# **Звіт про виконання лабораторної роботи № 2.3**

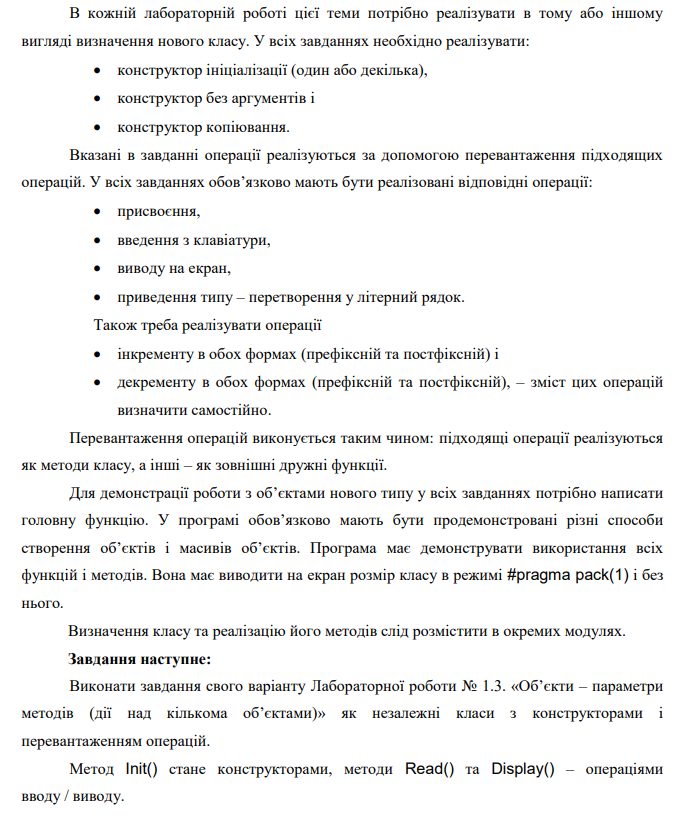
## «Конструктори та перевантаження операцій для класів» з дисципліни «Об’єктно-орієнтоване програмування»

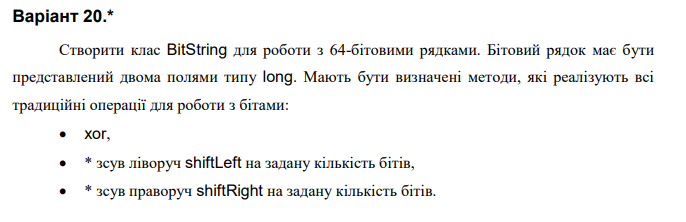
## Студента групи «**ІТ-12**» - **Степанчука Сергія**

## Мета роботи

Освоїти використання конструкторів та перевантаження операцій.

## Умова завдання

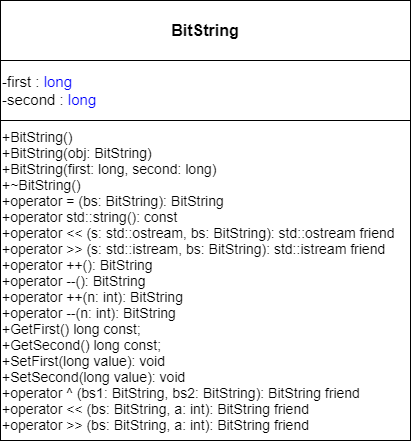




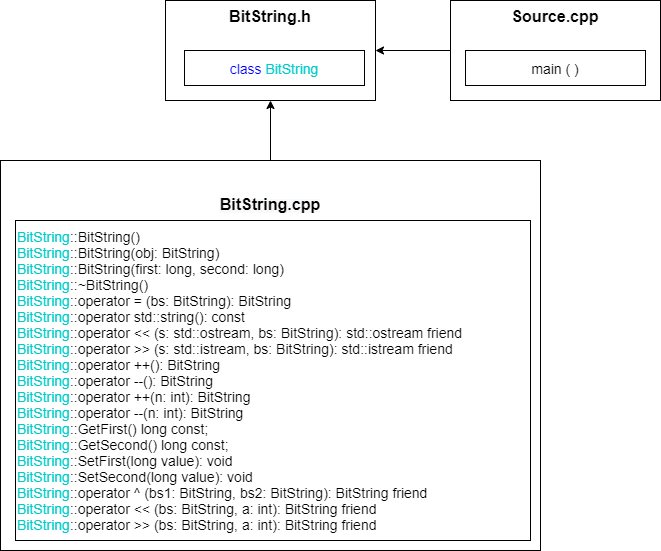
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_2.3>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // BitString.h

#pragma once

//визначення класу

#include <string>

#include <iostream>

#include <sstream>

class BitString

{

private:

long first = 0;

long second = 0;

public:

BitString() {};

BitString(const BitString& obj) :

first(obj.first), second(obj.second) {}

BitString(const long& first, const long& second);

~BitString() {};

BitString& operator = (const BitString&);

operator std::string() const;

friend std::ostream& operator << (std::ostream&, const BitString&);

friend std::istream& operator >> (std::istream&, BitString&);

// prefix

BitString& operator ++();

BitString& operator --();

// postfix

BitString operator ++(int);

BitString operator --(int);

long GetFirst() const;

long GetSecond() const;

void SetFirst(long value);

void SetSecond(long value);

friend BitString operator ^ (const BitString& a, const BitString& b);

friend BitString operator << (const BitString& a, int n);

friend BitString operator >> (const BitString& a, int n);

};

