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

**Структура хранения множества**

Выполнил:

студент ф-та ИТММ гр. 381808-2

Ратов Святослав.

Проверил:

ассистент каф. МО ЭВМ, ВМК

Панов А.

Нижний Новгород

2019 г.

Содержание

[Введение 3](#_Toc22161443)

[Постановка задачи 4](#_Toc22161444)

[Руководство пользователя 5](#_Toc22161445)

[Руководство программиста 6](#_Toc22161446)

[Описание структуры программы 6](#_Toc22161447)

[Описание структур данных 6](#_Toc22161448)

[Описание алгоритмов 7](#_Toc22161449)

[Эксперименты 11](#_Toc22161450)

[Заключение 12](#_Toc22161451)

[Литература 13](#_Toc22161452)

[Приложения 14](#_Toc22161453)

[Приложение 1 14](#_Toc22161454)

[Приложение 2 14](#_Toc22161455)

# Введение

Теория множеств – учение об общих свойствах множеств – преимущественно бесконечных. Явным образом понятие *множества* подверглось систематическому изучению во второй половине XIX века в работах немецкого математика Георга Кантора2.

Влияние теории множеств на развитие современной математики очень велико. Прежде всего, теория множеств явилась фундаментом ряда новых математических дисциплин (теории функций действительного переменного, общей топологии, общей алгебры, функционального анализа и др.).

Постепенно теоретико-множественные методы находят всё большее применение и в классических частях математики. Например, в области математического анализа они широко применяются в качественной теории дифференциальных уравнений, вариационном исчислении, теории вероятностей и др.

Активное применение аппарата теории множеств в современной науке приводит к необходимости создания соответствующих программных решений. Вместе с тем лишь в отдельных языках программирования предусмотрены встроенные средства для работы с множествами (примером может служить язык Pascal в реализации фирмы Borland).

Программная реализация *множества* может выполняться различными способами (в соответствии с требованиями конкретной задачи или с общих позиций) и обычно тесно связана с использованием битовых операций в выбранном языке программирования.

Данная работа посвящена изучению одного из возможных подходов к хранению и обработке множеств.

# Постановка задачи

Реализовать структуру хранения данных, представленную в виде класса TSet, которая позволяет применять к элементам множества следующие операции:

* Добавление/удаление элемента из множества;
* Проверка на наличие элемента в множестве;
* Сравнение элементов;
* Присваивание элемента;
* Разность с элементом;
* Объединение;
* Пересечение;
* Дополнение.

# Руководство пользователя

Для получения решета Эратосфена необходимо ввести верхнюю границу целых значений и нажать enter. После этого программа выведет в консоль список простых чисел из диапазона в отформатированном виде и также выведет общее количество этих чисел.

Для подключения библиотеки с реализациями методов для работы с битовым полем необходимо указать заголовок "tbitfield.h".

Для подключения библиотеки для работы с множествами необходимо указать заголовок “tset.h”.

# Руководство программиста

## Описание структуры программы

Программа содержит в себе:

* BitField.h, BitField.cpp – модуль с классом, реализующим операции над *Битовыми полями*;
* Set.h, Set.cpp – модуль с классом, реализующим обработку *Множеств*;
* sample\_prime\_numbers.cpp – модуль программы тестирования.

В файлах tbitfield.h и tset.h представлены прототипы классов TBitField и TSet соответственно. В файлах tbitfield.cpp и tset.cpp содержатся реализации этих классов. В файле sample\_prime\_numbers.cpp, представлено тестирование программ поддержки битового поля на примере решета Эратосфена.

## Описание структур данных

Битовое поле представляет из себя класс TBitField. Данный класс позволяет создать массив элементов типа TELEM, являющегося пользовательским типом данных.

Класс включает в себя следующие поля:

* 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 позволяет работать со следующими методами:

* int GetMemIndex(const int n) const; // индекс в pМем для бита n
* TELEM GetMemMask (const int n) const; // битовая маска для бита n
* int GetLength(void) const; // получить длину (к-во битов)
* void SetBit(const int n); // установить бит
* void ClrBit(const int n); // очистить бит
* int GetBit(const int n) const; // получить значение бита

Имеются следующие перегрузки битовых операций:

* 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

## Описание алгоритмов

**Класс TBitField:**

1. Конструктор



Сначала идет проверка на неотрицательное чило. Далее идёт подсчёт количества памяти, которое затем используется для создания динамического массива размером (BitLen / (sizeof(int) \* 8) + 1).

1. Конструктор копирования



В MemLen и BitLen присваиваем значения полей bf, затем выделяем новую память под необходимое количество бит. В цикле происходит поэлементное копирование данных.

1. Деструктор



Если есть элементы, то происходит очистка памяти pMem, затем присваиваивается BitLen и MemLen значение 0.

