

Московский Авиационный Институт
(Национальный Исследовательский Университет)

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

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
I семестр
Задание 1: «Простые классы»

Группа:	М8О-208Б-18, №6
Студент:	Хитриков Артемий Юрьевич
Преподаватель:	Журавлёв Андрей Андреевич
Оценка:	
Дата:	30.09.2019

Москва, 2019

1. Задание

(вариант № 6): Создать класс *BitString* для работы с 96-битовыми строками. Битовая строка должна быть представлена двумя полями: старшая часть `unsigned long long (uint64_t)`, младшая часть `unsigned int (uint32_t)`. 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.

2. Адрес репозитория на GitHub

https://github.com/DeZellt/oop_exercise_01

3. Код программы на C++

lab1.cpp

```
#include <iostream>
#include <cstdint>
#include "bitstring.h"
```

```
int main() {
```

```
    uint32_t first;
    uint64_t second;
    uint32_t first2;
    uint64_t second2;
    const uint32_t bitleft = 3;
    const uint32_t bitright = 5;
    uint32_t incfirst;
    uint64_t incsecond;
```

```
    std::cout << "Введите данные в следующем формате: *32-битовая строка* +
*64-битовая строка*.\n";
    std::cin >> first >> second;
    bitstring a{first, second};
    std::cout << "Введите данные для второй строки в следующем формате: *32-
битовая строка* + *64-битовая строка*.\n";
    std::cin >> first2 >> second2;
```

```
bitstring f{first2};
bitstring s{0, second2};
bitstring test{first2, second2};
bitstring zero{};
```

```
std::cout << "Сейчас будут выведены следующие строки через отступ - " <<
first2 << " 0, 0 " << second2 << ", 0 0." << std::endl;
f.print();
s.print();
zero.print();
```

```
std::cout << "Сейчас будут проведены стандартные битовые операции с
вашей строкой." << std::endl;
```

```
std::cout << a.getF() << " " << a.getS() << " & " << test.getF() << " " <<
test.getS() << " = ";
```

```
(a.And(test)).print();
```

```
std::cout << a.getF() << " " << a.getS() << " | " << test.getF() << " " <<
test.getS() << " = ";
```

```
(a.Or(test)).print();
```

```
std::cout << a.getF() << " " << a.getS() << " ^ " << test.getF() << " " <<
test.getS() << " = ";
```

```
(a.Xor(test)).print();
```

```
std::cout << "~(" << a.getF() << " " << a.getS() << ") = ";
```

```
(a.Not()).print();
```

```
std::cout << "Сейчас будет проведен сдвиг на " << bitleft << " битов влево и
на " << bitright << "битов вправо. После сдвига будет печататься состояние
строки." << std::endl;
```

```
a.shiftLeft(bitleft);
```

```
a.print();
```

```
a.shiftRight(bitright);
```

```
a.print();
```

```
std::cout << "А теперь проверим еще функции. Для того, чтобы проверить
функцию включения, введите еще одно число в следующем формате: *32-
битовая строка* + *64-битовая строка*." << std::endl;
```

```
std::cin >> incfirst >> incsecond;
```

```
bitstring inc{incfirst, incsecond};
```

```
std::cout << "Кол-во единиц равно " << a.countone() << std::endl;
```

```
std::cout << a.getF() << " " << a.getS() << " == " << inc.getF() << " " <<
inc.getS() << "? Это " << (a.equal(inc) ? "правда" : "неправда") << std::endl;
```

```
std::cout << "В " << a.getF() << " " << a.getS() << " включено " << inc.getF()
<< " " << inc.getS() << "? Это " << (a.include(inc) ? "правда" : "неправда") <<
std::endl;
```

```
return 0;
```

```
}
```

```
#ifndef D_BITSTRING_H_
#define D_BITSTRIGN_H_ 1
```

```
#include <iostream>
#include <cstdint>
```

```
class bitstring {
```

```
private:
```

```
    uint32_t m_first;
    uint64_t m_second;
    void shiftLeftOne();
    void shiftRightOne();
    int32_t findOne() const;
    int32_t findTwo() const;
    int32_t length() const;
```

```
public:
```

```
    bitstring(uint32_t first=0, uint64_t second=0);
    bitstring And(const bitstring& r) const;
    bitstring Or(const bitstring& r) const;
    bitstring Xor(const bitstring& r) const;
    bitstring Not() const;
    void shiftLeft(int32_t i);
    void shiftRight(int32_t i);
    int32_t countone() const;
    bool equal(const bitstring& r) const;
    bool include(const bitstring& r) const;
    uint64_t getS() const;
    uint32_t getF() const;
    void print();
```

```
};
```

```
#endif
```

```
bitstring.cpp
```

```
#include "bitstring.h"
#include <iostream>
#include <cstdint>
```

```
bitstring::bitstring(uint32_t first, uint64_t second) {
    m_first = first;
```

```

    m_second = second;
}

int32_t bitstring::findOne() const { // ищет длину второго слова
    int32_t secondlen = 0;
    for (int32_t i = 0; i < 64; i++) if (((m_second >> i) & 1) == 1) secondlen = i;
    return ++secondlen;
}

int32_t bitstring::findTwo() const {
    int32_t firstlen = 0;
    for (int32_t i = 0; i < 32; i++) if (((m_first >> i) & 1) == 1) firstlen = i;
    return ++firstlen;
}

int32_t bitstring::length() const {return (findOne() + findTwo());}

bitstring bitstring::And(const bitstring& r) const { // побитовое "И"
    bitstring result{};
    result.m_first = m_first & r.m_first;
    result.m_second = m_second & r.m_second;
    return result;
}

bitstring bitstring::Or(const bitstring& r) const { // побитовое "ИЛИ"
    bitstring result{};
    result.m_first = m_first | r.m_first;
    result.m_second = m_second | r.m_second;
    return result;
}

bitstring bitstring::Xor(const bitstring& r) const { // побитовое "ИСКЛ ИЛИ"
    bitstring result{};
    result.m_first = m_first ^ r.m_first;
    result.m_second = m_second ^ r.m_second;
    return result;
}

bitstring bitstring::Not() const { // побитовое "НЕ"
    bitstring result{};
    result.m_first = ~m_first;
    result.m_second = ~m_second;
    return result;
}

