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: структура хранения»

Выполнил:
студент группы 361706-1
Резанцев Сергей Алексеевич
_____Подпись

Научный руководитель:
ассистент каф. МОСТ ИИТММ

_____Лебедев И. Г.

Нижний Новгород
2018 г.

Содержание

1. Введение

Целью данной лабораторной работы является рассмотрения множеств и их представления в виде битовых полей и для реализации этой программы понадобится создать классы с различными функциями и перегрузить арифметические операторы.

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

Множества - это объект, состоящий из объектов (элементы данного множества), обладающих общим свойством.

Битовые поля полезны по нескольким причинам:

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.
4. Применяются для максимально плотной упаковки данных, если не важна скорость их получения.

2. Постановка задачи

Реализовать библиотеку с классами TBitField и TSet для работы с битовыми полями и множествам. Также должны присутствовать тесты на базе Google Test, покрывающие классы TBitField и TSet, и пример использования библиотеки.

3. Руководство пользователя

Чтобы начать работу с программой запустите приложение sample_prime_numbers. В начале на экране появится надпись “Тестирование программ поддержки битового поля. Решето Эратосфена. Введите верхнюю границу целых значений”. Затем вас попросят ввести число. Потом вас попросят ввести битовое поле размерности, равной числу, которое вы ввели. И тогда на экране появится битовое поле, которое вы ввели, и надпись:”Печать множества не кратных чисел. Печать простых чисел. В первых ... числах ... простых”. И выведут эти числа.

Или…

В начале на экране появится надпись “Тестирование программ поддержки множества. Решето Эратосфена. Введите верхнюю границу целых значений”. Затем вас попросят ввести число. Потом вас попросят ввести элементы множества количества, равного числу, которое вы ввели. И тогда на экране появится множество, которое вы ввели, и надпись:”Печать множества не кратных чисел. Печать простых чисел. В первых ... числах ... простых”. И выведут эти числа.

4. Руководство программиста

4.1. Описание структуры программы

В решении 3 проекта: главный проект, библиотека и тесты. В библиотеке 2 заголовочных файла, в котором реализованы классы:TBitField и TSet; и исполнительный файл, в котором реализованы методы классов. В главном проекте только 1 исполнительный файл.

4.2. Описание структур данных

В программе описаны классы:

TBitField:

В нем 3 поля:

bitLen // длина битового поля - макс. к-во битов

*pMem память для представления битового поля

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

И реализованы следующие методы:

```
GetMemIndex(const int n) const; // индекс в pMем для бита n
GetMemMask (const int n) const; // битовая маска для бита n
TBitField(int len);
 TBitField(const TBitField &bf);
  ~TBitField();
  // доступ к битам
  GetLength(void) const;      // получить длину (к-во битов)
  SetBit(const int n);       // установить бит
  ClrBit(const int n);       // очистить бит
  GetBit(const int n) const; // получить значение бита
  // битовые операции
```

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

TSet:

В нем 2 поля:

maxPower; // максимальная мощность множества

bitField; // битовое поле для хранения характеристического вектора

И реализованы следующие методы:

TSet(int mp);

TSet(const TSet &s); // конструктор копирования

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

operator TBitField(); // преобразование типа к битовому полю

Доступ к битам:

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

Теоретико-множественные операции:

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

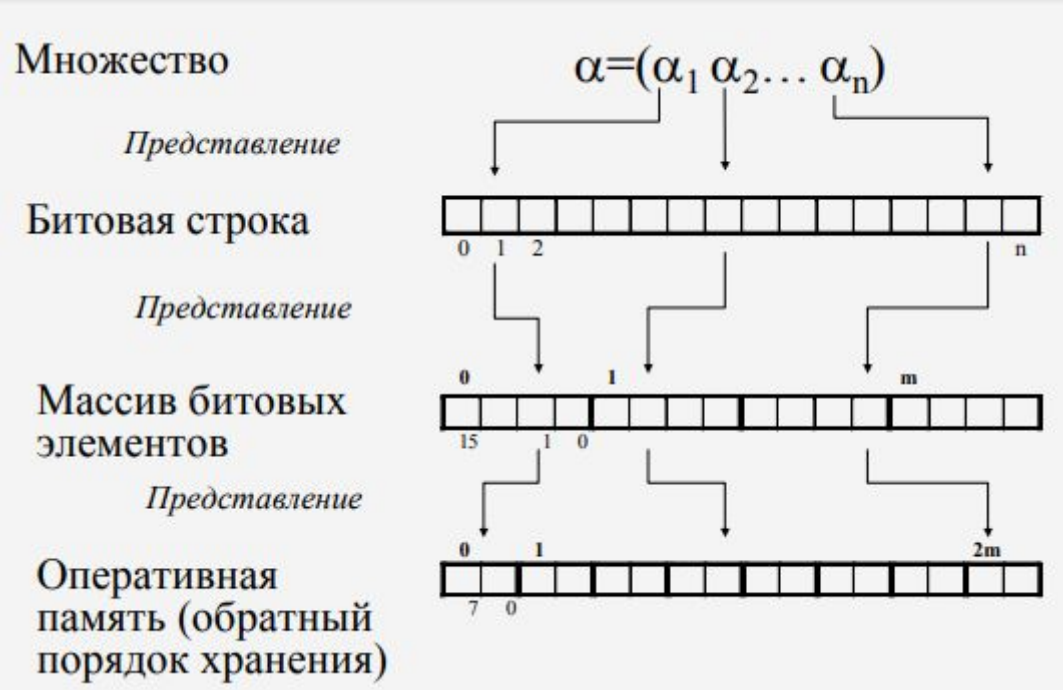
TSet operator- (const int Elem); // разность с элементом

TSet operator+ (const TSet &s); // объединение

TSet operator* (const TSet &s); // пересечение

TSet operator~ (void); // дополнение

4.3. Описание алгоритмов



Представление структуры данных.

На множество подается набор чисел. Если элемент с таким числом присутствует в множестве, то бит с этим значением принимает 1, если нет - 0.

TBitField:

Объединение: Логическое И для каждого бита битового поля

Дополнение: Логическое ИЛИ для каждого бита битового поля

Отрицание: Логическое НЕ для каждого бита битового поля

TSet:

Объединение: Логическое И для каждого бита битового поля

Дополнение: Логическое ИЛИ для каждого бита битового поля

Отрицание: Логическое НЕ для каждого бита битового поля

5. Заключение

В данном курсовом проекте при разработке программы были рассмотрены множества и их реализация с помощью битовых полей. Также я реализовал свою первую структуру данных и протестировал ее с помощью Google Test. И получил навыки для работы с этой структурой данных и для тестирования программ.

6. Литература

1. Васильев А.Н. Самоучитель C++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.

2. Т. А. Павловская C/C++ Программирование на языке высокого уровня. - СПб.:Питер, 2011. - 461 с.

3.Крапенко С. Н. и др. Методы объектно-ориентированного программирования. http://e-learning.unn.ru/course/view.php?id=251.

4.Страуструп. Б. Курс «Язык программирование C++ для профессионалов» http://www.intuit.ru/studies/courses/98/98/info

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