### // BitString.cpp

//реалізація методів класу

#include "BitString.h"

using namespace std;

long BitString::GetFirst() const

{

return first;

}

long BitString::GetSecond() const

{

return second;

}

void BitString::SetFirst(long value)

{

first = value;

}

void BitString::SetSecond(long value)

{

second = value;

}

BitString::BitString(const long& first, const long& second): first(first), second(second) {}

BitString& BitString::operator = (const BitString& pr)

{

first = pr.first;

second = pr.second;

return \*this;

}

BitString::operator std::string() const

{

stringstream ss;

ss << first << ", " << second;

return ss.str();

}

ostream& operator << (ostream& out, const BitString& r)

{

out << string(r);

return out;

}

istream& operator >> (istream& in, BitString& r)

{

cout << "first = "; in >> r.first;

cout << "second = "; in >> r.second;

return in;

}

BitString& BitString::operator ++()

{

first++;

return \*this;

}

BitString& BitString::operator --()

{

first--;

return \*this;

}

BitString BitString::operator ++(int)

{

BitString t(\*this);

second++;

return t;

}

BitString BitString::operator --(int)

{

BitString t(\*this);

second--;

return t;

}

BitString operator ^ (const BitString& a, const BitString& b)

{

BitString t;

t.first = a.first ^ b.first;

t.second = a.second ^ b.second;

return t;

}

BitString operator << (const BitString& a, int n)

{

BitString t = a;

long tmp[2];

const int last\_bit = sizeof(long) \* 8 - 1;

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

{

tmp[0] = t.first << 1;

tmp[1] = t.second << 1;

if (t.first & (1 << last\_bit))

tmp[1] |= (1 << 0);

else

tmp[1] &= ~(1 << 0);

if (t.second & (1 << last\_bit))

tmp[0] |= (1 << 0);

else

tmp[0] &= ~(1 << 0);

t.first = tmp[0];

t.second = tmp[1];

}

return t;

}

BitString operator >> (const BitString& a, int n)

{

BitString t = a;

long tmp[2];

const int last\_bit = sizeof(long) \* 8 - 1;

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

{

tmp[0] = t.first >> 1;

tmp[1] = t.second >> 1;

if (t.first & (1 << 0))

tmp[1] |= (1 << last\_bit);

else

tmp[1] &= ~(1 << last\_bit);

if (t.second & (1 << 0))

tmp[0] |= (1 << last\_bit);

else

tmp[0] &= ~(1 << last\_bit);

t.first = tmp[0];

t.second = tmp[1];

}

return t;

}

### // main.cpp

// функція main

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "BitString.h"

using namespace std;

int main()

{

BitString o;

cin >> o;

int n;

cout << o << endl << endl;

// ShiftLeft

cout << "How many bits do you want to shift left?" << endl;

cout << "Input: "; cin >> n;

BitString a = o << n;

cout << a << endl;

// ShiftRight

cout << "How many bits do you want to shift right?" << endl;

cout << "Input: "; cin >> n;

BitString b = o >> n;

cout << b << endl;

// Xor

cout << "Xor: " << endl;

cout << (a ^ b) << endl;

cin.get();

return 0;

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Program/BitString.h"

#include "../Program/BitString.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

TEST\_METHOD(Xorr)

{

BitString a, b;

a.SetFirst(4);

a.SetSecond(4);

b.SetFirst(8);

b.SetSecond(8);

BitString c = (a ^ b);

Assert::AreEqual(c.GetFirst(), 12l);

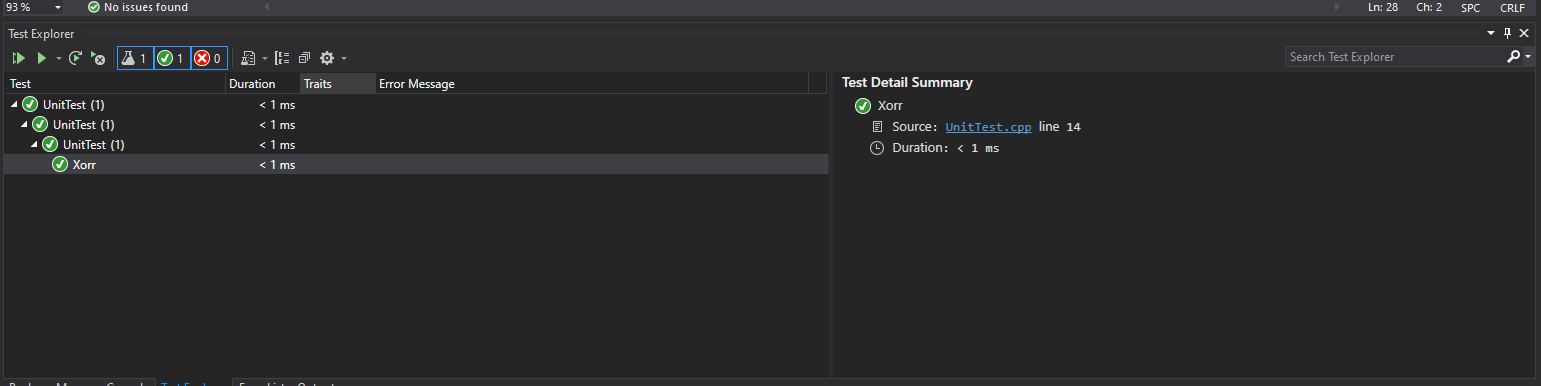
Assert::AreEqual(c.GetSecond(), 12l);

}

};

}

### Результат



## Висновок

Освоїв використання конструкторів та перевантаження операцій.