```

```

void bitstring::shiftLeft(int32_t i) { // сдвиг на произвольное кол-во бит влево
    for (int32_t k=0; k<i; k++) shiftLeftOne();
}

void bitstring::shiftLeftOne() { // сдвиг влево на один бит
    const uint64_t max = 0x80000000;
    m_first <<= 1;
    if((m_second & max) == max) m_first++;
    m_second <<= 1;
}

void bitstring::shiftRight(int32_t i) { // сдвиг вправо на i бит
    for (int32_t k=0; k<i; k++) shiftRightOne();
}

void bitstring::shiftRightOne() { // сдвиг вправо на один бит
    m_second >>= 1;
    if((m_first & 1) == 1) m_second += 0x80000000;
    m_first >>= 1;
}

int32_t bitstring::countone() const { // считает кол-во единиц
    int32_t count = 0;
    bitstring temp{m_first, m_second};
    for (int32_t i = 0; i < 96; i++) {
        if ((temp.getS() & 1) == 1) count++;
        temp.shiftRight(1);
    }
    return count;
}

bool bitstring::equal(const bitstring& r) const {return(countone() ==
r.countone());}

bool bitstring::include(const bitstring& r) const {return (((m_first & r.getF()) ==
r.getF()) && ((m_second & r.getS()) == r.getS()));}

uint64_t bitstring::getS() const {return m_second;}
uint32_t bitstring::getF() const {return m_first;}
void bitstring::print() {std::cout << "Строка равна " << getF() << " " << getS() <<
std::endl;}

```

```
cmake_minimum_required(VERSION 3.10.2)
project(lab1)
set(CMAKE_CXX_STANDARD 14)
add_executable(lab1
    lab1.cpp
    bitstring.cpp)

set(CMAKE_CXX_FLAGS
    "${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

4. Входные данные

Входные данные должны удовлетворять условиям, заданным лабораторной работе. Для данного варианта входные данные – две пары чисел, у обеих пар первое число формата unsigned int, второе unsigned long long.

Test01.txt

```
31 15
15 31
43 17
```

Test02.txt

```
0 1
13 169
0 0
```

Test03.txt

```
478 12839
1223 83829
119 1
```

5. Результаты выполнения тестов

Result_test01.txt

Введите данные в следующем формате: *32-битовая строка* + *64-битовая строка*.