1. GetMemIndex



Проверка на вход в диапазон массива. Возвращает индекс бита n.

1. GetMemMask



Возвращает битовуя маску числа.

1. GetLength



Возвращает длину битового поля.

1. SetBit



Проверка на вход в диапазон массива. В массив pMem записываем битовое представление числа n. Индекс и битовую маску для n получаем при помощи методов GetMemIndex и GetMemMask.

1. CLrBit



Проверка на вход в диапазон массива. В массив pMem на индекс числа n записывается нулевой набор битов или другими словами, происходит поразрядная конъюнкция с инверсией числа.

1. GetBit



Проверка на вход в диапазон массива. Возвращает битовое представление числа n умноженное на свою маску.

1. Перегрузка оператора присваивания



Выполняется проверка на уникальность чисел. Затем происходит очистка и выделение памяти по размеру bf. В цикле происходит копирование данных в новый массив. Возвращает новый массив.

1. Перегрузка оператора сравнения

  
Выполняется проверка на различную длину. Затем идет поэлементное сравнение элементов между собой до первого различия. Если такого нет возвращает true, иначе false.

1. Перегрузка оператора сравнения

  
Аналогичная реализация (11).

1. Перегрузка битового «или»



Создается временный массив для записи результатов. В массив tmp копируются значения массива bf. Затем в цикле в массив pMem, который относится к tmp, записываются значения дизъюнкции между дву битами. Возвращает tmp.

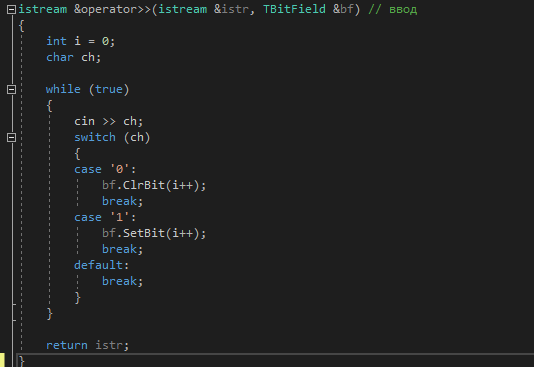
1. Перегрузка битового «и»



Аналогично (13)

1. Отрицание

  
Поэлементно инвертируются значения битового поля. Возвращает tmp.

1. Ввод  
     
   
2. Вывод



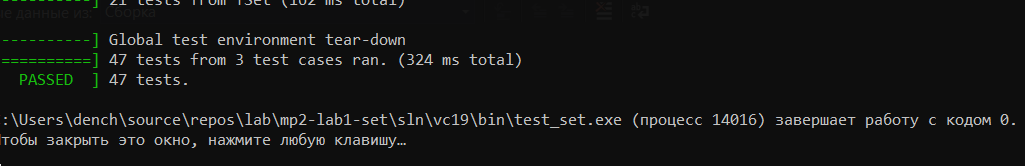
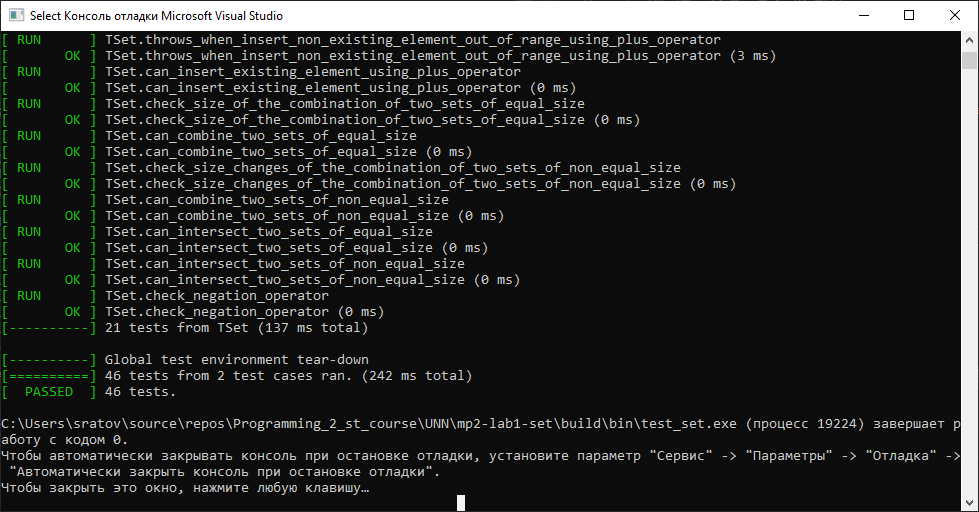
Поэлементный вывод всех значений с помощью метода GetBit.

