Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Простые классы.

Студент:	Касимов М.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	6
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

bits.h:

```
#ifndef OOP_EXERCISE_01_BITS_H
#define OOP_EXERCISE_01_BITS_H

#include <iostream>
#include <string>
#include <vector>

class bit {
private:
    unsigned long long a;
    unsigned int b;
public:
    void read(std::istream & is );
    void write_10(std::ostream& os);
    void write_2(std::ostream& os);
    bit();
    bit AND (const bit & lel) const ;
    bit OR (const bit & lel) const ;
    bit XOR (const bit & lel) const ;
    bit NOT () const ;
    void shiftLeft(std::istream& is);
    void shiftRight(std:: istream& is);
    int number_of_units() const ;
    int comparsion(const bit & lel) const;
    int inclusion(const bit & lel) const;
    void vvod(unsigned long long i, unsigned int j);
    unsigned long long vivod_star() const ;
    unsigned int vivod_mlad() const ;
};

#endif //OOP_EXERCISE_01_BITS_H
```

bits.cpp:

```
// Created by magom on 14.09.2019.
#include "bits.h"

int number(unsigned long long m) {
    int i = 0;
    while (m > 0) {
        i += m % 2;
        m = m / 2;
    }
    return i;
}

void bit::read(std::istream &is) {
    std::string s;
    is >> s;
```

```
    int n = s.size();
    std::string t(n, '0');
    std::vector<int> v;
    while (s != t) {
        int d = 0;
        for (int i = 0; i < s.size(); i++) {
            d *= 10;
            d += (s[i] - 48);
            s[i] = char(48 + d / 2);
            d %= 2;
        }
        v.push_back(d);
    }
    //   for (auto st : v)
    //     std::cout << st << ' ';
    //std::cout << std::endl;

    unsigned int b_step = 1;
    for (int i = 0; i < 32 && i < v.size(); i++) {
        b += v[i] * b_step;
        b_step *= 2;
    }
    unsigned long long a_step = 1;
    for (int i = 32; i < v.size(); i++) {
        a += v[i] * a_step;
        a_step *= 2;
    }
}

void to2(unsigned long long a, std::ostream &os) {
    if (a == 0)
        return;
    to2(a / 2, os);
    std::cout << a % 2;
}

void bit::write_10(std::ostream &os) {
    os << a << " " << b << std::endl;
}

void bit::write_2(std::ostream &os) {
    if (a != 0)
        to2(a, os);
    else
        std::cout << 0;
    std::cout << " ";
    if (b != 0)
        to2(b, os);
    else
        for (int i = 0; i < 32; ++i) {
            std::cout << 0;
        }
```

```
    os << std::endl;
}

bit bit::AND(const bit &lel) const {
    bit temp;
    unsigned long long temp_a = a & lel.vivod_star();
    unsigned int temp_b = b & lel.vivod_mlad();
    temp.vvod(temp_a, temp_b);
    return temp;
}

bit bit::OR(const bit &lel) const {
    bit temp;
    unsigned long long temp_a = a | lel.vivod_star();
    unsigned int temp_b = b | lel.vivod_mlad();
    temp.vvod(temp_a, temp_b);
    return temp;
}

bit bit::XOR(const bit &lel) const {
    bit temp;
    unsigned long long temp_a = a ^lel.vivod_star();
    unsigned int temp_b = b ^lel.vivod_mlad();
    temp.vvod(temp_a, temp_b);
    return temp;
}

bit bit::NOT() const {
    bit temp;
    unsigned long long temp_a = ~a;
    unsigned int temp_b = ~b;
    temp.vvod(temp_a, temp_b);
    return temp;
}

void bit::shiftLeft(std::istream &is) {
    int k = 0, i = 0;
    is >> i;
    while (k < i) {
        if (b >= 2147483648) {
            a = a << 1;
            ++a;
            b = b << 1;
            ++k;
        } else {
            a = a << 1;
            b = b << 1;
            ++k;
        }
    }
}
```

```
void bit::shiftRight(std::istream &is) {
    unsigned int k = 0, f = 2147483648, i = 0;
    is >> i;
    while (k < i) {
        if (a % 2 == 1) {
            a = a >> 1;
            b = b >> 1;
            b = b | f;
            ++k;
        } else {
            a = a >> 1;
            b = b >> 1;
            ++k;
        }
    }
}


void bit::vvod(unsigned long long i, unsigned int j) {
    a = i;
    b = j;
}

unsigned long long bit::vivod_star() const {
    return a;
}

unsigned int bit::vivod_mlad() const {
    return b;
}

int bit::number_of_units() const {
    return number(a) + number(b);

}

int bit::comparsion(const bit &lel) const {
    int i1 = number(a) + number(b);
    int i2 = lel.number_of_units();
    if (i1 == i2)
        return 1;
    else
        return 0;
}

int bit::inclusion(const bit &lel) const {
    std::vector<int> v1, v2;
    int result = 1;
    int min_size;
    unsigned int b1 = b;
    while (b1 > 0) {
        v1.push_back(b1 % 2);
```

```
        b1 = b1 / 2;
    }
    unsigned long long a1 = a;
    while (a1 > 0) {
        v1.push_back(a1 % 2);
        a1 = a1 / 2;
    }
    b1 = lel.vivod_mlad();
    while (b1 > 0) {
        v2.push_back(b1 % 2);
        b1 = b1 / 2;
    }
    a1 = lel.vivod_star();
    while (a1 > 0) {
        v2.push_back(a1 % 2);
        a1 = a1 / 2;
    }
    if (v1.size() >= v2.size()) {
        min_size = v2.size();
    } else
        return 0;
    for (int i = 0; i < min_size; ++i) {
        if (v1[i] != v2[i]) {
            result = 0;
            break;
        }
    }
    return result;
}

bit::bit() {
    a = 0;
    b = 0;
}
```

