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 shiftLeft(int shift);
    void shiftRight(int shift);
    void print_bits(); //вспомогательная функция для вывода побитово строки

    BigString BS_and(const BigString& s2); //возвращает результат побитовгого и(&) для двух строк
    BigString BS_or(const BigString& s2); //возвращает результат побитовгого или(|) для двух строк
    BigString BS_xor(const BigString& s2); //возвращает результат исключающего или для двух строк
    BigString BS_not(); //возвращает битовую инверсию

    int numof_units(); //вернет кол-во единиц в строке
  BigString* comparison(BigString& s2); // вернет указатель на стрку с наибольшим кол-вом единиц

private:
    unsigned long long lString; //левое поле строки
    unsigned long long rString; //правое поле строки
};

#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::shiftLeft(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::shiftRight(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& s2)
{
    BigString res(0, 0);
    unsigned long long value;
    unsigned long long mask = 0x8000000000000000;
    for(int i = 0; i < 64; ++i)
```

```
    {
        value = (this->lString & s2.lString) & mask;
        if (value)
        {
            res.lString = res.lString | value;
        }
        mask = mask >> 1;
    }
    mask = 0x8000000000000000;
    for(int i = 0; i < 64; ++i)
    {
        value = (this->rString & s2.rString) & mask;
        if (value)
        {
            res.rString = res.rString | value;
        }
        mask = mask >> 1;
    }
    return res;
}

BigString BigString::BS_or(const BigString& s2)
{
    BigString res(0, 0);
    unsigned long long mask = 0x8000000000000000;
    unsigned long long value;

    for(int i = 0; i < 64; ++i)
    {
        value = (this->lString | s2.lString) & mask;
        if (value)
        {
            res.lString = res.lString | value;
        }
        mask = mask >> 1;
    }
    mask = 0x8000000000000000;
    for(int i = 0; i < 64; ++i)
    {
        value = (this->rString | s2.rString) & mask;
        if (value)
        {
            res.rString = res.rString | value;
        }
        mask = mask >> 1;
```

```
    }
    return res;
}

BigString BigString::BS_xor(const BigString& s2)
{
    BigString res(0, 0);
    unsigned long long mask = 0x8000000000000000;
    unsigned long long value1;
    unsigned long long value2;
    for(int i = 0; i < 64; ++i)
    {
        value1 = this->lString & mask;
        value2 = s2.lString & mask;
        if ((value1 != 0 && value2 == 0) || (value1 == 0 && value2 != 0))
        {
            res.lString = res.lString | mask;
        }
        mask = mask >> 1;
    }
    mask = 0x8000000000000000;
    for(int i = 0; i < 64; ++i)
    {
        value1 = this->rString & mask;
        value2 = s2.rString & mask;
        if ((value1 != 0 && value2 == 0) || (value1 == 0 && value2 != 0))
        {
            res.rString = res.rString | mask;
        }
        mask = mask >> 1;
    }
    return res;
}

BigString BigString::BS_not()
{
    BigString res(0, 0);
    unsigned long long mask = 0x8000000000000000;
    unsigned long long value;

    for(int i = 0; i < 64; ++i)
    {
        value = this->lString & mask;
        if (!value)
        {
```

```
            res.lString = res.lString | mask;
        }
        mask = mask >> 1;
    }
    mask = 0x8000000000000000;
    for(int i = 0; i < 64; ++i)
    {
        value = this->rString & mask;
        if (!value)
        {
            res.rString = res.rString | mask;
        }
        mask = mask >> 1;
    }
    return res;
}

int BigString::numof_units()
{
    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& s2)
{
    int value1 = this->numof_units();
    int value2 = s2.numof_units();
```

```
    if (value1 >= value2)
    {
        return this;
    }
    else
    {
        return &s2;
    }
}

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";
}
```

main.cpp:

```
#include <iostream>
#include <fstream>
#include "bigString.hpp"

void separator()
{
   std::cout <<"==============";
   for(int i = 0; i < 128; ++i)
   {
      std::cout << "=";
   }
   std::cout << std::endl;
}

unsigned long long get_numfile(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;
}

int main(int argc, char* argv[])
{
   unsigned long long n1;
   unsigned long long n2;

   if (argc < 2)
   {
      return 1;
   }

   std::ifstream fin(argv[1], std::ios_base::in);
```

```
if (!fin.is_open())
{
   std::cout << "Нельзя отрыть файл!\n";
   return 2;
}

n1 = get_numfile(fin);
n2 = get_numfile(fin);
BigString str1(n1, n2);

n1 = get_numfile(fin);
n2 = get_numfile(fin);
BigString str2(n1, n2);

separator();
std::cout << "str1          ";
str1.print_bits();
std::cout << "str2          ";
str2.print_bits();
std::cout << "str1 and str2 ";
str1.BS_and(str2).print_bits();
separator();
std::cout << "str1          ";
str1.print_bits();
std::cout << "str2          ";
str2.print_bits();
std::cout << "str1 or str2  ";
str1.BS_or(str2).print_bits();
separator();
std::cout << "str1          ";
str1.print_bits();
std::cout << "str2          ";
str2.print_bits();
std::cout << "str1 xor str2 ";
str1.BS_xor(str2).print_bits();
separator();
std::cout << "str1          ";
str1.print_bits();
std::cout << "not str1      ";
str1.BS_not().print_bits();
separator();
std::cout << "str2          ";
str2.print_bits();
std::cout << "not str2      ";
str2.BS_not().print_bits();
```