**Класс TSet:**

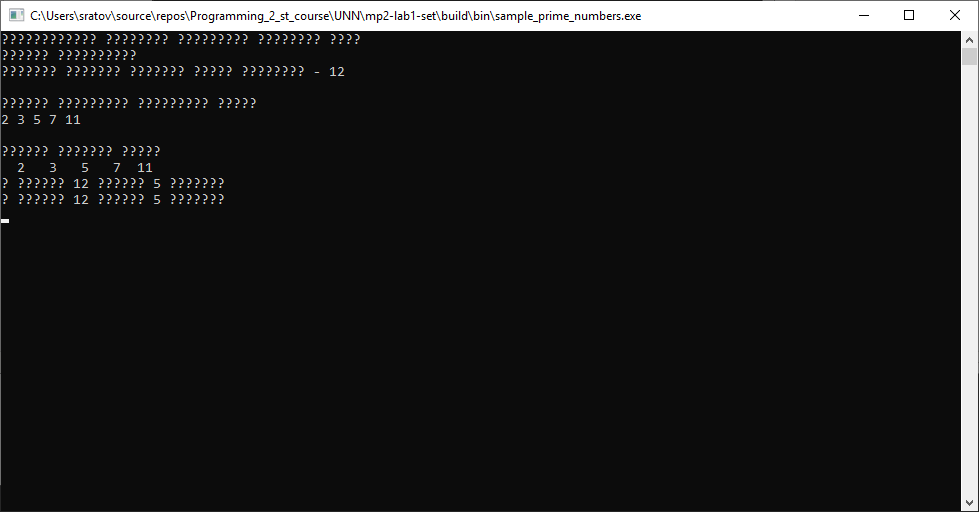
1. Конструктор  
     
   BitField и MaxPower инициализируются максимальной мощностью множества.
2. конструктор копирования  
     
   В BitField и MaxPower присваиваем значения s.
3. конструктор преобразования типа  
     
   BitField присваивается bf. MaxPower присваивается длина bf.
4. получить макс. к-во эл-тов  
     
   Возвращает MaxPower.
5. элемент множества?  
     
   Проверка на принадлежность с помощью метода GetBit. Возвращает True или false.
6. включение элемента множества  
     
   Добавление элемента в множество с помощью метода SetBit.
7. исключение элемента множества  
     
   Удаление элемента из множества с помощью метода ClrBit.
8. присваивание  
     
   BitField и MaxPower присваивается множество s.
9. сравнение  
     
   Сравнение BitField с множеством s.
10. сравнение  
      
    Аналагочно (9)
11. объединение  
      
     Поразрядное или BitField с множеством s. Возвращает результат операции.
12. объединение с элементом  
      
    Выделение временной памяти. Бит нового массива меняется на передаваемый. Создается и возвращается новое измененное множество.
13. разность с элементом  
      
    Выделение временной памяти. Бит нового массива удаляется. Создается и возвращается новое измененное множество.
14. пересечение  
      
    Возвращается новое множество измененное после поразрядной конъюнкции BitField с s.
15. дополнение  
      
    Возвращается новое инвертированное множество.
16. ввод  
      
    Циклический ввод элементов в множество
17. вывод  
      
    Поэлементный вывод всех элементов множества с помощью метода IsMember.

# Эксперименты

1. Программа проходит все тесты Google



1. Программа выводит правильный результат проверки простых чисел и имеет форматированный вывод (код символов сбит из-за англоязычной версии, но тем не менее все верно):



# Заключение

Получены навыки работы с GoogleTests и CMake.

# Литература

1. Столлингс, В. Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 896 с.: ил. — Парал. тит. англ.
2. Johnson M. Superscalar Microprocessor Design. — Englewood Cliff, New Jersey: Prentice Hall, 1991.
3. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.
4. Stone H. High performance Computer Architecture. — Reading, MA: Addison-Wesley, 1993.
5. Tullsen D.M., Eggers S.J. Effective Cache Prefetching on a Bus-Based Multiprocessor. — ACM Transactions on Computer Systems, pp. 57-88, Feb 1995.
6. Chandra D., Guo F., Kim S., Solihin Y. Predicting inter-thread cache contention on a chip multi-processor architecture. — Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA), pp. 340–351, Feb 2005.
7. Press W., Teukolsky S., Vetterling W., Flannery B. Numerical Recipes in C. The Art of Scientific Computing. Second Edition. — Cambridge University Press, 1992.
8. Камаев А.М., Сиднев А.А., Сысоев А.В. Об одном подходе к анализу эффективности приложений // Труды 50-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук»: Часть I. Радиотехника и кибернетика. - М.: МФТИ, 2007.
9. Debugging and performance monitoring. Intel® 64 and IA-32 Architectures Software Developer’s Manual. Volume 3B: System Programming Guide, Part 2. May 2007. — [http://www.intel.com/products/processor/manuals/]
10. Юнаковский А.Д. Начала вычислительных методов для физиков. – Н. Новгород: ИПФ РАН, 2007.