main.cpp:

```
#include <iostream>
#include "bits.h"

int main() {

    bit a;

    a.read(std::cin);
    a.write_10(std::cout);
    a.write_2(std::cout);
    a.shiftLeft(std::cin);
    a.write_10(std::cout);
    a.write_2(std::cout);
    std::cout<<a.number_of_units()<<std::endl;

    bit c;
```

```
    c.read(std::cin);
    c.write_10(std::cout);
    c.write_2(std::cout);

    bit f;
    f=a.AND(c);
    f.write_10(std::cout);
    f.write_2(std::cout);
    f=a.OR(c);
    f.write_10(std::cout);
    f.write_2(std::cout);
    f=a.NOT();
    f.write_10(std::cout);
    f.write_2(std::cout);
    f=a.XOR(c);
    f.write_10(std::cout);
    f.write_2(std::cout);
    std::cout<<c.inclusion(a)<<std::endl<<c.comparsion(a);
    return 0;
}
```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 3.10.2)
project(oop_exercise_01)

set(CMAKE_CXX_STANDARD 14)

add_executable(oop_exercise_01 main.cpp bits.h bits.cpp)
```

test.sh:

```
executable=$1

for file in test_??.test
do
  $executable < $file > tmp
  if cmp tmp ${file%%.test}.result
  then
    echo Test "$file": SUCCESS
  else
    echo Test "$file": FAIL
  fi
  rm tmp
done
```

2. Ссылка на репозиторий на GitHub.

https://github.com/magomed2000kasimov/oop_exercise_01

3. Набор тестов.

test_01.test:
5
1
7

test_02.test:
4294967296
1
1

4. Результаты выполнения тестов.

test_01.result:
0 5
0 101
0 10
0 1010
2
0 7
0 111
0 2
0 10
0 15
0 1111
18446744073709551615 4294967285
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111110101
0 13
0 1101
0
0
test_02.result:
1 0
1 00000000000000000000000000000000
2 0
10 00000000000000000000000000000000
1
0 1
0 1
0 0
0 00000000000000000000000000000000

2 1
10 1
18446744073709551613 4294967295
11111111111111111111111111111111111111111111111111111111111101
11111111111111111111111111111111
2 1
10 1
0
1

5. Объяснение результатов работы программы.

1) Метод **void read(std::istream & is)** является главной функцией в моей программе, именно она правильно «расскладывает» число в 96 бит. Для этого она использует вспомогательный вектор и с помощью простых операций переводит его на 1 и 0, а дальше первые 32 бита уходят в переменную типа uns int, остальные в uns. long.

2) Методы **void shiftLeft(std::istream& is)**, **void shiftRight(std:: istream& is)** производят левый и правый битовый сдвиг. При левом сдвиге учитывается 32 бит в переменной тип uns. int, а при правом сдвиге учитывается первый бит переменной типа uns. long.

3) **void bit::write_10(std::ostream &os)** и **void bit::write_2(std::ostream &os)** выводят младшую и старшую переменную и битовое представление строки соответственно. Для этого write_2 использует операцию деления на 2.

4) функции включения и равенства помещают двоичную строку в вектор и цикл считает количество единиц и совпадения.

5) Все остальные функции тривиальные и используют готовые битовые операции языка C++.

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий git, а также изучил основы работы с классами в C++. Создал класс, соответствующий варианту моего задания, реализовал его методы.

7. Литература.

1)лекции по ООП МАИ.
2) Г.Шилдт «C++».