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: | Козлов А.Д. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Журавлев А.А. |
| Вариант: | 7 |
| Оценка: |  |
| Дата: |  |

Москва

2019

**1. Код программы на языке C++:**

**bigString.hpp:**

#ifndef BIGSTRING\_HPP

#define BIGSTRING\_HPP

struct BigString{

BigString();

BigString(unsigned long long n1, unsigned long long n2);

~BigString();

void set\_lstring(unsigned long long n);

void set\_rstring(unsigned long long n);

unsigned long long get\_lstring() const;

unsigned long long get\_rstring() const;

void print\_bits();

void shift\_left(int shift);

void shift\_right(int shift);

BigString bs\_and(const BigString& s);

BigString bs\_or(const BigString& s);

BigString bs\_xor(const BigString& s);

BigString bs\_not();

int num\_of\_units() const;

BigString\* comparison(BigString& s);

int is\_include(const BigString& s) const;

private:

unsigned long long lString;

unsigned long long rString;

};

#endif

**bigString.cpp:**

#include <iostream>

#include "bigString.hpp"

BigString::BigString():lString(0), rString(0){}

BigString::BigString(unsigned long long n1, unsigned long long n2):lString(n1), rString(n2){}

BigString::~BigString(){}

void BigString::set\_lstring(unsigned long long n) {

this->lString = n;

}

void BigString::set\_rstring(unsigned long long n) {

this->rString = n;

}

unsigned long long BigString::get\_lstring() const {

return this->lString;

}

unsigned long long BigString::get\_rstring() const {

return this->rString;

}

void BigString::print\_bits() {

unsigned long long Mask\_firstBit = 0x8000000000000000; // первый бит 1 остальные 0

unsigned long long num = this->lString;

for(int i = 0; i < 64; ++i) {

if (num&Mask\_firstBit) {

std::cout << "1";

}

else {

std::cout << "0";

}

num = num << 1;

}

num = this->rString;

for(int i = 0; i < 64; ++i) {

if (num&Mask\_firstBit) {

std::cout << "1";

}

else {

std::cout << "0";

}

num = num << 1;

}

std::cout << "\n";

}

void BigString::shift\_left(int shift) {

unsigned long long Mask\_firstBit = 0x8000000000000000; // первый бит 1 остальные 0

int bit;

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

if(this->rString & Mask\_firstBit) {

bit = 1;

}

else {

bit = 0;

}

this->rString = this->rString << 1;

this->lString = this->lString << 1;

this->lString = this->lString | bit;

}

}

void BigString::shift\_right(int shift) {

unsigned long long Mask\_lastBit = 1; // последний бит 1 остальные 0

int bit;

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

if (this->lString & Mask\_lastBit) {

bit = 1;

}

else {

bit = 0;

}

this->lString = this->lString >> 1;

this->rString = this->rString >> 1;

if (bit) {

this->rString= this->rString | 0x8000000000000000;

}

}

}

BigString BigString::bs\_and(const BigString& s) {

BigString res;

res.set\_lstring(this->lString & s.get\_lstring());

res.set\_lstring(this->rString & s.get\_rstring());

return res;

}

BigString BigString::bs\_or(const BigString& s) {

BigString res;

res.set\_lstring(this->lString | s.get\_lstring());

res.set\_rstring(this->rString | s.get\_rstring());

return res;

}

///

BigString BigString::bs\_xor(const BigString& s) {

BigString res;

res.set\_lstring(this->lString ^ s.get\_lstring());

res.set\_rstring(this->rString ^ s.get\_rstring());

return res;

}

BigString BigString::bs\_not() {

BigString res;

res.set\_lstring(~(this->lString));

res.set\_rstring(~(this->rString));

return res;

}

int BigString::num\_of\_units() const {

unsigned long long mask = 0x8000000000000000;

int counter = 0;

for(int i = 0; i < 64; ++i) {

if (this->lString & mask) {

counter++;

}

mask = mask >> 1;

}

mask = 0x8000000000000000;

for(int i = 0; i < 64; ++i) {

if (this->rString & mask) {

counter++;

}

mask = mask >> 1;

}

return counter;

}

BigString\* BigString::comparison(BigString& s) {

int value1 = this->num\_of\_units();

int value2 = s.num\_of\_units();

if (value1 >= value2) {

return this;

}

else {

return &s;

}

}

int BigString::is\_include(const BigString& s) const {

if(((this->lString & s.get\_lstring()) == this->get\_lstring()) || ((this->rString & s.get\_rstring()) == this->get\_rstring())) {

return 1;

}

else {

return 0;

}

}

**main.cpp:**

#include <iostream>

#include <fstream>

#include "bigString.hpp"

void separator();

void meny();

unsigned long long get\_num\_in\_file(std::ifstream& fin);

int main(int argc, char\* argv[]) {

int key;

int shift;

unsigned long long n1;

unsigned long long n2;

