

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«Национальный исследовательский

**Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

Институт информационных технологий, математики и механики

**Кафедра математического обеспечения и суперкомпьютерных
технологий**

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

«Множества на основе битовых полей»

Выполнил: студент группы
381706-2

Крюков Дмитрий Алексеевич

_____ Подпись

Научный руководитель:

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

_____ Подпись

Нижний Новгород

2018

Содержание

1. Введение.....	3
2. Постановка задачи.....	4
3. Руководство пользователя.....	5
4. Руководство программиста.....	6
4.1 Описание структуры программы.....	6
4.2 Описание структур данных.....	6
4.3 Описание алгоритмов.....	7
5. Эксперименты.....	8
6. Заключение.....	9
7. Литература.....	10

1. Введение

Битовые поля обеспечивают удобный доступ к отдельным битам данных. Они позволяют формировать объекты с длиной, не кратной байту. Что в свою очередь позволяет экономить память, более плотно размещая данные.

Битовые поля применяются для максимально полной упаковки информации, если не важна скорость доступа к этой информации. Например, для увеличения пропускной способности канала при передаче информации по сети или для уменьшения размера информации при хранении. Также использование битовых полей оправдано, если процессор поддерживает специализированные инструкции для работы с битовыми полями, а компилятор использует эти инструкции при генерировании машинного кода.

Множество — тип и структура данных в информатике, которая является реализацией математического объекта множество.

Данные типа множество позволяют хранить ограниченное число значений определённого типа без определённого порядка. Повторение значений, как правило, недопустимо. За исключением того, что множество в программировании конечно, оно в общем соответствует концепции математического множества. Для этого типа в языках программирования обычно предусмотрены стандартные операции над множествами.

2. Постановка задачи

Реализация класса битового поля:

Для битового поля определены операции:

- проверка отдельного бита
- установка и снятие отдельного бита
- сравнение и присваивания битовых полей
- проведение с ними операций «&», «|», «~».

Реализация класса множество :

Для множества определены операции:

- проверка наличия элемента $a \in A$
- добавление элемента $A + a$
- удаление элемента $A - a$

