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 на основе битовых полей»**

**Выполнил:** студент группы 381706-1

Соболева Юлия Алексеевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018 г.

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 11](#_Toc531274526)

[6. Литература 12](#_Toc531274527)

# Введение

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как битовые поля.

**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.

Такая структура хранения довольно удобна и вот лишь несколько примеров, почему:

- при ограниченной памяти вы можете хранить несколько булевых (истина, ложь) переменных в одном байте;

- интерфейсы некоторых устройств передают информацию, которая кодируется в биты внутри одного байта;

- некоторые программы кодирования нуждаются в доступе к битам внутри байта.

**Цель данной работы**  разработка структуры данных для хранения множеств с использованием битовых полей, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация класса битового поля TBitField.
2. Реализация класса множества TSet.
3. Разработка интерфейса для данных классов.
4. Обеспечение работоспособности примера использования.
5. Реализация нескольких тестов на базе Google Test.

# 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

Проект “ sample\_prime\_numbers” состоит из следующих файлов**:**

1. sample\_prime\_numbers.cpp (в нём находится main)

Проект “set” состоит из следующих файлов:

1. tbitfield.h (описание класса “TBitField”)
2. tbitfield.cpp (реализация методов класса “ TBitField ” и перегрузка операций)
3. tset.h (описание класса “TSet”)
4. tset.cpp (реализация методов класса “ TSet ” и перегрузка операций)

Проект “ test\_set” состоит из следующих файлов:

1. test\_tbitfield.cpp (реализация тестов для класса TBitField)
2. test\_tset.cpp (реализация тестов для класса TSet)

## Описание структур данных

***Класс TBitField:***

*Поля:*

* BitLen - длина битового поля - макс. к-во битов
* \*pMem - массив для представления битового поля
* MemLen - количество элементов Мем для представления битового поля

*Конструкторы и деструктор:*

* TBitField(int len);
* TBitField(const TBitField &bf);
* ~TBitField();

*Методы:*

* int GetLength(void) const; - получить длину (количество битов)
* void SetBit(const int n); - установить бит
* void ClrBit(const int n); - очистить бит
* int GetBit(const int n) const; - получить значение бита

*Перегруженные операторы:*

* 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
* 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:***

*Поля:*

* MaxPower - максимальная мощность множества
* BitField - битовое поле для хранения характеристического вектора

*Конструкторы:*

* TSet(int mp);
* TSet(const TSet &s);
* TSet(const TBitField &bf);

*Методы*:

* operator TBitField(); - преобразование типа к битовому полю
* int GetMaxPower(void) const; - максимальная мощность множества
* 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

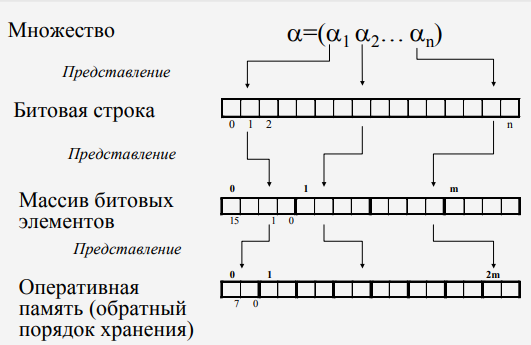
*Перегруженные операторы:*

* 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
* 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 слева направо, однако биты элемента массива нумеруются справа налево



Кроме того в программе были реализованы такие алгоритмы, как:

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

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

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

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

1. 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 работы я смогла реализовать такую структуру данных, как битовое поле. Вместе с ней, согласно заданному интерфейсу, я реализовала классы TBitField и TSet. Написание нескольких своих тестов в дополнение к уже имеющимся, а так же достижение одной из главных целей – того, чтобы все тесты были пройдены, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс битового поля TBitField и класс множества TSet
2. Разработать интерфейс для данных классов.
3. Обеспеченить работоспособность примера использования.
4. Реализовать некоторые тесты на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# Литература

* Книги

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.

* 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.