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**

**Отчет по учебной практике**

**«Множества но основе битовых полей»**

**Выполнил:** студент группы 381706-1

Остапович Денис Евгеньевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc531274519)

[2. Постановка задачи 4](#_Toc531274520)

[3. Руководство пользователя 5](#_Toc531274521)

[4. Руководство программиста 6](#_Toc531274522)

[4.1. Описание структуры программы 6](#_Toc531274523)

[4.2. Описание структур данных 7](#_Toc531274524)

[4.3. Описание алгоритмов 9](#_Toc531274525)

[5. Заключение 12](#_Toc531274526)

[6. Литература 13](#_Toc531274527)

# Введение

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике.

**Битовое поле** – некоторое количество бит, расположенных последовательно. Число связанных бит – ширина поля. Они применяются для максимально полной упаковки информации, если не важна скорость доступа к данным. 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.

**Цель лабораторной работы** — разработка структуры данных для хранения множеств с использованием битовых полей.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка класса битового поля TBitField.
2. Разработка класса множества TSet.
3. Продемонстрировать работу этих классов.
4. Реализовать набор автоматических тестов на Google C++ Testing Framework.

# Руководство пользователя

Пример использования классов битового поля и множества представлен в качестве решения задачи поиска простых чисел с помощью алгоритма "Решето Эратосфена".

При запуске программы с пользователя запрашивается верхняя граница целых значений , среди которых в дальнейшем будут найдены все простые числа. Затем происходит поиск и подсчет простых чисел После на экран выводится множество некратных чисел(битовая строка простых чисел) и простые числа от 0 до .

Далее работа программы различается для классов битового поля и множества.

1. При использовании Битовых полей.

Пользователю предлагается самостоятельно ввести битовую строку. Для этого необходимо задать размер битовой строки . Затем ввести последовательность битов длинной +1 (т.к. нужно учесть нулевой элемент). В конце на экран будет выведена полученная битовая строка и множество чисел, которое в ней отражено. На этом работа программы завершается.

1. При использовании класса множества.

Пользователю предлагается самостоятельно ввести неотрицательное множество целых чисел. Для этого необходимо задать наибольший элемент множества . Затем ввести набор чисел от 0 до . Для завершения ввода введите -1 или число более . В конце на экран будет выведено полученное множество и его битовая строка. На этом работа программы завершается.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:**:**

* sample\_prime\_numbers.cpp (реализация)
* tbitfield.h (описание класса TBitField)
* tbitfield.cpp (реализация методов класса TBitField и перегрузка операций)
* tset.h (описание класса TSet)
* tset.cpp (реализация методов класса TSet и перегрузка операций)
* Проект test\_set состоит из следующих файлов:
* test\_tbitfield.cpp (реализация тестов для класса TBitField)
* test\_tset.cpp (реализация тестов для класса TSet)

## Описание структур данныхКласс TbitField (шаблонный класс)

#### *Класс TBitField*

**private**:

* *int bitLen* - длина битового поля
* *unsigned int \*pMem* - память для представления битового поля
* *int memLen* - количество битов для представления битового поля
* *int GetMemIndex(const int n) const* – метод определяющий индекс бита n в массиве pMem
* *unsigned int GetMemMask(const int n) const* – метод возвращающий маску для бита n

**public:**

1. Описаны два конструктора и деструктор

* *TBitField(int len)* – конструктор инициализации. Принимает длину битового поля.
* *TBitField(const TBitField &bf)* – конструктор копирования. Принимает ссылку на объект TBitField.
* *~TBitField()* – деструктор.

1. Описаны методы

* *int GetLength() const* – возвращает длину битового поля
* *void SetBit(const int n)* – устанавливает n-й бит битового поля.
* *void ClrBit(const int n)* – очищает n-й бит битового поля.
* *int GetBit(const int n) const* – возвращает значение бита n.
* *int operator==(const TBitField &bf) const* – принимает ссылку на объект класса TBitField, проверяет на равенство два битовых поля. Если они идентичны, то возвращает 1, иначе 0.
* *int operator!=(const TBitField &bf) const* - принимает ссылку на объект класса TBitField, проверяет на неравенство два битовых поля. Если они различны, то возвращает 1, иначе 0.
* *TBitField& operator=(const TBitField &bf)* - принимает ссылку на объект класса TBitField, присваивает полученное битовое поле к исходному.
* *TBitField operator|(const TBitField &bf)* - принимает ссылку на объект класса TBitField, возвращает новый объект класса TBitField полученный путем использования операции побитового «ИЛИ» между исходным и пришедшим битовыми полями.
* *TBitField operator&(const TBitField &bf) -* принимает ссылку на объект класса TBitField, возвращает новый объект класса TBitField полученный путем использования операции побитового «И» между исходным и пришедшим битовыми полями.
* *TBitField operator~()* - возвращает объект битового поля, полученный путем применения операции побитового отрицания для исходного битового поля.
* *friend istream &operator>>(istream &istr, TBitField &bf)* – ввод битового поля с клавиатуры.
* *friend ostream &operator<<(ostream &ostr, const TBitField &bf)* – вывод битового поля.

