МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2

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

Студент *Каширин Кирилл Дмитриевич, группа М8О-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

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. Исходный код лежит в 3 файлах:

1. main.cpp: основная программа, взаимодействие с пользователем посредством команд из меню
2. BitString.h: описание класса адресов
3. BitString.cpp: реализация класса адреса

Протокол работы

BitString a:
00000000000000000000000000000000000000000000000000
01011011100000010000000000000000000000000000000000
00000000000000000111111010011101
BitString b:
00000000000000000000000000000000000000000000001010
01011011000111100000000000000000000000000000000000
00000000000001010100010100000010
AND
00000000000000000000000000000000000000000000000000
10110110000000000000000000000000000000000000000000
00000000000000010001000000000000
OR
00000000000000000000000000000000000000000000010100
10110111001111100000000000000000000000000000000000
00000000001010111111110011111
XOR
00000000000000000000000000000000000000000000010100
00000010011111000000000000000000000000000000000000
00000000001010011101110011111
NOT
11111111111111111111111111111111111111111111111111
01001000111111011111111111111111111111111111111111
11111111111111000000101100010
1 in a
18
0
0
Example with literal:00000000000000000000000000
00000000000001000111011110101011011000000000000000
00000000000000000000000000000011001110110011001100011
Example with literal:00000000000000000000000000
00000000000001000111011110101011011000000000000000
00000000000000000000000000000011001110110011001100011

Дневник отладки

Проблем и ошибок при написании данной работы не возникло.

Недочёты

Выводы

В процессе выполнения работы я на практике познакомился с пользовательскими литералами. Как оказалось, у них есть свои преимущества. Они очень удобны и практичны. Использование этого средства позволяет получать из заданных типов данных какие то данные с использованием специального оператора.

Исходный код:

adress.h
adress.cpp
main.cpp

```
#include <iostream>
#include "BitString.h"
#include <cmath>
using namespace std;

BitString operator "" _bit(const char* str, size_t size) {
    int cnt = 0;
    unsigned long long one=0;
    unsigned long long two = 0;
    while (str[cnt]!=' ') {
        one*=10;
        one+=str[cnt]-'0';
        cnt++;
    }
    cnt++;
    while (str[cnt]!='\0') {
        two*=10;
        two+=str[cnt]-'0';
        cnt++;
    }
    BitString a(one,two);
    return a;
}
int main() {
    BitString a(23425, 32413);
    BitString b(678686, 345346);
    //BitString c(cin);
    cout << "BitString a:" << endl << a << endl;
    cout << "BitString b:" << endl << b << endl;
    cout << "AND" << endl << (a&b) << endl;
    cout << "OR" << endl << (a||b) << endl;
    cout << "XOR" << endl << (a^b) << endl;
    cout << "NOT" << endl << ~a << endl;
    cout << "1 in a" << endl << a.count() << endl;
    cout << (b > a) << endl;
    cout << a.include(b) << endl;
```

```
    cout << "Example with literal:" << "2342235 423523"_bit << endl;
    BitString test(2342235,423523);
     cout << "Example with literal:" << test << endl;
}
```