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.

1. **Адрес репозитория на GitHub**

<https://github.com/DeZellt/oop_exercise_01>

1. **Код программы на С++**

lab1.cpp

#include <iostream>

#include <cstdint>

#include "bitstring.h"

int main() {

uint32\_t first;

uint64\_t second;

uint32\_t onlyfirst = 0x1234;

uint64\_t onlysecond = 0x12345678;

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};

bitstring f{onlyfirst};

bitstring s{0 , onlysecond};

bitstring test{onlyfirst, onlysecond};

bitstring zero{};

std::cout << "Сейчас будут выведены следующие строки через отступ - " << onlyfirst << " 0, 0 " << onlysecond << ", 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

#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) == 1) 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 { // проверяет равенство по кол-ву единиц

if (countone() == r.countone()) return true;

else return false;

}

bool bitstring::include(const bitstring& r) const {

if (((m\_first & r.getF()) == r.getF()) && ((m\_second & r.getS()) == r.getS())) return true;

return false;

}

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;}

CMakeLists.txt

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")

# **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

7 15

Test02.txt

31 15

7 0

Test03.txt

1 1

0 3

1. **Результаты выполнения тестов**

Result\_test01.txt

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

31 15

Сейчас будут выведены следующие строки через отступ - 4660 0, 0 305419896, 0 0.

Строка равна 4660 0

Строка равна 0 305419896

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

31 15 & 4660 305419896 = Строка равна 20 8

31 15 | 4660 305419896 = Строка равна 4671 305419903

31 15 ^ 4660 305419896 = Строка равна 4651 305419895

~(31 15) = Строка равна 4294967264 18446744073709551600

Сейчас будет проведен сдвиг на 3 битов влево и на 5битов 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\*.

7 15

Кол-во единиц равно 7

7 3221225475 == 7 15? Это правда

В 7 3221225475 включено 7 15? Это неправда

Result\_test02.txt

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

31 15

Сейчас будут выведены следующие строки через отступ - 4660 0, 0 305419896, 0 0.

Строка равна 4660 0

Строка равна 0 305419896

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

31 15 & 4660 305419896 = Строка равна 20 8

31 15 | 4660 305419896 = Строка равна 4671 305419903

31 15 ^ 4660 305419896 = Строка равна 4651 305419895

~(31 15) = Строка равна 4294967264 18446744073709551600

Сейчас будет проведен сдвиг на 3 битов влево и на 5битов 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\*.

7 0

Кол-во единиц равно 7

7 3221225475 == 7 0? Это неправда

В 7 3221225475 включено 7 0? Это правда

Result\_test03.txt

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

1 1

Сейчас будут выведены следующие строки через отступ - 4660 0, 0 305419896, 0 0.

Строка равна 4660 0

Строка равна 0 305419896

Строка равна 0 0

Сейчас будут проведены стандартные битовые операции с вашей строкой.

1 1 & 4660 305419896 = Строка равна 0 0

1 1 | 4660 305419896 = Строка равна 4661 305419897

1 1 ^ 4660 305419896 = Строка равна 4661 305419897

~(1 1) = Строка равна 4294967294 18446744073709551614

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

Строка равна 8 8

Строка равна 0 1073741824

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 3

Кол-во единиц равно 1

0 1073741824 == 0 3? Это неправда

В 0 1073741824 включено 0 3? Это неправда

1. **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.