#### *Класс Tset (шаблонный)*

**private:**

* *int maxPower* - максимальная мощность множества
* *TBitField bitField* - битовое поле для хранения характеристического вектора

**public:**

1. Описаны 3 конструктора:

* *TSet(int mp)* – конструктор инициализации. Принимает максимальную мощность множества.
* *TSet(const TSet &s)* - конструктор копирования. Принимает ссылку на объект класс TSet.
* *TSet(const TBitField &bf)* - конструктор преобразования типа. Принимает ссылку на объект класс TBitField.

1. Описан метод преобразования типа битовому полю:

* *operator TBitField()* – возвращает битовое поле характеристического вектора.

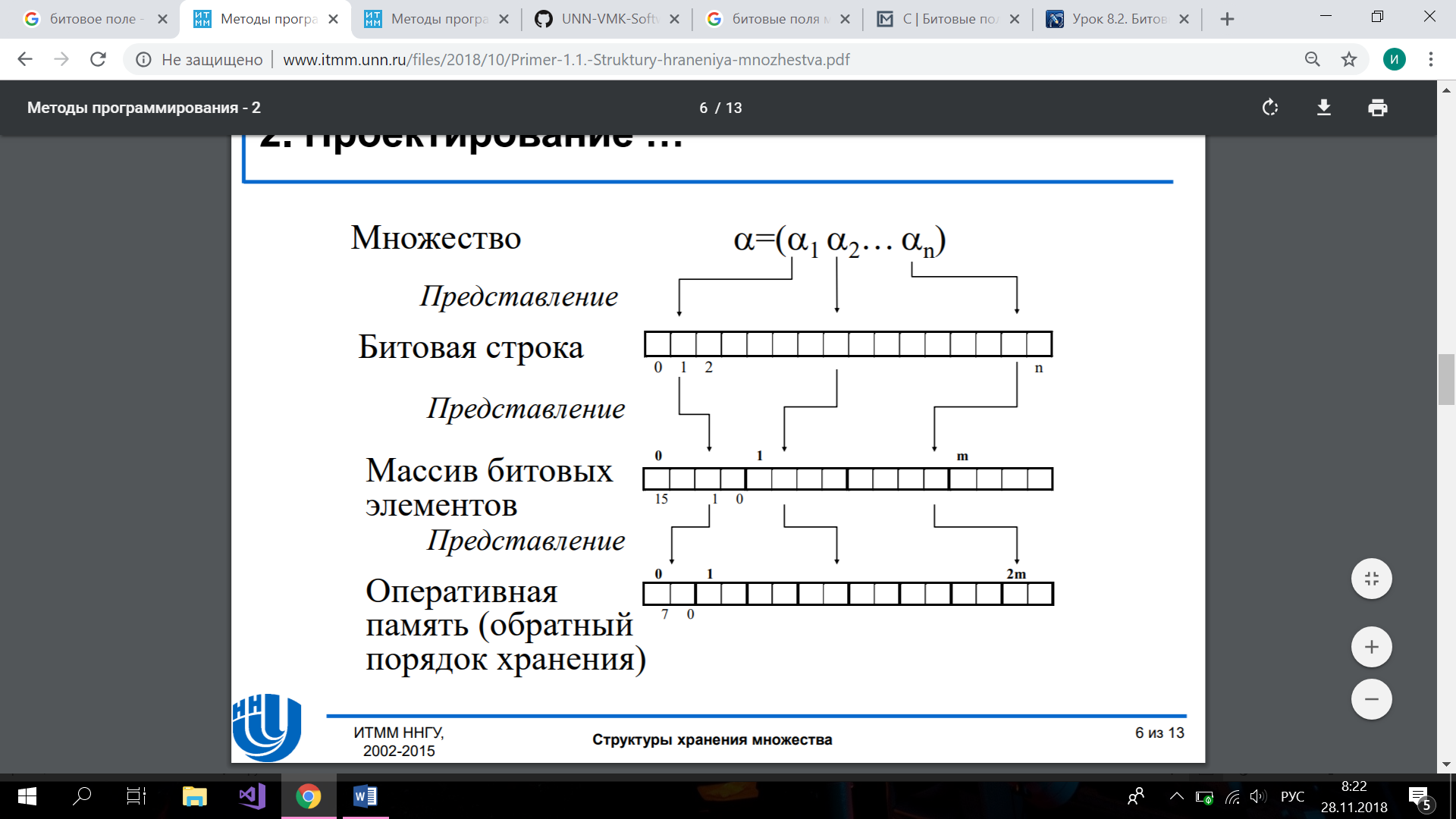
1. Описаны методы:

