МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**ЛАБОРАТОРНАЯ РАБОТА**

**«Битовые поля»**

**Выполнил:** студент группы 381706-2

Паузин Леонид Павлович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018

Содержание

[1.Введение 3](#_Toc8259886)

[2. Постановка задачи 4](#_Toc8259887)

[3.Руководство пользователя 5](#_Toc8259888)

[4.Руководство программиста 6](#_Toc8259889)

[4.1 Описание структуры программы 6](#_Toc8259890)

[4.2 Описание структур данных 6](#_Toc8259891)

[4.3 Описание алгоритмов 7](#_Toc8259892)

[5.Заключение 8](#_Toc8259893)

[6.Литература 9](#_Toc8259894)

# 1.Введение

При разработке современного программного обеспечения для десктопных устройств, разработчик, как правило, не старается экономить память, которую будет занимать программа при выполнении. Однако, в некоторых случаях это просто необходимо, ввиду ее недостатка на устройстве, под которое ведется разработка, или в связи с нехваткой вычислительной мощностью этого устройства. В отдельных случаях, необходимую информацию можно представить в виде последовательности бит. Это эффективно, когда основной задачей стоит определение присутствия элементов в множестве. Все возможные элементы нумеруются, затем каждому присваивается 1 если этот он присутствует в множестве и 0 если он не присутствует.

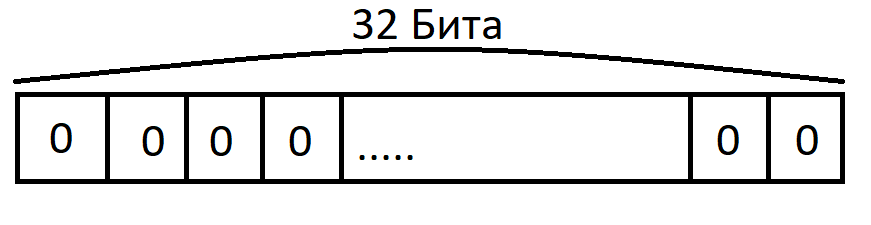
Разберем пример: пусть у нас есть переменная типа integer, она кодируется 4 байтами, в каждом байте 8 бит, соответственно всего у нас есть 32 бита. Если присвоить этой переменной значение 0, то и все биты «занулятся» (рис. 1), если же присвоить этой переменной значение 1, то 31 старших бит останутся равными 0, а последний 32 бит примет значение равное 1 (рис. 2). Соответственно при 2 у нас будет 30 старших бит с 0 значением, 31-ый с 1-ым и 32-ой с 0-ым (рис. 3). 

Рис.1 Рис.2



Рис. 3

Таким образом тратится гораздо меньше памяти на хранение информации. В C++ уже реализованы операции работы с битами, но только для конкретных типов данных. А что если элементов в множестве больше чем количество бит, которое требуется для представления этого элемента? Тогда нужно написать собственный класс битовых полей, который не будет зависеть от разрядности типа данных и в котором будут перегружены операции над битами.

# 2. Постановка задачи

Создать класс «Битовое поле», в котором будут перегружены основные битовые операции, от него унаследовать класс «Набор», в котором будут реализованы методы работы с множествами. Для проверки, реализовать «Решето Эратосфена» с помощью методов класса «Битовое поле» и «Набор».

# 3.Руководство пользователя

Здесь пользователю потребуется ввести любое натуральное число, после нажать кнопку Enter, после чего программа найдет все простые числа в интервале от 2 до заданного числа

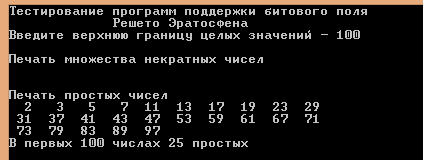


Рисунок 4. Демонстрация работы класса

# 4.Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из 4 основных модулей:

• Класс «TBitField»;

• Класс «TSet»;

• Проект, использующий фреймворк Google Test, для проверки правильности работы классов «TBitField» и «TSet»;

• Проект, реализующий решето Эратосфена

**Класс Tbitfield:**

Класс tbitfield содержит реализацию класса «Битовое поле», описанного во введении работы. В файле tbitfield.h написано объявление этого класса, а в tbitfield.cpp написана его реализация.

**Класс Tset:**

Класс tset содержит реализацию класса «Набор», который упрощает работу с множествами. В нем реализованы такие методы, как «включить элемент в множество», «проверить наличие элемента в множестве», «пересечение», «объединение», «дополнение» множеств и др. В файле tset.h написано объявление этого класса, а в tset.cpp написана его реализация.

**Класс gtest:**

Класс gtest реализует тестирование классов tbitfield и tset, по средствам фреймворка Google Test. Тесты пишутся для каждого метода классов, каждого ветвления этих методов и для всех возможных исключений этих методов. В файле «test\_tbitfield.cpp» реализованы тесты для класса tbitfield, а в файле «test\_tsetcpp» соответственно для класса tset.

**Проект sample\_prime\_numbers:**

В данном проекте реализован примет использования битовых полей и множеств, который в конечном итоге и будет доступен пользователю. Здесь используется предкомпиляция, поэтому можно увидеть, как будет вести себя программа при использовании битовых полей, и при использовании класса «Набор» (идентичное поведение).

## 4.2 Описание структур данных

**Класс TBitField:**

Поля:

typedef unsigned int TELEM

int bitLen - длина битового поля — максимальное количество битов

TELEM \*pMem - память для представления битового поля

int memLen - количество элементов Мем для представления битового поля

**Класс TSet:**

Поля:

int maxPower - максимальная мощность множества

TBitField bitField - битовое поле для хранения характеристического вектора

## Описание алгоритмов

TBitField::TBitField(int len) — конструктор с параметром, который принимает целое число, равное числу битов, необходимых пользователю для использования. BitLen = len, MemLen = len / (8 \* sizeof(TELEM)) + 1 — вычисляется количество элементов заданного типа, необходимые для представления нужного числа битов, выделяется память необходимого размера и зануляются все элементы: pMem = new TELEM[MemLen]; for (int i = 0; i < MemLen; i++) pMem[i] = 0.

TBitField::TBitField(const TBitField &bf) — конструктор копирования, который принимает константную ссылку на объект своего типа. Копирует поля другого элемента, где необходимо использует цикл.

TbitField::~TBitField() - деструктор, который необходим для очистки памяти от массива элементов TELEM.if (pMem != NULL) delete []pMem.

# 5.Заключение

В заключении можно сказать, что все поставленные цели и задачи были выполнены, а именно: созданы классы «TBitField» и «TSet», написаны к ним тесты, и они успешно пройдены. С помощью этих классов была реализована небольшая, но полезная программа – «Решето Эратосфена», которая помогает находить простые числа. По ходу выполнения работы получены навыки работы с GIT

# 6.Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.