Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики механики

БитОвые поля

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381706-2

Антипин А.С.

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г.

Нижний Новгород

2018 г.

Содержание:

[Введение 3](#_Toc530923072)

[Постановка задач и целей 4](#_Toc530923073)

[Постановка задачи: 4](#_Toc530923074)

[Цели работы: 4](#_Toc530923075)

# 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.

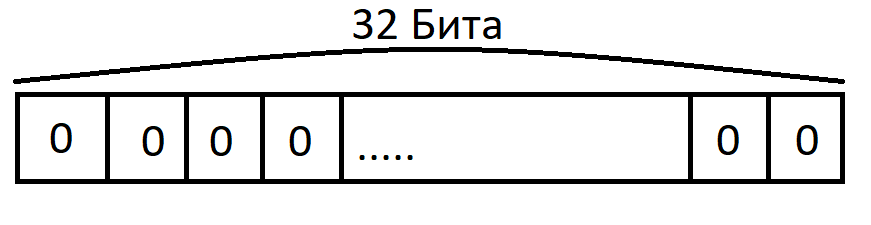
Разберем пример: пусть у нас есть переменная типа integer, она кодируется 4 байтами, в каждом байте 8 бит, соответственно всего у нас есть 32 бита. Если присвоить этой переменной значение 0, то и все биты «занулятся» (рис. 1), если же присвоить этой переменной значение 1, то 31 старших бит останутся 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. 2

рис. 1

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.

# Постановка задач и целей

## 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».

## Цели работы:

* Познакомиться с битовыми операциями;
* Познакомиться с системой контроля версий – GIT;
* Познакомится с фреймворком gtest и пройти все тесты, написанные для классов «Битовое поле» и «Набор».

# Руководство пользователя

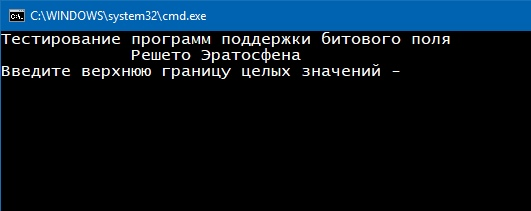
При запуске программы, в консоль будут написаны следующие слова (рис. 4):

рис. 4

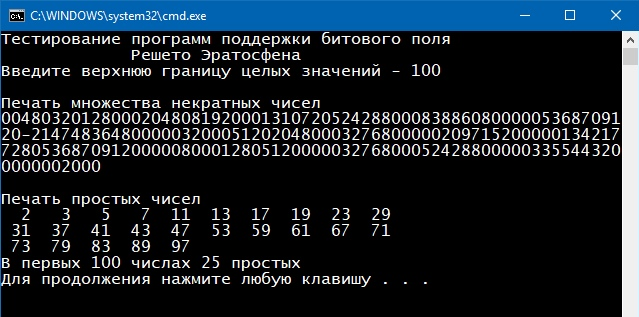
Здесь пользователю потребуется ввести любое натуральное число, после нажать кнопку Enter, после чего программа найдет все простые числа в интервале от 2 до заданного числа (рис. 5).

рис. 5

# Руководство программиста

## Описание структур 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.

Древо классов

**Класс gtest**

**test\_main.cpp**

**test\_tbitfield.cpp**

**test\_tset.cpp**

**Проект sample\_prime\_numbers**

**sample\_prime\_numbers.cpp**

**tset.h**

**tset.cpp**

**Класс tset**

**tbitfield.h**

**tbitfield.cpp**

**Класс tbitfield**

**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).