BigString str1;

BigString str2;

if (argc > 1) {

std::ifstream fin(argv[1], std::ios\_base::in);

if (!fin.is\_open()) {

std::cout << "Нельзя отрыть файл!\n";

return 2;

}

n1 = get\_num\_in\_file(fin);

n2 = get\_num\_in\_file(fin);

str1.set\_lstring(n1);

str1.set\_rstring(n2);

n1 = get\_num\_in\_file(fin);

n2 = get\_num\_in\_file(fin);

str2.set\_lstring(n1);

str2.set\_rstring(n2);

fin.close();

}

else {

std::cin >> n1 >> n2;

str1.set\_lstring(n1);

str1.set\_rstring(n2);

std::cin >> n1 >> n2;

str2.set\_lstring(n1);

str2.set\_rstring(n2);

}

meny();

std::cin >> key;

switch(key) {

case 1:

std::cout << "shift: ";

std::cin >> shift;

str1.print\_bits();

str1.shift\_left(shift);

str1.print\_bits();

break;

case 2:

std::cout << "shift: ";

std::cin >> shift;

str1.print\_bits();

str1.shift\_right(shift);

str1.print\_bits();

break;

case 3:

str1.print\_bits();

str2.print\_bits();

str1.bs\_and(str2).print\_bits();

break;

case 4:

str1.print\_bits();

str2.print\_bits();

str1.bs\_or(str2).print\_bits();

break;

case 5:

str1.print\_bits();

str2.print\_bits();

str1.bs\_xor(str2).print\_bits();

break;

case 6:

str1.print\_bits();

str1.bs\_not().print\_bits();

break;

case 7:

str1.print\_bits();

str2.print\_bits();

std::cout << "num\_of\_units :" << str1.num\_of\_units() <<std::endl;

std::cout << "num\_of\_units :" << str2.num\_of\_units() <<std::endl;

break;

case 8:

str1.print\_bits();

str2.print\_bits();

std::cout << "comparison \n";

str1.comparison(str2)->print\_bits();

break;

case 9:

str1.print\_bits();

str2.print\_bits();

if (str1.is\_include(str2)) {

std::cout << "Yes\n";

}

else {

std::cout << "No\n";

}

break;

default:

std::cout << "error" << std::endl;

break;

}

return 0;

}

void separator() {

std::cout <<"==============";

for(int i = 0; i < 128; ++i) {

std::cout << "=";

}

std::cout << std::endl;

}

void meny() {

std::cout << "1)shiftLeft" << std::endl;

std::cout << "2)shiftLeft" << std::endl;

std::cout << "3)AND" << std::endl;

std::cout << "4)OR" << std::endl;

std::cout << "5)XOR" << std::endl;

std::cout << "6)NOT" << std::endl;

std::cout << "7)num\_of\_units" << std::endl;

std::cout << "8)comparison" << std::endl;

std::cout << "9)is\_include" << std::endl;

}

unsigned long long get\_num\_in\_file(std::ifstream& fin) {

unsigned long long num = 0;

char ch;

while(fin.get(ch)) {

if (ch != ' ' && ch != '\n' && ch != EOF) {

num = num \* 10 + (ch - '0');

}

else {

break;

}

}

return num;

}

**CmakeLists.txt:**

cmake\_minimum\_required(VERSION 3.1)

project(oop\_exercise\_01)

add\_executable(oop\_exercise\_01 main.cpp bigString.cpp)

**2. Ссылка на репозиторий на GitHub.**

**https://github.com/ArtemKD/oop\_exercise\_01**

**3. Набор testcases.**

test\_01.txt:

12868426913406380000 5193880475836733000

12918000598641058000 11398999006967146000

test\_02.txt:

11093947366930840000 12809315413885479000

8591448501686567000 7091160512692316000

test\_03.txt:

5647118480700488000 15870949720163300000

13670168416534604000 8221050084544006000

test\_04.txt:

11023188960974280000 10712243251891943000

15260885590755232000 11658364395506348000

test\_05.txt:

17490701296889190000 1907006900119359500

696737765169393700 17953908319343747000

**4. Результаты выполнения тестов.**

test\_01.txt:

=============================================================

str1 10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

str2 10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

str1 and str2 10110010000001011101010000011000000010100000100010100000110000000000100000010000010000000000001001000000000001001000011000000000

=============================================================

str1 10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

str2 10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

str1 or str2 10110011110101011111011100011011000111111110100111100111111100001101111000110101011111111111011111001101010111111111011001011000

=============================================================

str1 10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

str2 10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

str1 xor str2 00000001110100000010001100000011000101011110000101000111001100001101011000100101001111111111010110001101010110110111000001011000

