Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Факультет Информационных технологий, математики и механики

Отчёт по лабораторной работе

**Множества на битовых полях**

Выполнил:

студент ИТММ гр. 381808-2

Некравцев А. А.

Проверил:

Нижний Новгород

2019 г.

**Содержание**

Введение ….............................................................................................................3

Постановка задачи ….............................................................................................4

Руководство пользователя …................................................................................5

Руководство программиста …..............................................................................5

Описание структуры программы ….........................................................5

Описание структур данных …..................................................................6

Описание алгоритмов …...........................................................................6

Заключение …........................................................................................................7

Литература ….........................................................................................................8

**Введение**

В лабораторной работе представлена программная реализация множества на основе битового поля.

Множество - это структура данных, позволяющая хранить ограниченное число значений определённого типа без определённого порядка.

Битовое поле - некоторое количество бит, расположенных последовательно в памяти, значение которых процессор не способен прочитать из-за особенностей аппаратной реализации.

Преимущество такой реализации в существенном уменьшении размера информации при хранении. Так как работаем мы с каждым битом по отдельности, ключевыми операциями являются логическое И(&), ИЛИ(|), отрицание(~) и логические сдвиги <<, >>.

**Постановка задачи**

**Цель 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.

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**

Пользователю предоставляется доступ к двум классам TBitfield, TSet. Опишем интерфейс данных классов.

## *TbitField:*

TBitField(int len) // конструктор создающий битовое поле длинны n

TBitField(const TBitField &bf) // конструктор копирования

GetMemIndex(const int n) const // функция получает номер бита и возвращает номер байта в котором он записан

GetMemMask(const int n) const // функция получает номер бита и возвращает его маску для

## *Tset:*

TSet(int mp) // конструктор создающий множество мощности mp

TSet(const TSet &s) // конструктор копирования

TSet(const TBitField &bf) // конструктор преобразования типа

operator+(const int Elem) // объединение с элементом

operator-(const int Elem) // разность с элементом

operator\*(const TSet &s) // пересечение множеств

operator~(void) // дополнение множества

**Руководство программиста**

**Описание структуры программы**

В разделе *set* лежат заголовочные файлы *tbitfield.h* и *tset.h*, в них определены классы битового поля *TBitField* и множества *TSet*. Их реализация содержится в *tbitfield.cpp* и *tset.cpp*.

Раздел *test\_set* содержит набор тестов, реализованный в файлах *test\_tbitfield.cpp* и *test\_tset.cpp* с помощью фреймворка Google Test.

Раздел *sample\_prime\_numbers* содержит пример использования множества и битового поля для поиска простых чисел с помощью алгоритма “Решето Эратосфена”.

**Описание структур данных**

**TBitField:**

class TBitField

int BitLen; // длина битового поля - макс. к-во битов

TELEM \*pMem; // память для представления битового поля

int MemLen; // к-во эл-тов Мем для представления бит.поля

Класс TSet наследуется от TBitField.

**TSet:**

class TSet

int MaxPower; // максимальная мощность множества

TBitField BitField; // битовое поле для хранения характеристического вектора

**Описание алгоритма**

Ключевые методы в классе TBitField представлены ниже. GetMemMask определяет битовую маску для принимаемого бита, а GetMemIndex для принимаемого бита определяет индекс элемента в массиве pMem, в котором хранится битовое поле.

int TBitField::GetMemIndex(const int n) const {

if (n < 0 || n >= BitLen) throw "Index is out of range";

else return n>>5;

}

TELEM TBitField::GetMemMask(const int n) const {

if (n < 0 || n >= BitLen) throw "Index is out of range";

else {

TELEM m = 1 << (n % ((sizeof(TELEM) \* 8)));

return m;

}

}

Далее будет приведена операция пересечения множеств в классе TSet. По ааналогии с ней реализованы объединение со множеством, объединение с элементом, дополнение, разность с элементом.

TSet TSet::operator\*(const TSet &s) {

return TSet(BitField & s.BitField);

}

Операция сравнения двух множеств:

int TSet::operator==(const TSet &s) const {

if(BitField == s.BitField && MaxPower == s.MaxPower) return 1;

else return 0;

}

Операция присваивания:

TSet& TSet::operator=(const TSet &s) {

if (s != \*this) {

MaxPower = s.MaxPower;

BitField = s.BitField;

}

return \*this;

}

**Заключение**

В работе успешно реализованы класс битового поля и класс множества, демонстрирующие экономичность хранения таким образом числового множества. Произведена проверка несколькими тестами с помощью Google Test и алгоритмом “решето Эратосфена”.

**Приложение**

Фрагменты исходного кода представлены в личном репозитории на Github:

<https://github.com/Alexey42/mp2-lab1-set/tree/master/src>