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
Задание 1: «Простые классы»

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

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.

2. Адрес репозитория на GitHub

3. Код программы на C++

Lab1.cpp

```
#include <iostream>
#include "Bitstring.h"


int main(){
  int m, n;
  bool l;
  std::cout << "Введите значения строки 1\n";
  Bitstring BS1;
  BS1.read(std::cin);
  std::cout << "Введите значения строки 2\n";
  Bitstring BS2;
  BS2.read(std::cin);
  std::cout << "Введите количество битов для сдвига\n";
  std::cin >> n;
  std::cout << "Первая строка\n";
  BS1.print(std::cout);
  std::cout << "Вторая строка\n";
  BS2.print(std::cout);
  std::cout << "and\n";
  BS1._and(BS2).print(std::cout);
  std::cout << "or\n";
  BS1._or(BS2).print(std::cout);
  std::cout << "xor\n";
  BS1._xor(BS2).print(std::cout);
  std::cout << "not\n";
  BS1._not().print(std::cout);
  std::cout << "BS1 shiftleft\n";
  BS1.shiftLeft(n).print(std::cout);
  std::cout << "BS2 shiftleft\n";
```

```
  BS2.shiftLeft(n).print(std::cout);
  std::cout << "BS1 shiftRight\n";
  BS1.shiftRight(n).print(std::cout);
  std::cout << "BS2 shiftRight\n";
  BS2.shiftRight(n).print(std::cout);
  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;
}
```

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);
        Bitstring _and(const Bitstring& bs2) const;
        Bitstring _or(const Bitstring& bs2) const;
        Bitstring _xor(const Bitstring& bs2) const;
        Bitstring _not() const;
        Bitstring shiftLeft(int m) const;
        Bitstring shiftRight(int m) const;
        int counter() const;
        bool compare(const Bitstring& bs2) const;
        bool includes(const Bitstring& bs2) const;
        void print(std::ostream& out) const;
        void read(std::istream& in);
};

#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(const Bitstring& bs2) const{
Bitstring bs3{(this->b1)&(bs2.b1),(this->b2)&(bs2.b2)};
return bs3;
}

Bitstring Bitstring::_or(const Bitstring& bs2) const{
  Bitstring bs3{(this->b1)|(bs2.b1),(this->b2)|(bs2.b2)};
  return bs3;
}

Bitstring Bitstring::_xor(const Bitstring& bs2) const{
  Bitstring bs3{(this->b1)^(bs2.b1),(this->b2)^(bs2.b2)};
return bs3;
}

Bitstring Bitstring::_not() const{
  Bitstring bs3{~(this->b1),~(this->b2)};
  return bs3;
}

Bitstring Bitstring::shiftLeft(int m) const{
  uint32_t a,t2 = b2;
uint64_t t1 = b1;
a = 1;
a <<= 31;
for(int i = 0; i < m; i++){
if((t2&a) > 0){
      t1 <<= 1;
t2 <<= 1;
      t1 = t1 + 1;
    }
    else{
t1 <<= 1;
t2 <<= 1;
    }
}
return Bitstring(t1,t2);
}

Bitstring Bitstring::shiftRight(int m) const{
  uint64_t a,t1 = b1;
  uint32_t b,t2 = b2;
b = 1;
b <<= 31;
a = 1;
for(int i = 0; i < m; i++){
if((t1&a) > 0){
t1 >>= 1;
 t2 >>= 1;
      t2 = t2 + b;
```

```
    }
    else{
      t1 >>= 1;
      t2 >>= 1;
}
  }
return Bitstring(t1,t2);
}

int Bitstring::counter() const{
  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(const Bitstring& bs2) const{
int a = this->counter();
int b = bs2.counter();
if(a == b)
return true;
return false;
}

bool Bitstring::includes(const Bitstring& BS2) const{
  if(((this->b1)&(BS2.b1)) == BS2.b1)
if(((this->b2)&(BS2.b2)) == BS2.b2)
return true;
return false;

}


void Bitstring::print(std::ostream& out) const{

        uint64_t a = 1;
        a<<=63;
uint32_t b = 1;
b <<=31;
for(int i = 0;i<64;i++){
```

```
out<< ((a & this->b1) > 0);
a>>=1;
        }
        for(int i = 0;i<32;i++){
                out<< ((b & this->b2) > 0);
                b>>=1;
}
        out<<std::endl;
}

void Bitstring::read(std::istream& in){
  in >> b1 >> b2;
}
```

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)
```

4. Результаты выполнения тестов

Тест 1

Введите значения строки 1

Введите значения строки 2

Введите количество битов для сдвига

Первая строка

Вторая строка

or
00000000000000000000000000000000000000000000000000000000000000000000000000001011
xor
00000000000000000000000000000000000000000000000000000000000000000000000000001000
not
11111111111111111111111111111111111111111111111111111111111111111111111111110100
BS1 shiftleft
00000000000000000000000000000000000000000000000000000000000000000000000000101100
BS2 shiftleft
00000000000000000000000000000000000000000000000000000000000000000000000000001100
BS1 shiftRight
00000000000000000000000000000000000000000000000000000000000000000000000000001011
BS2 shiftRight
00000000000000000000000000000000000000000000000000000000000000000000000000000011
count units BS1
3
count units BS2
2
comparing units
0
includes BS1 BS2

1

Тест 2

Введите значения строки 1

Введите значения строки 2

Введите количество битов для сдвига

Первая строка

Вторая строка

11111111111111111111111111111111111111111111111111111111110111111
11111111111111111111111
BS1 shiftleft
00000000000000000000000000000000000000000000000000000000001000
00000000000000000000000000000
BS2 shiftleft
00000000000000000000000000000000000000000000000000000000100000
00000000000000000000000010000
BS1 shiftRight
00000000000000000000000000000000000000000000000000000000000010
00000000000000000000000000000
BS2 shiftRight
00000000000000000000000000000000000000000000000000000000001000
00000000000000000000000000100
count units BS1
1
count units BS2
2
comparing units
0

Тест 3

Введите значения строки 1

Введите значения строки 2

Введите количество битов для сдвига

Первая строка

Вторая строка

00000000000000000000000000000000000000000000000000000000000001000
00000000000000000000000001000
not
11111111111111111111111111111111111111111111111111111111101111111
11111111111111111111100
BS1 shiftleft
00000000000000000000000000000000000000000000000000000000000010000
00000000000000000000000001100
BS2 shiftleft
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000101100
BS1 shiftRight
00000000000000000000000000000000000000000000000000000000000001000
00000000000000000000000000011
BS2 shiftRight
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000001011
count units BS1
3
count units BS2
3
comparing units
1

5. 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.

6. 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.