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 I семестр

Задание 1: «Простые классы»

|  |  |
| --- | --- |
| Группа: | М8О-108Б-18, №12 |
| Студент: | Васильева Василиса Евгеньевна |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 30.11.2018 |

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, 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.

1. **Адрес репозитория на GitHub**

<https://github.com/vasilisavasileva/oop_exercise_1>

1. **Код программы на С++**

Bitstring.h

|  |
| --- |
| #ifndef BITSTRING\_H |
|  | #define BITSTRING\_H |
|  | #include <inttypes.h> |
|  |  |
|  | class Bitstring { |
|  | private: |
|  | uint64\_t b1; |
|  | uint32\_t b2; |
|  | public: |
|  | Bitstring(); |
|  | Bitstring(uint64\_t b1, uint32\_t b2); |
|  | Bitstring \_and(Bitstring bs2); |
|  | Bitstring \_or(Bitstring bs2); |
|  | Bitstring \_xor(Bitstring bs2); |
|  | Bitstring \_not(); |
|  | void shiftLeft(int m); |
|  | void shiftRight(int m); |
|  | int counter(); |
|  | bool compare(Bitstring bs2); |
|  | bool includes(Bitstring bs2); |
|  | void print(); |
|  | }; |
|  |  |
|  | #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 Bitstring::\_and(Bitstring bs2){ |
|  | Bitstring bs3{(this->b1)&(bs2.b1),(this->b2)&(bs2.b2)}; |
|  | return bs3; |
|  | } |
|  |  |
|  | Bitstring Bitstring::\_or(Bitstring bs2){ |
|  | Bitstring bs3{(this->b1)|(bs2.b1),(this->b2)|(bs2.b2)}; |
|  | return bs3; |
|  | } |
|  |  |
|  | Bitstring Bitstring::\_xor(Bitstring bs2){ |
|  | Bitstring bs3{(this->b1)^(bs2.b1),(this->b2)^(bs2.b2)}; |
|  | return bs3; |
|  | } |
|  |  |
|  | Bitstring Bitstring::\_not(){ |
|  | Bitstring bs3{~(this->b1),~(this->b2)}; |
|  | return bs3; |
|  | } |
|  |  |
|  | void Bitstring::shiftLeft(int m){ |
|  | uint32\_t a; |
|  | a = 1; |
|  | a <<= 31; |
|  | for(int i = 0; i < m; i++){ |
|  | if(((this->b2)&a) > 0){ |
|  | this->b1 <<= 1; |
|  | this->b2 <<= 1; |
|  | this->b1 = this->b1 + 1; |
|  | } |
|  | else{ |
|  | this->b1 <<= 1; |
|  | this->b2 <<= 1; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void Bitstring::shiftRight(int m){ |
|  | uint64\_t a; |
|  | uint32\_t b; |
|  | b = 1; |
|  | b <<= 31; |
|  | a = 1; |
|  | for(int i = 0; i < m; i++){ |
|  | if(((this->b1)&a) > 0){ |
|  | this->b1 >>= 1; |
|  | this->b2 >>= 1; |
|  | this->b2 = this->b2 + b; |
|  | } |
|  | else{ |
|  | this->b1 >>= 1; |
|  | this->b2 >>= 1; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | 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; |
|  | } |
|  | for(int i = 0;i<32;i++){ |
|  | std::cout<< ((b & this->b2) > 0); |
|  | b>>=1; |
|  | } |
|  | std::cout<<std::endl; |
|  | } |
|  |  |

Lab1.cpp

|  |
| --- |
| #include <iostream> |
|  | #include "Bitstring.h" |
|  |  |
|  |  |
|  | int main(){ |
|  | 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"; |
|  | BS1.print(); |
|  | std::cout << "Вторая строка\n"; |
|  | BS2.print(); |
|  | std::cout << "and\n"; |
|  | BS3 = BS1.\_and(BS2); |
|  | BS3.print(); |
|  | std::cout << "or\n"; |
|  | BS3 = BS1.\_or(BS2); |
|  | BS3.print(); |
|  | std::cout << "xor\n"; |
|  | BS3 = BS1.\_xor(BS2); |
|  | BS3.print(); |
|  | std::cout << "not\n"; |
|  | BS3 = BS1.\_not(); |
|  | BS3.print(); |
|  | std::cout << "BS1 shiftleft\n"; |
|  | BS1.shiftLeft(n); |
|  | BS1.print(); |
|  | std::cout << "BS2 shiftleft\n"; |
|  | BS2.shiftLeft(n); |
|  | BS2.print(); |
|  | std::cout << "BS1 shiftRight\n"; |
|  | BS1.shiftRight(n); |
|  | BS1.print(); |
|  | std::cout << "BS2 shiftRight\n"; |
|  | BS2.shiftRight(n); |
|  | BS2.print(); |
|  | 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; |
|  | return 0; |
|  | } |

CMakeLists.txt

|  |
| --- |
| cmake\_minimum\_required(VERSION 2.8) # Проверка версии CMake. |
|  | # Если версия установленой программы |
|  | # старее указаной, произайдёт аварийный выход. |
|  |  |
|  | project(lab1) # Название проекта |
|  |  |
|  | set(SOURCE\_EXE lab1.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

0

11

Введите значения строки 2

0

3

Введите количество битов для сдвига

2

Первая строка

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

Вторая строка

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

and

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

or

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

xor

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000

not

111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110100

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101100

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001100

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

count units BS1

3

count units BS2

2

comparing units

0

includes BS1 BS2

1

Тест 2

Введите значения строки 1

0

2147483648

Введите значения строки 2

2

4

Введите количество битов для сдвига

2

Первая строка

000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000

Вторая строка

000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000100

and

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

or

000000000000000000000000000000000000000000000000000000000000001010000000000000000000000000000100

xor

000000000000000000000000000000000000000000000000000000000000001010000000000000000000000000000100

not

111111111111111111111111111111111111111111111111111111111111111101111111111111111111111111111111

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000010000

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000100

count units BS1

1

count units BS2

2

comparing units

0

includes BS1 BS2

0

Тест 3

Введите значения строки 1

1

3

Введите значения строки 2

0

11

Введите количество битов для сдвига

2

Первая строка

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000011

Вторая строка

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

and

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011

or

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000001011

xor

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000001000

not

111111111111111111111111111111111111111111111111111111111111111011111111111111111111111111111100

BS1 shiftleft

000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000001100

BS2 shiftleft

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101100

BS1 shiftRight

000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000011

BS2 shiftRight

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001011

count units BS1

3

count units BS2

3

comparing units

1

includes BS1 BS2

0

1. **Объяснение результатов работы программы**

Программа просит на вход значения битовых строк, которые представляются четырьмя десятичными числами, по два на каждую строку. Далее программа выполняет методы, описанные в классе, поочередно. Параллельно она выводит результаты в стандартный поток вывода.

1. **Вывод**

Изучили основы объектно-ориентированного программирования, методы, классы. Написала простой класс Bitstring, в котором реализована 96 битовая строка, разделенная на два поля. Узнала, как побитовые операторы работают со строками большой длины и реализовала методы класса с помощью этих логических операторов.