=============================================================

str1 10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

not str1 01001101011010100010100111100110111100000101011000011000000111111011011111101011101000010011100110111111111010100010100110110111

=============================================================

str2 10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

not str2 01001100101110100000101011100101111001011011011101011111001011110110000111001110100111101100110000110010101100010101100111101111

=============================================================

Num of units in str1: 57

Num of units in str2: 56

=============================================================

Max in str1 and str2:

10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

=============================================================

str1 >> 3

00010110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001

=============================================================

str1 << 5

11001010010101110101100001100100001111101010011110011111100000010010000001010001011110110001100100000000010101110101100100100000

=============================================================

test\_02.txt:

=============================================================

str1 10011001111101011001111001011010000110101100101011111001110000001011000111000011110101001000000010010100100101000110100001011000

str2 01110111001110101111010011100100100100010001011100000000010110000110001001101000110111001110101001101100010011010110011101100000

str1 and str2 00010001001100001001010001000000000100000000001000000000010000000010000001000000110101001000000000000100000001000110000001000000

=============================================================

str1 10011001111101011001111001011010000110101100101011111001110000001011000111000011110101001000000010010100100101000110100001011000

str2 01110111001110101111010011100100100100010001011100000000010110000110001001101000110111001110101001101100010011010110011101100000

str1 or str2 11111111111111111111111011111110100110111101111111111001110110001111001111101011110111001110101011111100110111010110111101111000

=============================================================

str1 10011001111101011001111001011010000110101100101011111001110000001011000111000011110101001000000010010100100101000110100001011000

str2 01110111001110101111010011100100100100010001011100000000010110000110001001101000110111001110101001101100010011010110011101100000

str1 xor str2 11101110110011110110101010111110100010111101110111111001100110001101001110101011000010000110101011111000110110010000111100111000

=============================================================

str1 10011001111101011001111001011010000110101100101011111001110000001011000111000011110101001000000010010100100101000110100001011000

not str1 01100110000010100110000110100101111001010011010100000110001111110100111000111100001010110111111101101011011010111001011110100111

=============================================================

str2 01110111001110101111010011100100100100010001011100000000010110000110001001101000110111001110101001101100010011010110011101100000

not str2 10001000110001010000101100011011011011101110100011111111101001111001110110010111001000110001010110010011101100101001100010011111

=============================================================

Num of units in str1: 59

Num of units in str2: 60

=============================================================

Max in str1 and str2:

01110111001110101111010011100100100100010001011100000000010110000110001001101000110111001110101001101100010011010110011101100000

=============================================================

str1 >> 3

00010011001111101011001111001011010000110101100101011111001110000001011000111000011110101001000000010010100100101000110100001011

=============================================================

str1 << 5

01100111110101100111100101101000011010110010101111100111000000101100011100001111010100100000001001010010010100011010000101100000

=============================================================

test\_03.txt:

=============================================================

str1 01001110010111101001100001010110101001101011001000111001010000001101110001000000111100001010111010111110011010110011001010100000

str2 10111101101101100011000110111111010001110100111100101000111000000111001000010111000010010111111010110001111100001100011101110000

str1 and str2 00001100000101100001000000010110000001100000001000101000010000000101000000000000000000000010111010110000011000000000001000100000

=============================================================

str1 01001110010111101001100001010110101001101011001000111001010000001101110001000000111100001010111010111110011010110011001010100000

str2 10111101101101100011000110111111010001110100111100101000111000000111001000010111000010010111111010110001111100001100011101110000

str1 or str2 11111111111111101011100111111111111001111111111100111001111000001111111001010111111110011111111010111111111110111111011111110000

=============================================================

str1 01001110010111101001100001010110101001101011001000111001010000001101110001000000111100001010111010111110011010110011001010100000

str2 10111101101101100011000110111111010001110100111100101000111000000111001000010111000010010111111010110001111100001100011101110000

str1 xor str2 11110011111010001010100111101001111000011111110100010001101000001010111001010111111110011101000000001111100110111111010111010000

=============================================================

str1 01001110010111101001100001010110101001101011001000111001010000001101110001000000111100001010111010111110011010110011001010100000

not str1 10110001101000010110011110101001010110010100110111000110101111110010001110111111000011110101000101000001100101001100110101011111

=============================================================

str2 10111101101101100011000110111111010001110100111100101000111000000111001000010111000010010111111010110001111100001100011101110000

not str2 01000010010010011100111001000000101110001011000011010111000111111000110111101000111101101000000101001110000011110011100010001111

=============================================================

Num of units in str1: 60

Num of units in str2: 67

=============================================================

Max in str1 and str2:

10111101101101100011000110111111010001110100111100101000111000000111001000010111000010010111111010110001111100001100011101110000