```
31 15
```

Введите данные для второй строки в следующем формате: *32-битовая строка* + *64-битовая строка*.

15 31

Сейчас будут выведены следующие строки через отступ - 15 0, 0 31, 0 0.

Строка равна 15 0

Строка равна 0 31

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

$31\ 15 \& 15\ 31 =$ Строка равна 15 15

$31\ 15 | 15\ 31 =$ Строка равна 31 31

$31\ 15 \wedge 15\ 31 =$ Строка равна 16 16

$\sim(31\ 15) =$ Строка равна 4294967264 18446744073709551600

Сейчас будет проведен сдвиг на 3 бита влево и на 5 битов вправо. После сдвига будет печататься состояние строки.

Строка равна 248 120

Строка равна 7 3221225475

А теперь проверим еще функции. Для того, чтобы проверить функцию включения, введите еще одно число в следующем формате: *32-битовая строка* + *64-битовая строка*.

43 17

Кол-во единиц равно 7

$7\ 3221225475 == 43\ 17?$ Это неправда

В $7\ 3221225475$ включено 43 17? Это неправда

Result_test02.txt

Введите данные в следующем формате: *32-битовая строка* + *64-битовая строка*.

0 1

Введите данные для второй строки в следующем формате: *32-битовая строка* + *64-битовая строка*.

13 169

Сейчас будут выведены следующие строки через отступ - 13 0, 0 169, 0 0.

Строка равна 13 0

Строка равна 0 169

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

$0\ 1 \& 13\ 169 =$ Строка равна 0 1

$0\ 1 | 13\ 169 =$ Строка равна 13 169

$0\ 1 \wedge 13\ 169 =$ Строка равна 13 168

$\sim(0\ 1) =$ Строка равна 4294967295 18446744073709551614

Сейчас будет проведен сдвиг на 3 бита влево и на 5 битов вправо. После сдвига будет печататься состояние строки.

Строка равна 0 8

Строка равна 0 0

А теперь проверим еще функции. Для того, чтобы проверить функцию включения, введите еще одно число в следующем формате: *32-битовая строка* + *64-битовая строка*.

0 0

Кол-во единиц равно 0

0 0 == 0 0? Это правда

В 0 0 включено 0 0? Это правда

Result_test03.txt

Введите данные в следующем формате: *32-битовая строка* + *64-битовая строка*.

478 12839

Введите данные для второй строки в следующем формате: *32-битовая строка* + *64-битовая строка*.

1223 83829

Сейчас будут выведены следующие строки через отступ - 1223 0, 0 83829, 0 0.

Строка равна 1223 0

Строка равна 0 83829

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

478 12839 & 1223 83829 = Строка равна 198 549

478 12839 | 1223 83829 = Строка равна 1503 96119

478 12839 ^ 1223 83829 = Строка равна 1305 95570

~(478 12839) = Строка равна 4294966817 18446744073709538776

Сейчас будет проведен сдвиг на 3 бита влево и на 5 битов вправо. После сдвига будет печататься состояние строки.

Строка равна 3824 102712

Строка равна 119 2147486857

А теперь проверим еще функции. Для того, чтобы проверить функцию включения, введите еще одно число в следующем формате: *32-битовая строка* + *64-битовая строка*.

119 1

Кол-во единиц равно 12

119 2147486857 == 119 1? Это неправда

В 119 2147486857 включено 119 1? Это правда

6. Вывод

Классы — удобный пользовательский тип данных, позволяющий хранить не только данные, но и способы работы с ними. Особенностью классов является инкапсуляция методов и объектов — ее наличие делает работу программы более стабильной.