* *int GetMaxPower(void) const* – возвращает максимальную мощность множества
* *void InsElem(const int Elem)* – добавляет элемент Elem в множество
* *void DelElem(const int Elem)* - удаляет элемент Elem из множества
* *int IsMember(const int Elem) const* - проверяет наличие элемента Elem в множестве
* Перегружены теоретико-множественные операторы:
* *int operator==(const TSet &s) const* – принимает ссылку на объект класса TSet, проверяет на совпадение два битовых поля. Если они идентичны, то возвращает 1, иначе 0.
* *int operator!=(const TSet &s) const* - принимает ссылку на объект класса TSet, проверяет на совпадение два битовых поля. Если они идентичны, то возвращает 1, иначе 0.
* *TSet& operator=(const TSet &s)* – принимает ссылку на объект класса TSet, присваивает полученное множество к исходному.
* *TSet operator+(const int Elem*) – к исходному множеству добавляет Elem (если его там нет) и возвращает новый объект класса TSet.
* *TSet operator-(const int Elem)* – из исходного множества удаляет Elem (если он там есть) и возвращает новый объект класса TSet.
* *TSet operator+(const TSet &s)* - принимает ссылку на объект класса TSet, возвращает новый объект класса TSet, который является объединением исходного и полученного множества.
* *TSet operator\*(const TSet &s)* - принимает ссылку на объект класса TSet, возвращает новый объект класса TSet, который является пересечением исходного и полученного множества.
* *TSet operator~(void)* – возвращает дополнение к исходному множеству.
* *friend istream &operator>>(istream &istr, TSet &bf)* – ввод множества с клавиатуры.
* *friend ostream &operator<<(ostream &ostr, const TSet &bf)* – вывод характеристического вектора множества.

## Описание алгоритмов

**Формирование битового поля:**

На входе мы имеем множество натуральных чисел. Его можно представить виде битовой строки, где каждый бит принимает значение «1», если элемент присутствует во входном множестве, и «0» в противном случае. Битовую строку удобнее всего хранить в массиве. Однако, стоит заметить одну особенность – в битовой строке нумерация бит происходит слева направо, в массиве нумерация элементов также происходит справа налево, однако биты элемента массива нумеруются справа налево.



Множество

Битовая строка

Битовое поле

Кроме того в программе были реализованы такие алгоритмы, как:

Чтобы установить бит на позицию используется побитовое «ИЛИ» между битами элемента массива pMem, в котором находится бит n, и между его битовой маской:

|  |  |
| --- | --- |
|  |  |

Чтобы очистить бит используем побитовое «И» между битами элемента массива pMem, в котором находится бит , и между дополнением его битовой маски:

|  |  |
| --- | --- |
|  |  |

Чтобы получить значение бита n используем побитовое «И» между битами элемента массива pMem, в котором находится бит , и между его битовой маской:

|  |  |
| --- | --- |
|  |  |

# Заключение

В ходе выполнения лабораторной была разработана структура данных – битовые поля. Реализованы классы TBitField и TSet, реализован пример использования этих классов, написаны недостающие тесты на базе Google C++ Testing Framework.

# Литература

* Книги

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Bjarne Stroustrup The C++ programming language

* Ссылки в Internet

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [http://www.itmm.unn.ru/files/2018/10/Primer-1.1.-Struktury-hraneniya-mnozhestva.pdf], 2015.
2. Википедия https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D0%BE%D0%B5\_%D0%BF%D0%BE%D0%BB%D0%B5]