```
    separator();
    std::cout << "Num of units in str1: " << str1.numof_units() << std::endl;
    std::cout << "Num of units in str2: " << str2.numof_units() << std::endl;
    separator();
    BigString* p = str1.comparison(str2);
    std::cout << "Max in str1 and str2:\n";
    p->print_bits();

    fin.close();
    return 0;
}
```

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

10110010000001011101010000011000000010100000100010100000110000000000100000010000000000000100100000000000010010000011000000000

==========

str1

10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

str2

10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

str1 or str2

10110011110101011111011100011011000111111110100111100111111000011011110001101010111111111110111110011010101111111110111001011000

==========

str1

10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

str2

10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

str1 xor str2

00000001110100000010001100000011000101011110000101000111001100001101011000110001001010011111111110101100011010101101101110000010110000

==========

str1

10110010100101011101011000011001000011111010100111100111111000000100100000010100010111101100011001000000000101011101011001001000

not str1

01001101011010100010100111100110111100000101011000011000000111111011011111101011101000010011100110111111111010100010100110110111

==========

str2

10110011010001011111010100011010000110100100100010100000110100001001111000110001011000010011001111001101010011101010011000010000

not str2
0100110010111010000010101110010111100101101110101111100101111011000
0111001110100111101100110000110010101100010101100111101111
==========
Num of units in str1: 57
Num of units in str2: 56
==========
Max in str1 and str2:
1011001010010101110101100001100100001111101010011110011111100000010010
0000010100010111101100011001000000000101011101011001001000
==========
str1 >> 3
0001011001010010101110101100001100100001111101010011110011111100000010
0100000010100010111101100011001000000000101011101011001001
==========
str1 << 5
1100101001010111010110000110010000111110101001111001111110000001001000
0001010001011110110001100100000000010101110101100100100000
==========

test_02.txt:
===
str1
1001100111110101100111100101101000011010110010101111100111000000101100
011100001111010100100000001001010010010100011010000101100
str2
0111011100111010111101001110010010010001000101110000000000101100001100
010011010001101110011101010011011000100110101100111011000000
str1 and str2
0001000100110000100101000100000000010000000001000000000010000000010
000001000000110101001000000000000100000001000110000001000000
===
str1
1001100111110101100111100101101000011010110010101111100111000000101100
011100001111010100100000001001010010010100011010000101100
str2
0111011100111010111101001110010010010001000101110000000000101100001100
010011010001101110011101010011011000100110101100111011000000
str1 or str2
1111111111111111111111011111110100110111101111111111001110110001111001
111010111101110011101010111111001101110101101111101111000
===
str1
1001100111110101100111100101101000011010110010101111100111000000101100
011100001111010100100000001001010010010100011010000101100
str2
0111011100111010111101001110010010010001000101110000000000101100001100
010011010001101110011101010011011000100110101100111011000000
str1 xor str2
1110111011001111011010101011111010001011110111011111100110011000110100
111010101100001000011010101111110001101100100001111001110000
===
str1
1001100111110101100111100101101000011010110010101111100111000000101100
011100001111010100100000001001010010010100011010000101100
not str1
0110011000001010011000011010010111100101001101010000011000111111010011
100011110000101011011111110110101101101011100101111010011
===
str2
0111011100111010111101001110010010010001000101110000000000101100001100
010011010001101110011101010011011000100110101100111011000000
not str2
1000100011000101000010110001101101101110111010001111111110100111100111
011001011100100011000101011001001110110010100110001001111

==========
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
0100111001011110100110000101011010100110101100100011100101000000110111000100000011110000101011101011110011010110011001010100000
str2
1011110110110110001100011011111101000111010011110010100011100000011100100001011100001001011111101011000111110000110001110111000
str1 and str2
0000110000010110000100000001011000000110000000100010100001000000010100000000000000000000010111010110000011000000000001000100000
==========
str1
0100111001011110100110000101011010100110101100100011100101000000110111000100000011110000101011101011110011010110011001010100000
str2
1011110110110110001100011011111101000111010011110010100011100000011100100001011100001001011111101011000111110000110001110111000
str1 or str2
1111111111111110101110011111111111100111111111110011100111100000111111100101011111111001111111101011111111111010111111011111110000
==========
str1
0100111001011110100110000101011010100110101100100011100101000000110111000100000011110000101011101011110011010110011001010100000
str2
1011110110110110001100011011111101000111010011110010100011100000011100100001011100001001011111101011000111110000110001110111000
str1 xor str2
1111001111101000101010011110100111100001111111010001000110100000101011100101111111100111010000000011111001101111110101110100000
