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

Соколов Андрей Дмитриевич

\_\_\_\_\_\_\_\_\_\_\_ 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 11](#_Toc531274526)

[6. Литература 12](#_Toc531274527)

# Введение

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

Если не важна скорость доступа к данным, то для максимально полной упаковки информации применяются битовые поля. Битовое поле – некоторое количество бит, расположенных последовательно. Они позволяют работать с отдельными битами и полезны по нескольким причинам. Наиболее важные из них:

1. Если ограничено место для хранения информации, можно сохранить несколько логических (истина/ложь) переменных в одном байте.

2. Некоторые интерфейсы устройств передают информацию, закодировав биты в один байт.

3. Некоторым процедурам кодирования необходимо получить доступ к отдельным битам в байте.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация класса битового поля TBitField согласно заданному интерфейсу.
2. Реализация класса множества TSet согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация нескольких простых тестов на базе Google Test.
5. Публикация исходных кодов в личном репозитории на GitHub.

# 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 задаётся верхняя граница целых значений , среди которых будут найдены все простые числа. Затем происходит поиск и подсчет простых чисел и на экран выводится множество некратных чисел (битовая строка простых чисел) и простые числа от 0 до .

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.

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

* Модуль tbitfield.

Он включает в себя заголовочный файл tbitfield.h, в котором определен интерфейс класса битового поля TBitField и файл содержащий его реализацию tbitfield.cpp.

* Модуль tset.

Он включает в себя заголовочный файл tset.h, в котором определен интерфейс класса битового поля TSet и файл содержащий его реализацию tset.cpp.

* Тесты

Для каждого класса реализован набор тесов в файлах test\_tbitfield.cpp и test\_tset.cpp.

* Пример использования.

Для класса битового поля и множества в файле sample\_prime\_numbers.cpp реализован алгоритм поиска простых чисел, называемый "Решетом Эратосфена".

## Описание структур данных

#### *Класс TBitField*

Класс TBitField является шаблонным классом. В нем определены три поля и два метода реализации со спецификатором доступа private:

int bitLen - длина битового поля

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 int GetMemMask(const int n) const – метод возвращающий маску для бита n

TBitField(int len) – конструктор инициализации. Принимает длину битового поля.

TBitField(const TBitField &bf) – конструктор копирования. Принимает ссылку на объект TBitField.

~TBitField() – деструктор.

int GetLength() const – возвращает длину битового.

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*

Класс TSet является шаблонным классом. В нем определены два поля со спецификатором доступа private:

int maxPower - максимальная мощность множества

TBitField bitField - битовое поле для хранения характеристического вектора

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.

operator TBitField() – возвращает битовое поле характеристического вектора.

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

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.

Set& 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

Аргументом конструктора является количество 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. Удобнее всего это хранить в массиве. Также стоит заметить, что в Нумерация бит в битовой строке – слева направо, а нумерация элементов в массиве – слева направо, биты элемента – справа налево.

Также в классе TBuildField дополнительно были прописаны два метода реализации битового поля:

GetMemIndex – – получение индекса элемента для n-ого бита.

