Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 2: «Операторы, литералы»

|  |  |
| --- | --- |
| Группа: | М8О-108Б-18, №6 |
| Студент: | Васильева Василиса Евгеньевна |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 14.10.2019 |

Москва, 2019

1. **Задание**

Создать класс BitString для работы с 96-битовыми строками. Битовая строка должна быть представлена двумя полями: старшая часть unsigned long long, младшая часть unsigned int. 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. Операции and, or, xor, not, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа BitString.

1. **Адрес репозитория на GitHub**

https://github.com/vasilisavasileva/oop\_excercise\_2

1. **Код программы на С++**

*main.cpp*

#include <iostream>

#include<locale>

#include "Bitstring.h"

int main() {

setlocale(LC\_ALL, "rus");

uint64\_t a;

uint32\_t b;

int m, n;

bool l;

std::cout << "Введите значения строки 1\n";

std::cin >> a >> b;

Bitstring BS1 = Bitstring(a, b);

std::cout << "Введите значения строки 2\n";

std::cin >> a >> b;

Bitstring BS2 = Bitstring(a, b);

std::cout << "Введите количество битов для сдвига\n";

std::cin >> n;

Bitstring BS3;

std::cout << "Первая строка\n";

std::cout<<BS1<<std::endl;

std::cout << "Вторая строка\n";

std::cout<<BS2<<std::endl;

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

BS3 = BS1 & BS2;

std::cout<<BS3<<std::endl;

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

BS3 = BS1 | BS2;

std::cout<<BS3<<std::endl;

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

BS3 = BS1 ^ BS2;

std::cout<<BS3<<std::endl;

std::cout << "not\n";

BS3 = ~BS1;

std::cout<<BS3<<std::endl;

std::cout << "BS1 shiftleft\n";

BS1 = BS1 << n;

std::cout<<BS1<<std::endl;

std::cout << "BS2 shiftleft\n";

BS2 = BS2 << n;

std::cout<<BS2<<std::endl;

std::cout << "BS1 shiftRight\n";

BS1 = BS1 >> n;

std::cout<<BS1<<std::endl;

std::cout << "BS2 shiftRight\n";

BS2 = BS2 >> n;

std::cout<<BS2<<std::endl;

std::cout << "count units BS1\n";

m = BS1.counter();

std::cout << m << std::endl;

std::cout << "count units BS2\n";

m = BS2.counter();

std::cout << m << std::endl;

std::cout << "comparing units\n";

l = BS1.compare(BS2);

std::cout << l << std::endl;

std::cout << "includes BS1 BS2\n";

l = BS1.includes(BS2);

std::cout << l << std::endl;

//--------------123456789-123456789-123456789-123

Bitstring BS4 = 10000000000000000000000000000000\_bs;

std::cout<<BS4<<std::endl;

return 0;

}

*Bitstring.h*

*#ifndef BITSTRING\_H*

#define BITSTRING\_H

#include <inttypes.h>

#include <iostream>

class Bitstring {

private:

uint64\_t b1;

uint32\_t b2;

public:

Bitstring();

Bitstring(uint64\_t b1, uint32\_t b2);

friend Bitstring operator& (Bitstring& a, Bitstring& b);

friend Bitstring operator| (Bitstring& a, Bitstring& b);

friend Bitstring operator^ (Bitstring& a, Bitstring& b);

friend Bitstring operator~ (Bitstring& a);

friend Bitstring operator<< (Bitstring& l, int m);

friend Bitstring operator>> (Bitstring& l, int m);

int counter();

friend bool operator< (Bitstring& a, Bitstring& b);

friend bool operator> (Bitstring& a, Bitstring& b);

friend bool operator== (Bitstring& a, Bitstring& b);

bool compare(Bitstring bs2);

bool includes(Bitstring bs2);

void print();

friend std::ostream& operator<<(std::ostream& out, Bitstring& Bs);

};

Bitstring operator "" \_bs(const char\* str);

#endif

*Bitstring.cpp*

*#include "Bitstring.h"*

#include <iostream>

#include <inttypes.h>

Bitstring::Bitstring() {

this->b1 = 0;

this->b2 = 0;

}

Bitstring::Bitstring(uint64\_t b1, uint32\_t b2) {

this->b1 = b1;

this->b2 = b2;

}

Bitstring operator&(Bitstring& a, Bitstring& b) {

Bitstring bs3{ (a.b1)&(b.b1),(a.b2)&(b.b2) };

return bs3;

}

Bitstring operator|(Bitstring& a, Bitstring& b) {

Bitstring bs3 = Bitstring( (a.b1) | (b.b1),(a.b2) | (b.b2) );

return bs3;

}

Bitstring operator^ (Bitstring& a, Bitstring& b) {

Bitstring bs3{ (a.b1) ^ (b.b1),(a.b2) ^ (b.b2) };

return bs3;

}

Bitstring operator~ (Bitstring& a) {

Bitstring bs3{ ~(a.b1),~(a.b2) };

return bs3;

}

Bitstring operator<<(Bitstring& l, int m) {

uint32\_t a;

a = 1;

a <<= 31;

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

if (((l.b2)&a) > 0) {

l.b1 <<= 1;

l.b2 <<= 1;

l.b1 = l.b1 + 1;

}

else {

l.b1 <<= 1;

l.b2 <<= 1;

}

}

return l;

}

Bitstring operator>>(Bitstring& l, int m) {

uint64\_t a;

uint32\_t b;

b = 1;

b <<= 31;

a = 1;

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

if (((l.b1)&a) > 0) {

l.b1 >>= 1;

l.b2 >>= 1;

l.b2 = l.b2 + b;

