МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ

ЛАБОРАТОРНАЯ РАБОТА №1

по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Белоусов Егор Владимирович, группа М8О-207Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Цель:

- Изучение системы сборки на языке C++, изучение систем контроля версии.
- Изучение основ работы с классами в C++;

Задание:

Создать класс BitString для работы с 128-битовыми строками. Битовая строка должна быть представлена двумя полями типа unsigned long long. Должны быть реализованы все традиционные операции для работы с битами: and, or, xor, not. Реализовать сдвиг влево shiftLeft и сдвиг вправо shiftRight на заданное количество битов. Реализовать операцию вычисления количества единичных битов, операции сравнения по количеству единичных битов. Реализовать операцию проверки включения.

Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp: основная программа
2. bitstring.h : описание класса bitstring
3. bitstring.cpp: реализация класса bitstring

Дневник отладки

В main.cpp выполняются самые различные операции с двумя битовыми строчками.

D:\oop\lab01\cmake-build-debug\lab01.exe

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

01100100

00000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000

00000000

00000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000

01100100

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

00000000

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000110010000

00000000

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

00000000

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000110010000

00000000

0

1

11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

11111111

128

Process finished with exit code 0

Недочёты

По моему мнению недочетов нет.

Выводы

В данной лабораторной работе я познакомился с простейшими элементами ООП. Я создал класс для работы 128 битными строчками, реализовал для него простейшие битовые операции: XOR, OR, AND, NOT и т.д. На данном примере я понял, что использование объектно-ориентированного программирования делает написание кода более простым и удобным занятием.

Исходный код

main.cpp

```cpp
#include "bitstring.h"

int main() {
    BitString a(0, 100);
    BitString b(1, 0);
    print(a);
    print(b);
    print(XOR(a, b));
    print(AND(a, b));
    a = ShiftLeft(a, 10);
    b = ShiftRight(b, 32);
    print(a);
    print(b);
    BitString c = OR(a, b);
    print(c);
    std::cout << (compare(a, b)) << "\n";

    a = BitString(5, 2);
    b = BitString(1, 2);
    std::cout << subMask(a, b) << "\n";

    a = OR(a, NOT(a));
    print(a);

    std::cout << a.count() << "\n";
    system("pause");// if you want to run exe
    return 0;
}
```

bitstring.h

```cpp
#ifndef LAB01_BITSTRING_H
#define LAB01_BITSTRING_H

#include "iostream"


class BitString {
private:
    unsigned long long first;
    unsigned long long second;
public:
    BitString();

    BitString(unsigned long long first, unsigned long long second);

    int count() const;

    friend BitString AND(const BitString &a, const BitString &b);

    friend BitString OR(const BitString &a, const BitString &b);

    friend BitString XOR(const BitString &a, const BitString &b);

    friend BitString NOT(const BitString &a);
```

```
    friend BitString ShiftLeft(const BitString &a, unsigned int cnt);

    friend BitString ShiftRight(const BitString &a, unsigned int cnt);

    friend bool operator==(const BitString &a, const BitString &b);

    friend bool subMask(const BitString &mask, const BitString &submask);

    friend bool compare(const BitString &a, const BitString &b);

    friend void print(const BitString &a);

    ~BitString() = default;
};


#endif //LAB01_BITSTRING_H
```

bitstring.cpp

```
#include "bitstring.h"

BitString::BitString() : first(0), second(0) {}

BitString::BitString(unsigned long long first, unsigned long long second) :
first(first), second(second) {}

int BitString::count() const {
    return __builtin_popcountll(first) + __builtin_popcountll(second);
}

BitString AND(const BitString &a, const BitString &b) {
    return BitString(a.first & b.first, a.second & b.second);
}

BitString OR(const BitString &a, const BitString &b) {
    return BitString(a.first | b.first, a.second | b.second);
}

BitString XOR(const BitString &a, const BitString &b) {
    return BitString(a.first ^ b.first, a.second ^ b.second);
}

BitString NOT(const BitString &a) {
    return BitString(~a.first, ~a.second);
}

BitString ShiftLeft(const BitString &a, unsigned int cnt) {
    return BitString((a.first << cnt) + (a.second >> (64 - cnt)), a.second << cnt);
}

BitString ShiftRight(const BitString &a, unsigned int cnt) {
    return BitString(a.first >> cnt, (a.second >> cnt) + (a.first << (64 - cnt)));
}

bool operator==(const BitString &a, const BitString &b) {
    return a.first == b.first && a.second == b.second;
}

bool subMask(const BitString &a, const BitString &b) {
```

```
    return OR(a, b) == a;
}

bool compare(const BitString &a, const BitString &b) {
    return a.count() < b.count();
}

void print(const BitString &a) {
    for (int i = 63; i >= 0; --i) {
        if ((a.first >> i) & 1) {
            std::cout << "1";
        } else {
            std::cout << "0";
        }
    }
    for (int i = 63; i >= 0; --i) {
        if ((a.second >> i) & 1) {
            std::cout << "1";
        } else {
            std::cout << "0";
        }
    }
    std::cout << "\n";
}
```