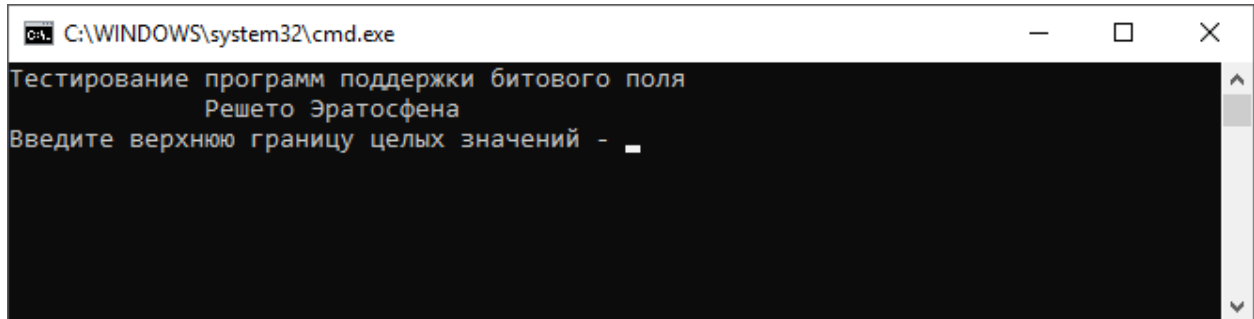
Теоретико-множественные операции

- объединение $A \cup B$
- пересечение $A \cap B$
- вычитание $A \setminus B$

3. Руководство пользователя

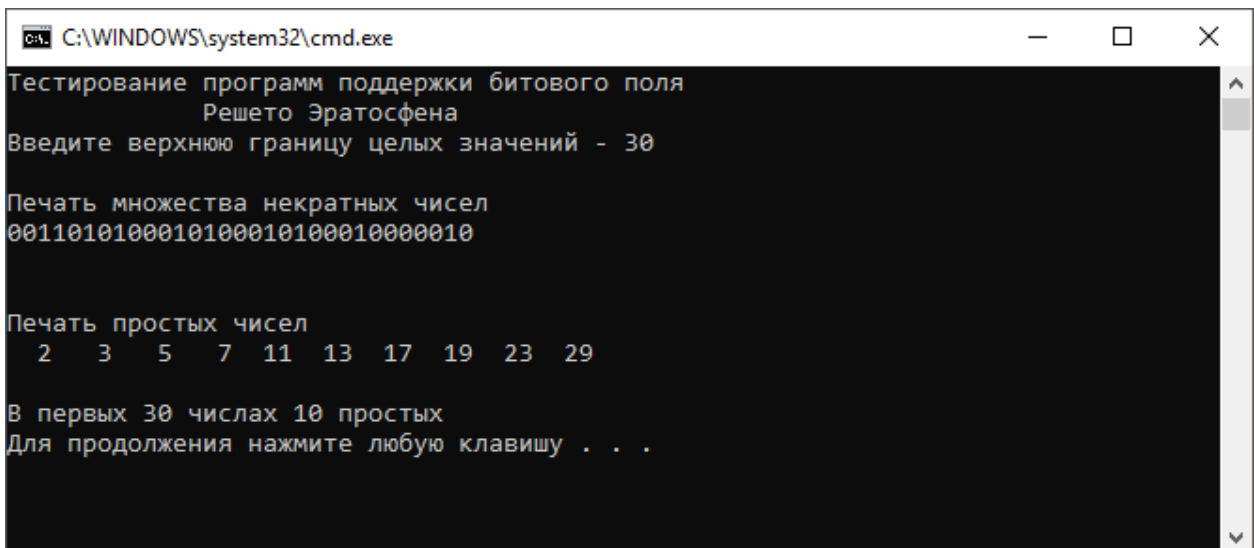
В программе используется алгоритм «Решето Эратосфена» для нахождения простых чисел.

От пользователя требуется ввести верхнюю границу значений



```
C:\WINDOWS\system32\cmd.exe
Тестирование программ поддержки битового поля
Решето Эратосфена
Введите верхнюю границу целых значений - _
```

На экран будет выведено множество простых чисел и их значения



```
C:\WINDOWS\system32\cmd.exe
Тестирование программ поддержки битового поля
Решето Эратосфена
Введите верхнюю границу целых значений - 30

Печать множества некратных чисел
0011010100010100010100010000010

Печать простых чисел
 2  3  5  7 11 13 17 19 23 29

В первых 30 числах 10 простых
Для продолжения нажмите любую клавишу . . .
```

4. Руководство программиста

4.1 Описание структуры программы

1. Модуль tbitfield (tbitfield.cpp tbitfield.h) – реализация класса битового поля
2. Модуль tset(tset.cpp tset.h) – реализация класса множества
3. sample_prime_numbers(sample_prime_numbers.cpp) – реализация решета Эратосфена при помощи классов битового поля и множества
4. tset_test(test_tset.cpp test_tbitfield.cpp) - тестирование классов битового поля и множества

4.2 Описание структур данных

Структура битового поля

Битовое поле - набор битов с номерами от 0 до BitLen, массив pMem рассматривается как последовательность MemLen элементов биты в эл-тах pMem нумеруются справа налево (от младших к старшим)

`int GetLength(void) const` - получить длину (к-во битов)

`void SetBit(const int n)` - установить бит

`void ClrBit(const int n)` - очистить бит

`int GetBit(const int n) const` - получить значение бита

Битовые операции

`int operator==(const TBitField &bf) const` - сравнение

`int operator!=(const TBitField &bf) const` - сравнение

`TBitField& operator=(const TBitField &bf)` - присваивание

`TBitField operator|(const TBitField &bf)` - операция "или"

`TBitField operator&(const TBitField &bf)` - операция "и"

`TBitField operator~(void)` - отрицание

Структура множества

Множество состоит из BitField - битового поле для хранения характеристического вектора и MaxPower - максимальной мощности множества