}

else {

l.b1 >>= 1;

l.b2 >>= 1;

}

}

return l;

}

int Bitstring::counter() {

uint64\_t a = 1;

uint32\_t b = 1;

uint64\_t l;

uint32\_t l1;

int count = 0;

for (int i = 0; i < 63; i++) {

l = (this->b1)&a;

if (l != 0) {

++count;

}

a <<= 1;

}

for (int i = 0; i < 32; i++) {

l1 = (this->b2)&b;

if (l1 != 0) {

++count;

}

b <<= 1;

}

return count;

}

bool Bitstring::compare(Bitstring bs2) {

int a = this->counter();

int b = bs2.counter();

if (a == b)

return true;

return false;

}

bool Bitstring::includes(Bitstring BS2) {

if (((this->b1)&(BS2.b1)) == BS2.b1)

if (((this->b2)&(BS2.b2)) == BS2.b2)

return true;

return false;

}

void Bitstring::print() {

uint64\_t a = 1;

a <<= 63;

uint32\_t b = 1;

b <<= 31;

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

std::cout << ((a & this->b1) > 0);

a >>= 1;

}

//std::cout<<'|';

for (int i = 0; i < 32; i++) {

std::cout << ((b & this->b2) > 0);

b >>= 1;

}

std::cout << std::endl;

}

bool operator< (Bitstring& a, Bitstring& b) {

if ((a.b1) < (b.b1))

return true;

else if ((a.b1) > (b.b1))

return false;

if ((a.b1) == (b.b1))

if ((a.b2) < (b.b2))

return true;

else

return false;

}

bool operator> (Bitstring& a, Bitstring& b) {

if ((a.b1) > (b.b1))

return true;

else if ((a.b1) < (b.b1))

return false;

if ((a.b1) == (b.b1))

if ((a.b2) > (b.b2))

return true;

else

return false;

}

bool operator== (Bitstring& a, Bitstring& b) {

return (((a.b1) == (b.b1)) && ((a.b2) == (b.b2)));

}

std::ostream& operator<<(std::ostream& out, Bitstring& Bs){

uint64\_t a = 1;

a <<= 63;

uint32\_t b = 1;

b <<= 31;

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

out << ((a & Bs.b1) > 0);

a >>= 1;

}

for (int i = 0; i < 32; i++) {

out << ((b & Bs.b2) > 0);

b >>= 1;

}

return out;

}

Bitstring operator "" \_bs(const char\* str) {

int size = strlen(str);

if(size > 96) exit(1);

uint64\_t a = 0;

uint32\_t b = 0;

int i = size-1;

for(i; i>=size-32 && i >=0;i--){

b += (str[i] - '0') << (size-1-i);

}

for(i; i>=0;i--){

a += (str[i] - '0')<<(size-33- i);

}

Bitstring result = Bitstring(a, b);

return result;

}

*CmakeLists.txt*

*cmake\_minimum\_required(VERSION 2.8) # Проверка версии CMake.*

*# Если версия установленой программы*

*# старее указаной, произойдёт аварийный выход.*

*project(2lab) # Название проекта*

*set(SOURCE\_EXE main.cpp) # Установка переменной со списком исходников для исполняемого файла*

*set(SOURCE\_LIB Bitstring.cpp) # Тоже самое, но для библиотеки*

*add\_library(bitstring STATIC ${SOURCE\_LIB}) # Создание статической библиотеки с именем foo*

*add\_executable(main ${SOURCE\_EXE}) # Создает исполняемый файл с именем main*

*target\_link\_libraries(main bitstring)*

1. **Результаты выполнения тестов**

Тест 1

Введите значения строки 1

1

1

Введите значения строки 2

1

1

Введите количество битов для сдвига

1

Первая строка

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

Вторая строка

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

and

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

or

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

xor

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

not

111111111111111111111111111111111111111111111111111111111111111011111111111111111111111111111110

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000010

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000010

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000001

count units BS1

2

count units BS2

2

comparing units

1

includes BS1 BS2

1

000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000

Тест 2

Введите значения строки 1

0

3

Введите значения строки 2

0

11

Введите количество битов для сдвига

3

Первая строка

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

Вторая строка

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

and

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

or

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

xor

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000

not

111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111100

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011000

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011000

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

count units BS1

2

count units BS2

3

comparing units

0

includes BS1 BS2

0

000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000

Тест 3

Введите значения строки 1

11

66

Введите значения строки 2

4

8

Введите количество битов для сдвига

3

Первая строка

000000000000000000000000000000000000000000000000000000000000101100000000000000000000000001000010

Вторая строка

000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000001000

and

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

or

000000000000000000000000000000000000000000000000000000000000111100000000000000000000000001001010

xor

000000000000000000000000000000000000000000000000000000000000111100000000000000000000000001001010

not

111111111111111111111111111111111111111111111111111111111111010011111111111111111111111110111101

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000101100000000000000000000000001000010000

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000010000000000000000000000000000001000000

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000101100000000000000000000000001000010

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000001000

count units BS1

5

count units BS2

2

comparing units

0

includes BS1 BS2

0

000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000

1. **Объяснение результатов работы программы**

Программа просит на вход значения битовых строк, которые представляются четырьмя десятичными числами, по два на каждую строку. Литерал, описанный в моей программе, создает из входных данных два объекта типа Bitstring. Потом над объектом производятся действия согласно моему варианту. Эти действия совершаются при помощи перегруженных операторов, таких как and, not, xor и т.д.

1. **Вывод**

В ходе работы я познакомилась с пользовательскими литералами и принципом их работы. Так же я освоила перегрузку операторов и поняла, насколько более удобной становится работа с объектами классов при их применении.