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

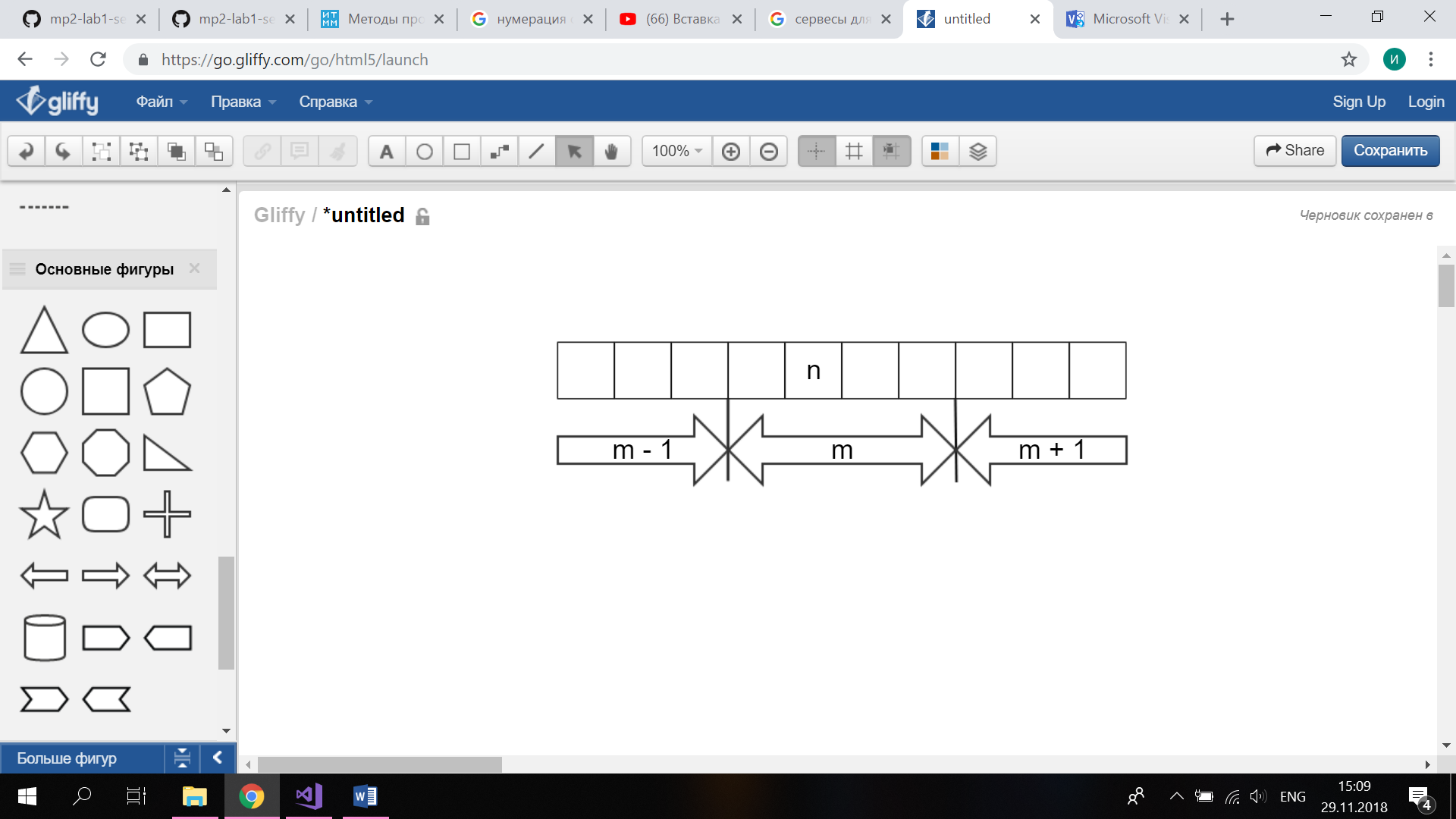


Рисунок 1. GetMemIndex. Возвращает индекс массива m, принимает номер бита n.

GetMemMask – – получение битовой маски для n-ого бита.

