a file is open

bool ifstream12::isOpen() const {

return fIStream.is\_open();

}

// Function to check if the input stream is in good state

bool ifstream12::good() const {

return fIStream.good();

}

// Function to check for end of file

bool ifstream12::eof() const {

return (fByteCount == 0);

}

// Overloaded input operator to read 12 bits from the input stream

ifstream12& ifstream12::operator>>(size\_t& aValue) {

aValue = 0;

for (int i = 0; i < 12; i++) {

auto bit = readBit();

if (bit == std::nullopt) {

break;

}

else if (bit == 1) {

aValue += (static\_cast<size\_t>(1) << i);

}

}

return \*this;

}

**Output**A computer screen shot of a computer

Description automatically generated