Доступ к битам

`void InsElem(const int Elem)` - включить элемент в множество

`void DelElem(const int Elem)` - удалить элемент из множества

`int IsMember(const int Elem) const` - проверить наличие элемента в множестве

Теоретико-множественные операции

`int operator== (const TSet &s) const` - сравнение

`int operator!= (const TSet &s) const` - сравнение

`TSet& operator=(const TSet &s)` - присваивание

`TSet operator+ (const int Elem)` - объединение с элементом

`TSet operator- (const int Elem)` - разность с элементом

`TSet operator+ (const TSet &s)` - объединение

`TSet operator* (const TSet &s)` - пересечение

`TSet operator~ (void)` – дополнение

4.3 Описание алгоритмов

Создание битового поля длины l

сохранить l как длину битового поля

вычислить число элементов массива для хранения битового поля по формуле:

$$M = l / N + 1;$$

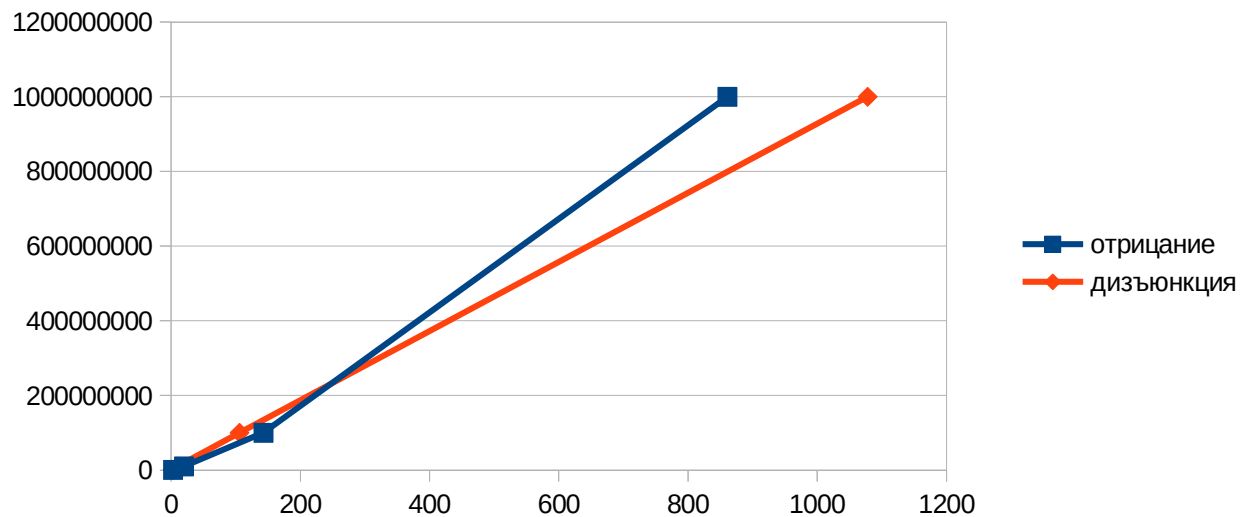
где N – размер типа данных используемого для хранения битового поля(в битах)

Выделить память для массива размера M

проинициализировать нулем все элементы массива

5. Эксперименты

операция	1000000	10000000	100000000	1000000000
отрицание	3	20	143	861
дизъюнкция	2	10	106	1078



Экспериментальная сложность алгоритмов отрицания и дизъюнкции — $O(n)$

Информация о компьютере:

Процессор:

Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz

Версия ОС:

Windows 10 (64 бит)

Видеокарта:

Модель: Intel(R) HD Graphics 5500

Память:

Оперативная память: 8104 МБ

6. Заключение

В ходе работы реализован класс битового поля, в нем реализованы функции для работы с битами, а так же операции «и» «или» и отрицание.

Реализован класс множества, в нем реализованы функции для работы с элементами, а так же основные теоретико — множественные операции.

7. Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.
2. Битовые поля | программирование на С и С++ - Режим доступа: <http://www.c-cpp.ru/books/bitovye-polya>