=============================================================

str1 >> 3

00001001110010111101001100001010110101001101011001000111001010000001101110001000000111100001010111010111110011010110011001010100

=============================================================

str1 << 5

00111001011110100110000101011010100110101100100011100101000000110111000100000011110000101011101011111001101011001100101010000000

=============================================================

test\_04.txt:

=============================================================

str1 10011000111110100011101111110100111001101110000100010101010000001001010010101001100010000111101101010000100110000011011001011000

str2 11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000

str1 and str2 10010000110010000000101010010100110000101100000100000001000000001000000010001000100000000010001000000000000100000010011001000000

=============================================================

str1 10011000111110100011101111110100111001101110000100010101010000001001010010101001100010000111101101010000100110000011011001011000

str2 11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000

str1 or str2 11011011111110111011111111110110111101101111011111011101010000001011010111101011110111001111111101111000101111111011111111111000

=============================================================

str1 10011000111110100011101111110100111001101110000100010101010000001001010010101001100010000111101101010000100110000011011001011000

str2 11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000

str1 xor str2 01001011001100111011010101100010001101000011011011011100010000000011010101100011010111001101110101111000101011111001100110111000

=============================================================

str1 10011000111110100011101111110100111001101110000100010101010000001001010010101001100010000111101101010000100110000011011001011000

not str1 01100111000001011100010000001011000110010001111011101010101111110110101101010110011101111000010010101111011001111100100110100111

=============================================================

str2 11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000

not str2 00101100001101100111000101101001001011010010100000110110111111110101111000110101001010110101100111010111110010000101000000011111

=============================================================

Num of units in str1: 59

Num of units in str2: 62

=============================================================

Max in str1 and str2:

11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000

=============================================================

str1 >> 3

00010011000111110100011101111110100111001101110000100010101010000001001010010101001100010000111101101010000100110000011011001011

=============================================================

str1 << 5

01100011111010001110111111010011100110111000010001010101000000100101001010100110001000011110110101000010011000001101100101100000

=============================================================

test\_05.txt:

=============================================================

str1 11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100

str2 00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000

str1 and str2 00000000101010110100010000001000010000001000000101010000001000000001100000100001000010000010110000000000010000000001000000001000

=============================================================

str1 11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100

str2 00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000

str1 or str2 11111011101110110111111101101110111110111000111101110110011101001111101101111111000111001111111000011001111111110101111110111100

=============================================================

str1 11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100

str2 00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000

str1 xor str2 11111011000100000011101101100110101110110000111000100110010101001110001101011110000101001101001000011001101111110100111110110100

=============================================================

str1 11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100

not str1 00001101010001001000101111010111000101100111010010101001100011111110010110001000111100110101001111111110001010101010111111110011

=============================================================

str2 00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000

not str2 11110110010101001011000010110001101011010111101010001111110110110000011011010110111001111000000111100111100101011110000001000111

=============================================================

Num of units in str1: 58

Num of units in str2: 59

=============================================================

Max in str1 and str2:

00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000

============================================================

str1 >> 3

00011110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001

============================================================

str1 << 5

11001010111011011101000010100011101001100010110101011001110000000110100111011100001100101011000000000111010101010100000000100000

============================================================

**5. Объяснение результатов работы программы.**

1) При запуске программы oop\_exercise\_01 с аргументами test\_??.txt объекты srt1, str2 получают данные из файлов test\_??.test.

2) Программа выполняет операции: and, or, xor, not, для строк полученных из тестов.

3) Побитовый сдвиг строк str1 и str2 вычисляется с помощью функций shiftLeft и shiftRight, затем выводится в стандартный поток вывода с помощью функции print\_bits.

4) Для строк srt1 и str2 вычисляется результат битовых операция(and, or, xor, not) с помощью функций AND, OR, XOR, NOT; результат выводится с помощью функции print\_bits.

5) Вычисление количества единиц в строке осуществляется с помощью функции num\_of\_units; функция возвращает значение типа int и выводится в стандартный вывод с помощью std::cout.

6) Сравнение строк по количеству единиц осуществляется с помощью функции comparison, которая использует предыдущую функцию num\_of\_units; функция возвращает указатель на объект класса BigString, затем с помощью метода класса print\_bits выводим слово с наибольшим количеством едининиц в стандартный вывод.

7) Проверка включения осуществляется с помощью функции is\_include.

**6. Вывод.**

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий GitHub, а также изучил основы работы с классами в C++. Создал класс, соответствующий варианту моего задания, реализовал для него битовые операции: «И»(and), «ИЛИ»(or), «ОТРИЦАНИЕ»(not), «Исключающее или»(xor), а также операции побитового сдвига(shiftLeft и shiftRight).