==========
str1
0100111001011110100110000101011010100110101100100011100101000000110111000100000011110000101011101011110011010110011001010100000
not str1
1011000110100001011001111010100101011001010011011100011010111111001000111011111000011110101000101000001100101001100110101011111
==========
str2
1011110110110110001100011011111101000111010011110010100011100000011100100001011100001001011111101011000111110000110001110111000
not str2
0100001001001001110011100100000010111000101100001101011100011111110001101111000010010111111010000001010011100000111100111000100011111

==========
Num of units in str1: 60
Num of units in str2: 67
==========
Max in str1 and str2:
1011110110110110001100011011111101000111010011110010100011100000011100
1000010111000010010111111010110001111100001100011101110000
==========
str1 >> 3
0000100111001011110100110000101011010100110101100100011100101000000011
0111000100000011110000101011101011110011010110011001010100
==========
str1 << 5
0011100101111010011000010101101010011010110010001110010100000011011100
0010000001111000010101110101111100110101100110010101000000
==========

test_04.txt:
==========
str1
10011000111110100011101111110100111001101110000100010101010000001001010010100110001000011110110101000010011000001101100101100
str2
11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000011011110101111110000
str1 and str2
10010000110010000000101010010100110000101100000100000001000000001000000010001000100000000010001000000000001000000010011001000000
==========
str1
10011000111110100011101111110100111001101110000100010101010000001001010010100110001000011110110101000010011000001101100101100
str2
11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000011011110101111110000
str1 or str2
11011011111110111011111111111011011110110111101111101110101000000101101011111010111011100111111110111100010111111101111111111000
==========
str1
10011000111110100011101111110100111001101110000100010101010000001001010010100110001000011110110101000010011000001101100101100
str2
11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000011011110101111110000
str1 xor str2
01001011001100111011010101100010001101000011011011011100010000000011010110001101011100110111010111100010101111100110011011000
==========
str1
10011000111110100011101111110100111001101110000100010101010000001001010010100110001000011110110101000010011000001101100101100
not str1
01100111000001011100010000001011000110010001111011101010101111110110101101001110111100001001010111101100111110010011010011
==========
str2
11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000011011110101111110000
not str2
00101100001101100111000101101001001011010010100000110110111111110101111000110100101011010110011101011111001000010100000011111

==========
Num of units in str1: 59
Num of units in str2: 62
==========
Max in str1 and str2:
11010011110010011000111010010110110100101101011111001001000000001010000111001010110101001010011000101000001101111010111111100000
==========
str1 >> 3
00010011000111110100011101111110100111001101110000100010101010000001001010010100110001000011110110101000010011000001101100101 1
==========
str1 << 5
01100011111010001110111111010011100110111000010001010101000000100101001010010011000100001111011010100001001100000110110010110 0000
==========

test_05.txt:
==========
str1
11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100
str2
00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000
str1 and str2
00000000101010110100010000001000010000001000000101010000001000000011000010000100001000001011000000000010000000001000000001000
==========
str1
11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100
str2
00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000
str1 or str2
11111011101110110111111110110111011111011100011110111011001110100111110110111101110011000111111010111111000111001111111000011001111111110101111110111100
==========
str1
11110010101110110111010000101000111010011000101101010110011100000001101001110111000011001010110000000001110101010101000000001100
str2
00001001101010110100111101001110010100101000010101110000001001001111100100101001000110000111111000011000011010100001111110111000
str1 xor str2
11111011000100000011101101100110101110110000111000100110010101001110001101011100001010011010010000110011011111101001111101101100
==========
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:
000010011010101101001111010011100101001010000101011100000010010011111
001001010010001100001111110000110000110101000011111110111000
==========
str1 >> 3
000111100101011101101110100001010001110100110001011010101100111000000
011010011101110000110010101100000000011101010101010000000001
==========
str1 << 5
110010101110110111010000101000111010011000101101010110011100000001101
001110111000011001010110000000001110101010101000000000100000
==========

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 BS_and, BS_or, BS_xor, BS_not; результат выводится с помощью функции print_bits.
5) Вычисление количества единиц в строке осуществляется с помощью функции numof_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 numof_units; функция возвращает указатель на объект класса BigString, затем с помощью метода класса print_bits выводим слово с наибольшим количеством едининиц в стандартный вывод.

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).