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

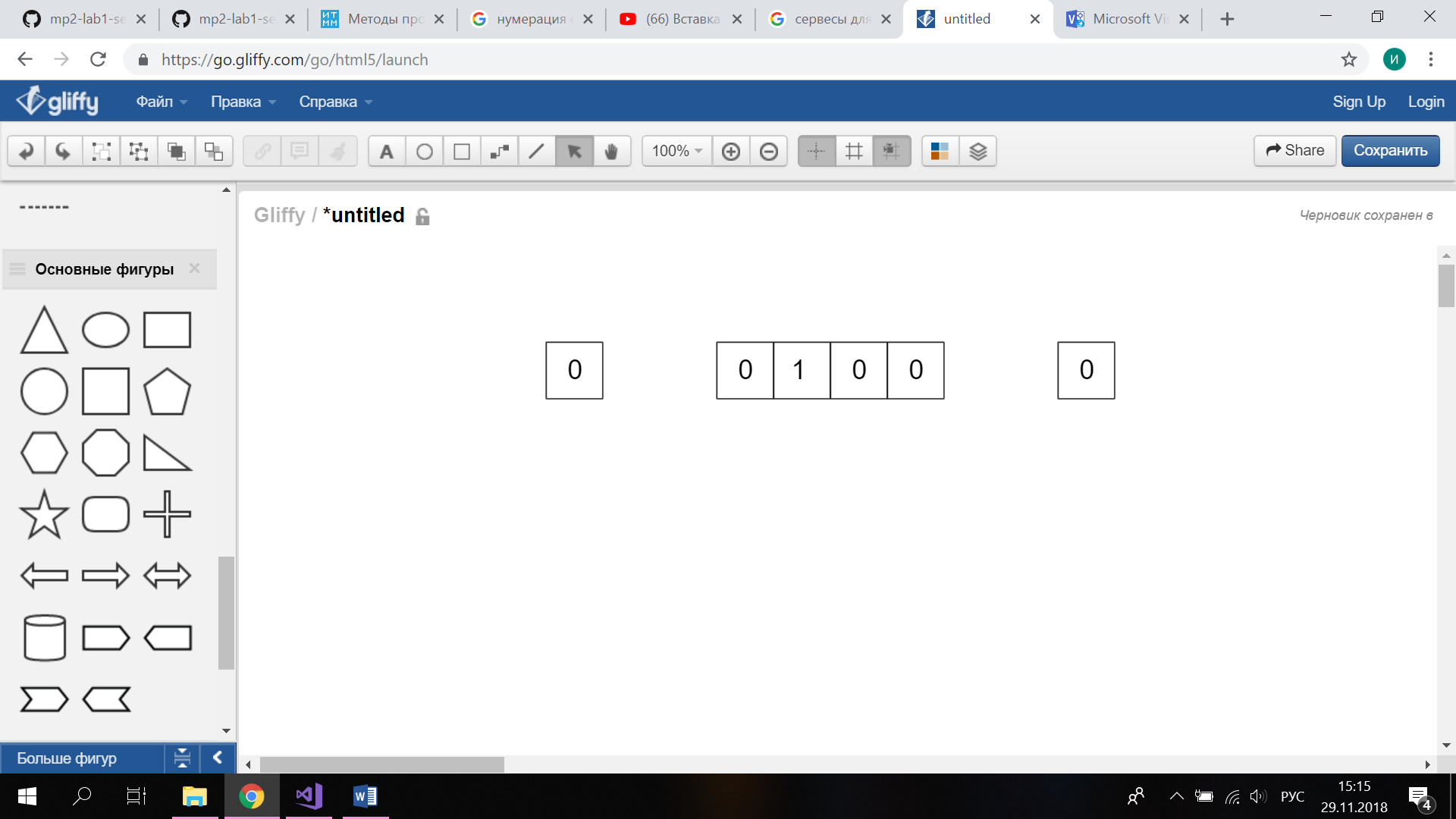


Рисунок 2. GetMemMask. Битовая маска для бита n из рисунка 2.

Эти методы используются в дальнейшем для реализации функции

SetBit(const int n), которая устанавливает бит на n-ую позицию. Для установки выполняется логическое сложение (операцию «битовое ИЛИ») числа с маской, у которой в позициях, соответствующих битовому полю, установлены единицы

Для установки в один или несколько битов нулей число операцией «битовое И» умножают на маску, у которой в позициях, соответствующих битовому полю, установлены нулевые биты.

Для получения значения бита умножают «битовую маску» на число с помощью операции «битовое И».

# Заключение

В процессе работы я понял принципы использования битовых полей. Они значительно уменьшают затраты на хранение информации и помогают решать многие практические задания. Освоена техника составления тестов путем самостоятельного составления нескольких из них.

# Литература

* Книги

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

* 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.